Numerical Hydrodynamics in GR

Some notes for the Southampton Summer School, July 2019.

Key differences with vacuum

Carsten has introduced numerical vacuum relativity. First I want to highlight the differences in principle when moving to the matter models.

Can't do the "right thing"

In vacuum we have Einstein's field equations that we believe to be "correct" at all relevant scales. Ignoring modified gravity, we take the continuum Einstein equations as being the "right" thing to do and assume we want to get this continuum solution.

In the matter case we know that we're approximating particles, and depending on the problems we want to tackle we have to think more or less about the particulate nature of what we're doing. Electromagnetic and neutrino emission require thinking harder about the individual particles involved; the bulk motion of the huge number of particles that go into a neutron star allow us to average over them, leading to a fluid approximation. When we take a numerical approximation the scales at which the model "fails" are much larger - still currently well below what we can simulate, but not so far out of reach that we should forget them.

Note that this necessarily means that we're throwing away the "correct" physics at the smallest scales. We have to incorporate these effects through closure relations. This comes in many forms. The equation of state, for example, links macro-scale observables based on micro-scale physics. In simulations, we'll also have to include effects like viscosity in similar ways through "sub-grid" models.

Variety of relevant models

With vacuum there's there are a few models used: full GR, approximations like CFC, and Newtonian gravity (maybe with modified potentials). However, it's now possible to use the full theory in most cases without approximation.

In the matter case there's a huge range of physical effects to consider, leading to a huge range of potential models that could be included. Starting from hydrodynamics, we can add magnetic fields through MHD or up to more complex EM couplings including, for example, resistivity, and we can add elasticity to model the crust, neutrinos to model cooling, and multifluids to model effects like superfluids or superconductivity. The list goes on. This makes the parameter space to cover horribly large, and parameter extraction in principle much harder.

Form of solutions

In vacuum the solutions are generically smooth, although there's some loss of regularity near singularities in the gauges currently used.

For matter it is *generic* that discontinuities will form. This is "obvious" when you think about neutron star mergers or supernovae: at the most dynamic points in their evolution, shocks will form. This puts restrictions on the numerical methods that we can use, and forces us to think carefully about the mathematical foundations of the solutions and the numerics.

Discontinuities have been used to emphasize the importance of having the equations of motion in conservation law form. We'll come back to why in a second, but we should note that there's no reason to assume that the equations we care about are conservative in the sense we want: the vacuum equations aren't, and the most general multifluid equations aren't either. However, we'll always have conservation of *total* stress-energy-momentum.

Summary

- Complex, interlinked models.
- Conservation laws will apply, but may need some additional non-conservative equations.
- Micro/meso-scale effects need including through closure relations.

Conservation laws

Stress-energy

Start from conservation of total stress energy,

$$\nabla_a T^{ab} = 0.$$

To get this into a form we can solve numerically, choose a tetrad $\{e_a^{(j)}\}$: four orthonormal vectors. Typically these will be associated with the computational coordinates, so you can loosely think of them as ∂_t, ∂_x and so on, but they generalize to more complex coordinate systems. Contract our conservation equation with a tetrad vector to get

$$\begin{aligned} \nabla_a \left(T^{ab} e_b^{(j)} \right) &= -T^{ab} \partial_a e_b^{(j)}, \\ \implies \quad \frac{1}{\sqrt{-g}} \partial_a \left(\sqrt{-g} T^{ab} e_b^{(j)} \right) &= -T^{ab} \partial_a e_b^{(j)}. \end{aligned}$$

We can rewrite this final equation by multiplying through the metric determinant and splitting off the time derivative to get the *balance law form*

$$\partial_t \mathbf{q} + \partial_i \mathbf{f}^{(i)}(\mathbf{q}) = \mathbf{s}(\mathbf{q}).$$

The crucial feature is that the spatial derivatives (∂_i) are in total derivative form. If there were no source terms ($\mathbf{s} \equiv \mathbf{0}$) then this would be a classical *conservation law*. In GR the source terms are geometric. In numerical methods theory we can (mostly!) focus on the principal part and ignore the source term when discussing shocks and other key features.

However, the stress-energy tensor only gives us four equations. Essentially it gives equations of motion for the total linear momentum and the total energy. For hydrodynamic models we also expect to need *at least* one more equation for the density of the fluid. In more general cases we'll need equations for the electromagnetic fields, for individual constituents, for elastic stresses, and so on. Some of these naturally give conservation or balance law forms - for example, ideal and resistive MHD and ideal elasticity can be written this way - but more complex models won't.

Shocks

To show why shocks form we move away from relativity for a second and consider the two toy models, both conservation laws, which result from linearizing hydrodynamics in different ways. First look at the advection equation, where the "density" q is advected to the right at a constant speed v:

$$\begin{aligned} & \partial_t q + \partial_x (vq) = 0, \\ \implies \qquad & \partial_t q + v \partial_x q = 0. \end{aligned}$$

Here the flux is $f = vq$ and the speed with which information moves is $\partial_q f = v$, as expected. We can draw a characteristic diagram in the $x - t$ plane showing the information moving to the right with speed v.

Now move on to Burger's equation,

$$\begin{aligned} & \partial_t q + \partial_x \left(\tfrac{q^2}{2} \right) = 0, \\ \implies \qquad & \partial_t q + q \partial_x q = 0. \end{aligned}$$

This approximates the acoustic modes in a fluid, driven by the fluid pressure. Here the flux is $f = q^2/2$ and the speed with which information moves is $\partial_q f = q$: it depends on the value of the solution itself. We can imagine putting down initial data like $q_0 \sim \sin(x)$. The information at the origin does not move, and nor does that at $x = \pi$. However, the information between $x = 0$ and $x = \pi$ moves to the right, as $q > 0$, whilst that between $x = \pi$ and $x = 2\pi$ moves to the left, as $q < 0$. Drawing a characteristic diagram we see that the characteristics *cross*.

This indicates that a shock will form. At a shock the solution jumps discontinuously, so the spatial derivative ∂_x no longer makes sense. We have to change what we think of as a solution in order to get something sensible. Looking at a more general conservation law,

$$\partial_t q + \partial_x f(q) = 0,$$

we instead interpret this in a weaker sense. Restrict attention to a small spatial volume V of width $|V|$. Then consider the *integral average* $\hat{q}$, defined as

$$\hat{q} = \frac{1}{|V|} \int_V q(x) \, \mathrm{d}x.$$

We then look at the *weak form*

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{q} + \oint_{\partial V} f(q) = 0.$$

Convert the surface integral into a volume integral, assuming that q is continuous within the volume, to get

$$\frac{1}{|V|} \int_V \left(\partial_t q + \partial_x f(q)\right) \, \mathrm{d}x = 0.$$

So the weak form is equivalent to the (strong) conservation law when q is continuous; it also makes sense when q is discontinuous.

This sort of trick can be generalised; we'll come back to it.

Choosing the volume V in a smart way can give us essential results, especially if we consider the volume in both space and time. In particular, if we assume that the volume is small (and take the limit as it goes to zero) and contains *exactly one* discontinuity, we can derive the *Rankine-Hugoniot* jump conditions:

$$V_s[\mathbf{q}] = [\mathbf{f}].$$

Here V_s is the speed of the shock, and the square brackets indicate the jump in the quantity across the shock. So $[\mathbf{q}] = \mathbf{q}_R - \mathbf{q}_L$, for example. Note that, for a system, this is multiple conditions: we hope that (for a system of size N) there are N solutions, but there is no guarantee.

With the Rankine-Hugoniot conditions we can look at one crucial, painful, result. Look at Burger's equation,

$$\partial_t q + \partial_x \left(\tfrac{1}{2} q^2\right) = 0.$$

The Rankine-Hugoniot conditions give us the shock speed (analytically in this case, which is rare):

$$\begin{aligned} V_s(q_R - q_L) &= \tfrac{1}{2}(q_R^2 - q_L^2) \\ \implies \qquad V_s &= \tfrac{1}{2}(q_R + q_L). \end{aligned}$$

That's fine. But next we want to look at the conservation laws

$$\partial_t q^n + \partial_x \left(\tfrac{n}{n+1} q^{n+1}\right) = 0.$$

A similar calculation leads to the shock speeds

$$V_s = \frac{n}{n+1} \frac{q_R^n - q_L^n}{q_R^{n+1} - q_L^{n+1}},$$

which are generically different for different n. *However*, these different conservation laws can be simplified when q is continuous, and they all simplify to

$$\partial_t q + q\partial_x q = 0.$$

The smooth solutions will be identical, independent of n. The discontinuous solutions will be different, with a shock speed depending on n.

The conclusion to draw from this is the weak, conservation law form is the fundamental form that matters: the strong form cannot, by itself, define the shock speed when the solution is discontinuous.

This is a serious problem for us, for example when using multifluids, where the equations can't be put in conservation law form. There is work on properly defining shocks and shock speeds for non-conservative products, but this is hard.

Euler equations

Let's look at the Newtonian Euler equations. They're fundamental to motivating how we model neutron stars as fluids. The conservation law form is

$$\partial_t \begin{pmatrix} \rho \\ \rho v \\ E \end{pmatrix} + \partial_x \begin{pmatrix} \rho v \\ \rho v^2 + p \\ (E+p)v \end{pmatrix} = \mathbf{0}.$$

Here ρ is the fluid density, v the velocity (in the x direction), p the fluid pressure, and E the total energy. E is given as $E = \rho(e + \frac{1}{2}v^2)$, where e is the internal energy. The key *primitive* quantities are ρ, v and e. The *conserved* quantities that we evolve are $\rho, \rho v$ and E: density, momentum and energy. To close the system we need an *equation of state* relating p to other thermodynamic quantities. We'll typically think of this naively as $p = p(\rho, e)$, but we need to take care when generalising this.

We can compare bits of the Euler equations to the advection and Burgers equations. For example, the equation for the density (the continuity equation) looks like an advection equation, and the equation for the energy does as well if the energy dominates over the pressure. Similarly, when the pressure and density are roughly constant, the momentum equation looks like Burgers equation.

If we want to add gravity via a Newtonian gravitational potential ϕ, where the acceleration is roughly $\nabla\phi$, we get

$$\partial_t \begin{pmatrix} \rho \\ \rho v \\ E \end{pmatrix} + \partial_x \begin{pmatrix} \rho v \\ \rho v^2 + p \\ (E+p)v \end{pmatrix} = \begin{pmatrix} 0 \\ -\rho\partial_x\phi \\ -\rho v\partial_x\phi \end{pmatrix}.$$

This adds a source term that doesn't depend on derivatives of the fluid quantities. This changes the bulk behaviour, but doesn't change the local behaviour, so we can keep the numerical methods roughly the same.

If we want to move to special relativity we get

$$\partial_t \begin{pmatrix} \rho W \\ \rho h W^2 v \\ E \end{pmatrix} + \partial_x \begin{pmatrix} \rho W v \\ \rho h W^2 v^2 + p \\ (E+p)v \end{pmatrix} = \mathbf{0}.$$

Here $W = (1-v^2)^{-1/2}$ is the Lorentz factor and accounts for the length contraction and time dilation effects in moving from the fluid frame to the lab frame where we will do the computation. The specific enthalpy $h = 1 + e + p/\rho$ accounts for the self-energy of the fluid - essentially how $E = mc^2$ comes in at relativistic speeds. The total energy $E = \rho h W^2 - p$ is written rather differently to the Newtonian case, again reflecting this self-energy effect.

There are two crucial features to note, one physical and one numerical. The physical one is the introduction of the Lorentz factor. This couples the equations much more tightly than the Newtonian case. Certain physical effects, include jet acceleration, can in principle be linked to this.

The numerical effect comes from the more complex relation between the primitive and conserved variables. Look back at the Newtonian equations. We are going to numerically solve for the conserved variables $\mathbf{q}$, which are the density, momentum and energy. We need to get back to the primitive variables (density, velocity and internal energy), in part because we need to compute the pressure from the equation of state that depends on them. To do this solely from $\mathbf{q}$ we divide the momentum by the density to get the velocity, then use the energy, density and velocity to get the internal energy.

In the relativistic case the more complex couplings between the terms make this much harder. Whilst it is possible in some cases to do this conversion in closed form, it's usual to solve this numerically. It's a nonlinear algebraic equation, so some sort of Newton-Raphson scheme is standard. This approach is definitely necessary as we move to GR and to more complex systems than a single fluid.

Finally, look at the GR Euler equations:

$$\partial_t \sqrt{\gamma} \begin{pmatrix} \rho W \\ \rho h W^2 v_j \\ \tau \end{pmatrix} + \partial_i \sqrt{-g} \begin{pmatrix} \rho W \left(v^i - \frac{\beta^i}{\alpha}\right) \\ \rho h W^2 v_j \left(v^i - \frac{\beta^i}{\alpha}\right) + p \delta^i_j \\ \tau \left(v^i - \frac{\beta^i}{\alpha}\right) + p v^i \end{pmatrix} = \sqrt{-g} \begin{pmatrix} 0 \\ T^{\mu\nu}\left(\partial_\mu g_{\nu j} - \Gamma^\mu_{\nu j}\right) \\ \alpha\left(T^{\mu 0}\partial_\mu \log\alpha - T^{\mu\nu}\Gamma^0_{\mu\nu}\right) \end{pmatrix}.$$

The geometric source terms are a mess and generally uninformative. The bulk effect of the geometry come in through the volume of the spacetime element, either through $\sqrt{-g}$ (for the full spacetime volume element) or $\sqrt{\gamma}$ (for the spatial 3-volume element). The coordinate effects, as we saw in the $3+1$ decomposition, lead to the "advective" velocities that appear in the fluxes being "corrected" to take into account the lapse and shift. Note that as the pressure is isotropic you'll see some of the "regularities" in the simpler cases break in GR.

Crucially, the general structure is still that of a balance law. If we can get techniques to work for the Newtonian Euler equations, they will often transfer across to full GR with only minor modifications.

Numerics

That was a lot of background theory. Let's look at the numerics, concentrating on implementing in small systems (like advection, Burgers, and the Euler equations), remembering that it should transfer to full GR.

Finite volume approach

We'll focus on one spatial dimension. Take the domain on which we want to solve, V, and split it into sub-domains V_i. This will often be called a grid, or a mesh. We can now apply our weak form to any sub-domain,

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{q}_i + \oint_{\partial V_i} f(q) = 0,$$

where

$$\hat{q}_i = \frac{1}{|V_i|}\int_{V_i} q(x)\,\mathrm{d}x.$$

As we're restricting to one dimension we can simplify this. Write $V_i = [x_{i-1/2}, x_{i+1/2}]$, and $|V_i| = |x_{i+1/2} - x_{i-1/2}| = \Delta x$. We can then write the weak form as

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{q}_i + \frac{1}{\Delta x}\left(f(q_{i+1/2}) - f(q_{i-1/2})\right) = 0.$$

The things we will evolve are $\hat{q}_i$. In each cell we will evolve the integral average of the state vector, such as the density. The form that we have is an *ordinary* differential equation, provided we have a way of going from the integral average $\hat{q}$ to the value of the function itself at the boundary of the domain, $x = x_{i\pm1/2}$. Solving an ODE is a standard problem in numerical methods and we usually don't do anything clever here. The standard in the field is to use Runge-Kutta methods of the form that you'll find in standard libraries or numerical methods courses.

The two problems are in getting the data at the cell boundaries in the first place, and then computing the fluxes f.

The problem with going from $\hat{q}_i$, which is a set of numbers that "live" at the centre of the cells x_i, to $q(x_{i\pm1/2})$, which is a set of numbers that live at the boundaries of the cells, is an interpolation problem. There are many standard interpolation methods. The simplest (often called the Godunov method in this context, or nearest-neighbour interpolation more generally), would set

$$q(x_{i\pm1/2}) = \hat{q}_i.$$

That is, it assumes that q is constant within the cell.

This works fine, but isn't very accurate. To do better is more complex, as we need a method that works at discontinuities. There is a general theorem that says, roughly, that trying to do better interpolation within a cell containing a discontinuity will lead to oscillations. This is linked to *Gibb's oscillations* that you may know from Fourier theory. This general problem leads to interpolation methods that try and detect, implicitly or explicitly, whether there might be a shock in a cell, and "do the best that they can". This is the *reconstruction* step in High Resolution Shock Capturing (HRSC) methods, and is probably the main source of inaccuracy in the schemes. The simplest algorithms that are better than Godunov are the *slope limited* schemes. The best general schemes right now are variants on the *Weighted Essentially Non-Oscillatory* (WENO) schemes.

Once we have the values at the cell boundary, the weak form makes it look like we can immediately compute $f(q(x_{i-1/2}))$ and hence update the ODE. However, this is misleading. We have *two* values for q at $x_{i-1/2}$: one from cell $i-1$ and one from cell i. These will not agree. Physically and mathematically there's no reason they should: there might be a shock at that point. So we need a method of calculating a single value at the cell boundary, either for the state vector $q_{i-1/2}$ or directly for the flux $f_{i-1/2}$. This is typically referred to as a *Riemann solver*: given two constant values $q_{L,R}$ either side of an interface at $x = 0$, compute the solution $q(x = 0)$ for all future time.

There are many methods of doing this of varying complexity. As the size of the system gets large it is often not possible, let alone practical, to exactly solve this problem. In many cases the simplest possible solution is good enough. We make the *assumption* (which is nearly always wrong) that the solution is described by two waves propagating to the left and right and the maximum speed compatible with stability: $V = \Delta x/\Delta t$. Using the Rankine-Hugoniot conditions and the conservation law form we can directly get the *Rusanov* or *global Lax-Friedrichs* flux

$$f(x=0) = \frac{1}{2}\left(f_L + f_R + \frac{\Delta x}{\Delta t}\left(q_L - q_R\right)\right).$$

As with many Riemann solvers this writes the solution as an average of the two neighbouring values and a correction term.

Improving the Riemann solver can give much better capturing of certain discontinuities (very weak shocks, or linear discontinuities, or certain magnetic discontinuities), often at considerable computational cost. Improving the reconstruction method typically gives much better bulk accuracy, particularly in smooth regions. In neutron star modelling it's typically the latter that we need, but the former may be important for capturing things like jet structure.

Discontinuous Galerkin methods

There's a couple of issues with finite volume methods (and the related finite

difference methods that are often used). They're related to the reconstruction step where we take the $\hat{q}_i$ to get the values of q at the cell boundaries.

The first issue is one of waste. When we interpolate the $\hat{q}_i$ we construct an approximate functional form for q everywhere. However, we then throw all that information away after computing the boundary values, only to have to compute it all again for the next step. Can't we use a better way of storing the information in the cell than just the cell average, that could be updated directly?

The second issue is one of computational cost. When we interpolate the $\hat{q}_i$ we need to use neighbouring cell values to build the reconstruction. Depending on how much accuracy we want we'll need more cell values. In one dimension for current typical approaches we'll need 4 (four) neighbours on either side, and to increase accuracy we'll need more. This information from the neighbouring cells needs communicating to the original cell. This can be very expensive, essentially because of how modern computers work. Particularly for near-future (*exascale*) machines, they are designed to do local calculations very fast. To communicate data that isn't in local memory, especially if it's on another processor on a parallel machine, can slow down the calculation by *orders of magnitude*.

Discontinuous Galerkin methods should get around this problem. There's a big push in certain groups to use these methods, particularly for the spacetime, but in some cases for fluids as well. The idea starts by revisiting the weak form.

Let's start from

$$\partial_t q + \nabla \cdot f = 0.$$

We know we need a weak form, which means we need to integrate away that divergence. Instead of doing this directly (as in the finite volume case), we'll first multiply by a smooth function ϕ. Integrating by parts over a volume V gives

$$\int_V \phi \partial_t q + \oint_{\partial V} \phi f - \int_V f \cdot \nabla \phi = 0.$$

The only term that is now differentiated is ϕ, which we get to choose, and which we'll always choose to be smooth. So this is well defined. We can now choose a *function basis* to represent q and ϕ, and we'll end up with a set of equations that we can solve.

The standard approach, called the *Galerkin* method, is to use the same function basis approximation for both q and ϕ. It is crucial that we ensure that the function basis is free to jump at the boundaries of each cell: we need this to capture shocks. This freedom gives us the *Discontinuous Galerkin* method. The simplest approximation is the choose a constant (zero order polynomial) approximation within each cell. That is, within cell V_i,

$$q = \hat{q}_i,$$
$$\phi = \hat{\phi}_i.$$

Plugging this into the generalised weak form gives us back the finite volume method.

The more general approach is to use, for example, Legendre polynomials. Then we write, within cell V_i

$$q = \sum_{m=0}^{M} \hat{q}_m P_m(x).$$

The weak form can then be written as an equation for the *mode coefficients* $\hat{q}_m$. With some tedious effort our weak form can now be written as

$$M\partial_t \hat{\mathbf{q}} + S^T f(\hat{\mathbf{q}}) = -[\phi \mathbf{F}]_{x_{i-1/2}}^{x_{i+1/2}}.$$

Here $\hat{\mathbf{q}}$ contains all the modes $\hat{q}_m$ within cell V_i. M is the *mass matrix* and S the *stiffness matrix*. They can both be pre-computed and depend only one the basis functions (here $P_m(x)$) and their derivatives.

The big advantage here is that the solution q is specified everywhere in terms of the basis functions. Evaluating it in order to, for example, get the data needed for the boundary fluxes on the right-hand-side, is immediate and cheap. This means we don't need information from any neighbours *except* the immediate neighbour (when computing the boundary flux). Therefore DG methods should be cheaper (as all the mode information is kept between steps) and more efficient on modern machines.

However, there's extra machinery that needs computing and storing, and the computational cost of solving for a single cell increases enormously. It's also problematic with shocks, as avoiding Gibbs' oscillations requires careful modification of the modal coefficients, and typically re-introduces the coupling between cells. It's unclear right now if moving to DG methods actually gains anything on complex relativistic systems with current hardware.

Gridding

Even the most accurate numerical method is approximating, and the error it makes is proportional to the size of its grid. Keeping with uniform grids we usually have

$$\mathcal{E} \sim C(\Delta x)^s,$$

where we only worry about the leading order (worst) term. For the Godunov method we'll have $s = 1$ and the error converges slowly. Typically people use PPM ($s = 3$) or high order WENO methods ($s \geq 3$) which have much better error convergence properties but are much more expensive. DG methods have $s \sim M$, but also tend to be much better in absolute terms (the constant C is typically much smaller). Near shocks, formally $s = 1$ in all cases, but we tend to *hope* that the shock is only a very small contributor to the total error (this isn't always true).

In practical terms a neutron star computation needs the grids to cover the stars and the near-zone of the gravitational waves at a minimum. The absolute

minimum would be roughly 600km from the centre of mass to the edge of the domain: more would be better. At the same time, we need enough cells across each star to resolve the motion: within the stars a resolution of 500m would be the absolute largest you could get away with. In three dimensions this leads to a bit over 10 billion cells. Within each cell you need to solve for the spacetime and the matter.

This isn't realistic. We need to cut down the number of cells massively. Exploiting symmetries can help but at best gains us a factor of 10. There are two tricks that are used to get the number down to something realistic.

First is *adaptive mesh refinement* (AMR). The full domain is covered by a very course grid (with cell resolution of say 10km). The parts of the domain with "interesting" behaviour (such as those containing the neutron stars) are covered by finer grids. The finer grids update the points on the coarse grid to maintain accuracy. The coarse grids provide boundary conditions for the fine. By nesting refined grids we can typically cut the number of cells needed down by three or four orders of magnitude.

Second is *non-Cartesian* grids, particularly spherical grids. Away from the neutron stars the spacetime is roughly axially symmetric. This means that a spherical coordinate system is better adapted to the problem, and we can use low resolution in the angular coordinates, and clever spacing (such as logarithmic) in the radial coordinate. In principle the coordinate freedom to work in arbitrary coordinates is built in to the equations we use: in practical cases there are some complications. Matching between different grids and different coordinate systems gets the benefits without the problems (such as coordinate singularities at the origin), but is technically painful.

Both of these tricks can be conceptually tricky to implement, especially on parallel machines. I'd strongly recommend using a library.

Constraints and MHD

Largely, adding more complexity increases the cost (by increasing the number of variables evolved) and the error (for the same reason) and the difficulty in interpreting the results, but does *not* change the underlying methods required. There is one qualitative step we haven't covered, which is constraints.

The prototypical constraint is the "no monopoles" condition in MHD,

$$\nabla \cdot \mathbf{B} = 0.$$

If this constraint is violated then it leads to the generation of spurious electric fields and spurious forces on the magnetic field lines. In numerical simulations, this typically leads to nonsensical results within only a very few timesteps. Similar constraints appear in a number of other models, such as elasticity. However, MHD has the "worst" behaviour and has seen the most study.

There are two methods typically used to ensure the constraints are satisfied "well enough" to not cause problems for the numerics.

The first is *constrained transport*. The numerical scheme is modified to ensure that the constraint is enforced to machine precision. There are variants on this, where sometimes the particular variables evolved are modified - for example, evolving the vector potential instead of the magnetic field directly - but typically it requires fixing some elements of the numerical scheme. This can make other aspects, such as AMR, more complex (or impossible) to implement.

The second is *constraint damping*. Additional fields corresponding to the constraints are added to the system, and source terms proportional to the constraints are added as well. The proportionality constants are chosen so that the constraints should be driven to zero exponentially quickly. This approach is flexible and works with any numerical scheme, but introduces free parameters and doesn't always interact well with boundaries (including AMR boundaries). It also allows the constraints to be violated, making numerical accuracy crucial.

At present a combination of both approaches is preferred for MHD. For more complex systems there's real questions as to what's the "best" approach to use.

Key phrases for near-future work

We've touched on the building blocks of most current work. Here are a few topics that may be important in the near future.

Well-balancing

In a static neutron star model the pressure gradient perfectly balances the geometric source terms. In the inspiral phase for a neutron star merger, we're nearly stationary so again these terms should nearly perfectly balance.

Numerically this is asking for a large term computed with one approach (HRSC) to (nearly) perfectly cancel another term computed with a different approach (often central differencing). This never happens and is a major source of error.

In Newtonian work, particularly with the shallow water equations (modelling tsunamis and the like), there are similar problems. Techniques have been developed to make sure these terms *perfectly* balance, and the discrete level, when the situation is stationary. Those techniques can't be directly translated to GR because of the form of the equations, but work in this direction could significantly improve the absolute accuracy of the simulations.

Positivity preserving schemes

We have to use conserved variables to solve for shocks. These make no sense outside the neutron star, in the "vacuum" exterior. Setting the density to zero

causes the methods to fail here, so a "low" density atomsphere is typically used, with effects on the final results that are hard to quantify.

Positivity preserving schemes get around the zero-density issues at the numerical level. Radice has used them in GR hydro simulations to effectively set the atmosphere arbitrarily small. Extending this to MHD is, however, hard.

Adaptive model refinement

Changing the grid resolution in different areas of space and time to match the accuracy requirements is one thing. A more complex approach would be to change the physical model that you're simulating.

The idea would be that, at the coarsest resolutions, we only need "simple" models (like GRMHD) to get the bulk features correct. In small regions of spacetime where we adapt the grid resolution we can also increase the complexity of the model by including resistivity or elasticity. This will only work with short-range effects (so will be difficult with, for example, neutrino transport), but is being used for some simulations of pulsar magnetospheres.

Path conservative methods

Reading list

This is a very incomplete list of sources I regularly use.

Reviews

- Font, Numerical Hydrodynamics and Magnetohydrodynamics in General Relativity, Living Review. Last updated in 2008 but crucial background.
- Marti & Müller, Grid-based Methods in Relativistic Hydrodynamics and Magnetohydrodynamics, Living Review. SR only but updated in 2015.
- Balsara, Higher-order accurate space-time schemes for computational astrophysics—Part I: finite volume methods, Living Review. Very methods heavy. Cutting edge but not easy going.

Theses

- Radice, Advanced Numerical Approaches in the Dynamics of Relativistic Flows. From 2013, touches on a number of important technical details.

Books

- Leveque, Finite Volume Methods for Hyperbolic Problems, CUP. No astrophysics but one of the standard numerical methods texts.
- Hesthaven, Numerical Methods for Conservation Laws: From Analysis to Algorithms, SIAM. Still no astrophysics and even more mathematical-technical, but goes deep into methods like Discontinuous Galerkin and spectral elements which may be the future direction of the field.
- Rezzolla & Zanotti, Relativistic Hydrodynamics, OUP. From 2013, its focus is on hydrodynamics, not MHD. Lots of detail.
- Alcubierre, Introduction to 3+1 Numerical Relativity, OUP. From 2012, its focus is really vacuum relativity, but introduces hydrodynamics well from that viewpoint.

Codes and tutorials

- Open Astrophysics Bookshelf. Relativity isn't a focus but the material covers a lot of numerics in great depth, with example codes throughout. Have a look at github.com/python-hydro for detailed examples in one and two dimensions.
- Einstein Toolkit. A production GRMHD code that runs on massively parallel machines. There's a steep learning curve and it's designed to do a broad range of things (so it's more complex than it needs to be to do any *one* thing), but this can be used for real research.