Discrete techniques for Numerical Relativity

Manuel Tiglio

Hearne Institute for Theoretical Physics

Louisiana State University

In collaboration with G. Calabrese, L. Lehner, D. Neilsen, J. Pullin, O. Reula, O. Sarbach

ITP, May 03

LSU Relativity

A far from complete list of numerical questions

Choosing the system to solve: coordinate conditions, free or constrained evolution, boundary conditions.

Once the problem has been defined, one needs to solve it in a numerically stable way.

Numerical stability = convergence: the property that errors go way with resolution.

Several issues that need to be considered for numerical stability:

- How does one discretize near boundaries?
- If doing extrapolation near a corner or a curvilinear boundary, in which direction should one do so? Does higher order extrapolation help?
- If using cubic boxes as computational domains, how does one impose boundary conditions at corners and edges?
- How shall one use dissipation near boundaries?
- Can one avoid artificial, numerical errors that grow fast in time?

Overview, main ideas

- Method of lines: from semidiscrete to fully discrete stability.
- Numerical stability as a discrete version of well posedness.
- Numerical stability through discrete energy estimates:
  Summation by parts
  Boundary conditions
- Going beyond numerical stability: preventing errors from growing.
- Application: black hole excision.
- Dynamical control of discrete constraint violations.
- Multi-patch evolution.

The method of lines

- Given a set of differential equations,

$$\partial_t u = A^i(u,\vec{x},t)D_i u + B^i(u,\vec{x},t)u$$

- Semidiscrete problem: first discretize space but not time:

$$\frac{d}{dt}u_{i,j,k} = A^i(u,x_{i,j,k},t)D_i u_{i,j,k} + B^i(u,x_{i,j,k},t)u_{i,j,k}$$

- Prove numerical stability for the semidiscrete problem (system of ODE's).

- If now you use appropriate time integrators, stability for the fully discrete problem follows. The details of what you did in the semidiscrete case do not matter, provided that the semidiscrete problem was stable.

- Get spatial discretizations that give semidiscrete stability, and time integrators that are "locally stable". Combine them at will.

Numerical stability: a discrete version of well posedness

Two reasons for looking at numerical stability for linear problems:

It is a necessary condition for stability in the non-linear case.

The solution to the nonlinear problem $\partial_t u = A^i(u,\vec{x},t)D_i u + B^i(u,\vec{x},t)u$

can sometimes be shown to be the solution to an iteration of linear problems:

$$\partial_t u^{n+1} = A^i(u^n,\vec{x},t)D_i u^{n+1} + B^i(u^n,\vec{x},t)u^{n+1}, \quad n = 0,1,2,...$$

Well posedness: existence and uniqueness of a solution with some smoothness, plus an energy estimate.

Define a scalar product $(u,v) = \int <u, Hv> dx$

and a norm or "energy" E(t) = (u,u), and show that

$$E(t) \leq f(t)E(0) + \text{boundary terms} \leq f(t)E(0)$$

Same f(t) for all initial data!

- Numerical stability: replace in the above expressions the integral by sum over grid points,

$$(u,v)_{\Delta x} = \Delta x \sum_{i,j} < u_i, Hv_j > \sigma_{ij}$$

and demand that f(t) does not depend on the resolution.

- Key idea to construct numerically stable schemes: construct discrete operators that allow you to repeat the steps done at the continuum when showing well posedness.

Some details on the key idea: summation by parts and discrete boundary conditions

- Steps followed at the continuum when deriving an energy estimate:
  - Integration by parts
  - Controlling boundary terms through (for example) maximally dissipative boundary conditions.
- Similarly, at the discrete level need one needs to:
  - Construct difference operators that satisfy summation by parts
  - Impose boundary conditions in a way that not only is consistent with the continuum, but also one controls the boundary terms left after summation by parts
  - These 2 steps guarantee numerical stability (roughly, there are more details here).

Summation by parts (SBP): a discrete version of integration by parts

- Integration by parts:

$$(u, \partial_x v) + (v, \partial_x u) = \tfrac{1}{2} uv\big|_0^1$$

- Say the domain goes from 0 to 1. Construct a difference operator D approximating d/dx, and a discrete scalar product such that

$$(u, Dv)_{\square} + (v, Du)_{\square} = \tfrac{1}{2} uv\big|_0^1 \quad \longleftarrow \boxed{\text{SBP}}$$

- The difference operator and scalar product depend only on the geometry of the computational domain, not on the equation you are solving.

- Accuracy: D has less accuracy at the boundary (compared to the interior). Global convergence rate better than the accuracy at the boundary, though (Gustaffsson). High order differencing (or extrapolation) at boundaries can introduce instabilities.

- Existence, properties: Kreiss + Scherer

- Explicit high order constructions in simple domains: Bo Strand 90's.

Discrete boundary conditions

- They have to be consistent with the continuum boundary conditions that guaranteed well posedness.

- But this is **not** enough for numerical stability. One needs to control the boundary terms left after summation by parts.

- One way of doing this is by imposing boundary conditions through an orthogonal projector P (Olsson '95).
- P acts on the space of gridfunctions: P: GF:→GF

- P is a projector: $P^2=P$, and self-adjoint $P=P^T$ → Key property

- Acts on the right hand side of the equation: $u_t = A^i D_i u + Bu$ → $u_t = P(A^i D_i u + Bu)$

Note: Can handle edges and vertices in a numerically stable way (provided you know what to do at the continuum).

Other ways (e.g., Carpenter et. Al.)

Beyond numerical stability: guiding the numerical solution

- Numerical stability guarantees that as resolution is increased the numerical errors go to zero.
- But still, at fixed resolution the errors can grow exponentially (or worse) in time.

$E_t(t)$ = (u,f(x)u)$_{□x}$ + boundary terms + Numerical terms

Continuum terms: obtained through summation by parts and appropriately representing boundary conditions.

Bounded for all resolutions, But potentially dangerous.

Example: say $E_t(t)$ = E(t)□x, then E(t) = exp(t □x) E(0). The norm of the solution stays constant in the limit □x □0, but at fixed resolution it grows in time.

We have to do something!

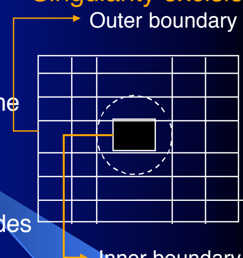
Strict stability: rearrange the discrete equations so that not only you have a bound for the energy, but also this bound is optimal.

□ The numerical solution stays close to the exact one, at each resolution.

□ Need some good energy estimate to guide the numerical simulation.

Application: black hole excision

Singularity excision



- "Hide" the singularity by placing an inner boundary inside the black hole.
- One wants to do that in a well posed way. In particular, make sure that the number of zero speed modes does not change as one moves along the boundary.
- Consider the simplest possible case: 2 concentric cubic boxes.
- We have constructed, for this geometry, dissipative operators, a scalar product and difference operators that satisfy summation by parts.
- Several ongoing projects using these numerical techniques for black hole excision.

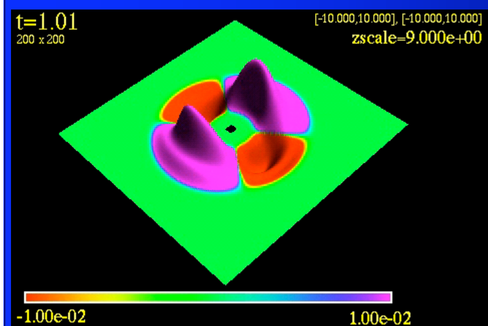
Fields propagating in stationary spacetimes

Discretizing with a numerically stable scheme, without further work, in general gives exponentially growing errors.

These are avoided by constructing the scheme such that the physical energy is non-increasing (or preserved) at the semidiscrete level.

Consider a time indepent system: $u_t = A^i D_i u + Bu$. For any H that satisfies

$\partial_i (HA^i) = HB + (HB)^T$ non-increasing energies E =(u,Hu) can be constructed:



$$\frac{d}{dt} E = \text{boundary terms} \le 0$$

Rearranging the discrete system as

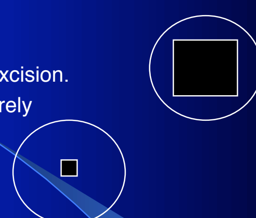
$$\partial_t u = \frac{1}{2} A^i D_i u + \frac{1}{2} H^{-1} D_i (HA^i u) + \left[B - \frac{1}{2} H^{-1} D_i (HA^i)\right] u$$

Implies that the semidiscrete energy is also non-increasing:

$$\frac{d}{dt} E_{\Delta x} = \text{boundary terms} \le 0$$

Square wheels

- Cubic boxes are not very well suited for black hole excision.
- An arbitrary cubic box inside a black hole is not a purely outflow boundary. It has to be small enough.

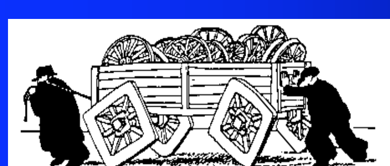


For example, for a Schwarzschild black hole, L~ 0.37M

One can use a piecewise cubic boundary.
A 2D example (Engquist '78): Say P_1 has grid coordinates (0,0).

$$D_x u(P_1) = \frac{1}{2\Delta x}\left(4u_{(0,1)} - 2u_{(0,0)} - u_{(-2,2)} - u_{(2,0)}\right)$$

The difference operator at P_1 depends in a highly non-intuitive way on the stencil near P_1



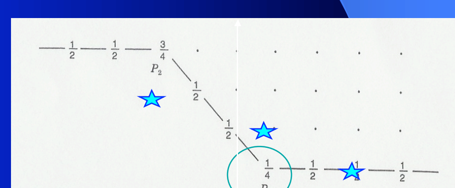
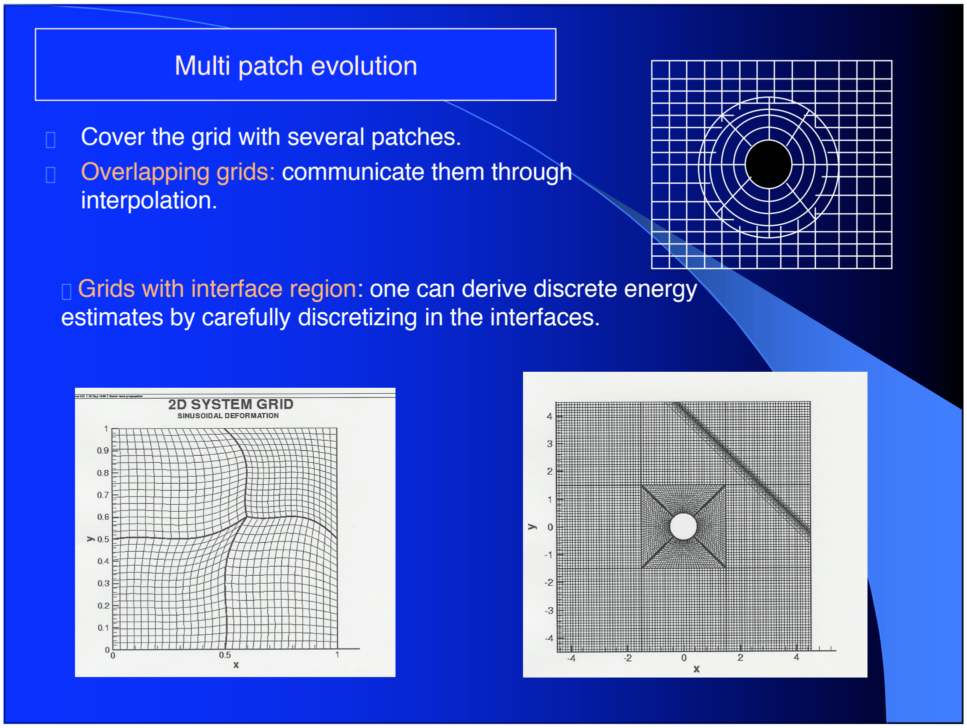
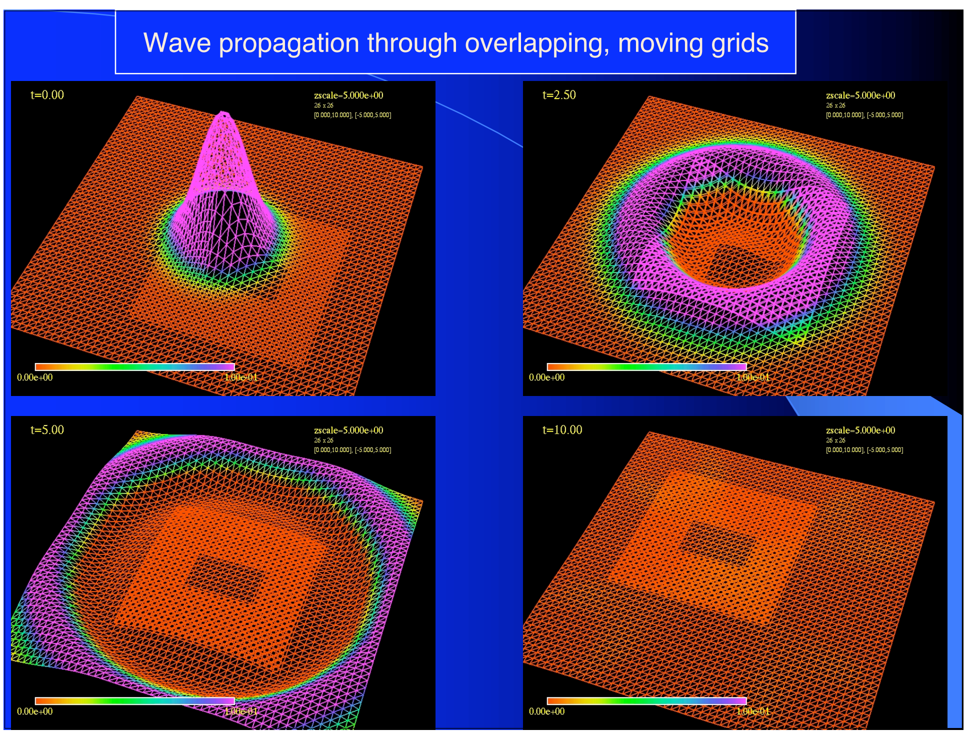


Figure 11.5.2 Domain and weighting coefficients for 45-degree corners.


Multi patch evolution
Cover the grid with several patches.
Overlapping grids: communicate them through interpolation.
Grids with interface region: one can derive discrete energy estimates by carefully discretizing in the interfaces.
2D SYSTEM GRID
SINUSOIDAL DEFORMATION


Wave propagation through overlapping, moving grids
t=0.00
t=2.50
t=5.00
t=10.00

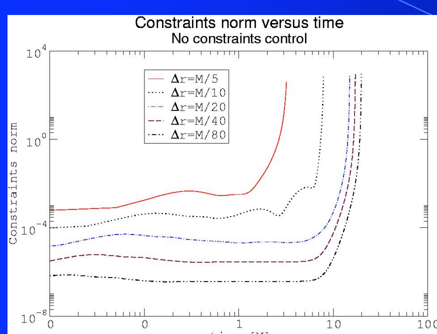
Dynamical control of discrete constraint-violations

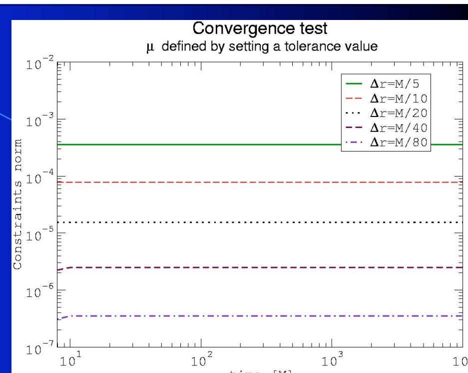
- Given a set of evolution equations, add the constraints $C_i = C_i$ (u, Du), to the right hand side
$$\partial_t u = A^i(u,x^j,t)D_i u + B^i(u,x^j,t)u \rightarrow A^i(u,x^j,t)D_i u + B^i(u,x^j,t)u + \mu C$$
- These evolution equations determine how these constraints propagate
$$\partial_t C = \tilde{A}^i(u,\vec{x},t,\mu)D_i C + \tilde{B}^i(u,\vec{x},t,\mu)C$$
- Define an energy for these constraints E_c = (C,C). It satisfies the evolution equation
$$\frac{d}{dt}E_c = I_1 + \mathrm{trace}(\mu I_2)$$
- Choose m to get the desired behaviour, e.g.
$$\mu = -\frac{I_1}{I_2 \dim[I_2]} \Rightarrow \frac{d}{dt}E_c(t) = 0$$

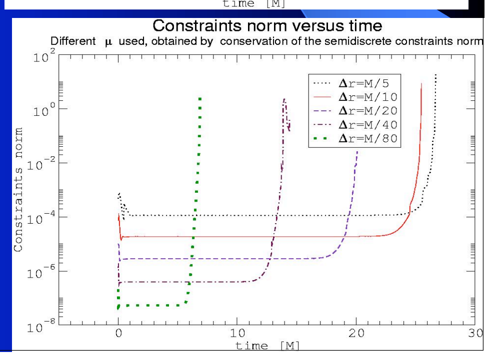
The discretized constraints will not grow in time

- Do it on the fly, during evolution. In order to achieve symmetric hyperbolicity, do it for one resolution, interpolate, and keep that μ(t) from there on. It is then a priori given.
- If you control the constraints (=they do not grow) for one resolution, and you are in the convergence regime, you are controlling them for higher resolutions as well

A proof of principle

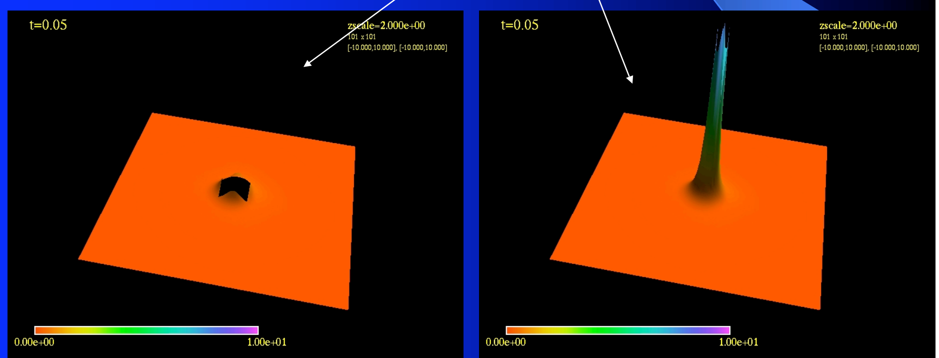






Dynamical minimization of constraints growth for a single black hole in 3D

- Symmetric hyperbolic formulation with live gauges.
- Non-constraint preserving boundary conditions for the moment.
- Physical characteristic speeds.
- Shown the initial discretized constraints, with a “good” inner boundary (small enough so that it is purely outflow) and a “bad” one (not purely outflow).



Conclusions, future directions

- Spherical boundaries: be patient, we will have them for you.
- Well posedness can actually be worked out for first order in time, and second order in space we plan to work out the numerics as well.
- Computing things and trying to answer questions (e.g.,are non-constraint preserving boundary conditions triggering the “typical” constraint instabilities?).
- Fluids.
- Axisymmetric systems.
- Resolution, resolution…