A multi-block infrastructure for three-dimensional time-dependent numerical relativity

Erik Schnetter,[1,2,*] Peter Diener,[3,1,†] Ernst Nils Dorband,[3,1,‡] and Manuel Tiglio[3,1,§]

[1]*Center for Computation and Technology, 302 Johnston Hall, Louisiana State University, Baton Rouge, LA 70803, USA*[¶]

[2]*Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut, Am Mühlenberg 1, D-14476 Golm, Germany*[**]

[3]*Department of Physics and Astronomy, 202 Nicholson Hall, Louisiana State University, Baton Rouge, LA 70803, USA*[††]

(Dated: 2006-02-24)

We describe a generic infrastructure for time evolution simulations in numerical relativity using multiple grid patches. After a motivation of this approach, we discuss the relative advantages of global and patch-local tensor bases. We describe both our multi-patch infrastructure and our time evolution scheme, and comment on adaptive time integrators and parallelisation. We also describe various patch system topologies that provide spherical outer and/or multiple inner boundaries.

We employ *penalty* inter-patch boundary conditions, and we demonstrate the stability and accuracy of our three-dimensional implementation. We solve both a scalar wave equation on a stationary rotating black hole background and the full Einstein equations. For the scalar wave equation, we compare the effects of global and patch-local tensor bases, different finite differencing operators, and the effect of artificial dissipation onto stability and accuracy. We show that multi-patch systems can directly compete with the so-called fixed mesh refinement approach; however, one can also combine both. For the Einstein equations, we show that using multiple grid patches with penalty boundary conditions leads to a robustly stable system. We also show long-term stable and accurate evolutions of a one-dimensional non-linear gauge wave. Finally, we evolve weak gravitational waves in three dimensions and extract accurate waveforms, taking advantage of the spherical shape of our grid lines.

PACS numbers: 04.25.Dm, 02.70.Bf, 02.60.Cb

arXiv:gr-qc/0602104 v1 25 Feb 2006

I. INTRODUCTION

Many of the spacetimes considered in numerical relativity are asymptotically flat. An ideal kind of domain for these has its boundaries at infinity, and at the same time has to handle singularities which exist or develop within the domain. In a realistic setup, the outer boundary is either placed at infinity, which is topologically a sphere, or one introduces an artificial outer boundary at some large distance from the origin. In both cases, the ideal boundary shape is a sphere.

There are several possible ways to deal with singularities. One of the most promising is *excision*, which was first used by J. Thornburg [1], where he acknowledges W. G. Unruh for the idea. Excision means introducing a inner boundary, so that the singularity is not in the computational domain any more. If done properly, all characteristic modes on this inner boundary are leaving the domain, so that no physical boundary condition is required. [2] applies this idea for the first time in a spherically symmetric time evolution.

A well posed initial boundary value problem requires in general smooth boundaries [3]. Using spherical boundaries satisfies all conditions above. Spherical boundaries have not yet been successfully implemented in numerical relativity with Cartesian grids. However, there were many attempts to approximate spherical boundaries. For example, the Binary Black Hole Grand Challenge Alliance used *blending* outer boundary conditions [4, 5], where the blending zone was approximately a spherical shell on a Cartesian grid. Excision boundaries often approximate a sphere by having a Lego (or staircase) shape [5, 6, 7, 8]. Some of the problems encountered with excision are attributed to this staircase shape. A spherical boundary would be smooth in spherical coordinates, but these are undesirable because they have a coordinate singularity on the z axis. A multi-block scheme allows smooth spherical boundaries without introducing coordinate singularities.

Using multiple grid patches is a very natural thing to do in general relativity. When one starts out with a manifold and wants to introduce a coordinate system, then it is a priori not clear whether a single coordinate system can cover all the interesting parts of the manifold. One usually introduces a set of overlapping maps, each covering a part of the manifold. After discretising the manifold, one arrives naturally at multiple grid patches.

Methods using multiple grid patches in numerical relativity were pioneered in 1987 by J. Thornburg [1, 9], where he also introduced excision as inner boundary condition for black holes. It was only in 2004 that a fully three-dimensional time evolution using multiple grid patches was successful [10], and a stationary Kerr

*Electronic address: schnetter@cct.lsu.edu
†Electronic address: diener@cct.lsu.edu
‡Electronic address: dorband@cct.lsu.edu
§Electronic address: tiglio@cct.lsu.edu
¶URL: http://www.cct.lsu.edu/
**URL: http://numrel.aei.mpg.de/
††URL: http://relativity.phys.lsu.edu/

black hole could be evolved in a stable and convergent manner.

In this paper, we describe a generic infrastructure for time evolution simulations in numerical relativity using multiple grid patches, and we show example applications of this infrastructure. We begin by defining our notation and terminology in section II, where we also discuss various choices that one has to make when using multi-patch systems. We describe our infrastructure in section III and the patch systems we use in section IV. We test our methods with a scalar wave equation on flat and curved backgrounds and with the full vacuum Einstein equations in sections V and VI. We close with some remarks on future work in section VII.

II. MULTI-PATCH SYSTEMS

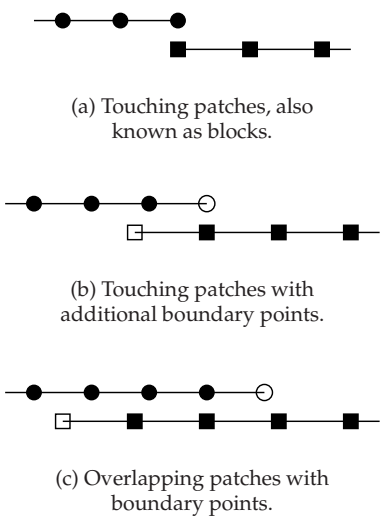
Our main motivation for multi-patch systems is that they provide smooth boundaries without introducing coordinate singularities. This allows us to implement symmetric hyperbolic systems of equations for well posed initial boundary value problems. However, multi-patch systems have three other large advantages when compared to using a uniform Cartesian grid.

No unnecessary resolution. In multi-patch systems the angular resolution does not necessarily increase with radius. An increasing angular resolution is usually not required, and multi-patch systems are therefore more efficient by a factor $O(n^2)$ when there are $O(n^3)$ grid points.

No CFL deterioration for co-rotating coordinates. When a co-rotating coordinate system is used, the increasing angular resolution in Cartesian grids forces a reduction of the time step size. Near the outer boundary, the co-rotation speed can even be superluminal. This does not introduce any fundamental problems, except that the time step size has to be reduced to meet the CFL criterion. This makes Cartesian grids less efficient by a factor of $O(n)$ for co-rotating coordinate systems when there are $O(n^3)$ grid points.

Convenient radially adaptive resolution. In multi-patch systems, it is possible to vary the radial resolution without introducing coordinate distortions [10]. This is not possible with Cartesian coordinates. Fish-eye coordinate transformations [11] lead to distorted coordinate systems. In practise, there is a limit to how large a fish-eye transformation can be, while there is no such limit for a radial rescaling in multi-patch systems.

These advantages are so large that we assume that fixed mesh refinement methods may even be superfluous for many applications in numerical relativity. Mesh refinement can be used adaptively, e.g. to resolve shock waves in a star. It would be difficult to handle this with adaptive patch systems. However, mesh refinement is also used statically to have higher resolution in the centre and lower resolution near the outer boundary. This case is very elegantly handled by multi-patch systems. Multi-patch systems could also be used to track moving features, such as orbiting black holes.



(a) Touching patches, also known as blocks.

(b) Touching patches with additional boundary points.

(c) Overlapping patches with boundary points.

Figure 1: Schematics of the difference between touching and overlapping grids for different inter-grid relations. The black grid points are evolved in time, the hollow grid points are determined via a boundary condition.

A. Terminology

Methods using multiple grid patches are not yet widely used in numerical relativity, and this leads to some confusion in terminology.

The notions of multiple patches, multiple blocks, or multiple maps are all virtually the same. In differential geometry, one speaks of maps covering a manifold. In computational physics, one often speaks of multi-patch systems when the patches overlap, and of multi-block systems if they only touch, i.e., if the blocks only have their boundaries in common. However, when discretised, there is an ambiguity as to what part of the domain is covered by a grid with a certain resolution. See figure 1 for an illustration.

In the following, we say that a grid extends from its first to its last grid point. This is different from Thornburg's notation [10]: He divides the grid into *interior* and *ghost* points.[1] The interior points are evolved in time, and a small number of ghost points are defined through an inter-patch boundary condition, which is interpolation in his case. He defines the grid extent as ranging from the first to the last *interior* point, ignoring the ghost points for that definition. Thus, when there are n ghost

[1] Just to demonstrate that terminology can be really confusing, we note that Thornburg's notion of "ghost points" is different from what Cactus [12] calls "ghost points".

points, he calls "touching" what we would call an "overlap of $2n$ points". When he speaks of overlapping grids according to his definition, then there are parts of the domain covered multiple times, and these overlap regions do not vanish in the continuum limit.

B. Coordinates and tensor bases

Although not strictly necessary, it is nevertheless very convenient to have one global coordinate system covering the whole domain. This makes it very easy to set up initial conditions from known analytic solutions, and it also makes it possible to visualise the result of a simulation. While a true physics-based visualisation would not rely on coordinates, currently used methods always display fields with respect to a coordinate system.

Since our calculations concern quantities that are not all scalars, we have to make a choice as to how to represent these. One elegant solution is to use a tetrad (or triad) formalism, where one represents vectorial and tensorial quantities by their projections onto the tetrad elements. However, this still leaves open the choice of tetrad elements.

In a multi-patch environment there are two natural choices for a coordinate basis. One can either use the global or the patch-local coordinate system. Both have advantages and disadvantages.

In a way, using a patch-local coordinate basis is the more natural choice. Given that one presumably knows how to evolve a system on a single patch, it is natural to view a multi-patch system just as a fancy outer boundary condition for each patch. In this way, one would continue to use patch-local coordinates everywhere, while the inter-patch boundary conditions involve the necessary coordinate transformations. It should be noted that these coordinate transformations mix the evolved variables, because (e.g. for a vector v_i) what is v_1 on one patch is generally a linear combination of all v_i in the other patches' coordinate system.

Using a global coordinate basis corresponds to defining a global triad that is smooth over the entire domain. This simplifies the inter-patch boundary conditions significantly, since there is no coordinate transformation necessary. Instead, one then has to modify the time evolution mechanism on the individual patches:

Let the letters $a, b, c \ldots$ denote abstract indices for quantities in the patch-local tensor basis, and letters $i, j, k \ldots$ abstract indices in the global tensor basis. The triad that defines the global tensor basis is given by e^i_a, which is a set of one-forms e_a labelled by a global index i. A vector field $\mathbf{v}$ is denoted v^a in the patch-local tensor basis and $v^i := e^i_a v^a$ in the global tensor basis. Note that v^i transforms as a scalar with respect to the patch-local tensor basis, as a change in the patch-local tensor basis does not change v^i.

When one calculates partial derivatives, e.g. through finite differences, one obtains these always in the patch-local coordinate basis. It is then necessary to transform these to the global coordinate basis. Since the evolved variables are scalars with respect to the patch-local coordinate basis, their first derivatives are co-vectors, and their transformation behaviour is straightforward:

$$\partial_i \;=\; \frac{\partial x^a}{\partial x^i}\,\partial_a \tag{1}$$

where $\partial x^a/\partial x^i = (\partial x^i/\partial x^a)^{-1} = (e^i_a)^{-1}$ is the (inverse) *Jacobian* of the coordinate transformation. This means that using a global tensor basis effectively changes the system of evolution equations.

Using a global coordinate basis also has advantages in visualisation. Visualisation tools generally expect non-scalar quantities to be given in a global coordinate basis. Often, one also wants to examine certain components of non-scalar quantities, such as e.g. the radial component of the shift vector. When the shift vector is given in different coordinate bases on each patch, then the visualisation tool has to perform a non-trivial calculation.

It should be noted that there are many quantities which have a non-tensorial character, such as e.g. the quantities d_{kij} of the formulation introduced in [13] which we use below; d_{kij} are partial derivatives of the three-metric. The quantities ϕ and $\tilde{\Gamma}^i$ of the BSSN formalism [14] have an even more complex transformation behaviour. $\phi = \ln \sqrt[12]{\det(\gamma_{ij})}$ is the logarithm of a scalar density, and $\tilde{\Gamma}^i = \tilde{\gamma}^{jk}\tilde{\Gamma}^i_{jk}$ is a partial derivative of a tensor density.

It is also necessary to define the set of characteristic variables at an inter-patch boundary in an invariant manner. One convenient way to do so is again to refer to a global coordinate basis. That is, one transforms the elements of the state vector to the global coordinate basis, and can then define the characteristic variables in a unique way.

Last but not least, there is one more compelling argument for using a global coordinate basis to represent the state vector. Since one, presumably, already knows how to evolve the system within a single patch, it may be unwise to place all the new complications that a multi-patch system brings into the inter-patch boundary condition. By changing the evolved system to use a global coordinate basis, one simplifies the inter-patch boundaries significantly, and furthermore one can implement and test both steps separately.

III. INFRASTRUCTURE

We base our code on the Cactus framework [12, 15] using the Carpet infrastructure [16, 17]. Cactus is a framework for scientific computing. As a framework, the core ("flesh") of Cactus itself contains no code that does anything towards solving a physics problem; it contains only code to manage modules ("thorns") and let them

interact. Cactus comes with a set of core thorns for basic tasks in scientific computing, including time integrators and a parallel driver for uniform grids. (A driver is responsible for memory allocation and load distribution on parallel machines.)

By replacing and adding thorns, we have extended Cactus' capabilities for multi-patch simulations. The mesh refinement driver Carpet can now provide storage, inter-processor communication, and I/O for multi-patch systems as well, and the multi-patch and mesh refinement infrastructures can be used at the same time. The definitions of the patch systems (see below) and the particular inter-patch boundary conditions are handled by additional thorns.

A. Infrastructure description

A computation that involves multiple patches requires implementing several distinct features. We decided to split this functionality across multiple modules, where each module is implemented as a Cactus thorn. These are

Driver (D): The driver is responsible for memory allocation and load distribution, for inter-processor communication, and for I/O. It also contains the basic time stepping mechanism, ensuring that the application e.g. updates the state vector on each patch in turn.

Multi-patch system (MP): The patch system selects how many patches there are and how they are connected, i.e., which face is connected to what face of which other patch, and whether there is a rotation or reflection necessary to make the faces match. The patch system also knows where the patches are located in the global coordinate space, so that it can map between patch-local and global coordinates.

Penalty boundaries (P): This thorn applies a penalty boundary condition to the right hand side (RHS) of the state vector of one face of one patch. We have described the details in [18]. It calls other routines to convert the state vector and its RHS on the patch and on its neighbour to and from their characteristic variables; it is thus independent of the particular evolution system.

Finite differences (FD): As a helper module, we implemented routines to calculate high order finite differences on the patches [18]. These operators use one-sided differencing near the patch boundaries.

Additionally, we make use of the following features that Cactus provides:

Time integrator (TI): The time integrator calls user-provided routines that evaluate the RHS of the state vector, advances the state vector in time, and calls boundary condition routines.

Boundary conditions (BC): The boundary condition infrastructure keeps track of what boundary conditions should be applied to what faces and to what variables. It distinguishes between *inter-processor boundaries* (which are synchronised by the driver), *physical boundary conditions* (where the user applies a condition of his/her choice), and *symmetry boundary conditions* (which are determined through a symmetry of the computational domain, e.g. a reflection symmetry about the equatorial plane). We have extended the notion of symmetry boundaries to also include inter-patch boundaries, which are in our case handled by the penalty method.

Together, this allows multi-patch systems to be evolved in Cactus within the existing infrastructure. Existing codes, which presumably only calculate the right hand side (RHS) and apply boundary conditions, can make use of this infrastructure after minimal changes, and after e.g. adding routines to convert the state vector from and to the characteristic modes. Existing codes which are not properly modular will need to be restructured before they can make use of this multi-patch infrastructure.

We use the penalty method to enforce the inter-patch boundary conditions. The penalty method for finite differences is described in [**? ? ?**], and we describe our approach and notation in [18].

In order to treat systems containing second (or higher) temporal derivatives with our current infrastructure, one needs to rewrite them to a form where they only contain first temporal derivatives by introducing auxiliary variables. This is always possible. Strongly or symmetric hyperbolic systems containing second (or higher) spatial derivatives [19, 20, 21, 22, 23, 24] could in principle also be handled without reducing them to first order. The definition of the characteristic modes has then to be adapted to such a formulation, and may then contain derivatives of the evolved variables.

This multi-patch approach is not limited in any way to finite differencing discretisations. It would equally be possible to use e.g. spectral methods to discretise the individual patches (as was done in [25].) It may even make sense to use structured finite element or finite volume discretisations on the individual patches, using a multi-patch system to describe the overall topology.

B. Initial condition, boundary conditions, and time evolution

We now describe how the initial conditions are set up and how the state vector is evolved in time. This explains in some more detail how the different modules interact, and what parts of the system have to be provided

by a user of this multi-patch infrastructure. Each step is marked in parentheses (e.g. "(MP)") with the module that performs that step.

Setting up the initial conditions proceeds as follows:

1. *Initialise the patch system.* (MP) The patch system is selected, and the location and orientation of the patches is determined depending on run-time parameters. If desired, the patch system specification is read from a file (see below).

2. *Set up the global coordinates.* (MP) For all grid points on all patches, the global coordinates are calculated. At this time we also calculate and store the Jacobian of the coordinate transformation between the global and the patch-local coordinate systems. If necessary, we also calculate derivatives of the Jacobian. Derivatives may be required to transform non-tensorial quantities when a patch-local tensor basis is used. The Jacobian can be calculated either analytically (if the coordinate transformation is known analytically) or numerically via finite differences.

3. *Initialise the three-metric.* (User code) Even when the evolved system does not contain the Einstein equations, we decided to use a three-metric. This is a convenient way to describe the coordinate system, which —even if the spacetime is flat— is non-trivial in distorted coordinates. For example, polar-spherical coordinates can be expressed using the three-metric $\gamma_{ij} = \mathrm{diag}\left(1, r^2, r^2 \sin^2\theta\right)$, and doing so automatically takes care of all geometry terms.

   This step is performed by the user code, and it is not necessary to use an explicit three-metric in order to use this infrastructure. At this time, we initialise the metric at the grid point locations, specifying its Cartesian components in the *global* tensor basis.

4. *Initialise the state vector.* (User code) Here we initialise the state vector of the evolution system, also again specifying its Cartesian components in the *global* tensor basis. When evolving Einstein's equations, this step and the previous are combined.

5. *Convert to local tensor basis (if applicable)* (User code) It is the choice of the user code whether the state vector should be evolved in the global or in the patch-local tensor bases. We have discussed the advantages of either approach above. If the evolution is to be performed in patch-local coordinates, then we transform the three-metric and the state vector at this time. Note that this transformation does not require interpolation, since we evaluated the initial condition already on the grid points of the individual patches.

At this stage, all necessary variables have been set up, and the state vector is in the correct tensor basis.

The time evolution steps occur in the following hierarchical manner:

1. *Loop over time steps.* (D) The driver performs time steps until a termination criterion is met. At each time step, the time integrator is called.

   (a) *Loop over substeps.* (TI) We use explicit time integrators, which evaluate the RHS multiple times and calculate from these the updated state vector.

      i. *Evaluate RHS.* (User code) The RHS of the state vector is calculated, using e.g. the finite differencing thorn.

      ii. *Apply boundary conditions to RHS.* (BC) We decided to apply boundary conditions not to the state vector, but instead to its RHS. This is necessary for penalty boundaries, but is a valid choice for all other boundaries as well. In our simulations, we do not apply boundary conditions to the state vector itself, although this would be possible. Both the inter-patch and the outer boundary conditions are applied via the multi-patch infrastructure in a way we explain below.

      iii. *Update state vector.* (TI) Calculate the next —or the final— approximation of the state vector for this time step.

      iv. *Apply boundary conditions to the state vector.* (User code) In our case, nothing happens here, since we apply the boundary conditions to the RHS of the state vector instead.

   (b) *Analyse simulation state.* (D) The driver calls various analysis routines, which e.g. evaluate the constraints, or calculate the total energy of the system, or output quantities to files.

The multi-patch thorn knows which faces of which patches are inter-patch boundaries and which are outer boundaries. Inter-processor boundaries are handled by the driver and need not be considered here. Symmetry boundary conditions (such as e.g. a reflection symmetry) are currently not implemented explicitly, but they can be trivially simulated by connecting a patch face to itself. This applies the symmetry boundary via penalties, which is numerically stable, but differs on the level of the discretisation error from an explicitly symmetry condition.

The boundary conditions are applied in the following way:

1. *Loop over all faces of all patches.* (MP) Traverse all faces, determining whether this face is connected to another patch or not. Apply the boundary condition for this face, which is in our case always a penalty condition.

(a) *Apply a penalty to the RHS.* (P) Apply a penalty term to the RHS of all state vector elements. This requires calculating the characteristic variables at the patch faces.

   i. *Determine characteristic variables.* (User code) Calculate the characteristic variables from the state vector on the patch to which the penalty should be applied. If applying an inter-patch boundary condition, calculate the characteristic variables from the other patches' state vector as well. If this is an outer boundary, then specify characteristic boundary data.

   ii. *Apply penalty.* (P) Apply the penalty correction to the characteristic variables.

   iii. *Convert back from characteristic variables.* (User code) Convert the characteristic variables back to the RHS of the state vector.

The edges and corners of the patches require some care. In our scheme, the edges and corners of the patches have their inter-patch condition applied multiple times, once for each adjoining patch. In the case of penalty boundary conditions, the edge and corner grid points are penalized multiple times, and these penalties are added up. This happens for each patch which shares the corresponding grid points. We have described this in more detail in [18].

For nonlinear equations, there is an ambiguity in the definitions of the characteristic variables and characteristic speeds on the inter-patch boundaries. Since we use the penalty method, the state vector may be different at the boundary points on both sides of the interface. When the metric is evolved in time, then it is part of the state vector, and it may be discontinuous across the interface. The definition of the characteristics depends on the state vector in the nonlinear case. It can thus happen that the characteristic speeds are all positive when calculated at the boundary point on one side of the interface, and all negative when calculated on the other side of the interface. One has to pay attention to apply the penalty terms in a consistent manner even if this is the case. In our scheme (as described above), we always calculate the characteristic information using the state vector on that side of the interface to which the penalty terms are applied. This scheme does not prefer either side of the interface, but it is not fully consistent for nonlinear equations.

Instead of using penalty terms to apply boundary conditions, one could also apply boundary conditions in other ways, e.g. through interpolation from other patches, or by specifying Dirichlet or von Neumann conditions. Our infrastructure is ready to do so, but we have not performed a systematic study of the relative advantages of e.g. penalty terms vs. interpolation.

C. Time integration

It is common in numerical relativity to use explicit time integration methods. These limit the time step size to a certain multiple of the smallest grid spacing in the simulation domain. In non-uniform coordinate systems, one has to determine the smallest grid spacing explicitly, since it is the proper distance between neighbouring grid points that matters, not the grid spacing in the patch-local coordinate systems, and the proper distance can vary widely across a patch. If the three-metric, lapse, or shift are time-dependent, then the proper distances between the grid points also vary with time.

Furthermore, the maximum ratio between the allowed time step size and the grid spacing depends not only on the system of evolution equations; it depends also on the spatial discretisation operators that are used, on the amount of artificial dissipation, and on the strength of the penalty terms. While all this can in principle be calculated a priori, it is time consuming to do so.

Instead, we often use adaptive time stepping, for example using the adaptive step size control of the Numerical Recipes [26, chapter 16.2]. One can specify a time integration accuracy that is much higher than the spatial accuracy, and thus obtain the convergence order according to the spatial discretisation. In practise, one would rather specify a time integration accuracy that is comparable to the spatial accuracy. Adaptive time stepping would also have the above advantages when used on a single Cartesian grid.

We would like to remark on a certain peculiarity of adaptive time stepping. With a fixed time step size, instabilities manifest themselves often in such a way that certain quantities grow without bound in a finite amount of simulation time. Numerically, one notices that these quantities become infinity or nan (not a number) at some point when IEEE semantics [27] are used for floating point operations. With an adaptive step size, this often does not happen. Instead, the step size shrinks to smaller and smaller values, until either the step size is zero up to floating point round-off error, or the time integrator artificially enforces a certain, very small minimum step size. In both cases, computing time is used without making any progress. This case needs to be monitored and detected.

D. Parallelisation

Our current implementation parallelises a domain by splitting and distributing each patch separately onto all available processors. This is not optimal, and it would be more efficient to split patches only if there are more processors than patches, or if the patches have very different sizes. This is a planned future optimisation.

We have performed a scaling test on multiple processors. We solve a simple test problem on a patch system

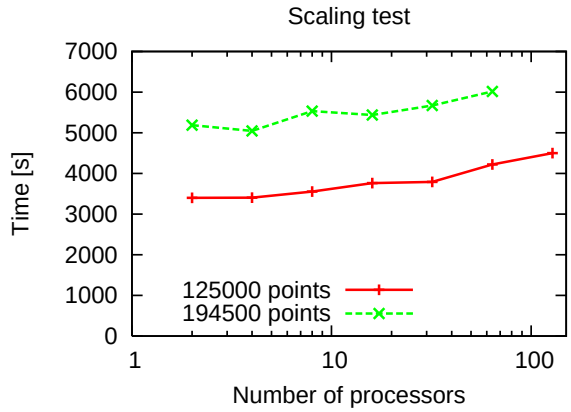


Figure 2: Wall clock time vs. numbers of processors for 100 time steps of a test problem. We keep the load of each processor approximately constant at 125000 and 194500 grid points, respectively. Our implementation scales up to at least 128 processors.

with multiple patches and measure the time it takes to take 100 time steps.[2] As we increase the number of processors, we also increase the number of grid points, so that the load per processors remains approximately constant. This is realistic, because one chooses the number of processors that one uses for a job typically depending on the problem size. Figure 2 shows the results of the scaling tests for two such problem sizes. We find that our implementation scales well up to at least 128 processors, and would probably continue to scale to larger numbers. See [28] for a comparison of other benchmarks using Cactus.

It would also be possible to distribute the domain onto the available processors by giving (at least) one domain to each processor. This would mean that one splits domains when one adds more processors, introducing additional inter-patch boundary conditions. Penalty inter-patch boundary conditions are potentially more efficient than using ghost zones, since they require an overlap of only one single grid point. An nth order accurate finite differencing scheme, on the other hand, requires in general an overlap of $2n$ grid point. Penalty boundary conditions thus require less communication between the patches. A disadvantage of this scheme is that the exact result of a calculation then depends on the number of processors. Of course, these differences are only of the order of the discretisation error.

Such differences are commonly accepted when e.g. elliptic equations are solved. Many efficient algorithms for solving elliptic equations apply a domain decomposition, assigning one domain to each processor, and using different methods for solving within a domain and for coupling the individual domains. The discretisation error in the solution depends on the number of domains. For hyperbolic equations that are solved with explicit time integrators, it is often customary to not have such differences. On one hand, this may not be necessary to achieve an efficient implementation, and on the other hand, it simplifies verifying the correctness of a parallel implementation if the result is independent of the number of processors. However, there are no fundamental problems in allowing different discretisation errors when solved on different numbers of processors, especially if this may lead to a more efficient implementation.

IV. PATCH SYSTEMS

We have implemented a variety of patch systems, both for testing and for standard application domains. It is also possible to read patch systems from file.

Simple testing patch systems are important not only while developing the infrastructure itself, but also while developing applications later on. Since the application has to provide certain building blocks, such as e.g. routines that convert to and from the characteristic representation, it is very convenient to test these in simple situations. Many patches have distorted local coordinate systems, and it is therefore also convenient to have patches with simple (one-dimensional) coordinate distortions. This can be used to test the tensor basis transformations — keeping in mind that some variables will not be tensorial, but will rather be tensor densities, or partial derivatives of tensors, with correspondingly more involved transformation behaviours.

We have currently two types of realistic patch systems implemented:

a. Six patches: This system consists of six patches that cover a spherical shell, i.e., a region with $r_{\text{min}} \le r \le r_{\text{max}}$. We use the same patch-local coordinates as in [29] and [18]. This system is useful if the origin is not part of the domain, e.g. for a single black hole. See figure 3 for an illustration.

b. Seven patches: This system consists of one cubic patch that covers the region near the origin, and six additional patches that cover the exterior of the cube until a certain radius r_{max}. We use the same patch-local coordinates as in [18], which are derived from the six-patch coordinates above. This system is useful if the origin should be part of the domain, e.g. for a single neutron star. See figure 4 for an illustration.

In addition to these two types, we have variations thereof, e.g. a system consisting of only one of the six patches assuming a sixfold reflection symmetry. We have individual patch types as generic building blocks,

[2] This test was performed with a 4th order Runge–Kutta integrator, the scalar wave equation formulated in a patch-local tensor basis, a seven-patch system, the D_{6-5} differencing operators, and a Mattsson–Svärd–Nordström dissipation operator. We varied the number of grid points per patch from 65^3 to 253^3 to keep the load per processor approximately constant. See section V A below, where these details are explained.

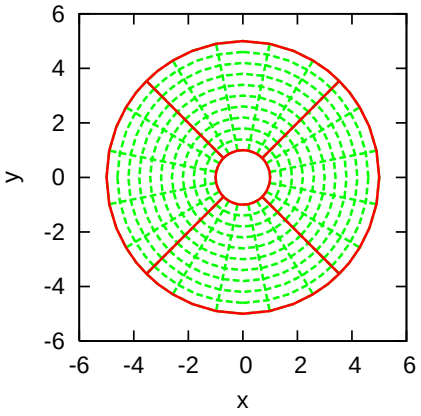


Figure 3: A cut in the equatorial plane of six patches, in which four patches are visible. The outer and inner domain boundaries are spheres. There is one radial coordinate spanning $r = \text{const}$ surfaces, and two angular coordinates perpendicular to that. The radial coordinate is smooth across patch boundaries.

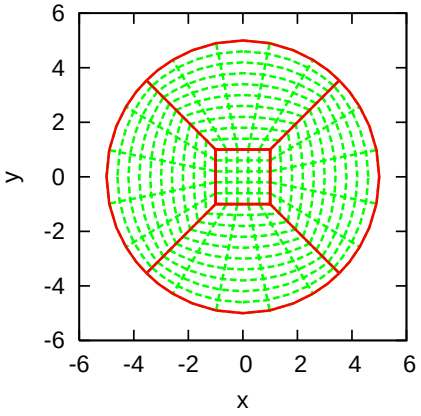


Figure 4: A cut in the equatorial plane of seven patches, in which five patches are visible. The outer boundary is a sphere, the inner patch is a cube. There is again one radial coordinate, but it does not span $r = \text{const}$ surfaces and it is not smooth across patch boundaries except at the outer boundary. The two angular coordinates are the same as in the six-patch system.

and we can glue them together to form arbitrary patch systems.

After seeking input from the computational fluid dynamics community, where multi-block systems are commonly used to obtain body-fitted coordinate systems, we decided that setting up patch systems by hand is too tedious, and that commercial tools should be used for that instead.[3] We therefore implemented a patch system reader that understands the GridPro [30] data format.

[3] We are indebted to our esteemed colleague F. Muldoon for teaching us about the state of the art in grid generation.

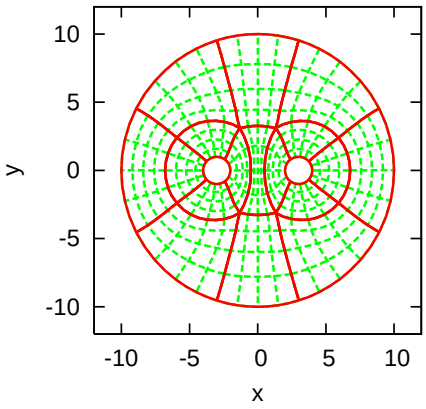


Figure 5: A cut in the equatorial plane of the imported binary black hole patch system. The outer and inner boundaries are spheres. Near the boundaries, the coordinate system is similar to spherical coordinates, i.e., there is one coordinate direction perpendicular to and two direction tangential to the boundary.

This is a straightforward ASCII based format which is specified in the GridPro documentation, and support for other data formats could easily be implemented as well.

Using GridPro, we could easily import patch systems with two holes and 27 patches (for a generic binary black hole system; see figure 5) and with e.g. 30 holes and 865 patches (for demonstration purposes; see figure 6). Another advantage of a tool like GridPro is that the grid points are automatically evenly distributed over the domain, which may be difficult to ensure if the grid is constructed by hand.

V. TESTS WITH THE SCALAR WAVE EQUATION

We test our multi-patch infrastructure with a scalar wave equation on an arbitrary, time-independent background. Since we express the coordinate distortions via a generic three-metric, there is —from the point of view of the code— no difference between a flat and a curved spacetime. A stationary black hole background requires non-trivial lapse and shift functions, but these are desirable even for flat spacetimes: a non-zero shift makes for moving or rotating coordinate systems (implementing fictitious forces), and a non-unity lapse could be used to advance different parts of the domain with different speeds in time, which can improve time integration efficiency (although we did not use it for that purpose).

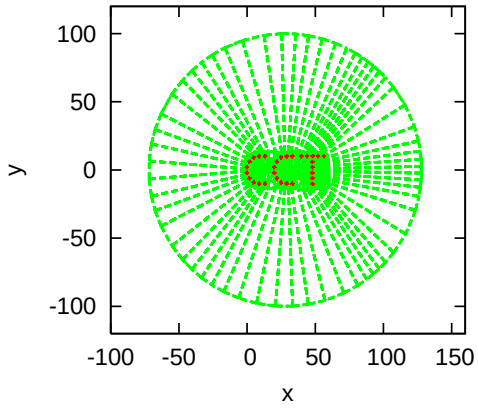
We use the notation α for the lapse, β^i for the shift vector, γ_{ij} for the three-metric, γ^{ij} for its inverse, and $\gamma = \det(\gamma_{ij})$ for its determinant.

We evolve the scalar wave equation

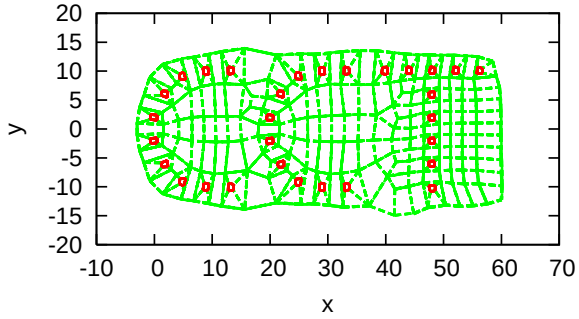
$$\Box u = 0 \tag{2}$$

by introducing the auxiliary variables

$$\rho = \partial_t u \tag{3}$$



(a) whole domain



(b) region near the origin

Figure 6: A cut in the equatorial plane of the imported demonstration patch system with 30 spherical holes in arbitrary positions close to the centre. The resolution in the centre is much higher than near the outer boundary, demonstrating how to achieve the same effect as fixed mesh refinement with multiple patches.

$$v_i \;=\; D_i u. \tag{4}$$

This renders the system into a first order form, leading to the time evolution equations

$$\partial_t u \;=\; \rho \tag{5}$$

$$\partial_t \rho \;=\; \beta^i \partial_i \rho + \frac{\alpha}{\sqrt{\gamma}} \partial_i \left[\frac{\sqrt{\gamma}}{\alpha} \left(\beta^i \rho + \alpha^2 H^{ij} v_j \right) \right] \tag{6}$$

$$\partial_t v_i \;=\; \partial_i \rho \tag{7}$$

with

$$H^{ij} \;=\; \gamma^{ij} - \beta^i \beta^j / \alpha^2. \tag{8}$$

If discretised with operators that satisfy summation by parts, this system is numerically stable with respect to the energy

$$E \;=\; \frac{1}{2} \int \frac{\sqrt{\gamma}}{\alpha} \left[\rho^2 + \alpha^2 H^{ij} v_i v_j \right] \, dV \tag{9}$$

for $|\beta| < \alpha$. In this case, this system is symmetric hyperbolic, and its characteristic variables and speeds are listed in [29].

We present below evolutions of the scalar wave equation on a flat background with the seven patch system and on a Kerr–Schild background with the six patch system. We compare the respective benefits of using a global or a patch-local tensor basis, and we study the behaviour of scalar waves on a Kerr–Schild background as a test problem.

A. Comparing global and patch-local tensor bases

We present here time evolutions of the scalar wave equation on a flat background with the seven-patch system. We compare two formulations, one based on a global, the other based on a patch-local tensor basis. We also apply a certain amount of artificial dissipation to the system, which is necessary because our formulation has non-constant coefficients. We use two different kinds of artificial dissipation, which were introduce by Kreiss and Oliger [**?**] and by Mattsson, Svärd, and Nordström [31], respectively.

We set the initial condition from an analytic solution of the wave equation, namely a traveling plane wave. We also impose the analytic solution as penalty boundary condition on the outer boundaries. This is in the continuum limit equivalent to imposing no boundary condition onto the outgoing characteristics and imposing the analytic solution as Dirichlet condition onto the incoming characteristics. The patch system has the outer boundary at $r = 3$, and the inner, cubic patch has the extent $[-1; +1]$. We use the penalty strength $\delta = 0$ at the inter-patch and at the outer boundaries. See [18] for our notation for the penalty terms.

The traveling plane wave is described by

$$u(t, x_i) \;=\; A \, \cos \left[2\pi \left(k_i x^i + \omega t \right) \right] \tag{10}$$

with $\omega^2 = \delta^{ij} k_i k_j$. We set $A = 1$ and $k_i = [0.2, 0.2, 0.2]$, so that the wave length is $5/\sqrt{3}$. We construct the solutions for ρ and v_i from u via their definitions (3) and (4), and evaluate these at $t = 0$ to obtain the initial condition. The flat background has $\alpha = 1$, $\beta^i = 0$, and $\gamma_{ij} = \delta_{ij}$ in the global tensor basis; the patch-local metric is constructed from that via a coordinate transformation.

We use dissipation operators that are compatible with summation by parts (SBP) finite difference operators. Introduced by Mattsson, Svärd, and Nordström in [31], we call them "MSN" operators. They are constructed according to

$$A_{2p}^{\mathrm{MSN}} \;=\; -\frac{\epsilon}{2^{2p}} \, h^{2p} \, \Sigma^{-1} D_p^T B_p D_p, \tag{11}$$

where $2p$ is the order of the interior derivative operator, ϵ is the dissipation strength, h is the grid spacing,

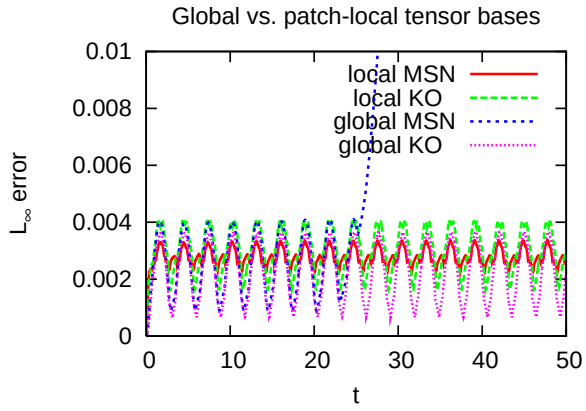


Figure 7: Comparing global and patch-local tensor bases, and Mattsson–Svärd–Nordström (MSN) and Kreiss–Oliger (KO) dissipation operators. These graphs show the L_∞ norm of the solution error vs. time for a coarse resolution on a seven-patch system. Some artificial dissipation is necessary to stabilise the system, since it has non-constant coefficients. For this particular value of the dissipation strength ϵ, using a global tensor basis is unstable with the MSN dissipation operators, but stable with the KO operators. With higher values of ϵ, the system is stable for both dissipation operators.

Σ is the norm with respect to which the derivative operator satisfies SBP, D_p is a consistent approximation of a pth derivative, and B_p is a diagonal matrix. The scaling with grid spacing is in contrast to standard Kreiss–Oliger (KO) dissipation operators [**?**], where the scaling h^{2p-1} is used. Experience has shown that with the MSN scaling it is sometimes necessary to increase the dissipation strength when resolution is increased in order to maintain stability. For this reason we have implemented SBP compatible KO dissipation operators constructed according to

$$A_{2p}^{\mathrm{KO}} \;=\; -\frac{\epsilon}{2^{2p+2}}\, h^{2p+1}\, \Sigma^{-1} D_{p+1}^T B_p D_{p+1}, \qquad (12)$$

where as before Σ is the norm of the $2p$th order accurate SBP derivative operator. This yields a dissipation operator with KO scaling that has the same accuracy as the SBP derivative operator near the boundary (and one order higher in the interior). The price is having to use a slightly wider stencil.

We use 21 grid points in the angular and in the radial directions. The central patch also has 21 grid points in each direction. This is a very coarse resolution. We use the D_{6-5} stencil of [18], which is globally sixth order accurate, and add compatible artificial dissipation to the system of both MSN and KO type as described above. We choose a dissipation coefficient $\epsilon = 3.0$. We use a transition region that is 0.3 times the size of the patch. The overall system is then sixth order accurate.

With a patch-local tensor basis and diagonal norm operators the system is strictly stable, i.e., the numerical error is at any given resolution bounded (up to boundary terms) by a constant (see also [18]), while with a global tensor basis, a small amount of artificial dissipation is required. However, since we are using restricted full norm operators for this test, dissipation is required for both the patch-local and patch-global case.

Figure 7 shows the L_∞ norm of the solution error vs. time up to $t = 50$. The discretisation using MSN dissipation is unstable for $\epsilon = 3.0$ when a global tensor basis is used, but it is stable when a local tensor basis is used. Larger values of ϵ also stabilise the global tensor basis discretisation. For the KO dissipation, a dissipation strength $\epsilon = 3.0$ is sufficient to stabilise both the local and global tensor basis formulations. Note that the error levels are very similar in all cases, showing that the main difference between the patch-local and patch-global tensor basis implementations is that more dissipation is necessary in order to stabilise the system.

B. Scalar wave equation on a Kerr–Schild background

We also present time evolutions of the scalar wave equation on a Kerr–Schild [32, section 3.3] background with six patches, which we use to excise the singularity. We choose the mass $M = 1$ and the spin $a = 0.9$ for the background. We place the inner boundary at $r = 1.4$ and the outer boundary at $r = 201.4$.

We use as initial condition $u(0, x^i) = 0$, $v_i(0, x^i) = 0$, and a modulated Gaussian pulse

$$\rho(0, x^i) \;=\; Y_{\ell m}\, A \exp\left[-\frac{(r-R)^2}{W^2}\right], \qquad (13)$$

where we choose the multipole $\ell = 2$, $m = 2$, the amplitude $A = 1$, the radius $R = 20$, and the width $W = 1$. We use conventions such that

$$Y_{22} \;=\; \sqrt{\frac{15}{32\pi}}\, \sin^2\theta \left(\cos^2\phi - \sin^2\phi\right). \qquad (14)$$

We impose $u = 0$, $\rho = 0$, and $v_i = 0$ with the penalty method as outer boundary conditions. This means that this condition is imposed onto the incoming characteristic modes. Since the inner boundary is an outflow boundary, no boundary condition is imposed there. At the outer boundary, ρ is indistinguishably close to zero at $t = 0$ (much closer than the floating point round-off error), so that there is no noticeable discontinuity to the initial condition. We use again the penalty strength $\delta = 0$ for both inter-patch and outer boundaries.

We use the patch-local tensor basis for this example. We use 21 grid points in the angular directions and 1001 grid points in the radial direction. We use here —for no particular reason— different discretisation parameters. It is our experience that the stability of the system does not depend on the particular choice of stencil, as long as it satisfies summation by parts [18]. We use here the D_{8-4} stencil, which is globally fifth order accurate, and add compatible MSN artificial dissipation to the system.

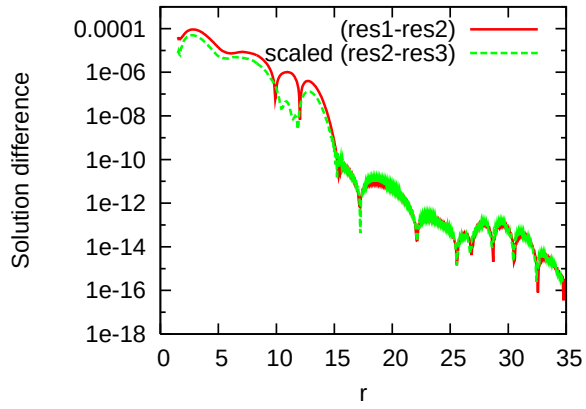


Figure 8: Self convergence test for a scalar field on a Kerr background with $a = 0.9$ at $t = 19.2$. The coarsest resolution has $21 \times 21 \times 1001$ grid points per patch, and the two higher resolutions have $1.5\times$ and $2\times$ more resolution. The coarsest radial resolution is thus $M/20$. The higher resolution graph has been scaled for 5th order convergence. For $r < 15$ we see about 6th to 7th order convergence, for $r > 15$ we see 5th order convergence.

We choose a dissipation coefficient $\epsilon = 0.2$, and we do not scale the dissipation with the grid spacing h. The overall system is then fifth order accurate. We use a fixed time step size $\Delta t = 0.05$ with a fourth order accurate Runge–Kutta integrator. Figure 8 shows that this setup converges to fifth order.

Figure 9 shows a snapshot of the simulation at $t = 92.2$. At that time, the wave pulse has traveled approximately half the distance to the outer boundary. The inter-patch boundaries are smooth, although the configuration is not axisymmetric. Figure 10 shows extracted wave forms from this simulation for the $\ell = 2$, $m = 2$ and for the $\ell = 4$, $m = 2$ modes. The $\ell = 2$ mode is present in the initial condition. The $\ell = 4$ mode should not be present; it is only excited via discretisation errors, since our grid setup is not spherically symmetric, but has an $m = 2$ structure. We have verified that the $\ell = 4$ modes converge away in the continuum limit.

We determine the complex quasi normal frequency $\omega = \omega_R + i\omega_I$ from the extracted wave form of the $\ell = 2$, $m = 2$ mode. We fit the wave seen by an observer at radius $r = 5$ to a function

$$f(t) \;=\; A \sin(\omega_R t - \phi) \exp(\omega_I t). \qquad (15)$$

This fit is performed for the real and imaginary part of the complex frequency as well as for the amplitude A and a phase ϕ. For a spin of $a = 0.9$, we obtain a frequency $\omega = 0.783269 - 0.0723018i$. This is in good agreement with the known exact values $\omega_{\text{exact}} = 0.781638 - 0.0692893i$. For the $\ell = 2$, $m = -2$ mode, we measured a frequency $\omega = 0.387875 - 0.0939477i$, while the exact value is $\omega_{\text{exact}} = 0.387710 - 0.0935902i$ [33, 34]. A detailed study of scalar wave evolutions on a Kerr–Schild background is in preparation [35].

For reasons of comparison between a code using the global tensor basis and one using the patch-local tensor basis, we evolved a similar physical system using both of these methods. We now choose a spin of $a = 0.5$ and initial data with an $\ell = 2$, $m = 0$ angular dependency. The frequency obtained from a simulation using the global tensor basis is $\omega = 0.491153 - 0.09496i$, the one computed using the local tensor basis is $\omega = 0.491347 - 0.095004i$, while the exact value is $\omega_{\text{exact}} = 0.491962 - 0.094630i$ [33, 34]. We find that the choice of tensor basis has little influence on the accuracy of the results.

VI. EVOLVING THE VACUUM EINSTEIN EQUATIONS

We evolve the vacuum Einstein equations using the symmetric hyperbolic formulation introduced in [13]. It includes as variables the three metric g_{ij}, the extrinsic curvature K_{ij}, the lapse α, and extra variables denoted by d_{kij} and A_i. When all the constraints are satisfied, these are related to the three-metric and lapse by $d_{kij} = \partial_k g_{ij}$ and $A_k = \partial_k \ln \alpha$. The formulation admits any of the Bona–Masso slicing conditions while still being symmetric hyperbolic. The shift has to be specified in advance as an arbitrary function of the spacetime coordinates t and x^i. In the tests below, we use a time harmonic slicing condition and a time-independent shift. The characteristic modes and speeds are listed in [13].

Previous $3D$ black hole simulations using this formulation were presented in [36], using a low order Cartesian code and cubic excision. In [37], constraint-preserving boundary conditions for this formulation were constructed; this paper then studies the well-posedness of the resulting initial-boundary value problem, and tests a numerical implementation of those boundary conditions in fully non-linear $3D$ scenarios as well, again with a Cartesian code.

After some initial (and quite lengthy) experiments with a patch-local tensor basis, we decided to use a global tensor basis instead.[4] We find that patch-local tensor bases increase the complexity of the inter-patch boundary conditions very much, because the characteristic decomposition of the field variables needs to be combined with the tensor basis transformation at the patch boundaries. On the other hand, converting the partial derivatives into the global tensor basis is trivial in comparison. In addition to that, analysing and visualising the output of a simulation is also made much easier when a global tensor basis is used.

[4] We are grateful to O. Sarbach for suggesting and insisting on this.

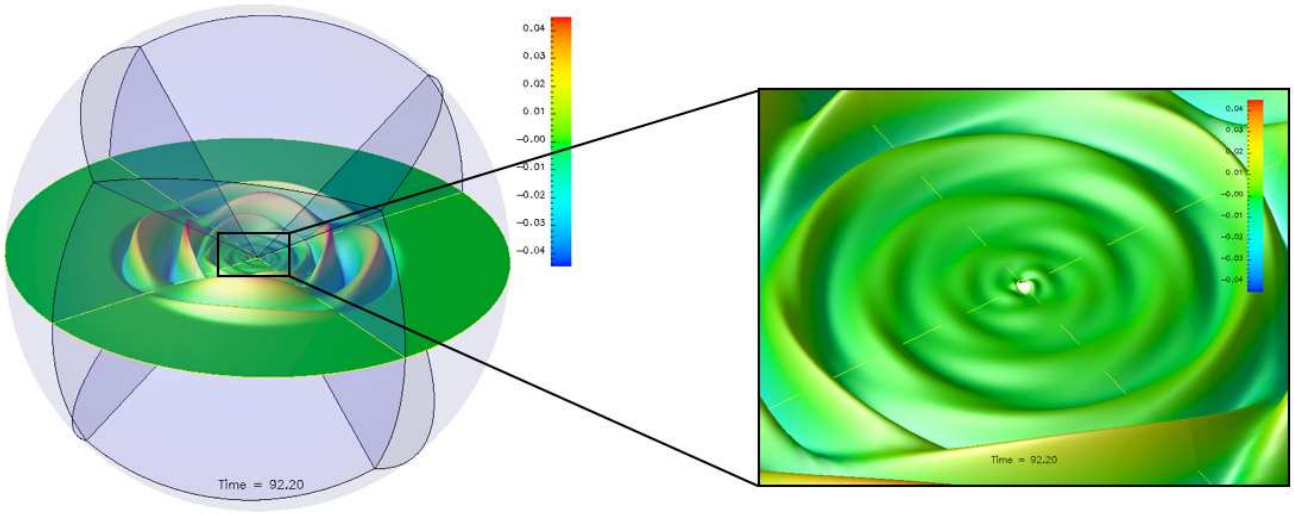
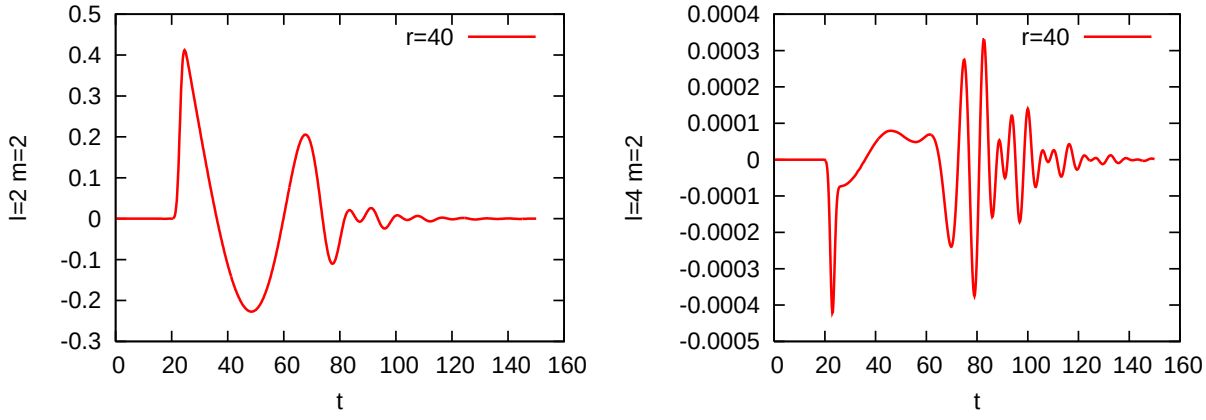


Figure 9: The patch system and the scalar wave configuration at $t = 92.2$, also enlarging the region near the excision boundary inside the horizon. The background is a rotating black hole with $a = 0.9$, the initial condition is an $\ell = 2$, $m = 2$ multipole. Note the large scale difference between the outer and the inner boundary, which is handled "naturally" and without mesh refinement. There are no artifacts visible at the inter-patch boundaries.



(a) $\ell = 2$, $m = 2$

(b) $\ell = 4$, $m = 2$

Figure 10: The $\ell = 2$, $m = 2$ and the $\ell = 4$, $m = 2$ modes, extracted at $r = 40$. The $\ell = 4$, $m = 2$ mode is excited through mode–mode coupling. Its amplitude is 10^3 times smaller than the $\ell = 2$, $m = 2$ mode. All other modes with $\ell \leq 4$ are zero up to floating point round-off error.

A. Robust stability test

The robust stability test in numerical relativity consists of evolving featureless initial conditions to which random noise has been added. This test was initially suggested in [38, 39] and later refined in the so-called Mexico tests [40]; see also [41]. The first stage of this test has a domain that is periodic in all directions, i.e., has a T^3 topology. The most difficult stage of the test has a spherical outer boundary through which noise is injected.

We implement the T^3 topology with a single patch, using penalty inter-patch boundary conditions to give the system a toroidal topology. We also use a six-patch topology and set the incoming modes to Minkowski on both the inner and outer boundary. The single patches have outer boundaries at $x^i \in [-1; +1]$, the six-patch system has the inner boundary at $r = 1.9$ and the outer boundary at $r = 11.9$. We use a Minkowski spacetime as background and add random noise with an amplitude

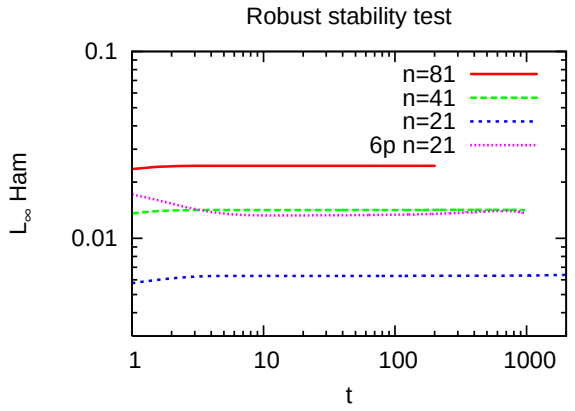


Figure 11: Robust stability test for the Einstein equations. This graph compares three different resolution of a single patch and the six-patch system. Random noise is initially added to all variables. At late times, the Hamiltonian constraint does not grow with time; the system is strictly stable. The two runs marked $n = 81$ and $6p$ $n = 21$ have the highest resolutions; we aborted them before they reached $t = 2000$, which corresponds to 1000 crossing times for the single patch.

of 10^{-8} to all variables. Since the Minkowski spacetime has no intrinsic scale, this amplitude should be compared to our floating point accuracy of approximately 10^{-16}. Terms that are quadratic in the noise amplitude have then the same order of magnitude as floating point inaccuracies.

In the runs shown below, we choose the penalty parameter $\delta = 0.5$. We discretise the domain with 21^3, 41^3, and 81^3 grid points per patch. We use the D_{8-4} derivative operator and its associated KO dissipation with a parameter $\epsilon = 0.5$. We also use an adaptive Runge–Kutta time integrator. For comparison, we also show results using a six-patch system with the D_{6-5} operator, using the same dissipation parameter.

Figure 11 shows the L_∞ norm of the Hamiltonian constraint as a function of time. The constraints remain essentially constant. Note that the constraints do not converge to zero with increasing resolution, since the random data are different for each run and do not have a continuum limit.

B. One-dimensional gauge wave

One of the most difficult of the Mexico tests [40, 41] is the gauge wave test. This is a one-dimensional non-linear gauge wave, i.e., flat space in a non-trivial coordinate system. This setup lives in a T^1 domain, i.e., it has again periodic boundaries, which we implement either as manifestly periodic boundary conditions or as penalty inter-patch boundary conditions. We use the slightly modified gauge wave which has the line element

$$ds^2 = -H\,dt^2 + H\,dx^2 + dy^2 + dz^2 \tag{16}$$

with

$$H = \exp\left[A \sin\left(\frac{2\pi(x-t)}{L}\right)\right]. \tag{17}$$

We choose the wave length L to be the size of our domain, and set the amplitude A to 0.5.

Our domain is one-dimensional with $x \in [-1;+1]$. Different from [40], we place grid points onto the boundaries. We use either 41 or 81 grid points. Figure 12 compares the shape of the wave form at late times to its initial shape. Our system is stable and very accurate even after 1000 crossing times when we use manifestly periodic boundary conditions. When we impose periodicity via penalties, the system is less accurate, and these inaccuracies lead to a drift which finally leads to a negative g_{11}, which is unstable. With 81 grid points per wave length, however, our system is both stable and very accurate after 1000 crossing times even with penalty boundaries.

C. Weak gravitational waves

We now consider perturbations of a flat spacetime, using the Regge–Wheeler (RW) perturbation theory to construct an exact solution to the linearised constraints of Einstein's equations, which we evolve with the fully non-linear equations. We linearise about the Minkowski spacetime.

For simplicity we consider an $\ell = 2$, $m = 0$ odd parity perturbation. The resulting metric in the Regge–Wheeler gauge and in spherical coordinates is

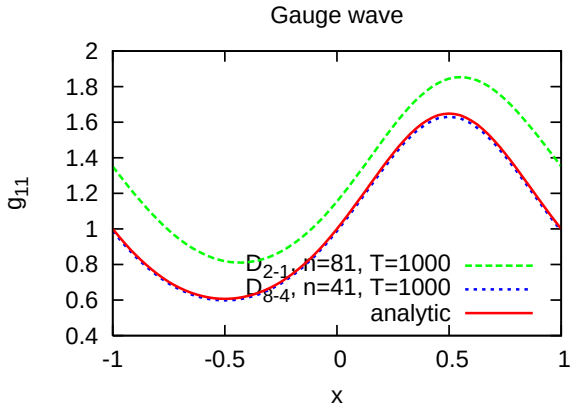
$$\begin{aligned} ds^2 = &-dt^2 + dr^2 + r^2(\sin^2\theta\, d\theta^2 + d\phi^2) \\ &- 6\,\delta\, r\Psi\, \sin^2\theta\, \cos\theta\, dr\, d\phi \\ &- 6\,\delta(\Psi + r\Psi')\sin^2\theta\, \cos\theta\, dt\, d\phi \end{aligned} \tag{18}$$

where a prime denotes derivative with respect to r, and where δ is a parameter that determines the "strength" of the perturbation (not to be confused with the penalty term, for which we use the same symbol). This metric satisfies the linearised constraints for *any* functions $\Psi(t = 0, r)$ and $\dot\Psi(t = 0, r)$, and satisfies all the linearised Einstein equations if Ψ satisfies the Regge-Wheeler equation

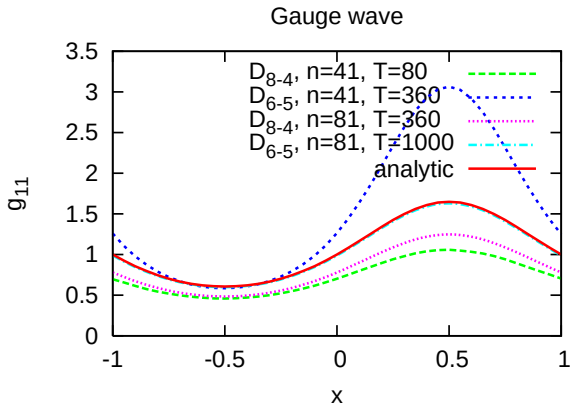
$$\ddot\Psi = \Psi'' - \frac{6}{r^2}\Psi. \tag{19}$$

It is simple to construct a purely outgoing, exact solution to the previous equation:

$$\Psi(u, r) = \frac{d^2F}{du^2} + \frac{3}{r}\frac{dF}{du} + \frac{3}{r^2}F \tag{20}$$



(a) Manifestly periodic



(b) Periodic via penalties

Figure 12: Results for the gauge wave test case, comparing differencing operators and resolutions. With manifestly periodic boundaries, the system is both stable and highly accurate after 1000 crossing times. With periodicity imposed via penalty boundary conditions, the system is less accurate, and lower resolutions are finally unstable after g_{11} becomes negative. However, with sufficient resolution and high order derivatives, our system is still highly accurate after 1000 crossing times.

where $F(u)$ is an arbitrary function of $u = r - t$. The above metric is very similar to the one in [42], except that ours is in the Regge–Wheeler gauge.

However, we use a different coordinate condition for our evolutions. We construct from the previous metric initial conditions in Cartesian coordinates for g_{ij}, K_{ij}, and d_{kij}. We set the initial lapse to one and evolve it through the time harmonic slicing condition, and set the shift to zero at all times.

We now want to check at what point non-linear effects begin to have an effect on the constraints. That is, we want to find out what resolutions are required to see that

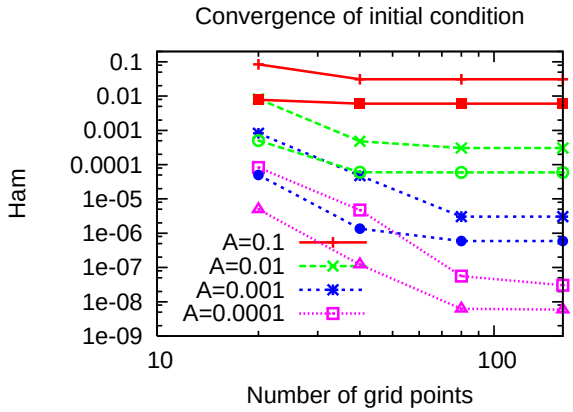


Figure 13: Norms of the Hamiltonian constraint for initial conditions with different amplitudes as a function of the number of grid points on each patch in each direction. For each initial amplitude we show both the L_2 and the L_∞ norm. Since the initial conditions only satisfy the linearised constraints, we clearly see the non-linear constraint violation at the highest resolutions.

the full, non-linear constraints do not actually converge to zero. As initial condition for the potential and its time derivative we choose

$$\Psi(t=0,r) = A \exp\left(-\frac{(r-r_0)^2}{\sigma^2}\right) \quad (21)$$

$$\dot{\Psi}(t=0,r) = -2\frac{(r-r_0)}{\sigma^2} B \exp\left(-\frac{(r-r_0)^2}{\sigma^2}\right). \quad (22)$$

The non-linear constraint violations should have an approximate quadratic dependence on the amplitude parameter (called A below). Figure 13 quantifies this violation. It displays the L_2 and L_∞ norms of the non-linear Hamiltonian constraint, measured in local coordinates, for different families of initial conditions as a function of resolution. We use the seven-patch system described in section IV with the outer boundary at $r = 6$, while the inner, cubic patch has an extent of ± 1. The initial condition parameters are $B = -A$, $\sigma = 0.6$, and $r_0 = 3$, with amplitudes $\delta = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$. We use the D_{6-5} derivative with N^3 grid points on each patch for $N = 21, 41, 81, 161$. At the highest resolution, the numerical constraints reach their non-zero continuum values. This highest resolution corresponds to 24 grid points per σ in the radial direction. This number is comparable to the *coarsest* resolutions that we use in the simulations presented below. This means that those simulations use constraint-violating initial conditions, and cannot expect the constraints to decrease with increasing resolution.

The non-linear constraint violation should be approximately a quadratic function of the amplitude of the perturbation δ, at least for small values of δ. Figure 14 shows that this is indeed the case. It displays the Hamiltonian constraint violation in the L_2 norm for the high-

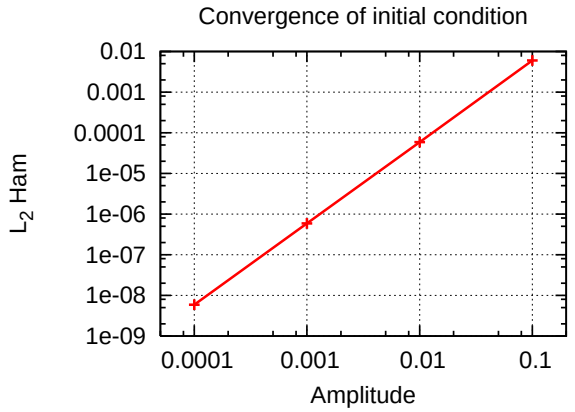


Figure 14: Hamiltonian constraint violation in the L_2 norm as a function of the amplitude of the perturbation for the highest resolution. This shows the expected quadratic dependence on the amplitude.

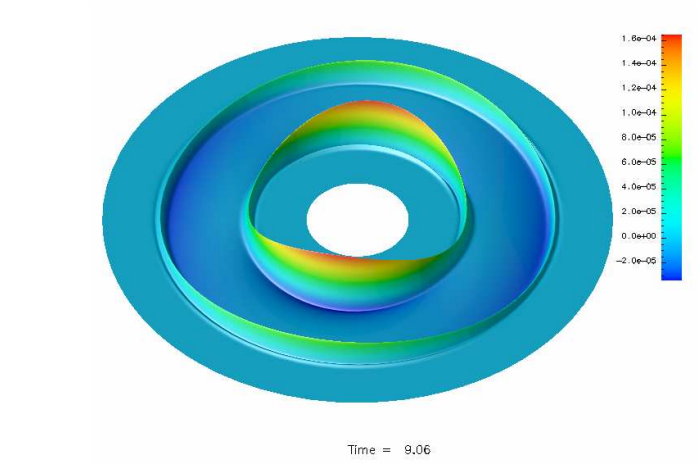


Figure 15: Evolution of weak gravitational waves. This shows the component K_{xx} of the extrinsic curvature in the equatorial plane at $t = 9.06$. The gravitational wave packet started as a spherical shell approximately in the middle between the inner and outer boundaries, and has then split into two packets which travel outwards and inwards, respectively.

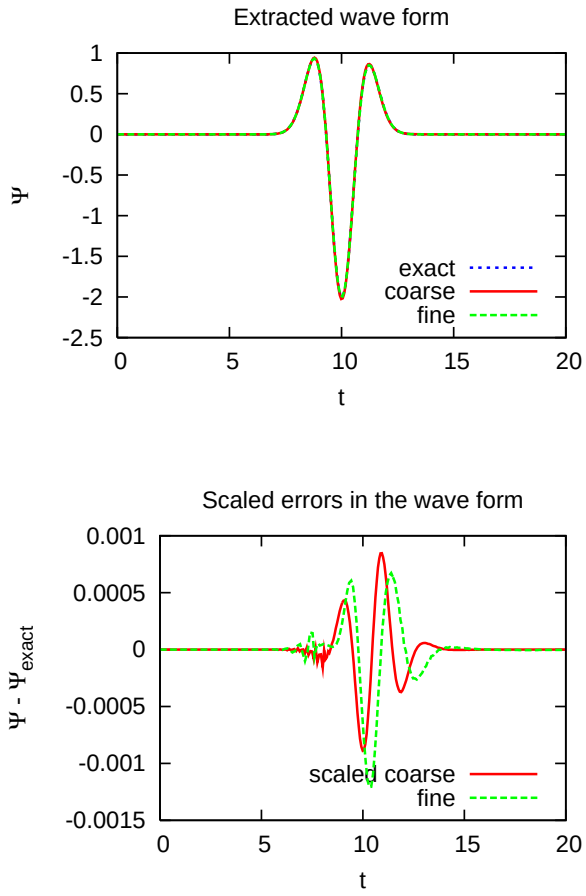


Figure 16: The upper panel shows the Regge–Wheeler potential Ψ vs. time, comparing the numerical and the exact solution at $r = 40$. The agreement is excellent, especially for a three-dimensional, fully nonlinear code. The lower panel shows the scaled differences to the exact solution for two resolutions with 16^2 and 22^2 grid points in the angular direction. This demonstrates fifth order convergence, as should be the case for the D_{8-4} operator.

est resolution of the previous figure as a function of the amplitude δ. The measured slope is 2.0002. Figure 15 shows a sample evolution of this odd parity initial condition family, using the six-patch geometry.

In order to evaluate the accuracy of our code we now choose an initial condition corresponding to an exact solution of the outgoing type described above. We do so by choosing

$$F(u) = \exp\left(-\frac{(r-r_0)^2}{\sigma^2}\right). \qquad (23)$$

In these evolutions the parameters that determine the initial condition are $\sigma = 1$, $r_0 = 30$, and amplitude $\delta = 10^{-3}$, with inner and outer boundaries at $r = 10$ and $r = 60$, respectively.

We extract the wave forms from the numerical results by calculating the Regge–Wheeler function at each grid point. We then average over one radial shell. There is no need for interpolating to a sphere, which simplifies the extraction procedure greatly, and probably also improves its accuracy. Figure 16 shows the numerically extracted Ψ_e at $r = 40$, and compared to its exact value Ψ, for an evolution using the D_{8-4} derivative, with the dissipation parameter $\epsilon = 0.05$ and $15 \times 15 \times 1001$ grid points on each patch. The agreement is excellent.

VII. CONCLUSIONS

We have motivated the use of multi-patch systems in general relativity, and have described a generic infrastructure for multi-patch time evolutions. Their main advantages are smooth boundaries and constant angular resolution, which makes them very efficient for representing systems requiring high resolution in the centre and having a radiative zone far away. They may even render fixed mesh refinement unnecessary in many cases.

We use the penalty method for inter-patch boundary conditions. It would equally be possible to use e.g. interpolation between the patches. A direct comparison of these different approaches would be very interesting. Our evolution systems are first order symmetric hyperbolic, but second order systems could be used as well. We use this infrastructure with high-order finite differencing operators, but other discretisations such as e.g. pseudo-spectral collocation methods can also be used. Our infrastructure is based on Cactus and Carpet and runs efficiently in parallel.

We have discussed the relative advantages of using global and patch-local tensor bases, and we have compared the accuracy and stability of both approaches. We suggest that using a global basis is substantially more convenient, both in the implementation of the code and in the post-processing of the generated output.

We have tested this infrastructure with a scalar wave equation on a fixed, stationary background, and with a symmetric hyperbolic formulation of the Einstein equations. We have shown that our multi-patch system with penalty boundary conditions is robustly stable and can also very accurately reproduce the nonlinear gauge wave of the *Apples with Apples* tests, which has been the most difficult of these tests for other codes.

Finally, we have simulated three-dimensional weak gravitational waves in three dimensions, using the same nonlinear code, and have accurately extracted the gravitational radiation. The latter is made especially simple since the wave extraction spheres are aligned with the numerical grid.

We believe that multi-patch systems, which provide smooth boundaries, will be an essential ingredient for discretising well-posed initial boundary value problems.

Acknowledgments

We thank Olivier Sarbach for suggesting to use a global tensor basis; our work would have been much more complicated if we had not taken this route. We thank Jonathan Thornburg for many inspiring discussions and helpful hints about multiple grid patches. We also engaged in valuable discussions with Burkhard Zink, especially regarding penalty inter-patch conditions for nonlinear equations, and with Emanuele Berti regarding scalar perturbations of a Kerr spacetime. We thank Enrique Pazos for his contributions to the design of our Regge–Wheeler code. We are very grateful to Frank Muldoon for his suggestions about automatic grid generation, and for the grids he generated for us. We thank Ian Hawke, Christian Ott, Enrique Pazos, Olivier Sarbach, and Ed Seidel for proofreading a draft of this paper.

During our work on this project, we enjoyed hospitality at Cornell University, at the Cayuga Institute of Physics, at the Albert–Einstein–Institut, and at the Center for Computation & Technology at LSU.

As always, our numerical calculations would have been impossible without the large number of people who made their work available to the public: we used the Cactus Computational Toolkit [12, 15] with a number of locally developed thorns, the LAPACK [43, 44] and BLAS [45] libraries from the Netlib Repository [46], and the LAM [47, 48, 49] and MPICH [50, 51, 52] MPI [53] implementations.

Computations for this work were performed on Peyote at the AEI, on Helix, Santaka, and Supermike at LSU, at NCSA under allocation MCA02N014, and at NERSC.

This work was partially supported by the SFB/TR-7 "Gravitational Wave Astronomy" of the DFG, the NSF under grant PHY0505761 and NASA under grant NASA-NAG5-1430 to Louisiana State University, and by the NSF under grants PHY0354631 and PHY0312072 to Cornell University. This work was also supported by the Albert–Einstein–Institut, Cornell University, by the Horace Hearne Jr. Institute for Theoretical Physics at LSU, and by the Center for Computation & Technology at LSU.

[1] J. Thornburg, *Coordinates and boundary conditions for the general relativistic initial data problem*, Class. Quantum Grav. **4**, 1119 (1987), URL `http://stacks.iop.org/0264-9381/4/1119`.

[2] E. Seidel and W.-M. Suen, *Towards a singularity-proof scheme in numerical relativity*, Phys. Rev. Lett. **69**, 1845 (1992), gr-qc/9210016.

[3] P. Secchi, *The initial boundary value problem for linear symmetric hyperbolic systems with characteristic boundary of constant multiplicity*, Differential Integral Equations **9**, 671 (1996).

[4] L. Rezzolla, A. M. Abrahams, R. A. Matzner, M. E. Rupright, and S. L. Shapiro, *Cauchy-perturbative matching and outer boundary conditions: computational studies*, Phys. Rev. D **59**, 064001 (1999), gr-qc/9807047.

[5] S. Brandt, R. Correll, R. Gómez, M. F. Huq, P. Laguna, L. Lehner, P. Marronetti, R. A. Matzner, D. Neilsen, J. Pullin, et al., *Grazing collisions of black holes via the ex-*

cision of singularities, Phys. Rev. Lett. **85**, 5496 (2000), gr-qc/0009047.

[6] D. Shoemaker, K. L. Smith, U. Sperhake, P. Laguna, E. Schnetter, and D. Fiske, *Moving black holes via singularity excision*, Class. Quantum Grav. **20**, 3729 (2003), gr-qc/0301111.

[7] M. Alcubierre et al., *Testing excision techniques for dynamical 3d black hole evolutions* (2004), gr-qc/0411137.

[8] U. Sperhake, B. Kelly, P. Laguna, K. L. Smith, and E. Schnetter, *Black hole head-on collisions and gravitational waves with fixed mesh-refinement and dynamic singularity excision*, Phys. Rev. D **71**, 124042 (2005), gr-qc/0503071.

[9] J. Thornburg, *Numerical relativity in black hole spacetimes*, Ph.D. thesis, University of British Columbia, Vancouver, British Columbia (1993).

[10] J. Thornburg, *Black hole excision with multiple grid patches*, Class. Quantum Grav. **21**, 3665 (2004), gr-qc/0404059.

[11] J. Baker, M. Campanelli, and C. O. Lousto, *The Lazarus project: A pragmatic approach to binary black hole evolutions*, Phys. Rev. D **65**, 044001 (2002), gr-qc/0104063.

[12] Cactus Computational Toolkit home page, URL `http://www.cactuscode.org/`.

[13] O. Sarbach and M. Tiglio, *Exploiting gauge and constraint freedom in hyperbolic formulations of Einstein's equations*, Phys. Rev. D **66**, 064023 (2002), gr-qc/0205086.

[14] M. Alcubierre, B. Brügmann, T. Dramlitsch, J. A. Font, P. Papadopoulos, E. Seidel, N. Stergioulas, and R. Takahashi, *Towards a stable numerical evolution of strongly gravitating systems in general relativity: The conformal treatments*, Phys. Rev. D **62**, 044034 (2000), gr-qc/0003071.

[15] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf, *The Cactus framework and toolkit: Design and applications*, in *Vector and Parallel Processing – VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science* (Springer, Berlin, 2003), URL `http://www.cactuscode.org/Publications/`.

[16] E. Schnetter, S. H. Hawley, and I. Hawke, *Evolutions in 3D numerical relativity using fixed mesh refinement*, Class. Quantum Grav. **21**, 1465 (2004), gr-qc/0310042.

[17] Mesh Refinement with Carpet, URL `http://www.carpetcode.org/`.

[18] P. Diener, E. N. Dorband, E. Schnetter, and M. Tiglio, *New, efficient, and accurate high order derivative and dissipation operators satisfying summation by parts, and applications in three-dimensional multi-block evolutions* (2005), gr-qc/0512001.

[19] H. O. Kreiss and O. E. Ortiz, *Some mathematical and numerical questions connected with first and second order time dependent systems of partial differential equations*, Lect. Notes Phys. **604**, 359 (2002), gr-qc/0106085.

[20] G. Nagy, O. E. Ortiz, and O. A. Reula, *Strongly hyperbolic second order Einstein's evolution equations*, Phys. Rev. D **70**, 044012 (2004), gr-qc/0402123.

[21] C. Gundlach and J. M. Martin-Garcia, *Hyperbolicity of second-order in space systems of evolution equations* (2005), gr-qc/0506037.

[22] M. C. Babiuc, B. Szilagyi, and J. Winicour, *Harmonic initial-boundary evolution in general relativity* (2006), gr-qc/0601039.

[23] O. Sarbach, G. Calabrese, J. Pullin, and M. Tiglio, *Hyperbolicity of the BSSN system of Einstein evolution equations*, Phys. Rev. D **66**, 064002 (2002), gr-qc/0205064.

[24] H. Beyer and O. Sarbach, *On the well posedness of the Baumgarte-Shapiro- Shibata-Nakamura formulation of Einstein's field equations*, Phys. Rev. D **70**, 104004 (2004), gr-qc/0406003.

[25] M. A. Scheel, A. L. Erickcek, L. M. Burko, L. E. Kidder, H. P. Pfeiffer, and S. A. Teukolsky, *3d simulations of linearized scalar fields in Kerr spacetime*, Phys. Rev. D **69**, 104006 (2004), gr-qc/0305027.

[26] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes* (Cambridge University Press, Cambridge, England, 1986), URL `http://www.nr.com/`.

[27] D. Goldberg, *What every computer scientist should know about floating-point arithmetic*, ACM Computing Surveys **23**, 5 (1991), URL `http://citeseer.ist.psu.edu/goldberg91what.html`.

[28] Cactus benchmarking results, URL `http://www.cactuscode.org/benchmarks/`.

[29] L. Lehner, O. Reula, and M. Tiglio, *Multi-block simulations in general relativity: high order discretizations, numerical stability, and applications*, Class. Quantum Grav. **22** (2005), gr-qc/0507004.

[30] GridPro Program Development Company, URL `http://www.gridpro.com/`.

[31] K. Mattsson, M. Svärd, and J. Nordström, *Stable and accurate artificial dissipation*, J. Sci. Comput. **21**, 57 (2004).

[32] G. B. Cook, *Initial data for numerical relativity*, Living Rev. Rel. **3**, 5 (2000), URL `http://relativity.livingreviews.org/Articles/lrr-2000-5/index.html`.

[33] E. Berti, V. Cardoso, and S. Yoshida, *Highly damped quasinormal modes of Kerr black holes: A complete numerical investigation*, Phys. Rev. D **69**, 124018 (2004), gr-qc/0401052.

[34] E. Berti and K. D. Kokkotas, *Quasinormal modes of Kerr–Newman black holes: Coupling of electromagnetic and gravitational perturbations*, Phys. Rev. D **71**, 124008 (2005), gr-qc/0502065.

[35] N. Dorband, E. Berti, P. Diener, E. Schnetter, and M. Tiglio, *Scalar perturbations of Kerr black holes: three-dimensional time evolutions with a multi-patch code* (2006), in preparation.

[36] M. Tiglio, L. Lehner, and D. Neilsen, *3d simulations of Einstein's equations: symmetric hyperbolicity, live gauges and dynamic control of the constraints*, Phys. Rev. D **70**, 104018 (2004), gr-qc/0312001.

[37] O. Sarbach and M. Tiglio, *Boundary conditions for Einstein's field equations: analytical and numerical analysis*, Journal of Hyperbolic Differential Equations **2**, 839 (2005), gr-qc/0412115.

[38] B. Szilágyi, R. Gómez, N. T. Bishop, and J. Winicour, *Cauchy boundaries in linearized gravitational theory*, Phys. Rev. D **62**, 104006 (2000), gr-qc/9912030.

[39] B. Szilagyi, B. Schmidt, and J. Winicour, *Boundary conditions in linearized harmonic gravity*, Phys. Rev. D **65**, 064015 (2002), gr-qc/0106026.

[40] M. Alcubierre, G. Allen, T. W. Baumgarte, C. Bona, D. Fiske, T. Goodale, F. S. Guzmán, I. Hawke, S. Hawley, S. Husa, et al., *Towards standard testbeds for numerical relativity*, Class. Quantum Grav. **21**, 589 (2004), gr-qc/0305023.

[41] Apples With Apples: Numerical Relativity Comparisons and Tests, URL `http://www.ApplesWithApples.org/`.

[42] S. A. Teukolsky, *Linearized quadrupole waves in general relativity and the motion of test particles*, Phys. Rev. D **26**, 745 (1982).

[43] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling,

A. McKenney, et al., *LAPACK Users' Guide* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999), 3rd ed., ISBN 0-89871-447-8 (paperback).

[44] LAPACK: Linear Algebra Package, URL `http://www.netlib.org/lapack/`.

[45] BLAS: Basic Linear Algebra Subroutines, URL `http://www.netlib.org/blas/`.

[46] Netlib Repository, URL `http://www.netlib.org/`.

[47] G. Burns, R. Daoud, and J. Vaigl, *LAM: An Open Cluster Environment for MPI*, in *Proceedings of Supercomputing Symposium* (1994), pp. 379–386, URL `http://www.lam-mpi.org/download/files/lam-papers.tar.gz`.

[48] J. M. Squyres and A. Lumsdaine, *A Component Architecture for LAM/MPI*, in *Proceedings, 10th European PVM/MPI Users' Group Meeting* (Springer-Verlag, Venice, Italy, 2003), no. 2840 in Lecture Notes in Computer Science, pp. 379–387.

[49] LAM: LAM/MPI Parallel Computing, URL `http://www.lam-mpi.org/`.

[50] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, *A high-performance, portable implementation of the MPI message passing interface standard*, Parallel Computing **22**, 789 (1996).

[51] W. D. Gropp and E. Lusk, *User's Guide for* `mpich`*, a Portable Implementation of MPI*, Mathematics and Computer Science Division, Argonne National Laboratory (1996), ANL-96/6.

[52] MPICH: ANL/MSU MPI implementation, URL `http://www-unix.mcs.anl.gov/mpi/mpich/`.

[53] MPI: Message Passing Interface Forum, URL `http://www.mpi-forum.org/`.