Contents

Introduction

Motivation & background

Fifteen years after the big breakthrough in black hole simulations [45], numerical relativity is a mature established field. The relevant equations are somewhat understood. Advanced numerical techniques are commonly used. Multiple comprehensive textbooks are available [4, 10]. Large-scale numerical simulations of black hole or neutron star collisions are routine and even used to guide experiments. Outside of this "mainstream", however, there exist barely-touched areas that are still highly relevant to fundamental theoretical questions. One such is the question of critical gravitational collapse of gravitational waves.

In 1993, at the dawn of numerical relativity, Choptuik discovered [21] that if one took a one-parameter family of spherically symmetric scalar field configurations and tuned the value of the parameter to the threshold between dispersal and collapse into a black hole, the resulting evolution would have very interesting properties reminiscent of critical phase transitions known from statistical mechanics. Spacetime evolution in the immediate vicinity of the threshold — the *critical point* — eventually approaches the same *universal* profile, i.e. independent of the initial data family chosen (though dependent on the matter model). This profile is *discretely self-similar* (DSS) — consisting of a sequence of *echoes*, each a regularly scaled copy of the previous echo. The echoes repeat periodically in logarithmic time. These phenomena are collectively called *critical behavior* (see [28] for a review of the subject).

Considerably less is known about critical collapse beyond spherical symmetry. This can be attributed to the significantly higher cost of numerical simulations and a richer space of possible behaviors. Axially symmetric collapse is especially interesting, as it is the highest degree of symmetry that admits a vacuum (i.e. pure gravitational radiation) collapse. This allows studying the properties of gravity by itself, with no influence from the chosen matter model.

So far, only two groups reported observing critical behavior in vacuum collapse. First there was a series of papers [1, 2] by Abrahams and Evans, very shortly after Choptuik's discovery. But though almost 30 years have passed, these results have not been independently reproduced to date. The second report was that of Sorkin in 2011 [50], using a different initial data family. However this later result has apparently been falsified by Hilditch [33]; Sorkin's stated critical point location also contradicts both earlier literature and our own data.

The cited study of Hilditch et al. is the only recent line of research in this area we are aware of. They report significant new progress with Brill wave critical collapse, but despite approaching the critical point to the same order of magnitude as Abrahams and Evans (while spending orders of magnitude more computational resources), they do not see conclusive evidence of universality or echoing. They do, however, report

major differences from the critical behavior of Choptuik, namely a "bifurcation" of the evolution into two collapse centers that, in supercritical spacetimes, result in two disjoint apparent horizons.

The current status of vacuum gravitational collapse is thus deeply unclear, neither of the critical behavior claims stands on particularly firm ground, and more independent verification is needed.

Furthermore, we believe that understanding the gravitational collapse in pure vacuum is interesting on its own, even outside of its critical behavior aspect. Nonlocality of gravitational waves makes it hard to understand how the collapse unfolds and the extreme departure from spherical symmetry tends to break down some common assumptions.

Plan & goals

The goal of this thesis is to advance the understanding of vacuum critical collapse in axial symmetry using numerical simulations. We attempt to reproduce previously reported results and consequently push past them closer to the critical point.

Our approach is steered by the numerical codes available to us. Not being a part of an established numerical relativity group, we only have access to public open source codes — most prominently the Einstein Toolkit — and whatever we are able to implement ourselves. Thus for the most part we rely on robust established techniques commonly used for simulating black-hole spacetimes, such as finite differencing on a regular grid, fixed mesh refinement, or the BSSN evolution equations.

Our work can be contrasted with the parallel research of Hilditch et al. [31, 32, 33], who tried to apply the aforementioned techniques in 2013, only to see them fail because of coordinate singularities [31]. Their response was to write a completely new code using different numerical methods (pseudospectral, rather than finite differences) and a different formulation of GR equations (generalized harmonic, rather than BSSN). We, on the other hand, attempt a minimal viable modification of the failing coordinate condition, keeping the rest of the numerical machinery unchanged. The existence of these two parallel approaches is beneficial, since producing the same results with two independent codes using different techniques increases their credibility.

In light of the abovementioned reproducibility issues, we place an emphasis on making our results verifiable and reproducible. That is achieved through:

- Comparing to previous studies where feasible; especially with the concurrent work [32, 33].
- Using invariant quantities to state our results, so that the same numbers can be extracted from simulations describing our spacetimes in different coordinates.
- Releasing all our codes as free software, so anyone can see and replicate our simulations exactly.

Due to personal proclivities of the author, this work leans towards the programming side of numerical relativity, perhaps at the expense of some mathematical rigor. We spend much effort on crafting the necessary software and optimizing it for high performance, making it publicly available as a part of this work. Comparatively less attention is given to the "mathematical" side of things — for the most part we rely on already established results.

Structure & layout

This thesis is divided into three main parts. In the first, Continuum systems, we describe the equations we are solving. We give a brief overview of the relevant numerical relativity topics in Chapter 1. These are all standard textbook results and are included merely for completeness. Chapter 2 contains details about our initial data families — both those used in previous studies (Brill waves) and those unique to our work (time-asymmetric waves). In Chapter 3 we deal with the choice of the time coordinate in our simulations. We review some existing slicing methods and introduce a new method we call *quasi-maximal* slicing (QMS), which is one of the major results of our work. The part concludes with Chapter 4, where we describe the tools we use to extract physically meaningful information from our spacetimes. These include invariants and horizons.

The second part, Numerical codes, contains detailed information on the numerical codes we use. In Chapter 5 we describe the relevant features of the Einstein Toolkit — a bundle of open source codes on which our simulations are based — along with our modifications to them. In Chapters 6 and 7 we describe the pseudospectral solvers we wrote to construct our initial data. Chapter 8 describes the multigrid solver used by the slicing codes that are the subjects of Chapter 9. Finally Chapter 10 deals with our horizon finders.

In the last part, Simulations, we describe the simulations we ran and the results we extracted from them. In Chapter 11 we validate our code in order to demonstrate its correctness and convergence. Then in Chapter 12 we describe the results obtained from our near-critical simulations.

Supplementary information is provided in the Appendices. Appendix A.1 contains tables listing the simulations we ran and their global properties. In Appendix A.2 we describe the data package attached to the electronic version of this work, containing complete source codes and parameter files. Appendices A.3 and A.4 contain copies of our published papers.

Conventions & formalism

Our notation and formalism are aligned with most other current NR work, e.g. the textbook [4]. That mainly encompasses standard general relativity in 4 dimensions, geometric units $c = G = 1$, metric signature $(-,+,+,+)$, and the 3+1 ADM split of spacetime. We assume axial symmetry (effectively reducing the problem to 2+1), vacuum, and zero cosmological constant. All our initial data families also possess reflection symmetry with respect to the equatorial plane.

Unless mentioned otherwise, repeated indices are assumed to be summed over, as per the Einstein summation convention. Greek indices are used for 4-dimensional quantities and run over $0-3$; Latin indices are used for 3-dimensional spatial quantities and run over $1-3$. In the chapters describing numerical code we also use Latin indices to index grid points, basis functions, etc.; their range is then either clear from the context or mentioned explicitly.

Part I

Continuum systems

Chapter 1

Numerical relativity

In this chapter we summarize the main numerical relativity results relevant to our work. A more thorough overview of the subject can be found in [4] or [10].

We start with standard general relativity in a 4-dimensional spacetime imbued with a metric tensor $g_{\mu\nu}$. The spacetime curvature is governed by the *Einstein field equations*

$$ {}^{(4)}R_{\mu\nu} - \frac{1}{2}{}^{(4)}R g_{\mu\nu} = 8\pi T_{\mu\nu}. \tag{1.1} $$

The right-hand side contains the *stress-energy tensor* $T_{\mu\nu}$, which we consider to vanish from here on, as we work in vacuum. The left-hand side contains the *Ricci scalar*

$$ {}^{(4)}R = {}^{(4)}R^{\alpha}_{\ \alpha} \tag{1.2} $$

and the *Ricci tensor*

$$ {}^{(4)}R_{\mu\nu} = {}^{(4)}R^{\alpha}_{\ \mu\alpha\nu}, \tag{1.3} $$

which are contractions of the *Riemann curvature tensor* ${}^{(4)}R^{\alpha}_{\ \beta\mu\nu}$, given as

$$ \nabla_\mu \nabla_\nu V^\alpha - \nabla_\nu \nabla_\mu V^\alpha = {}^{(4)}R^{\alpha}_{\ \beta\mu\nu} V^\beta. \tag{1.4} $$

The ${}^{(4)}$ decoration serves the purpose of distinguishing the 4-dimensional quantities from their 3-dimensional counterparts introduced later.

The covariant derivative ∇_μ associated with $g_{\mu\nu}$ acts on vectors as

$$ \nabla_\mu V^\nu = \partial_\mu V^\nu + {}^{(4)}\Gamma^{\nu}_{\mu\kappa} V^\kappa, \tag{1.5} $$

with the *Christoffel symbols* (connection coefficients) associated with $g_{\mu\nu}$

$$ {}^{(4)}\Gamma^{\alpha}_{\mu\nu} = \frac{1}{2} g^{\alpha\beta} \left(\partial_\mu g_{\nu\beta} + \partial_\nu g_{\mu\beta} - \partial_\beta g_{\mu\nu} \right). \tag{1.6} $$

One can write the Riemann tensor in terms of ${}^{(4)}\Gamma^{\alpha}_{\mu\nu}$ as

$$ {}^{(4)}R^{\alpha}_{\ \beta\mu\nu} = \partial_\mu {}^{(4)}\Gamma^{\alpha}_{\beta\nu} - \partial_\nu {}^{(4)}\Gamma^{\alpha}_{\beta\mu} + {}^{(4)}\Gamma^{\alpha}_{\kappa\mu} {}^{(4)}\Gamma^{\kappa}_{\beta\nu} - {}^{(4)}\Gamma^{\alpha}_{\kappa\nu} {}^{(4)}\Gamma^{\kappa}_{\beta\mu}. \tag{1.7} $$

1.1 3+1 split

To recast equations (1.1) as an initial-boundary value problem that can be evolved forward in time from some initial data, we take the spacetime to be foliated by a sequence of space-like *slices*. Each slice is labeled by a time coordinate t and

$$n^\mu = \frac{-1}{\sqrt{-(\nabla_\nu t)(\nabla^\nu t)}} \nabla^\mu t \tag{1.8}$$

is the future pointing unit normal vector to the slices. By construction $n^\mu n_\mu = -1$. The normalization factor in (1.8)

$$\alpha = \left[-(\nabla_\mu t)(\nabla^\mu t)\right]^{-1/2} \tag{1.9}$$

is called the *lapse* function — it measures the proper time that elapses between the neighboring time slices along the normal vector n^μ. Similarly, given coordinates x^i in a spatial slice, the 3-dimensional shift vector β^i measures the velocity of coordinate lines from one slice to the next

$$x^i_{t+\mathrm{d}t} = x^i_t - \beta^i \mathrm{d}t. \tag{1.10}$$

The lapse and the shift together are called the *gauge* functions, as they determine the choice of coordinates.

Specifically, the choice of the lapse — called the *slicing* — determines the spacetime foliation. We discuss our slicing choice in Chapter 3. The shift determines how the spatial coordinates evolve with time. While we experimented with several shift conditions during the course of this work, ultimately zero shift proved a simple and adequate choice, so we settled on $\beta^i = 0$.

The spatial induced metric within a slice is

$$\gamma_{\mu\nu} = g_{\mu\nu} + n_\mu n_\nu. \tag{1.11}$$

The *extrinsic curvature* of a slice is the derivative of the spatial metric along the normal

$$K_{\mu\nu} = -\frac{1}{2}\mathscr{L}_n \gamma_{\mu\nu}. \tag{1.12}$$

In adapted coordinates $\{t, x^i\}$, the time components of spatial vectors vanish and the 4-dimensional metric can be decomposed as

$$g_{\mu\nu} = \begin{pmatrix} -\alpha^2 + \beta_k\beta^k & \beta_i \\ \beta_j & \gamma_{ij} \end{pmatrix}, \tag{1.13a}$$

$$g^{\mu\nu} = \begin{pmatrix} -1/\alpha^2 & \beta^i/\alpha^2 \\ \beta_j/\alpha^2 & \gamma^{ij} - \beta^i\beta^j/\alpha^2 \end{pmatrix}. \tag{1.13b}$$

1.2 ADM equations

Contracting the vacuum Einstein equations twice with the normal to spatial slices n^μ yields the equation known as the *Hamiltonian constraint*

$$H = R + K^2 - K_{ij}K^{ij} = 0. \tag{1.14a}$$

Here $\boldsymbol{R}$ is the Ricci scalar associated with the spatial metric γ_{ij} and $K = \gamma^{ij}\boldsymbol{K}_{ij}$ is the trace of the extrinsic curvature. Considering also spatial projections of (1.1) once contracted with n^μ gives us the *momentum constraints*

$$\boldsymbol{M}_i = \mathrm{D}_j \boldsymbol{K}_i^j - \mathrm{D}_i K = 0, \tag{1.14b}$$

where D_i is the covariant derivative associated with γ_{ij}. Equations (1.14a) and (1.14b) together are called the *ADM constraints*, as they *constrain* the physical $\{\gamma_{ij}, \boldsymbol{K}_{ij}\}$ configurations on a slice to a hypersurface in the space of all such possible configurations. In other words, γ_{ij} and $\boldsymbol{K}_{ij}$ on each slice must satisfy the constraints in order to be a solution of the Einstein equations (1.1). The Bianchi identities guarantee the constraints remaining satisfied during evolution if they were satisfied initially, however this only holds at the continuum level.

Finally, the vacuum Einstein equations combined with spatial projections of the Riemann tensor twice contracted with the slice normal n^μ give us the evolution equation for the extrinsic curvature

$$(\partial_t - \mathscr{L}_\beta)\boldsymbol{K}_{ij} = -\mathrm{D}_i\mathrm{D}_j\alpha + \alpha(\boldsymbol{R}_{ij} + K\boldsymbol{K}_{ij} - 2\boldsymbol{K}_{ik}\boldsymbol{K}_j^k). \tag{1.15}$$

Together with (1.12), they form the *ADM evolution equations* for γ_{ij}, $\boldsymbol{K}_{ij}$. Given some initial values at $t = 0$ that satisfy the constraints and the choice of the gauge, these evolution equations could in theory be used to construct the spacetime as $\gamma_{ij}(t)$, $\boldsymbol{K}_{ij}(t)$.

The existence of the constraints (1.14) means the evolution equations are non-unique, since one can add arbitrary multiples of the constraint equations to them and thus obtain any number of alternative *formulations* with the same physical (i.e. constraint-satisfying) solutions, but different mathematical properties.

Such formulations can be grouped into two classes, called *free* and *constrained* evolution schemes. In the former the constraints are only solved on the initial slice. Afterwards, the constraint violations resulting from numerical inaccuracies behave according to the properties of the evolution equations. The values of $\boldsymbol{H}$ and $\boldsymbol{M}_i$ can be used as indicators of evolution accuracy. The constrained evolution schemes involve solving the constraint equations at each time step, so constraint-violating modes do not exist within numerical precision. However such schemes tend to be more complicated and computationally expensive.

Some free-evolution formulations are said to be *constraint-damping*, meaning they are constructed such that constraint violations arising from numerical inaccuracies are damped away.

1.3 BSSN formulation

Unfortunately it turns out that the ADM equations (1.12) and (1.15) do not constitute a well-posed Cauchy problem and do not lead to stable long-term evolutions. There are several free-evolution reformulations that are well-posed and stable — the one we use is called BSSN (or sometimes BSSNOK) [9, 49]. One alternative formulation, used e.g. in the parallel research by Hilditch et al. [32], is the generalized harmonic formulation, which also includes the choice of the gauge.

The first step in the BSSN formulation is conformally rescaling the spatial metric

$$\bar{\gamma}_{ij} = \varphi^2 \gamma_{ij}, \tag{1.16}$$

such that $\varphi = \det(\gamma_{ij})^{-1/6}$ and $\overline{\gamma}_{ij}$ has a unit determinant. Further we split off the trace K into a separate evolved variable and also conformally rescale the remaining traceless part

$$\overline{A}_{ij} = \varphi^2 \left(K_{ij} - \frac{1}{3} K \gamma_{ij} \right). \tag{1.17}$$

Finally we introduce the *conformal connection functions*

$$\overline{\Gamma}^i = \overline{\gamma}^{jk} \overline{\Gamma}^i_{jk}, \tag{1.18}$$

where $\overline{\Gamma}^i_{jk}$ are the Christoffel symbols associated with $\overline{\gamma}_{ij}$. The evolved variables in BSSN are $\{\overline{\gamma}_{ij}, \overline{A}_{ij}, \overline{\Gamma}^i, \varphi, K\}$ with their corresponding evolution equations

$$\left(\partial_t - \mathscr{L}_\beta\right) \varphi = \frac{1}{3} \varphi \alpha K, \tag{1.19a}$$

$$\left(\partial_t - \mathscr{L}_\beta\right) \overline{\gamma}_{ij} = -2\alpha \overline{A}_{ij}, \tag{1.19b}$$

$$\left(\partial_t - \mathscr{L}_\beta\right) K = -\mathrm{D}^i \mathrm{D}_i \alpha + \alpha \left(A_{ij} A^{ij} + \frac{1}{3} K^2 \right), \tag{1.19c}$$

$$\left(\partial_t - \mathscr{L}_\beta\right) \overline{A}_{ij} = \varphi^2 \left[-\mathrm{D}_i \mathrm{D}_j \alpha + \alpha R_{ij} \right]^{\mathrm{TF}} + \alpha \left(K \overline{A}_{ij} - 2 \overline{A}_{ik} \overline{A}^k_j \right), \tag{1.19d}$$

$$\begin{aligned} \left(\partial_t - \mathscr{L}_\beta\right) \overline{\Gamma}^i = {} & \overline{\gamma}^{jk} \partial_j \partial_k \beta^i + \frac{1}{3} \overline{\gamma}^{ij} \partial_j \partial_k \beta^k - 2 \overline{A}^{ij} \partial_j \alpha \\ & + 2\alpha \left(\overline{\Gamma}^i_{jk} \overline{A}^{jk} + 6 \overline{A}^{ij} \partial_j \varphi - \frac{2}{3} \overline{\gamma}^{ij} \partial_j K \right), \end{aligned} \tag{1.19e}$$

where TF denotes the tracefree part of the bracketed expression. The conformally related metric $\overline{\gamma}_{ij}$ and its inverse $\overline{\gamma}^{ij}$ are used to raise and lower indices on all the quantities with a bar. Terms involving D_i and R_{ij} are assumed to be computed from $\overline{\gamma}_{ij}$ and φ in the straightforward manner.

Compared to the ADM evolution equations, we have an extra constraint (1.18), which is typically enforced numerically by replacing $\overline{\Gamma}^i$ with the right-hand side of (1.18) whenever it appears undifferentiated. We also have two new algebraic constraints

$$\det(\overline{\gamma}_{ij}) = 1, \tag{1.20a}$$

$$\overline{A}_{ij} \overline{\gamma}^{ij} = 0, \tag{1.20b}$$

which also tend to be enforced at each time step.

The BSSN formulation with an appropriate gauge choice has been shown to be strongly hyperbolic and is successfully used in many codes today. One weak point of BSSN is that its characteristic decomposition contains so-called "zero-speed modes" corresponding to constraint violations. So constraint violations that arise from numerical inaccuracies may remain on the grid, polluting the simulations in the long term. To avoid this, there have been suggestions to replace BSSN with CCZ4 [7] — a similar formulation where the constraint violations disperse as waves and can also be dissipated using constraint damping.

Chapter 2

Initial data

In this chapter we describe the initial data we use in our critical collapse simulations. We divide them into two "classes" — the Brill data and what we call the "time-asymmetric data" (in contrast to the time-symmetric Brill waves). The former have been used in almost all studies of vacuum critical collapse published so far, including the current state-of-the-art [33]. The latter resulted from our attempts to reproduce the seminal results of Abrahams and Evans [2]. While we were not successful in that goal, the initial data we managed to obtain was interesting enough to be studied on its own merits.

Each of the abovementioned classes can be concretized into multiple one-parameter initial data families with a dimensionless parameter we always call A. Every such family f has its own collapse-dispersal threshold — the critical point $A = A^f_*$. The initial data is specified up to an overall scale σ, which we set equal to one in our numerical codes.

Since the "strength" (as characterized e.g. by the ADM mass) of the data does not always grow monotonously with A, we introduce notation $A_0 \prec A_1$ to indicate that initial data with $A = A_0$ is "weaker" than that with $A = A_1$.

Our numerical codes constructing those initial data families are described in Chapters 6 and 7. The specific setups we use in our simulations are described in Chapter 12.

2.1 Brill waves

The Brill waves are a class of time-symmetric initial data introduced by Brill [15] to prove that gravitational waves have positive energy. Among the virtues of this initial data is its simplicity — the only nontrivial construction requirement is numerically solving a single 2D linear elliptic equation.

The Brill waves were first numerically constructed by Eppley [24], who studied the properties of the initial slices in themselves. Later on, Brill waves of a slightly different form were evolved to study critical collapse in [5, 25, 31, 33, 47].

Brill waves are constructed by assuming an axially symmetric nonrotating spatial slice at the moment of time symmetry, which implies initially vanishing extrinsic curvature $K_{ij}(t = 0) = 0$. The spatial metric is then taken to have the form (in standard

spherical / cylindrical coordinates $\{r, \theta, \varphi\}$ / $\{\rho, \varphi, z\}$)

$$\begin{aligned} \gamma_{ij} \mathrm{d}x^i \mathrm{d}x^j &= \psi^4 \left[e^{2q} \mathrm{d}r^2 + r^2 \mathrm{d}\theta^2 + r^2 \sin^2\theta \mathrm{d}\varphi^2 \right] \\ &= \psi^4 \left[e^{2q} \left(\mathrm{d}\rho^2 + \mathrm{d}z^2 \right) + \rho^2 \mathrm{d}\varphi^2 \right]. \end{aligned} \tag{2.1}$$

Here ψ is an undetermined conformal factor and q — called a "seed function" — is a freely chosen axially symmetric function of spatial coordinates, which determines the initial wave packet shape. With these choices the momentum constraints are trivially satisfied, leaving us with only the Hamiltonian constraint of the form

$$\Delta\psi + \frac{1}{4} \left(\partial_{\rho\rho} q + \partial_{zz} q \right) \psi = 0, \tag{2.2}$$

where Δ is the flat-space Laplace operator. For some chosen form of q this is a linear elliptic PDE to be solved numerically for ψ. Initial data is then constructed by substituting this numerical solution into (2.1) and setting $K_{ij} = 0$.

We choose the same quadrupole form of q that is used in the cited critical collapse studies

$$q = A \left(\frac{r}{\sigma} \right)^2 \sin^2\theta e^{-r^2/\sigma^2} = A \left(\frac{\rho}{\sigma} \right)^2 e^{-(\rho^2+z^2)/\sigma^2}, \tag{2.3}$$

allowing us to compare results directly. Here A is the amplitude parameter that determines the initial "concentration" of the wave packet. Clearly for $A = 0$ we obtain a flat initial slice. It is also notable that both positive and negative values of A can be chosen and will produce different initial data families. Both were studied in [31], where they were called "geometrically prolate" and "geometrically oblate" respectively.

Profiles of the conformal factor ψ for near-critical and AH-containing initial data with positive and negative values of A, as produced by our solver, are shown in Figures 2.1 and 2.2. Dependence of the ADM mass on A is shown in Figure 2.3. The mass seems to diverge towards $A \sim -6.5$ and $A \sim 20$.

2.2 Time-asymmetric waves

The pioneering papers of Abrahams and Evans [1, 2] were, among other things, unique in not using the Brill wave initial data. Instead, their initial data was derived from the linearized quadrupole waves of Teukolsky [51]. Since their results were never reproduced independently, despite massive advances in easily available computational power [1], replicating their initial data as accurately as possible was of great interest to us.

Two different one-parameter families are used in each of the papers [1, 2] — called by the authors "ingoing-pulse" and "time-antisymmetric" respectively. The first of these had the properties of

- discontinuity in the 6th derivatives of the metric variables
- the initial wave packet being concentrated at a significant distance from coordinate origin

[1] And, as anecdotal evidence would have it, several attempts being made by different people.

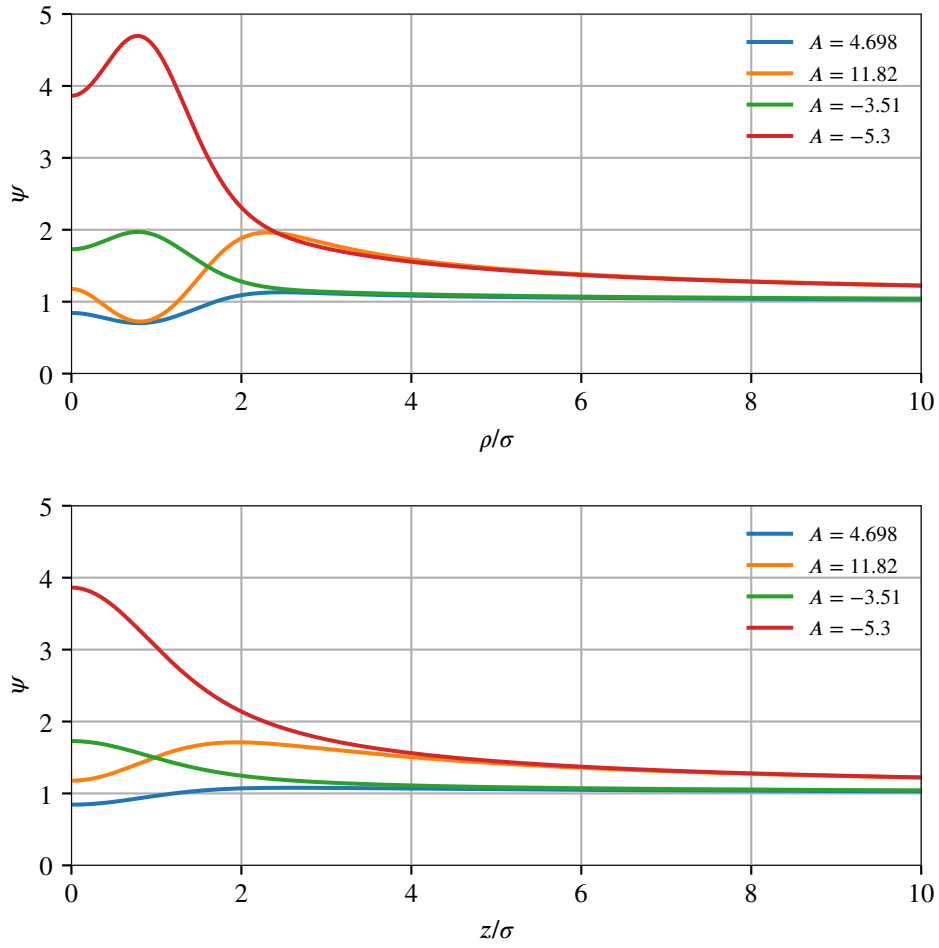

FIGURE 2.1: Profiles of the conformal factor ψ obtained by solving (2.2) and (2.3) with different values of A. Top: equatorial plane $z = 0$; bottom: axis of symmetry $\rho = 0$.

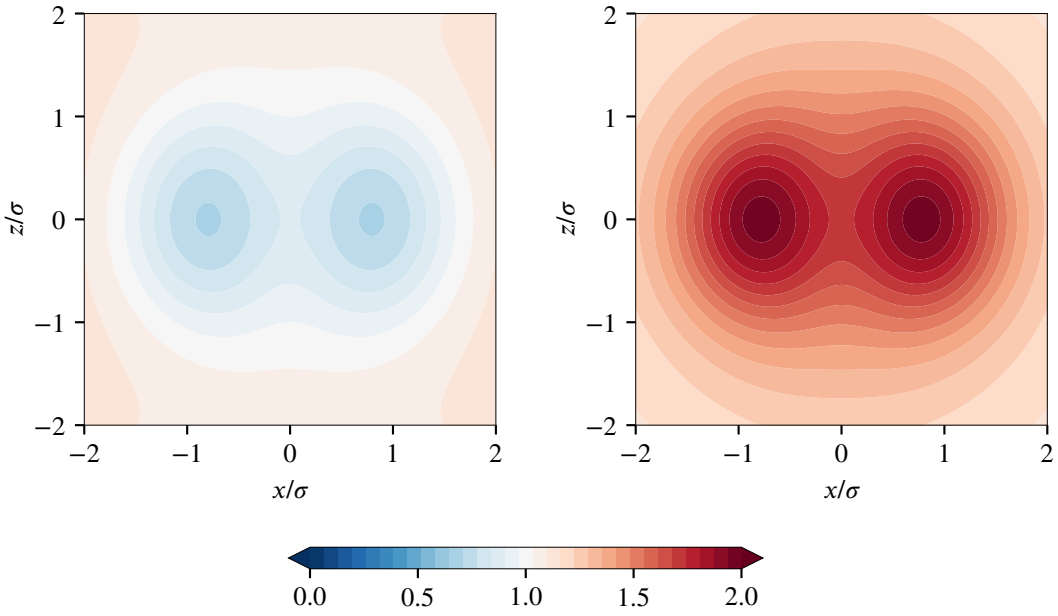

FIGURE 2.2: Contours of the conformal factor ψ obtained by solving (2.2) and (2.3), in the $x - z$ plane. Left: $A = 4.698$; right: $A = -3.51$.

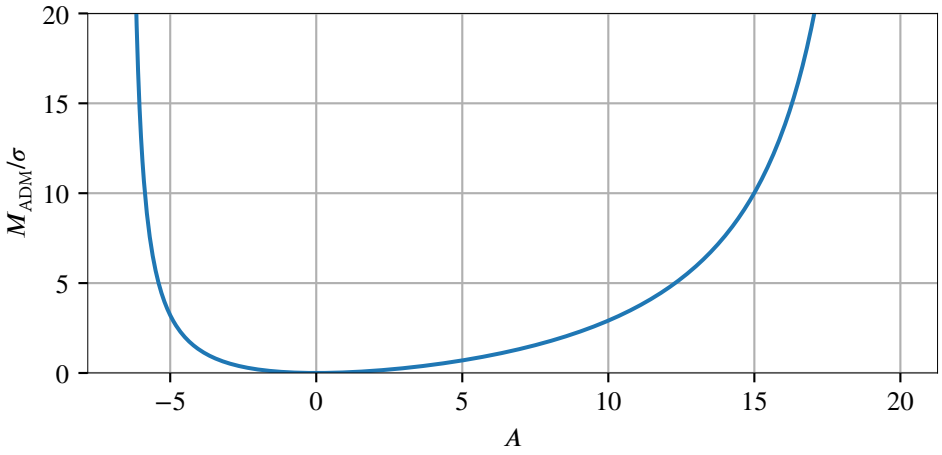

FIGURE 2.3: ADM mass of Brill wave initial data as a function of A.

which were challenging for our pseudospectral solver optimized for resolving the origin. For this reason, we focused on the "time-antisymmetric" data of [2], which was supposed to be smooth and concentrated close to the origin.

"Time symmetry" of the initial data means that the metric is momentarily stationary — its time derivative K_{ij} vanishes. By analogy "time antisymmetry" should mean that the metric itself is in some sense trivial — specifically conformally flat

$$\gamma_{ij} = \psi^4 \left(\mathrm{d}r^2 + r^2 \mathrm{d}\theta^2 + r^2 \sin^2\theta \mathrm{d}\varphi^2 \right). \tag{2.4}$$

In order for the data to be non-trivial, the extrinsic curvature now must be non-vanishing. Our interpretation of the choices made in [2] is as follows. Working with mixed-index spherical components K^i_j, the initial slice is chosen to be non-rotating — $K^r_\varphi = K^\theta_\varphi = 0$ — and maximal $K^\theta_\theta = -\left(K^r_r + K^\varphi_\varphi\right)$. Finally $K^r_\theta = K^r_\theta(r,\theta)$ is chosen to be a prescribed spatial function, which leaves us with two non-trivial momentum constraints to solve for the two remaining components K^r_r and K^φ_φ. The constraints for this data then reduce to a set of three coupled non-linear elliptic equations for $u^k = \left\{\psi, K^r_r, K^\varphi_\varphi\right\}$ that can be written as

$$0 = -\frac{8}{\psi^5}\Delta\psi - K^i_j K^j_i, \tag{2.5a}$$

$$0 = \partial_r K^r_r + \frac{6K^r_r \partial_r\psi}{\psi} + \frac{6K^r_\theta \partial_\theta\psi}{r^2\psi} + \frac{3}{r}K^r_r + \frac{1}{r^2}\partial_\theta K^r_\theta + \frac{1}{r^2\tan\theta}K^r_\theta, \tag{2.5b}$$

$$0 = -\partial_\theta K^r_r - \partial_\theta K^\varphi_\varphi - \frac{6\left(K^r_r + K^\varphi_\varphi\right)\partial_\theta\psi}{\psi} - \frac{K^r_r + 2K^\varphi_\varphi}{\tan\theta} + \frac{6K^r_\theta\partial_r\psi}{\psi} + \frac{2K^r_\theta}{r} + \partial_r K^r_\theta. \tag{2.5c}$$

While the exact profile of K^r_θ is not given explicitly in [2], a straightforward calculation from their equations (5) and (6) leads to (taking $r_0 = 0, \Lambda = \sigma$)

$$K^r_\theta = -60\sqrt{\frac{2}{\pi}}A\left[3\frac{r}{\sigma} - 2\left(\frac{r}{\sigma}\right)^3\right]\exp\left[-\left(\frac{r}{\sigma}\right)^2\right]\sin(2\theta). \tag{2.6}$$

It is here that we encountered the first signs of trouble: profile (2.6) (plotted in Figure 2.4) is visibly different from the one shown in the upper panel of Fig. 1 in [2]. Though

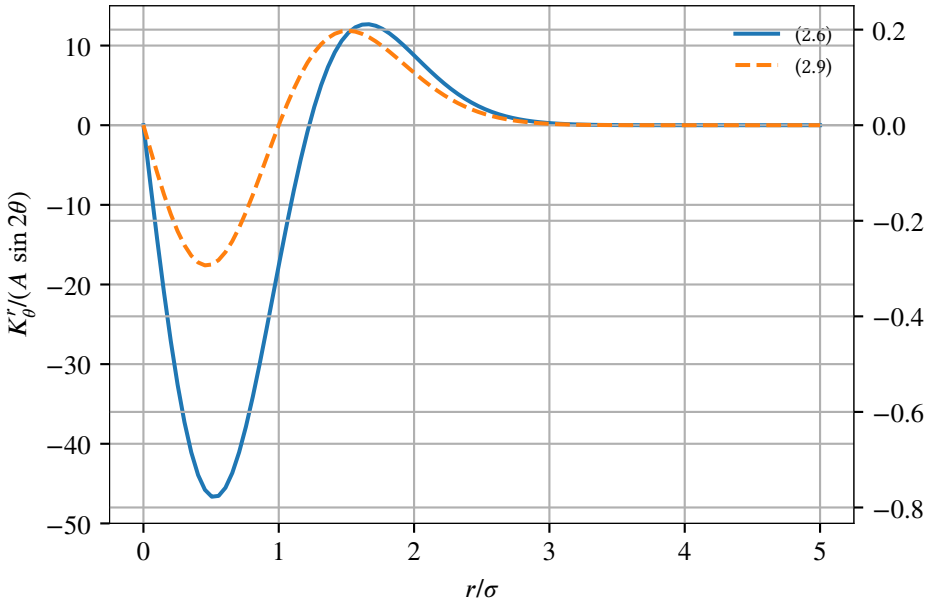

FIGURE 2.4: Profiles of the "seed function" K^r_θ: the one from Abrahams&Evans [2], given by (2.6) (left axis, solid); our "simplified" version (2.9) (right axis, dashed).

both profiles have two opposite-sign peaks followed by rapid decay to zero, their peaks have the ratio of ~ 2, while ours reach ~ 4 and are opposite in sign.

Another, more serious, disagreement with [2] became apparent once we tried solving equations (2.5) numerically (using our solver described in Chapter 7). We discovered that our solver, starting with a flat initial guess, would only find solutions up to $A_{\max} \approx 0.00921$ and would fail to converge for $A > A_{\max}$. Furthermore initial data close to $A_{\max}$ was clearly subcritical when evolved, while the critical point in [2] was stated to be $A \approx 6.3875$.

Investigating the immediate vicinity of $A = A_{\max}$ we discovered that the ADM mass of the initial slice M_{ADM} starts to grow rapidly when approaching $A_{\max}$:

$$\lim_{A \nearrow A_{\max}} \frac{\mathrm{d}M_{\text{ADM}}}{\mathrm{d}A} = \infty. \tag{2.7}$$

Conjecturing the dependency $M_{\text{ADM}}(A)$ to be parabolic around $A = A_{\max}$, we tried extrapolating from two solutions $u^k_{A_0}, u^k_{A_1}$ for some $A_0 > A_1$ close to $A_{\max}$

$$\tilde{u}^k_{A_1} = 2u^k_{A_0} - u^k_{A_1}. \tag{2.8}$$

Then using these extrapolated values $\tilde{u}^k_{A_1}$ as an initial guess for $A = A_1$, our solver converges to a *second* solution $\overline{u}^k_{A_1}$ with a higher M_{ADM} than $u^k_{A_1}$. Furthermore using this new solution as an initial guess we can construct an entire new branch of solutions, where M_{ADM} grows with decreasing A, apparently diverging as $A \to 0$. As $A \to A_{\max}$ both branches approach the same solution, so we can consider them together as a single one-parameter initial data family. Since A is no longer enough to uniquely identify a solution, we mark the "upper-branch" solutions with a bar: $A = A_1$ is a lower-branch (lower mass) solution, while $A = \overline{A}_1$ is an upper-branch (higher mass) solution.

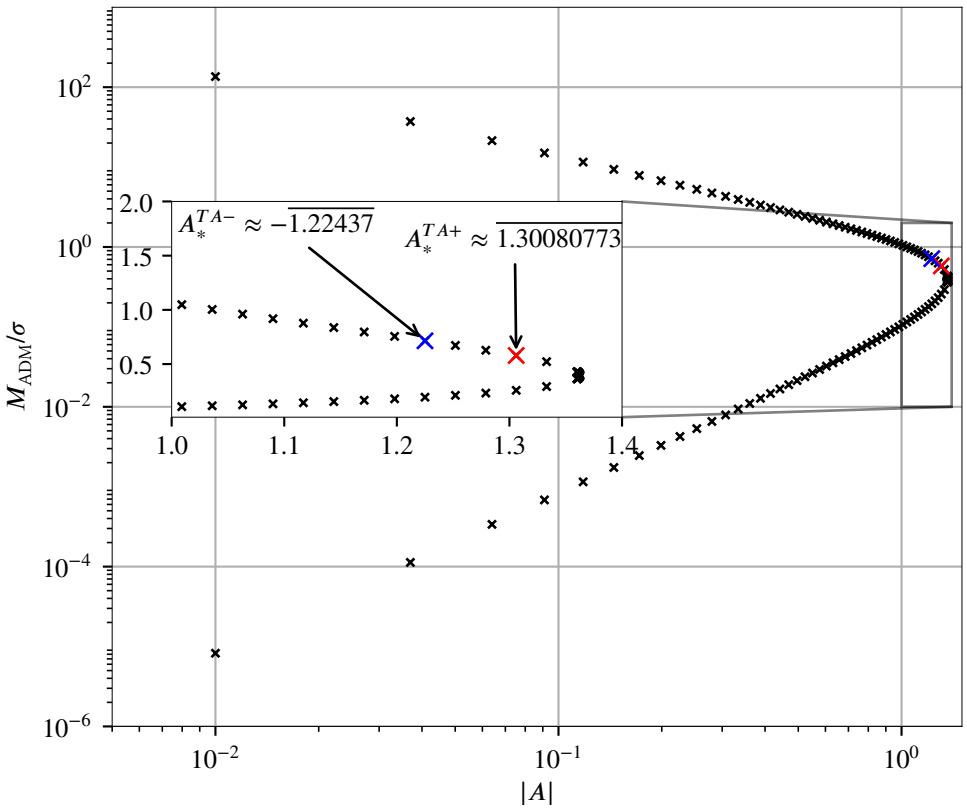

FIGURE 2.5: Dependence of the ADM mass M_{ADM} on the amplitude parameter A for the time-asymmetric data (2.9). Inset shows the approximate location of the critical point for forward and backward evolution (i.e. positive and negative A).

The origin of this non-uniqueness can be understood by estimating the A–M_{ADM} dependence in the following way. The Hamiltonian constraint is $8\Delta\psi = -\psi^5 K_{ij}K^{ij}$. The individual terms then roughly behave as $\psi \sim 1 + M$, $\Delta\psi \sim -M$, $K_{ij}K^{ij} \sim A^2$, which gives us $A^2 \sim M/(1+kM)^5$. This relation has a qualitatively similar form to the actual dependence (Figure 2.5) and predicts the existence of A_{max}. It was later pointed out to us by D. Hilditch that very similar non-uniqueness behavior for different kinds of initial data was observed in [43].

Since evolving this data shows the critical point A_* to be on the upper branch and the paper [2] mentions nothing about such a nontrivial structure of the solution space, it seems clear that the constraints solutions we managed to obtain by solving (2.5) with (2.6) are not the same that were used in [2]. We did not manage to determine why that is so. Despite this setback, we decided to investigate this data anyway, as an example of an ID family that is significantly different from Brill waves. However, since the specific form of (2.6) was guided by the goal of reproducing [2] — which was no longer on the table — we simplified the "seed function" to

$$K^r_\theta = A\left[\left(\frac{r}{\sigma}\right)^3 - \frac{r}{\sigma}\right]\exp\left[-\left(\frac{r}{\sigma}\right)^2\right]\sin(2\theta). \tag{2.9}$$

This is the form that we actually use in our simulations described in 12. As it is no longer guided by correspondence to the linearized waves [51], it does not seem appropriate to call this data "Teukolsky waves". Furthermore, as evolving this data

forward and backward in time produces different results (e.g. with different critical amplitudes), the "time-antisymmetric" label is also misleading. We will thus call the data (2.5), (2.9) "time-asymmetric waves" (TA).

The TA waves again have the non-unique structure of the solution space described above, with $A_{\max} \approx 1.3625$. As for Brill waves, we can also consider negative values of the parameter A. However here replacing $A \to -A$ merely flips the sign of K_{ij}, so we get the same initial slice evolved backward in time.

We show the dependence of the initial slice's mass on A in Figure 2.5, indicating the approximate location of the critical point for positive and negative values of A (details on locating the critical point are given in Chapter 12). Since $A_*^{TA+} \prec |A_*^{TA-}|$, the complete spacetime evolution follows one of these possibilities:

- $0 < A \prec A_*^{TA+}$: Both A and $-A$ evolutions are subcritical, so waves converge from infinity and disperse into infinity, leaving behind flat space.
- $A_*^{TA+} \prec A \prec |A_*^{TA-}|$: Evolution for A is supercritical, evolution for $-A$ is subcritical, so waves converge from infinity and collapse into a black hole.
- $|A_*^{TA-}| \prec A > 0$: Evolution for both A and $-A$ is supercritical, so waves emerge from a white hole and collapse into a black hole.

We also show the profiles of the unknown functions ψ, K_r^r, K_φ^φ found by our solver in Figures 2.6 and 2.7. Interestingly, K_r^r is discontinuous at the origin $r = 0$, having different limits along the ρ and z axes. We discuss this further in Chapter 7.

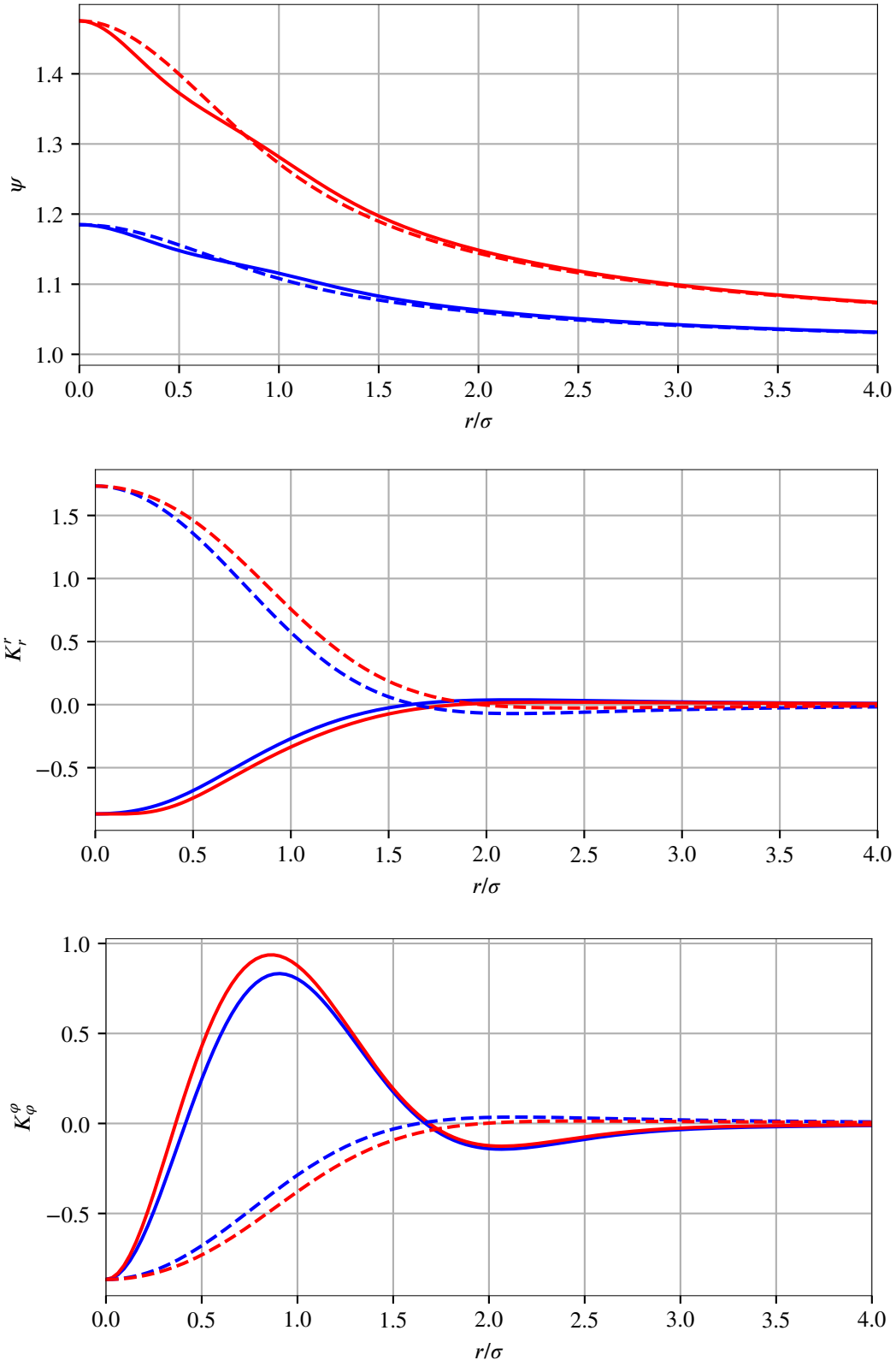

FIGURE 2.6: Solutions of the constraints (2.5), (2.9) for $A = 1.3$ (lower-branch, subcritical, plotted in blue) and $A = \overline{1.3}$ (upper-branch, supercritical, plotted in red). Solid: equatorial plane $z = 0$; dashed: axis of symmetry $\rho = 0$.

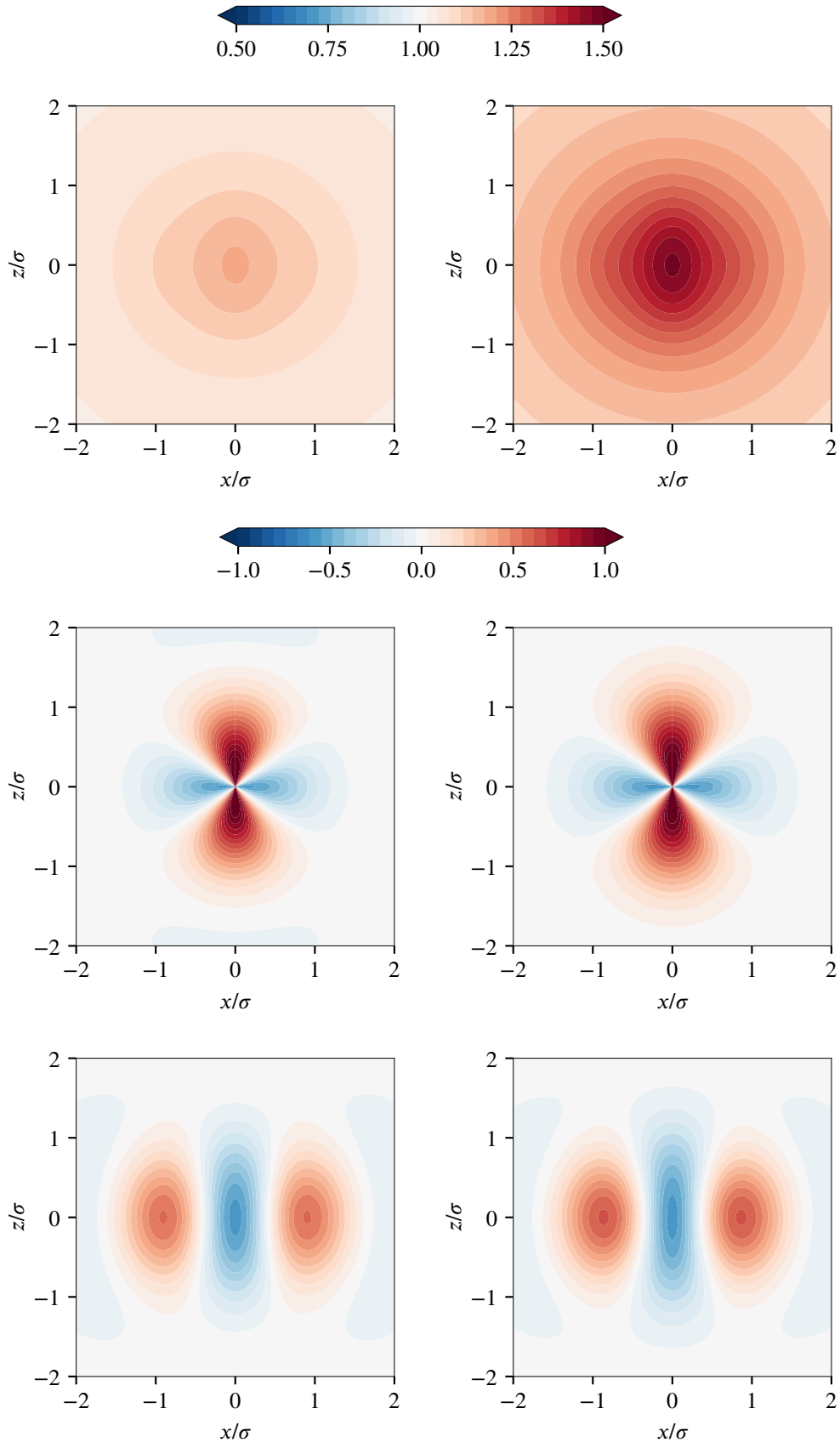

FIGURE 2.7: Solutions of the constraints (2.5), (2.9) for $A = 1.3$ (left) and $A = \overline{1.3}$ (right) in Cartesian coordinates $\{x, y, z\}$ restricted to the $y = 0$ plane. Shown are, from top to bottom: ψ, K^r_r, K^φ_φ.

Chapter 3

Slicing

Slicing is the choice of the lapse function α, which determines the foliation of the evolved spacetime into spatial slices. Experience shows that choosing an appropriate slicing condition is of crucial importance for evolving extreme spacetimes successfully. In this chapter we describe the slicing methods relevant to our simulations. The first two methods we mention are well-known textbook results, the third has been developed as a part of this work specifically to treat critical gravitational collapse of gravitational waves.

3.1 Maximal slicing

Maximal slicing is defined by requiring the trace of the extrinsic curvature $K = K^i_i$ to be identically zero throughout the simulation. Substituting $\partial_t K = 0$ into the evolution equation (1.19c), we get

$$-\mathrm{D}^i \mathrm{D}_i \alpha + K_{ij} K^{ij} \alpha = 0, \tag{3.1}$$

which is an elliptic equation to be solved for the lapse α at each spatial slice. Maximal slicing tends to produce smooth well-behaved slices and is strongly singularity avoiding. It is often considered the slicing of choice whenever technical difficulties do not prevent its use. These difficulties include:

- The evolution system becomes of a mixed elliptic-hyperbolic type, which is more complicated to handle, especially in combination with mesh refinement. We discuss this further in Chapter 9.

- A common strategy for handling outer boundaries with hyperbolic systems is simply making the domain so large that the area of interest is causally disconnected from the boundary. This can be achieved cheaply by using mesh refinement with very coarse outer grids. Then the boundary can be treated in a naive unphysical manner, e.g. by holding all the values fixed there. Because information propagates with finite velocity in hyperbolic systems, unphysical modes appearing at the boundaries will not pollute the central region during the lifetime of the simulation. In elliptic systems the information propagates instantly, so this strategy becomes less viable.

- Solving an elliptic equation requires much larger amounts of computing resources than doing a single hyperbolic evolution step, so a maximal slicing

solver will typically account for a large part of a simulation's runtime (over 50% in our setups).

- As is shown in [27], the constraint $K = 0$ needs to be enforced during the evolution for the system to be stable. Because of that, any inaccuracies in solving the equation (3.1) show up as constraint violations.

For these reasons we find maximal slicing impractical to use in our simulations, especially for near-critical spacetimes where high accuracy is required. While we do implement a maximal slicing module in our code, we do not use it for the simulations that produce the bulk of our results. It is mainly valuable to us as the starting point for the quasi-maximal slicing (QMS) described below.

3.2 1+log slicing

The so-called "1+log" slicing belongs to the Bona-Massó class of slicing conditions [12] and is widely used for evolving black-hole spacetimes. It derives from the desire to use a hyperbolic lapse condition — thus avoiding many of the abovementioned problems with maximal slicing — while preserving, inasmuch as possible, its advantages. The condition takes the form of a hyperbolic evolution equation for the lapse

$$(\partial_t - \mathscr{L}_\beta)\alpha = -2\alpha K, \tag{3.2}$$

placing α on equal footing with the evolved metric variables. With zero shift it asymptotically approaches maximal slicing, assuming the evolution settles down to a stationary state.

While the 1+log slicing has been successfully used in evolving black hole spacetimes, it has proven less robust for gravitational collapse of gravitational waves. In [31] it was discovered that 1+log slicing fails with near-critical Brill waves, forming a numerical discontinuity that was conjectured to correspond to a coordinate singularity. In our own simulations, described in Chapter 11, we confirm this failure for both Brill and TA waves and show new data supporting the coordinate-singularity theory. To run simulations close to the critical point we develop a modified version of the 1+log slicing, described in the next section.

3.3 Quasi-maximal slicing

The slicing we developed for our critical collapse simulations is intended to be a compromise between the smoothness of the maximal slicing and the high performance of the 1+log slicing. It consists of adding an extra source term to the 1+log evolution equation (3.2) that brings the slices closer to the maximal ones. For that reason we call it the "quasi-maximal slicing" (QMS).

The motivation for the quasi-maximal slicing can be summarized as follows:

- Maximal slicing is well-behaved, but slow.
- 1+log slicing is fast, but sometimes crashes.
- Lapse is a gauge function, so any condition we choose does not in principle need to be satisfied exactly, as long as the result is numerically well-behaved.

Our goal then was to construct a source term that would capture the "rough shape" of the maximal slicing and add it to the evolution equation (3.2). We start with the maximal slicing equation (3.1) and take its time derivative

$$0 = \left(\partial_t - \mathscr{L}_\beta\right) \left[\gamma^{ij} \mathrm{D}_i \mathrm{D}_j \alpha - K_{ij} K^{ij} \alpha\right], \tag{3.3}$$

which straightforwardly expands to

$$\begin{aligned} &\gamma^{ij} \mathrm{D}_i \mathrm{D}_j \left[(\partial_t - \mathscr{L}_\beta)\alpha\right] - K_{ij} K^{ij} \left[(\partial_t - \mathscr{L}_\beta)\alpha\right] = \\ &- \left[\left(\partial_t - \mathscr{L}_\beta\right) \gamma^{ij}\right] \mathrm{D}_i \mathrm{D}_j \alpha + \gamma^{ij} \left(\partial_t \Gamma^k_{ij}\right) \partial_k \alpha \\ &- \left(\gamma^{ij} \mathrm{D}_i \mathrm{D}_j \beta^k\right) \mathrm{D}_k \alpha - \beta^j R^i_j \mathrm{D}_i \alpha + \alpha \left(\partial_t - \mathscr{L}_\beta\right) \left(K_{ij} K^{ij}\right). \end{aligned} \tag{3.4}$$

Now we replace the time derivative of the lapse with a new function W

$$W = (\partial_t - \mathscr{L}_\beta)\alpha \tag{3.5}$$

and use the evolution equations to replace the time derivatives of γ_{ij} and K_{ij} with purely spatial quantities known on one slice. We obtain

$$\begin{aligned} &\mathrm{D}^i \mathrm{D}_i W - K_{ij} K^{ij} W = \\ &- 2\alpha K^{ij} \mathrm{D}_i \mathrm{D}_j \alpha + \gamma^{ij} \left(\partial_t \Gamma^k_{ij}\right) \partial_k \alpha - \left(\gamma^{ij} \mathrm{D}_i \mathrm{D}_j \beta^k\right) \mathrm{D}_k \alpha \\ &- \beta^j R^i_j \mathrm{D}_i \alpha + \alpha \left(2 \dot{K}_{ij} K^{ij} + 4\alpha K^i_j K^k_i K^j_k\right), \end{aligned} \tag{3.6}$$

where $\dot{K}_{ij}$ is a shorthand for the right-hand side of (1.15). This is an elliptic equation for the function W, with the same structure as the maximal slicing condition (3.1), but a more complicated right-hand side.

By construction, solving (3.6) for W and using it to evolve the lapse according to (3.5) causes the acceleration of K to vanish. If we start the evolution with $K = 0$ and the appropriate initial value for α, we would get precisely the maximal slicing (for now leaving aside the question of numerical stability of such a system).

The basic idea behind our quasi-maximal slicing is then to use an *approximate solution* W_a as a source term in the 1+log slicing condition

$$(\partial_t - \mathscr{L}_\beta)\alpha = -2\alpha K + \kappa W_a. \tag{3.7}$$

Here κ is some function of coordinate time t that allows us to "switch off" the source term when it is no longer needed. It is typically set equal to one at the beginning and then sent smoothly to zero after the waves in the central region disperse or a horizon forms.

It is known that for PDE systems such as BSSN we are not free to modify even the gauge conditions arbitrarily — adding combinations of evolved functions and their derivatives may lead to loss of well-posedness of the system. The source term added in (3.7) does not modify the principal part of the PDEs linearized around flat space, so it does not affect the hyperbolicity of the PDE system. This follows from the fact that (3.6) has a trivial principal part — $\Delta W = 0$. Furthermore, the source term does not contain the exact solution W, but some approximation W_a that does not approach W as simulation resolution is increased. We give the details on the construction of W_a in section 9.3.

Chapter 4

Spacetime analysis

The evolved BSSN quantities (1.19) are inherently coordinate-dependent and do not provide us with direct physical information about what is going on in the spacetime. Extracting physically meaningful quantities from the data then requires a bit more work, which is the subject of this Chapter. Our primary tools for this purpose are spacetime invariants and apparent horizons.

4.1 Invariants

Other than the Ricci scalar, which is zero in vacuum, the most obvious curvature invariant is the Kretschmann scalar

$$I_K = R_{\alpha\beta\gamma\delta} R^{\alpha\beta\gamma\delta}. \tag{4.1}$$

Its dimension is minus fourth power of length and in vacuum it can be written in terms of the ADM variables as

$$\begin{aligned} I_K = &\left(R_{ijkl} + K_{ik}K_{jl} - K_{il}K_{jk}\right)\left(R^{ijkl} + K^{ik}K^{jl} - K^{il}K^{jk}\right) \\ &+ 8\left[(\mathrm{D}_i K_{jk})(\mathrm{D}^j K^{ik}) - (\mathrm{D}_i K_{jk})(\mathrm{D}^i K^{jk})\right] \\ &+ 4\left(R_{ij} + KK_{ij} - K_{ik}K^k_j\right)\left(R^{ij} + KK^{ij} - K^{ik}K^j_k\right). \end{aligned} \tag{4.2}$$

For a Schwarzschild black hole (which should be the endstate of our supercritical spacetimes) with mass M, the Kretschmann scalar is

$$I_K = \frac{48M^2}{R^6}, \tag{4.3}$$

where R is the areal radius.

Values computed according to (4.2) within a numerical simulation tend to be susceptible to noise, because I_K contains second derivatives of the evolved variables squared and has a very fast falloff towards infinity (faster than that of the individual terms in (4.2)). Other independent invariants can be constructed from higher derivatives of the metric, but evaluating them accurately would be even harder.

Simpler invariants can be obtained from the fact that our spacetimes are axially symmetric. That implies the existence of the spacelike Killing vector field

$$\eta^i = \left(\frac{\partial}{\partial\varphi}\right)^i. \tag{4.4}$$

From this vector we can construct the circumferential radius $\overline{\rho}$ and the norm of its gradient

$$\overline{\rho}^2 = \eta_i \eta^i, \tag{4.5}$$

$$\overline{\rho}'^2 = g^{\mu\nu}(\partial_\mu \overline{\rho})(\partial_\nu \overline{\rho}). \tag{4.6}$$

Note that despite the notation $\overline{\rho}'^2$ can be negative, while $\overline{\rho}^2$ is always non-negative. One disadvantage of these invariants is that their values are uninteresting on the axis of symmetry: $\overline{\rho}^2\big|_{\rho=0} = 0$, $\overline{\rho}'^2\big|_{\rho=0} = 1$. Since the most interesting dynamics in our critical collapse simulations happens precisely on the axis, we want an invariant that has nontrivial values there. One such can be constructed as

$$\zeta = \frac{1 - \overline{\rho}'^2}{\overline{\rho}^2}, \tag{4.7}$$

completed by the appropriate limit on the axis. It can also be shown that on the axis I_K and ζ are related by $I_K = 12\zeta^2$, so ζ can be positive or negative, while I_K is always non-negative.

4.2 Apparent horizons

Another very useful diagnostic for black-hole spacetimes are apparent horizons. Unlike event horizons, which are global and can only be determined after the entire spacetime history is known, apparent horizons can be computed from data on a single spatial slice.

We start by considering a smooth closed two-dimensional surface S within a given spatial slice, with the outer unit normal vector s^i. Then the induced metric on S is

$$m_{ij} = \gamma_{ij} - s_i s_j \tag{4.8}$$

and the *expansion* Θ_o of outgoing null geodesics on S is

$$\sqrt{2}\Theta_o = m^{ij}(\mathrm{D}_i s_j - K_{ij}). \tag{4.9}$$

The expansion of *ingoing* null geodesics Θ_i is obtained by replacing $s^i \to -s^i$. Following [29] we call S a *marginally outer trapped surface* (MOTS) when $\Theta_o = 0$ everywhere on S. The outermost MOTS is then called the *apparent horizon* (AH).

Locating apparent horizons is typically done by looking for solutions of equation (4.9) with $\Theta_o = 0$ and selecting the outermost one. In the following text we keep Θ_o as an arbitrary constant, allowing us to locate any constant-expansion surfaces.

Having found an apparent horizon, one can assign a mass to it as

$$M_{\mathrm{AH}} = \sqrt{\frac{A_{\mathrm{AH}}}{16\pi}}, \tag{4.10}$$

with A_{AH} the area of the surface.

4.2.1 Spherical parametrization

The usual approach in axial symmetry [3] is choosing some origin point and constructing spherical coordinates $\{r, \theta, \varphi\}$ centered on this point. The surface is then parametrized in those coordinates as a level set $F(r, \theta) = 0$ of the function

$$F(r, \theta) = r - h(\theta). \tag{4.11}$$

Then the unit normal vector can be written as

$$s_i = \frac{\partial_i F}{w} = \frac{1}{w}\left\{1, -h', 0\right\}, \tag{4.12a}$$

$$w = \sqrt{\gamma^{ij}(\partial_i F)(\partial_j F)}, \tag{4.12b}$$

where the prime denotes differentiation with respect to θ. Substituting the normal into equation (4.9) gives us

$$\begin{aligned}\sqrt{2}\Theta &= m^{ij}\left(\partial_i s_j - \Gamma^k_{ij} s_k - K_{ij}\right) \\ &= m^{ij}\left(\frac{\partial_{ij} F}{w} - \frac{\partial_i w}{w} s_j - \Gamma^k_{ij} s_k\right) \\ &= -\frac{m^{\theta\theta}}{w} h'' - m^{ij}\left(\Gamma^k_{ij} s_k + K_{ij}\right).\end{aligned} \tag{4.13}$$

Isolating the highest derivative turns this equation into

$$h'' = \frac{-w^3}{\gamma^{rr}\gamma^{\theta\theta} - \left(\gamma^{r\theta}\right)^2}\left[m^{ij}\left(\Gamma^k_{ij} s_k + K_{ij}\right) + \sqrt{2}\Theta_o\right], \tag{4.14}$$

a second-order non-linear ODE to be solved for $h(\theta)$. All the metric variables are understood to be evaluated at $r = h(\theta)$. The boundary conditions follow from the requirement that S is smooth, so $h'(0) = h'(\pi) = 0$.

The origin of the parametrization coordinates is typically chosen to lie on the axis of symmetry [1] , but need not coincide with the origin of the simulation coordinates. E.g. in [33] it was discovered that Brill waves sufficiently close to A_* collapse into twin AHs above and below the equatorial symmetry plane.

4.2.2 Cartesian parametrization

One major limitation of the parametrization (4.11) is that it assumes the surface to be a *ray-body*, where every coordinate ray from the center $r = 0$ hits the surface exactly once. It turns out that some apparent horizons in our numerically constructed spacetimes violate this assumption and so cannot be located using this method. For that reason we develop an alternative parametrization that avoids the ray-body limitation.

This second parametrization is formulated in Cartesian coordinates restricted to the $y = 0$ plane, in line with the Cartoon approach described in section 5.3.1. Within this plane the surface S can be described as a parametrized curve $X^i(\lambda) = \{X(\lambda), 0, Z(\lambda)\}$ with a parameter λ. The meridian tangent vector is then

$$u^i = \frac{\mathrm{d}X^i}{\mathrm{d}\lambda}. \tag{4.15}$$

[1] Unless one is trying to locate toroidal apparent horizons, which we do not do in this work.

The parameter λ can be normalized to give the tangent vector unit norm $u^i u_i = 1$. The induced metric on the surface can then be built from this tangent and the angular Killing vector

$$m^{ij} = u^i u^j + \frac{1}{\overline{\rho}^2}\eta^i \eta^j. \tag{4.16}$$

The unit normal vector is orthogonal to both of them

$$s_i = \pm \epsilon_{ijk} \frac{\eta^j}{\overline{\rho}} u^k. \tag{4.17}$$

The choice of the sign determines the surface orientation. Equation (4.9) then transforms to

$$\begin{aligned} \sqrt{2}\Theta_o &= u^i u^j \mathrm{D}_i s_j + \frac{1}{\overline{\rho}^2}\eta^i \eta^j \mathrm{D}_i s_j - m^{ij} K_{ij} \\ &= -s_j u^i \mathrm{D}_i u^j + \frac{1}{2\overline{\rho}^2} s^i \mathrm{D}_i \overline{\rho}^2 - m^{ij} K_{ij}, \end{aligned} \tag{4.18}$$

where we used orthogonality $u^i s_i = 0$ in the first term and the Killing equation in the second. Expanding the directional covariant derivative and again isolating the highest derivative gives us

$$\frac{\mathrm{d}^2 X^k}{\mathrm{d}\lambda^2} = s^k \left(\frac{1}{2\overline{\rho}^2} s^i \mathrm{D}_i \overline{\rho}^2 - m^{ij} K_{ij} - \sqrt{2}\Theta_o \right) - \Gamma^k_{ij} u^i u^j. \tag{4.19}$$

These are two coupled non-linear ODEs for $X(\lambda)$, $Z(\lambda)$. As before, the metric variables are to be evaluated at $\{x = X(\lambda), y = 0, z = Z(\lambda)\}$. The boundary conditions are $u^z(0) = u^z(\lambda_{\max}) = 0$, where $\lambda_{\max} > 0$ is the value of the parameter λ such that $X(\lambda_{\max}) = 0$.

Part II

Numerical codes

Chapter 5

The Einstein Toolkit/Cactus framework

Our numerical simulations of spacetime evolution are based on the Einstein Toolkit [22, 23] (2013 release), which is in essence a distribution of the Cactus [17, 18, 26] grid computation environment and NR-oriented Cactus modules (called *thorns*). The most important thorns for our purposes are the Carpet [19, 48] *driver* implementing Berger-Oliger-style mesh refinement [11], and the McLachlan [16, 36, 39] code implementing the BSSN formulation of the Einstein equations.

In this chapter we describe the main features of these codes as they pertain to our simulations, along with the changes we made to adapt them to our use case. See Appendix A.2 for obtaining the complete source code as we used it for this work.

5.1 Overview

Cactus consists of the core part, called the *flesh*, and a number of modules called *thorns*. The flesh and the selected thorns are compiled into single executable that performs the requested computation. The executable accepts as input a *parameter file*, which is a text file specifying which of the compiled thorns are to be activated and what parameters are to be passed to them. The parameter file thus determines the precise simulation that is to be run.

The flesh does little more than parsing the parameter file, setting up some basic environment, and loading the specified thorns. Everything else, including all numerical code and most of the "infrastructure", is distributed across the thorns themselves. One special thorn is called the *driver* — it sets up the numerical grid(s) and manages the evolution loop, multi-process parallelism via MPI, I/O and other "core" tasks. The driver we use is Carpet, which sets up a hierarchy of nested grids at different resolutions and evolves them following the Berger-Oliger mesh refinement algorithm, outlined below.

The primary mechanism for executing thorn code is built around so-called *schedule groups* and involves a number of ordered trees (i.e. trees where each node's children are arranged in *traversal order*). Each node has a label, then a schedule group X consists of the node with the label X and all its children, recursively. Each root schedule group (i.e. the root of the corresponding tree) is named according to its function, such as STARTUP for initialization code, INITIAL for initial data setup, or EVOL for evolution. Individual thorns then may add either new nodes (sub-groups) or leaves (functions

to be called) to existing groups, together with ordering constraints (e.g. before or after a specific child in the same group). The bulk of the simulation then consists of traversing the schedule groups in appropriate order.

For example, the following code (written in the Cactus-specific mini-language for describing schedule groups)

```
SCHEDULE msa_mg_eval IN MoL_CalcRHS BEFORE ML_BSSN_evolCalcGroup {
    SYNC: ML_lapse
    LANG: C
} ""
```

in our maximal slicing thorn (described in Chapter 9) causes the msa_mg_eval C function (which sets up the lapse for the current time step) to be executed in the MoL_CalcRHS group (which contains all the right-hand-side computations for the quantities evolved with the method of lines) before the ML_BSSN_evolCalcGroup group (which computes the right-hand sides of the BSSN evolution equations, which involve the lapse).

The thorns also declare *grid functions*, which are spatial fields represented by discrete samples on a grid. The driver manages the memory arrays where the grid functions are stored and makes them available to the thorn code.

The simulation process can then be outlined as follows:

1. The parameter file specifies which thorns are to be activated.
2. The schedule groups are populated according to the active thorns.
3. The driver traverses the startup group(s), initializing all the thorns.
4. The driver sets up the numerical grids and grids functions on them, corresponding to the refinement levels specified in the parameter file.
5. The driver traverses the initial data schedule group(s), filling each refinement level with initial data. In our case this invokes libbrilldata or libteukolsky-data.
6. The evolution loop starts. The driver repeatedly traverses the evolution schedule groups on each refinement level according to the Berger-Oliger mesh refinement algorithm.
7. Once the termination criterion, specified in the parameter file, is satisfied (e.g. reaching a certain evolution time), the loop is terminated and the simulation ends.

5.2 Parallelization

The Einstein Toolkit contains facilities for running simulations on multiple processor cores simultaneously, in order to compute the results faster. This may involve multiple cores within a single physical processor, several processors within a single *multi-socket* machine, multiple interconnected machines, or any combinations of these.

Two distinct techniques can be used for this purpose: shared-memory *multithreading* and multi-process *MPI* parallelization. The former involves multiple threads of execution in a single process. These threads share the same memory address space

and so can access the same working-memory data. In a typical simulation, most of the runtime is spent within grid function operations, where a large output data array is computed from one or more input data arrays. Each output element is computed independently, making the operation *trivially parallelizable*: multiple threads can compute different output elements concurrently. Cactus thorns typically use the OpenMP [41, 42] technology to multi-thread grid operations with very little extra code.

Conversely MPI (Message Passing Interface) [40] parallelization involves the use of multiple processes that communicate by passing messages. Within Cactus, every refinement level (see next section) comprising the numerical domain is partitioned into rectangular subdomains — one per process. Each process then evolves its own subdomain, using MPI communication to synchronize required data with the other processes. The actual method employed by the MPI library for passing the messages depends on the process *topology*: the other processes may be located on the same machine (then OS-level memory sharing can be used) or on different machines reachable via *Ethernet* or a low-latency interconnect such as *InfiniBand*.

Since MPI has much higher overhead than multithreading, it is typically preferable to use the latter when possible and the former when necessary. However, in setups such as a multi-socket machine with processor-local physical memory banks, it is often advantageous to run one MPI process per physical processor (using multithreading within each process to fully load all the cores in the processor). The overhead of MPI communication is in this case more than balanced out by processor working on its local memory, which it can access with much lower latency.

5.3 Carpet & mesh refinement

The Carpet project consists of a collection of Cactus thorns that provide mesh refinement and associated infrastructure. While the (by now quite antique) version of the code we are using has some support for *adaptive* mesh refinement, we did not find it easily usable for our work. In our simulations we thus use *fixed* mesh refinement, where the levels are manually specified in the parameter file and do not change during the simulation. Another feature of Carpet that we do not use are multiple refined patches within a level — there is exactly one grid at each refinement level in our simulations.

The simulation domain thus consists of D square nested grids, numbered from 0 (coarsest) to $D-1$ (finest). Each finer level halves the spacing between the grid points and the time step, as required by the CFL condition. The grids are *properly nested* — a refined grid at l-th level with all of its *ghost* zones (described below) is entirely contained within its parent $l-1$-th level. Carpet manages the grid function storage at each level and *prolongation* and *restriction* between the levels. Evolving the l-th level from time $t \to t + \Delta t$ is conceptually done as follows:

1. Make a single time evolution step on the level l, using the supplied time step Δt. This is performed by setting up the environment that is to be available to thorn code on l-th level and then traversing the evolution schedule group once.

2. If $l = D - 1$, finish the step and return.

3. Invoke this algorithm twice on the $l+1$-th level with the time step $\Delta t/2$.

4. *Restrict* the grid functions from the $l+1$-th level onto the intersecting points of the l-th level.

5. *Prolongate* the grid functions from the l-th level onto the *buffer zones* (described further on) of the $l+1$-the level.

There are two kinds of grid boundaries in this scheme — *physical* and *refinement*. The former delimit the entire computation domain — we do not perform the simulation beyond them. The latter are located at the edges of a refinement level — the simulation is performed beyond a refinement boundary, but at reduced resolution. Both kinds of boundaries are treated similarly using the notion of *ghost zones*. A ghost zone is a thin layer of points along a grid boundary that is accessed during the evolution process (e.g. by finite difference or dissipation operators), but is not evolved by itself. Instead, it is filled through some other means. The thickness g (in points) of a ghost zone is specified in the parameter file and the value is chosen depending on the order of the various grid operators used.

5.3.1 Physical boundaries & Cartoon

The physical boundaries are treated using a single ghost layer that is updated after each time integrator sub-step and can be further divided into *outer* and *symmetry* boundaries. The former exist only on the coarsest grid and represent the outermost edge of the simulated system. The corresponding ghost zone is filled according to the boundary condition prescribed in the parameter file (e.g. static or radiative). The symmetry boundaries are used to enforce some symmetry condition and exist on all the refinement levels that touch the relevant boundary.

We implemented the analytical *Cartoon* method [32] to treat axial symmetry in Cartesian coordinates $\{x, y, z\}$. It consists of restricting the numerical domain to the $y = 0, x \geq 0$ half-plane. Reflection symmetry condition is then applied on the symmetry axis $x = 0$. As our spacetimes have an additional reflection symmetry with respect to the equatorial plane, the $z = 0$ axis is also a symmetry boundary in all our simulations and we only evolve the $x \geq 0, z \geq 0$ quadrant. However this is orthogonal to the use of the Cartoon method.

All the partial derivatives in the y direction (which would need to access points at $y \neq 0$ when replaced by finite difference operators) are then replaced by combinations of function values and their x/z derivatives, as follows from axial symmetry. The precise replacement for each quantity depends on its tensorial type and is given explicitly e.g. in [32]. We perform this replacement manually in all the components that compute partial derivatives of grid functions — BSSN right-hand sides, constraints computation, QMS, and dissipation.

The Einstein Toolkit also contains the Cartoon2D thorn implementing the "original" Cartoon method [6], in which the evolved $y = 0, x \geq 0$ half-plane is sandwiched between layers of ghost points in the positive and negative y directions. Those ghost points are filled by applying rotation symmetry to the values at $y = 0$. The advantage of this method is that it does not require as much code to be explicitly "Cartoon-aware". It is, however, less accurate due to the use of spatial interpolation. Even more importantly, it requires significantly more memory and computational time, so we avoid using it.

5.3.2 Refinement boundaries

The refinement boundaries serve as the equivalent of outer physical boundaries for the refined levels $l > 0$, except the boundary information is provided by the coarser levels. We make use of the Carpet feature called *tapered grids*, which allows treating the refinement boundaries without resorting to time interpolation (typically used otherwise). It involves placing at each refinement boundary a *buffer zone* consisting of $2s$ ghost-point layers, where s is the number of sub-steps used by the time integrator (i.e. the number of times it evaluates the right-hand sides of the evolution equations). This buffer zone is then filled using interpolation from the coarser level each time the two levels are in sync. The individual ghost layers are then progressively invalidated as time integration on the finer level proceeds, so that by the time the two levels are in sync again the entire buffer zone is "used up".

Prolongation from a coarser grid onto a buffer zone uses Lagrange interpolation of the order specified in the parameter file; typically we choose one higher than the order of the finite difference operators. Conversely, restriction from a fine grid onto a coarser one is done by *injection* (i.e. simply copying the values), since every overlapping coarse-grid point also exists on the finer grid.

5.3.3 MPI & synchronization

Carpet also does most of the heavy lifting required for MPI parallelism: partitioning the domain between the processes, synchronizing the required data, interpolating between different grids, etc., so that most of the numerical code does not need to perform any explicit MPI operations. The boundaries between components located in different processes are handled using *synchronization ghost zones*: they work similarly to physical-boundary ghost zones, except they are filled via MPI synchronization operations.

One limitation of Carpet's MPI handling is that the code is written with purely hyperbolic systems in mind. Adapting it to a mixed elliptic-hyperbolic case then requires performing certain synchronization operations manually in our own thorns, as described in Chapter 9.

5.4 Evolution & output

Time evolution in the Einstein Toolkit follows the *method of lines*, which separates time integration from spatial discretization. The PDE system under consideration is assumed to be written in the first-order-in-time form

$$\partial_t u^a = F^a[u^b], \tag{5.1}$$

with evolved spacetime quantities $u^a = u^a(t, x^j)$ and the right-hand side operators F^a that do not contain time derivatives of the evolved quantities u^b. Then we discretize the system in space by sampling u^a on a grid, obtaining grid functions $u^a_I = u^a_I(t)$, where the multi-index I identifies grid points. The right-hand sides F^a are also discretized by replacing the spatial derivatives with finite difference operators of the chosen order o, resulting in

$$\partial_t u^a_I(t) = F^a_o[u^b_I(t)]. \tag{5.2}$$

This equation then may be regarded as a large collection of ODEs — one for every grid point I and evolved variable u^a. These ODEs then may be integrated using standard techniques, such as the Runge-Kutta method.

Within Cactus, the method of lines is implemented by the appropriately-named MoL thorn, which provides a variety of time integrators. Other thorns then may register evolved variables and schedule right-hand side evaluation code into the schedule group provided for that purpose by MoL. Right-hand sides evaluated in this manner are then combined by MoL to step forward in time.

The ML_BSSN thorn from the McLachlan project implements the right-hand sides F_o^a corresponding to the BSSN formulation of the Einstein equations, along with the right-hand sides for the 1+log slicing and the Γ-driver shift condition (we set the corresponding factor to zero in our parameter files, so shift is not evolved). We modified this thorn by:

- Splitting off 1+log lapse evolution, so it can be enabled or disabled independently of the other equations. This allows maximal slicing to be used.
- Adding the W_a source term from equation (3.7) to the original 1+log condition.
- Replacing the y derivatives according to the analytic Cartoon method.

In order for this kind of an evolution scheme to be stable, it is important to damp high-frequency components that spuriously appear due to numerical inaccuracies. That is accomplished through the use of Kreiss-Oliger dissipation [37], implemented in the aptly-named Dissipation thorn. As for the evolution equations, we modify the dissipation scheme to be compatible with the Cartoon method.

Carpet also includes capabilities to output simulation data into persistent storage. We use the CarpetIOHDF5 thorn to periodically store values of the grid functions into HDF5 [30] files — both on the entire 2D grid (with low frequency) or 1D data along the x and z axes (with higher frequency). Carpet can also checkpoint a running simulation state into a file and later restart from it, which is useful for running very long simulations.

Chapter 6

LIBBRILLDATA — Brill wave initial data solver

This chapter describes the code we wrote to construct the Brill wave initial data (described in section 2.1) by solving equation (2.2). The solver is implemented as a standalone library called LIBBRILLDATA, written mostly in C with small amounts of optional vectorized x86 assembly code [1]. The library exports a C API and also contains a simple Python wrapper. To construct Brill waves in a CACTUS simulation we also wrote a very simple CACTUS thorn called BRILLDATA, which contains the "glue code" to bridge between CACTUS and LIBBRILLDATA. We found the flexibility provided by this approach to be very useful, as the library can be easily verified and reused outside of CACTUS. See Appendix A.2 for obtaining the source code of LIBBRILLDATA.

The solver is a mostly straightforward application of the pseudospectral (collocation) method, which is thoroughly explained e.g. in the excellent book [14]. It can internally work with either cylindrical or spherical coordinates, though the current version will always use spherical. Cylindrical coordinates are mainly a development artifact — they were easier to implement since they map directly to the coordinates used in simulations. Spherical coordinates are significantly more efficient, requiring far fewer coefficients to reach the same accuracy.

Due to axial symmetry the problem is effectively 2D, with the Laplace operator from (2.1) taking the form

$$\Delta\psi = \partial_{\rho\rho}\psi + \frac{1}{\rho}\partial_\rho\psi + \partial_{zz}\psi \tag{6.1}$$

$$= \partial_{rr}\psi + \frac{2}{r}\partial_r\psi + \frac{\cos\theta}{r^2\sin\theta}\partial_\theta\psi + \frac{1}{r^2}\partial_{\theta\theta}\psi. \tag{6.2}$$

in cylindrical and spherical coordinates respectively. The coordinate singularity on the symmetry axis $\rho = 0$ is easily treated by applying the l'Hôpital's rule to the singular term. The coordinate singularity at the origin $r = 0$ in spherical coordinates is more complicated, but we avoid it by simply not placing any collocation points there.

We expand the unknown conformal factor as a truncated series

$$\psi = 1 + \sum_{i=0}^{N_0-1}\sum_{j=0}^{N_1-1} C_{ij} B_0^i(x^0) B_1^j(x^1) \tag{6.3}$$

[1] While the processor time spent in the initial data solver is negligible compared to the total runtime of near-critical simulations, speeding up the evaluation of ψ made it more convenient to analyze initial slices (e.g. for locating AHs) and served as practice for the more important QMS code.

where B_0^i, B_1^j are the basis functions for each spatial direction, C_{ij} the unknown expansion coefficients, and N_0, N_1 the expansion orders along each spatial dimension. $\{x^0, x^1\}$ are either $\{\rho, z\}$ for cylindrical coordinates or $\{r, \theta\}$ for spherical. Since ψ is expected to behave as $1 + M_{\mathrm{ADM}}/2r + O(r^{-2})$ towards infinity, the functions

$$\mathrm{SB}_{2k}(x) = \sin\left((2k+1)\mathrm{arccot}\frac{x}{L}\right) \tag{6.4}$$

introduced in [13] are an excellent fit for the radial basis. They are symmetric with respect to $x = 0$ and decay as $1/x$ towards infinity. Here L is the compactification scale that needs to be tuned to the problem at hand. We discuss our choice of L below. When using cylindrical coordinates, we take $B_0^i = B_1^i = \mathrm{SB}_{2i}$. With spherical coordinates we take the obvious choice of even cosines for the angular part: $B_0^i = \mathrm{SB}_{2i}$, $B_1^i = \cos(2i\theta)$. Both choices automatically satisfy the symmetry and decay properties expected from the solution, so we do not need to impose explicit boundary conditions.

The pseudospectral method consists of requiring the equation to be satisfied exactly at $N = N_0 N_1$ collocation points, which implies N linear equations for the N coefficients C_{ij}. The collocation points are chosen as the tensor product of 1D grids associated with each basis function. For even cosines the obvious collocation grid is

$$\theta_b = \frac{b\pi}{2N_1}; \quad b \in \{0 \dots N_1 - 1\}. \tag{6.5}$$

The situation is slightly more complicated for the SB functions — we want to avoid placing any points at infinity and the origin. There are multiple viable grids, the one we found to converge fastest with N_0 is

$$x_a = L\cot\left(\pi\frac{(a+2)}{2N_0+3}\right); \quad a \in \{0 \dots N_0 - 1\}. \tag{6.6}$$

The linear system to be solved then takes the form

$$\mathscr{M}^{ij}_{ab} C_{ij} = S_{ab}, \tag{6.7}$$

where a, b and i, j both run over N_0, N_1 and index, respectively, the collocation grid points and the unknown expansion coefficients.

$$S_{ab} = -\frac{1}{4}\left[\partial_{\rho\rho} q + \partial_{zz} q\right](x^0_a, x^1_b) \tag{6.8}$$

contains the values of the right hand side at the collocation points, while the entries of the pseudospectral matrix are

$$\mathscr{M}^{ij}_{ab} = \left[\Delta(B_0^i B_1^j) + \frac{1}{4}(\partial_{\rho\rho} q + \partial_{zz} q) B_0^i B_1^j\right](x^0_a, x^1_b). \tag{6.9}$$

To express (6.7) as a matrix-vector problem, we need to "pack" the pairs of indices into a single integer index. This is done straightforwardly as

$$I = N_0 j + i \tag{6.10}$$

and analogously for $A \leftrightarrow a, b$, with the inverse operation

$$j = \mathrm{floor}(\frac{I}{N_0}), \tag{6.11}$$

$$i = I \% N_0, \tag{6.12}$$

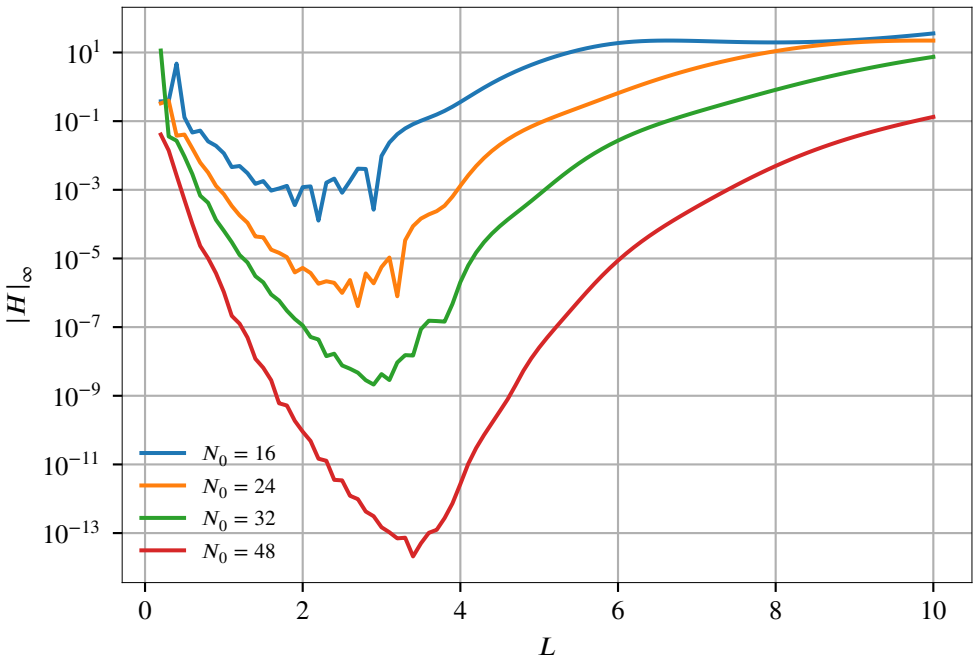

FIGURE 6.1: Dependence of the solver accuracy on the compactification factor L. Plotted is the maximum value of the residual against L for different radial solver order N_0. Angular solver order is $N_1 = 10$; amplitude parameter $A = 5$.

where % is the *modulo* operator (remainder after integer division). Then the linear system

$$\mathscr{M}^I_A C_I = S_A \tag{6.13}$$

is solved using LU decomposition, implemented in LAPACK [8]. We can then evaluate ψ at an arbitrary location using (6.3).

Computation constraints on this method stem from the fact that $\mathscr{M}^I_A$ is a full $N \times N$ matrix. Then in 64-bit floating point precision we need $8N^2$ bytes of working memory to store the matrix and $\sim 2/3N^3$ FLOPs to solve the linear system. A high-end modern system might have $\sim 10^{11}$ bytes of working memory and $\sim 10^2$ cores, each performing $\sim 10^{10}$ FLOP/second. Both the memory constraints and requiring the solution to not take more than $\sim 10^3$ seconds (assuming perfect parallelism) imply $N \lesssim 10^5$. Various practical considerations (such as other data being present in memory, or the code not being perfectly parallel) can be expected to reduce this limit by another order of magnitude in a real-world computation. While $N \sim 10^4$ is far beyond what is needed — constructing critical Brill data requires only ~ 50 radial and ~ 10 spherical basis functions for roundoff error to become dominant — these bounds will be relevant in the following Chapter.

The compactification factor L may be provided to the solver as a parameter and defaults to $L = 3$, which follows from testing shown in Figure 6.1. Figures 6.2 and 6.3 show clean spectral convergence of the residual (Hamiltonian constraint violation) with increasing N_0 and N_1, until the roundoff error of $\lesssim 10^{-13}$ becomes dominant.

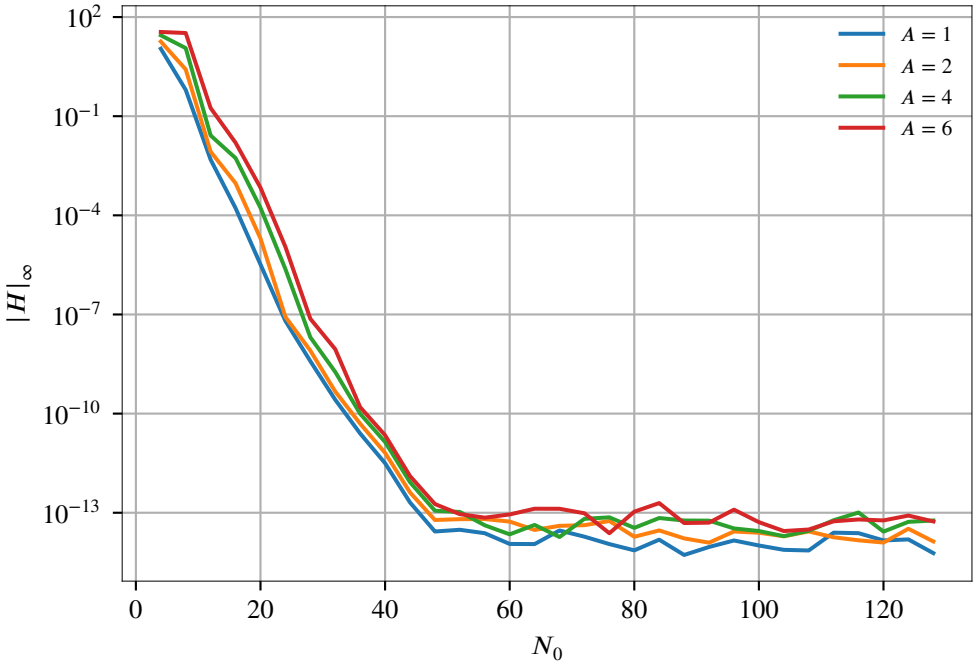

FIGURE 6.2: Dependence of the solver accuracy on the radial solver order N_0 for several different values of A. Angular solver order is $N_1 = 16$.

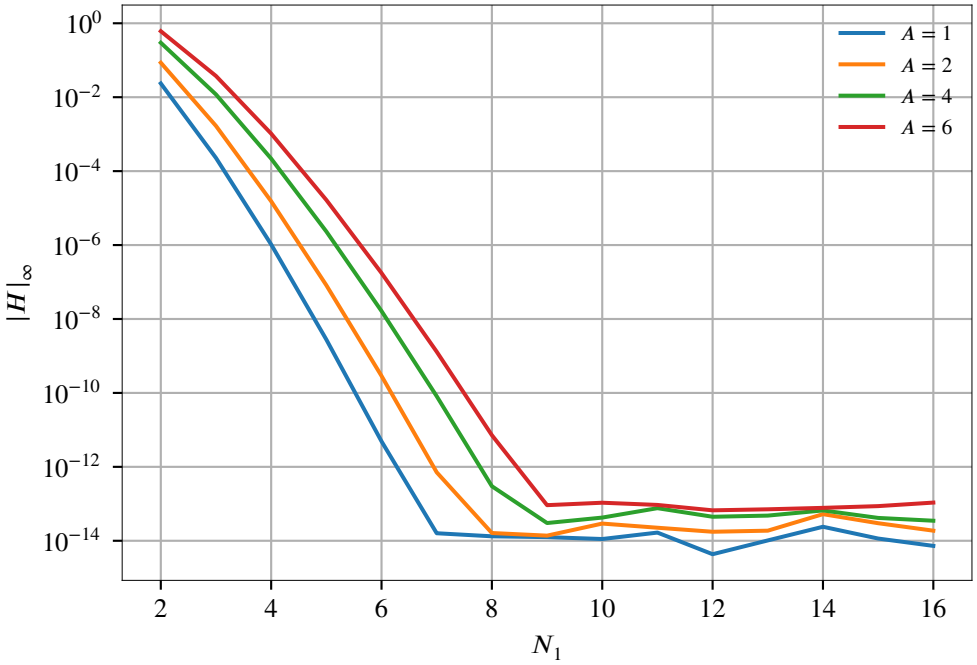

FIGURE 6.3: Dependence of the solver accuracy on the angular solver order N_1 for several different values of A. Radial solver order is $N_0 = 80$.

Chapter 7

LIBTEUKOLSKYDATA — Teukolsky/time-asymmetric initial data solver

In this chapter we describe our code for constructing the time-asymmetric initial data of section 2.2. Since it historically grew out of our attempts to reconstruct the Teukolsky waves used in [2], the solver is called LIBTEUKOLSKYDATA. Similarly to LIBBRILLDATA, LIBTEUKOLSKYDATA is also a standalone library written in C, with an additional Python wrapper, and a simple "glue" CACTUS thorn called TEUKOLSKYDATA.

To construct the data we need to solve equations (2.5). In some respects they are similar to the Brill wave equation (2.2): both are boundary-value problems on the same semi-infinite domain. The unknown functions all have the same parity behavior on the symmetry boundaries and polynomial decay towards infinity. For those reasons our solver is derived from LIBBRILLDATA, is again based on the pseudospectral method, and uses the same basis functions (though now only spherical coordinates are implemented).

There are, however, three key differences:

1. there are three coupled equations for three unknown functions, rather than one;
2. the equations are non-linear;
3. the solution space is more complicated, as described in section 2.2.

To deal with these points, the solver now has a more complicated architecture composed of several layered components. They are described in the following sections.

7.1 PSSOLVE — linear vector pseudospectral PDE solver

The innermost component is called PSSOLVE and is an evolution of the core of LIBBRILLDATA. It solves a sequence of P linear second-order equations

$$E^{ab}_{ik}\partial_{ab}u^k + F^{a}_{ik}\partial_{a}u^k + G_{ik}u^k = S_i, \tag{7.1}$$

where

- i numbers the equations and runs over $\{0, \dots, P-1\}$;

- k numbers the unknown functions and also runs over $\{0, \ldots, P-1\}$;
- a, b number the spatial directions and run over 0, 1;
- u^k are the unknown functions;
- $E^{ab}_{ik}, F^a_{ik}, G_{ik}$ are the (spatially varying) coefficients in front of the second, first and zeroth derivatives of the unknown functions;
- S_i are the right-hand sides of the equations.

A PSSOLVE instance is initialized for a given value of P, a choice of basis functions $B^k_a(x^a)$ and spectral expansion orders N^k_a. While the code is written to allow varying spectral bases and orders for different unknown functions, in practice we always use the same $B_a(x^a)$ and N_a for all k and never investigated solver behavior when any of those would be unequal.

PSSOLVE differs from the LIBBRILLDATA code in three main ways. First, the equation coefficients are provided to it as input, rather than being hardcoded. Second, there are P unknown functions represented by $N_0 N_1 P$ coefficients C^k_{ij}, changing the "packing" scheme (6.10) to

$$I = N_0 N_1 k + N_0 j + i \tag{7.2}$$

and analogously for "unpacking". In principle the same applies to collocation points, but in practice we use the same collocation points for all the equations — each collocation point corresponds to P rows in the pseudospectral matrix, one row per equation.

The third difference is that an initialized solver instance will typically be used multiple times to obtain solutions for varying equation coefficients and right-hand sides. As our problem requires solving successive linear systems with slowly changing coefficients, we make use of the BiCGSTAB method [52]. BiCGSTAB — biconjugate gradient stabilized method, related to a better-known conjugate gradient method for positive-definite problems — is an iterative method for solving linear systems. Given a good preconditioner it can converge in just a few iterations (each involving several matrix-vector multiplications), significantly outperforming an LU-based solve. Since the ideal preconditioner is the matrix inverse $\mathscr{M}^{-1}$ itself, the algorithm for a single PSSOLVE call is as follows:

1. Accept on input the equation coefficients $E^{ab}_{ik}, F^a_{ik}, G_{ik}$, the right-hand sides S_i and the initial guesses $\tilde{u}^k$ evaluated on the collocation grid. Construct the pseudospectral matrix $\mathscr{M}$.
2. If this is the first solve or many (by default 1024) solves have been performed since the preconditioner was updated, continue to step 4.
3. Try solving the system with the BiCGSTAB method, using the previously computed inverse as the preconditioner. If it converges to desired tolerance (by default 10^{-15}) within a few (by default 64) iterations, return the solution.
4. Perform LU decomposition to compute the inverse $\mathscr{M}^{-1}$. Store it as the preconditioner, then also use it to solve the linear system. Return the solution.

For our case BiCGSTAB is typically at least an order of magnitude faster than the LU decomposition, so it is well worth the extra code.

7.2 NLSOLVE — non-linear vector PDE solver

Above PSSOLVE in the order of things sits NLSOLVE — the component handling non-linearity in our PDEs. It is based on the iterative Newton-Kantorovich method described in the Appendix C of [14], which is a generalization of the well-known Newton's root-finding method.

NLSOLVE assumes a sequence of second-order equations

$$0 = F_i\left(u^k, \partial_a u^k, \partial_{ab} u^k\right). \tag{7.3}$$

At n-th iteration we have an approximate solution $\tilde{u}^k_n$. Defining the solution error Δ^k_n

$$u^k = \tilde{u}^k_n + \Delta^k_n \tag{7.4}$$

we expand the equation (7.3) to first order around the approximate solution

$$0 = F_i + \frac{\delta F_i}{\delta u^l}\Delta^l_n + \frac{\delta F_i}{\delta \partial_c u^l}\partial_c \Delta^l_n + \frac{\delta F_i}{\delta \partial_{cd} u^l}\partial_{cd}\Delta^l_n + \dots, \tag{7.5}$$

where F_i and their Fréchet derivatives $\delta F_i/\delta \dots$ are taken to be evaluated for the approximate solution $\tilde{u}^k_n$ and we neglect the terms quadratic in Δ^l_n and its derivatives. We can view (7.5) as a sequence of second-order linear PDEs for the corrections Δ^k_n, which can be solved with PSSOLVE. The algorithm for NLSOLVE is then as follows

1. Accept on input an initial guess $\tilde{u}^k_0$ and a callback (i.e. a function pointer in C) that evaluates F_i for provided values of u^k and its first and second derivatives. Optionally accept a callback evaluating the Fréchet derivatives of F_i; if one is not provided then use a finite difference approximation derived from F_i.

2. Evaluate F_i and its Fréchet derivatives for the current approximate solution $\tilde{u}^k_n$. Use those values to form the equation coefficients and right-hand sides for PSSOLVE. Use PSSOLVE to solve (7.5) for the spectral expansion of Δ^k_n.

3. If the absolute value of the largest spectral coefficient of Δ^k_n is below the tolerance (by default 10^{-12}), finish iteration and return the current approximate solution.

4. If the absolute value of the largest spectral coefficient of Δ^k_n is very large (by default 10^{12}), consider the iteration to have diverged and return a failure condition.

5. Compute the next approximate solution $\tilde{u}^k_{n+1} = \tilde{u}^k_n + \Delta^k_n$ and repeat from step 2.

7.3 Solving the constraints

NLSOLVE and PSSOLVE are generic elliptic solvers with no built-in knowledge of the system being solved. Such specifics are handled in the surrounding layers of code. The physics is mainly contained in a small piece of code that straightforwardly calculates

- the "seed function" K^r_θ and its first derivatives at the specified coordinates according to (2.9);

- the right-hand sides of the constraints (2.5), given the spatial coordinates and the values of $\psi - 1, K^r_r, K^\varphi_\varphi$.

We found that it is not necessary to calculate the Fréchet derivatives from (7.5) explicitly — the finite-difference approximation provides adequate results and is much simpler to compute.

We use the same basis functions as in LIBBRILLDATA for all three unknown functions $\psi - 1, K^r_r, K^\varphi_\varphi$: $B^i_0(r) = \mathrm{SB}_{2i}(r)$, $B^i_1(\theta) = \cos(2i\theta)$. Since some of the constraints are singular on the axis of symmetry $\rho = 0$, we choose the angular collocation grid to avoid the axis

$$\theta_b = \frac{(b+1)\pi}{2(N_1+1)}. \tag{7.6}$$

The radial collocation grid is again (6.6), avoiding the origin $r = 0$.

It is notable that K^r_r found by the solver is discontinuous at the origin — it attains different values for different θ. This is merely a coordinate effect and transforming the solution to Cartesian coordinates produces smooth components of K_{ij}. No special treatment is required in the solver, other than transforming to Cartesian coordinates as $K_{xx}(r = 0) = \psi^4 K^r_r(r = 0, \theta = \pi/2)$, $K_{zz}(r = 0) = \psi^4 K^r_r(r = 0, \theta = 0)$ in TEUKOLSKYDATA.

The value of $A_{\max}$ can be straightforwardly found by bisection. For $A < A_{\max}$ the solver will rapidly converge from a flat-space (i.e. all-zero spectral coefficients) initial guess, with $|\Delta^k_n|_\infty$ monotonically decreasing with n. While convergence slows down close to $A_{\max}$, even $A_{\max} - A \approx 10^{-8}$ converges in 14 Newton-Kantorovich iterations. For $A > A_{\max}$, $|\Delta^k_n|_\infty$ will stop decreasing after a few iterations and will either oscillate or diverge. So we judge A to be above $A_{\max}$ if the solver fails to converge in 32 iterations.

7.4 Getting to the "upper" branch

To construct initial data from the "upper" solution branch for some $A = A_t$, we pick $A_0 \approx \mathrm{sgn}(A_t)(A_{\max} - 10^{-8})$ and $A_1 = A_0(1 - 10^{-3})$ and compute the corresponding lower-branch solutions $u^k_{A_0}, u^k_{A_1}$. From those we construct $\tilde{u}^k_{A_1} = 2u^k_{A_0} - u^k_{A_1}$ and use it as an initial guess for a solve with $A = A_1$. The solver will then converge to the upper-branch solution $\overline{u}^k_{A_1}$.

Convergence on the upper branch is more precarious — simply using $\overline{u}^k_{A_1}$ as the initial guess for distant values of A will likely diverge. To solve for an arbitrary A_t we gradually move from A_1 to A_t using the following iterative algorithm:

1. Start with $n = 1$ and the step size $\epsilon = (A_t - A_1)/128$.
2. At n-th iteration, set $A_{n+1} = \max(A_n + \epsilon, A_t)$. Try solving for $A = A_{n+1}$ using $\overline{u}^k_{A_n}$ as the initial guess.
3. If the solver fails to converge in a few (by default 32) iterations, reduce ϵ by half and retry step 2 (without increasing n).
4. If we reached $A = A_t$, return the solution.
5. Multiply ϵ by 1.75, increase n by one, and repeat from step 2.

7.5 Accuracy & convergence

The solver converges far slower than LIBBRILLDATA. This can be attributed to the fact that the system is more complicated: not only are there multiple equations, but they are also non-linear and degenerate at certain locations. In addition one component of the solution is discontinuous. The solver thus requires many more coefficients per variable for the residual to reach the roundoff level (which is itself higher due to more operations being performed). Furthermore, the fact that there are three unknown variables triples the number of coefficients and increases the pseudospectral matrix size by almost an order of magnitude.

In light of these factors, the computational limitations discussed at the end of the previous Chapter become more practically relevant. As can be seen from Figures 7.1 and 7.2, roughly $N_0 N_1 P \approx 128 \times 16 \times 3 = 6144$ spectral coefficients are needed to accurately resolve a near-critical $A = \overline{1.3}$ solution, which is quite close to the "practical" upper bound of 10^4 coefficients estimated previously. So while we are able to construct accurate near-critical initial data for the seed function (2.9), handling more complicated data would require more sophistication — a better collocation grid or a basis choice, or perhaps even a multi-domain method. That is the main reason we do not attempt to reproduce the "ingoing-pulse" data of [1].

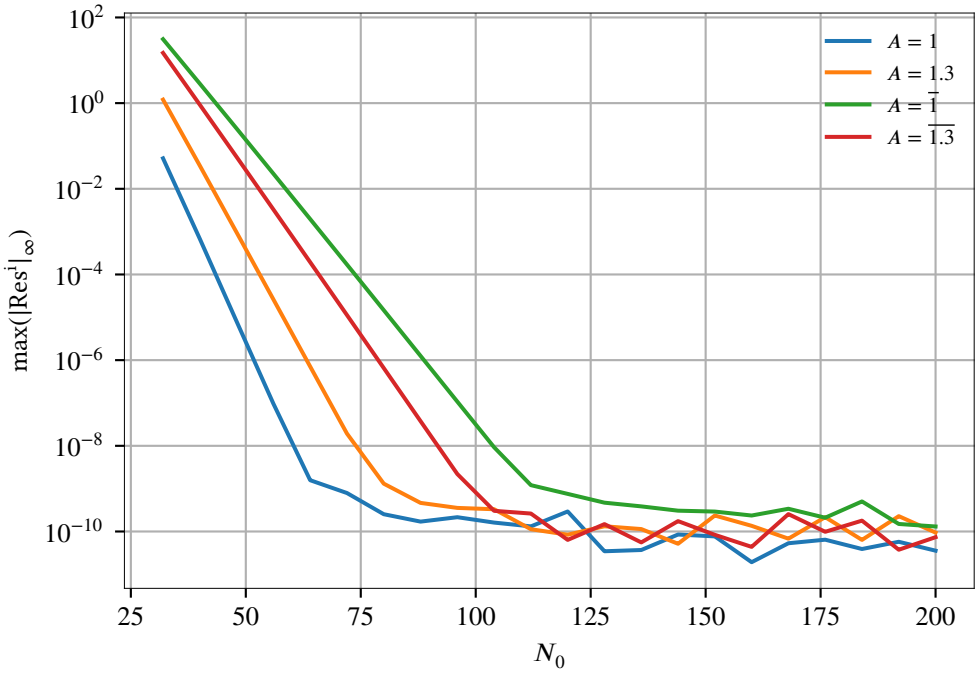

FIGURE 7.1: Dependence of the solver accuracy on the radial solver order N_0 for several different values of A. Angular solver order is $N_1 = 16$. Res^i is the residual of the i-th constraint equation (2.5)

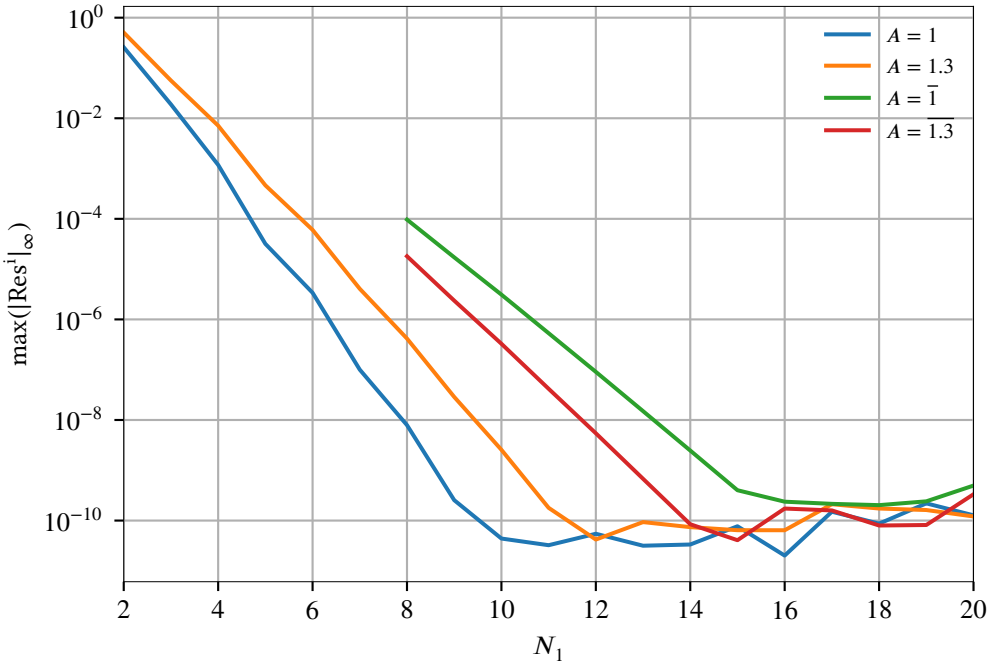

FIGURE 7.2: Dependence of the solver accuracy on the angular solver order N_1 for several different values of A. Radial solver order is $N_0 = 180$. The solver fails to find the upper branch for very low expansion orders, so the plots for the upper-branch initial data start at $N_1 = 8$.

Chapter 8

LIBMG2D — 2D multigrid elliptic solver

This chapter describes the LIBMG2D code, which forms the foundation for our slicing methods — maximal and QMS. LIBMG2D is a standalone library for solving linear scalar elliptic 2D PDEs using the multigrid method. For an introduction to multigrid methods, see section 20.6 in [44] and references therein. This text is mainly concerned with those aspects of our code that deviate from the textbook treatments, specifically our use of higher-order finite differences and "unaligned" grids.

LIBMG2D is written mostly in C, with some performance-critical parts having an optional vectorized x86 assembly implementation. Computations can be further parallelized using shared-memory multithreading (PTHREADS or OpenMP) and MPI. See Appendix A.2 for obtaining the source code of LIBMG2D.

8.1 Discretization

The solver assumes a PDE of the form

$$L[u] \equiv E^{ab}\partial_{ab}u + F^{a}\partial_{a}u + Gu = S, \tag{8.1}$$

with $a, b \in \{0, 1\}$ indexing the spatial directions, $\{E^{ab}, F^{a}, G\}$ the spatially varying equation coefficients defining the linear operator L, S the right-hand side, and $u = u(x_0, x_1)$ the unknown function being solved for. The problem is discretized on a square equidistant $N \times N$ grid as

$$L_N[u_N] = S_N. \tag{8.2}$$

Here L_N is an approximation of L where partial derivatives are replaced with centered finite differences FD^o of configurable even order o. The solver accepts as input the values of the coefficients, the right-hand side, and an initial guess for u sampled on this grid and produces the solution on the same grid. $u_N = u_N^{ij}$ and $S_N = S_N^{ij}$ are 2D arrays holding the grid values of u and S.

As the method involves interpolating between multiple grids at different resolutions, u and S are assumed to be continuous and sufficiently smooth for the finite difference operators used. Since the equations we solve with LIBMG2D involve the Laplacian in cylindrical coordinates, the continuity condition would be too strict for the equation coefficients making up L. We thus relax it to allow each coefficient

to have either a discontinuous jump or a first-order pole on any of the four grid boundaries. Outside of the boundaries the equation coefficients are assumed to be smooth.

Unlike many textbook descriptions of the multigrid method, we do not require $N - 1$ to be a power of 2, since we want the LIBMG2D discretization to match the CACTUS one and CACTUS grid sizes are computed according to nontrivial rules. For this reason, we allow N to be any integer larger than 3. The price for this flexibility is the more complex form of the inter-grid transfer operators, as described later.

8.2 Relaxation scheme

At the core of the multigrid solver is the following iterative relaxation scheme. For some approximate solution $\tilde{u}_N$ define its *residual*

$$\mathrm{Res}[\tilde{u}_N] = L_N[\tilde{u}_N] - S_N. \tag{8.3}$$

The residual vanishes for the exact solution u_N of the discretized problem (8.2), so we can view the solution as the endstate of an evolution process in a virtual time t

$$\partial_t \tilde{u}_N(t) = \mathrm{Res}[\tilde{u}_N(t)]. \tag{8.4}$$

Approximating the time derivative with the first-order forward finite difference operator, we get the iterative scheme

$$\tilde{u}_N^{i+1} = \tilde{u}_N^i + \Delta t \mathrm{Res}[\tilde{u}_N^i], \tag{8.5}$$

which is started with some initial guess $\tilde{u}_N^0$ and then iterated until convergence.

Two points are of note here compared to standard treatments of this method. The first is that (8.5) is written in terms of the residual, rather than explicitly expanded finite differences. We found that structuring the code around the residual makes it significantly cleaner and easier to reason about. It also allows us to "decouple" the iterative structure from the discretization of the derivatives, so higher order finite differences can be easily plugged in. Finally, with this approach we concentrate most of the solver runtime into the residual computation function, so it becomes a convenient target for performance optimization.

The second point is that (8.5) corresponds to the Jacobi method, rather than the typically-recommended Gauss-Seidel method. While the latter converges faster, it cannot be expressed as elegantly in terms of the residual, and is also significantly harder to parallelize due to "non-local" data dependencies. We concluded that the difference in convergence rate is not significant enough to be worth the extra code complexity.

The scheme (8.5) is "pointwise" — every value in the result array can be computed independently of others, so it is trivially parallelisable. To amortize multithreading overhead and make efficient use of memory caching, our residual computation function computes entire lines of output, with different lines potentially computed concurrently in different threads of execution. We also make use of the vector processing capabilities of modern x86 CPUs — on systems that support the AVX2 instructions we leverage them to compute 4 adjacent residual values simultaneously.

To estimate the upper bound on the "time step" Δt we can use the von Neumann stability analysis (see e.g. Chapter 20 in [44]), which leads to a condition of the form

$$\Delta t \leq \frac{\Delta x^2}{K_{ab} E^{ab} + \Delta x K_a F^a + \Delta x^2 K G} \tag{8.6}$$

with $K_{...}$ some constants depending on the chosen FDO order. Since the equation coefficients are variable, Δt should be chosen to satisfy (8.6) at all grid points. In practice we use a simplified form

$$\Delta t = \frac{F_r \Delta x^2}{\max(|E^{00}|, |E^{11}|) + \Delta x^2 \max |G|/8} \tag{8.7}$$

with a solver parameter F_r, which has proven to work well enough for our purposes.

Boundary conditions are again implemented using ghost zones — the numeric arrays storing the grid functions are allocated with a thin layer of extra points at each boundary. These points are updated appropriately so that any algorithm that requires out-of-grid data points (which include computing the residual, prolongation, and restriction, as described later) will produce correct results satisfying the boundary conditions. The conditions implemented are

- FIXVAL: The function assumes a prescribed value on the boundary. The ghost zone is simply filled with the provided values.
- REFLECT: The function has a reflection symmetry across the boundary. Each ghost layer is filled by copying the corresponding layer from inside the grid.
- FALLOFF: The function falls off as K/r across the boundary, with a constant K and $r = (x_0^2 + x_1^2)^{1/2}$. The ghost zone is filled by extrapolating from the values of u_N on the boundary.

8.3 Multigrid

It is well known that relaxation schemes like (8.5) converge far too slowly to be useful on their own. A multigrid method employs relaxation on the eponymous multiple grids to vastly improve convergence. Consider again an approximate solution $\tilde{u}_N$. Define the *solution error* Δ_N as

$$u_N = \tilde{u}_N - \Delta_N, \tag{8.8}$$

then substituting (8.8) into (8.2) yields

$$L_N[\Delta_N] = \mathrm{Res}[\tilde{u}_N]. \tag{8.9}$$

In other words, to compute the solution error Δ_N we need to solve the original PDE with the residual on the right-hand side. An iteration of the multigrid method consists of solving (8.9) approximately on a coarser grid, interleaved with several applications of the relaxation procedure (8.5) Since each of those processes primarily affects different Fourier components of $\tilde{u}_N$, this significantly speeds up convergence over pure relaxation.

More specifically, we cover the problem domain with D levels of $N_d \times N_d$ grids of progressively lower resolution, $d \in \{0, \dots, D-1\}$. The finest grid with $N = N_0$ is the one where we want to obtain the final solution, while the coarsest grid $N = N_{D-1}$ should be small enough that the discretized linear problem (8.2) can be solved exactly (though this condition is not enforced). Then the recursive algorithm for d-th level is as follows:

1. Accept as input the PDE coefficients $E^{ab}_{N_d}, F^{a}_{N_d}, G_{N_d}$, the right-hand side S_{N_d}, and the initial guess $\tilde{u}_{N_d}$ sampled on the d-th grid. Initialize the cycle count $c = 0$.

2. If $N_d \leq N_{\text{exact}}$, where N_{exact} is a solver parameter, explicitly construct the linear system corresponding to (8.2) and solve it using a combination of LU decomposition and the BiCGSTAB method, as described in section 7.1. Return the solution u_{N_d}.

3. Perform K_{pre} steps of the relaxation procedure (8.5).

4. If $d < D - 1$, *restrict* the residual and the equation coefficients onto the coarser $d+1$-th grid as per section 8.5. Recursively invoke this algorithm on the $d+1$-th grid with the restricted residual as the right-hand side to obtain an approximate correction $\tilde{\Delta}_{N_{d+1}}$. *Prolongate* the correction to the d-th grid as per section 8.4 and apply it to the current approximate solution according to (8.8).

5. Perform K_{post} steps of the relaxation procedure (8.5).

6. Increase the cycle count c by 1. If $c < C$, where C is a solver parameter, repeat from step 2.

7. Return the approximate solution $\tilde{u}_{N_d}$.

The solver repeatedly invokes this algorithm with $d = 0$ until the residual norm $|\text{Res}[u_{N_0}]|_\infty$ falls below the specified tolerance.

As mentioned above, the finest grid size N_0 is typically determined by external factors and may in principle be arbitrary. The coarser levels $d > 0$ are all chosen by the solver and their sizes are $N_d = 2^{k_1 - d + 1} + 1$. k_1 is taken to be the largest integer such that $N_0 \geq 1.5 N_1$ (the factor of 1.5 is picked to avoid having two grids with too similar resolutions, which would be inefficient). The number of grids D is chosen such that the coarsest grid has $N_{D-1} = N_{\text{exact}}$ (by default 5), up to a configurable maximum $D_{\max}$ that defaults to 16.

8.4 Prolongation

To prolongate the correction from a coarser grid onto a finer one we use standard Lagrange polynomial interpolation. With finite differences of order o we use a polynomial of order $O = o + 1$ in each direction, i.e. defined by $o + 2$ source points. Since o is always even, the interpolation process also uses an even number of points. For each destination (fine-grid) point, the source (coarse-grid) points are chosen such that there is an equal number of them on each side of the destination point (unless the destination point happens to lie directly on some point of the source grid, in which case interpolation for that point reduces into copying the corresponding source value).

A single multigrid solve will typically require prolongating between the same two grids many times. Furthermore we typically use the same solver instance many times with different equation coefficients. It thus makes sense to optimize the algorithm for many invocations of the prolongation process between the same two grids, with different function values.

Consider a coarse M-point grid with grid spacing H, a fine N-point grid with grid spacing h, and a coarse-grid function u_M. Then the prolongated result $\mathscr{P}_N[u_M]^{ij}$ at the fine-grid point (i, j) can be written as

$$u_N^{ij} = \mathscr{P}_N[u_M]^{ij} = \sum_{K=0}^{O} A_K^i \sum_{L=0}^{O} A_L^j u_M^{(P_i+K)(P_j+L)}. \tag{8.10}$$

The indices P_i that map i-th fine-grid point to its corresponding source point are computed as

$$P_i = \text{floor}\left(\frac{h}{H} i\right) - \frac{O-1}{2}. \tag{8.11}$$

The corresponding interpolation coefficients are

$$A_K^i = \prod_{\substack{L=0 \\ L \neq K}}^{O} \frac{x_i - X_{iL}}{X_{iK} - X_{iL}} \tag{8.12}$$

where $x_i = ih$ and $X_{iK} = (P_i + K)H$. The prolongation process for general ("unaligned") grids is illustrated in Fig. 8.1.

The indices and coefficients are computed once when the solver is initialized, prolongation is then performed by executing (8.10) for each point in the destination grid. Prolongation is again trivially parallel, since each point can be computed independently from others. As for the residual, we compute entire lines in parallel to reduce threading overhead. Unlike the residual computation, interpolation code cannot be as easily vectorized since adjacent output points do not necessarily use adjacent input points. For that reason our vectorized AVX2 function computes only one output value per iteration, which may not be optimal.

8.5 Restriction

Restriction is used to transfer the equation coefficients and the residual from a finer grid onto a coarser one. The obvious choice for restriction would again be employing Lagrange interpolation (for "aligned" grids, where every coarse-grid point also lies on the fine grid, this would be a plain copy of the corresponding values — usually called "injection"), but this choice turns out to be suboptimal. The outline of the issues can be glimpsed in the case of injection from the fact that it completely disregards the fine-grid points that do not also exist on the coarse grid.

A better restriction operator is called "full weighting" and is derived as "adjoint" to a given prolongation operator in the following way. On an N-point grid define a scalar product of two grid functions u_N^{ij}, v_N^{ij} as

$$(u_N, v_N)_N = h^2 \sum_{i,j=0}^{N-1} u_N^{ij} v_N^{ij}. \tag{8.13}$$

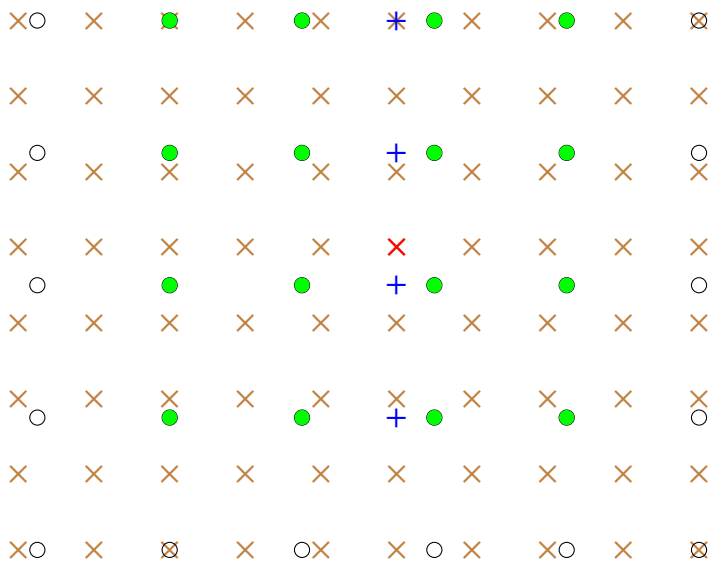

FIGURE 8.1: Prolongation between two grids with $H/h = 1.75$ using 3rd order Lagrange interpolation. ○ are the coarse (source) grid points; × are the fine (destination) grid points. ● are the coarse-grid points used to compute the fine-grid value at ×. 2D interpolation is equivalent to a sequence of 1D interpolations: first interpolate within each horizontal line to obtain 4 intermediate values +, then interpolate from those intermediate values to obtain the final result.

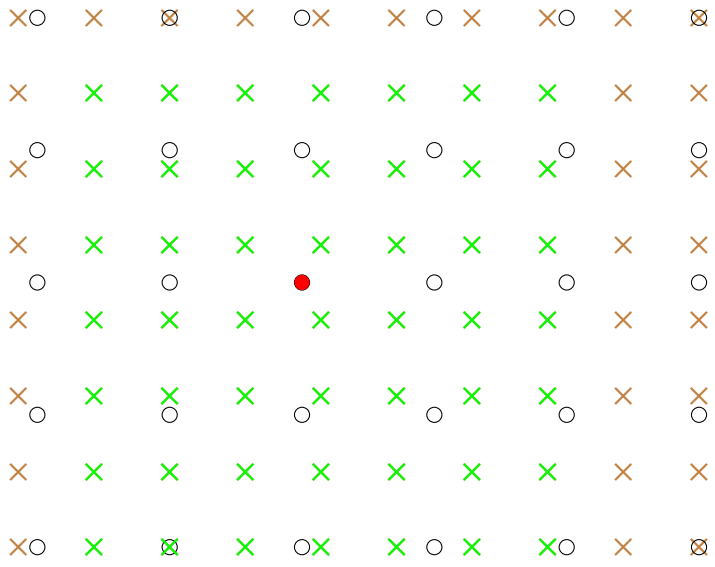

FIGURE 8.2: Restriction adjoint to the prolongation of Fig. 8.1. × are the fine (source) grid points used to compute the coarse-grid value at ●

Then as in the previous section, consider an N-point fine grid, an M-point coarse grid, a fine-grid function u_N, a coarse-grid function v_M, and a prolongation operator $\mathscr{P}_N[v_M]$. Then the full-weighted restriction operator $\mathscr{R}_M[u_N]$ is defined by requiring it to preserve the above scalar product

$$(u_N, \mathscr{P}_N[v_M])_N = (\mathscr{R}_M[u_N], v_M)_M. \tag{8.14}$$

To obtain the explicit form of this operator, consider a trial function V_M^{IJ} that has the value of one at the coarse-grid point (I, J) and zero everywhere else. Then for restricting an arbitrary fine-grid function u_N we obtain

$$u_M^{IJ} = \mathscr{R}_M[u_N]^{IJ} = \frac{h^2}{H^2} \sum_{i,j=0}^{N-1} u_N^{ij} \mathscr{P}_N[V_M^{IJ}]^{ij}. \tag{8.15}$$

Taking 2D Lagrangian interpolation of order O as $\mathscr{P}_N$, prolongating V_M^{IJ} may produce non-zero values only at an $R \times R$ sub-grid with $R = \text{ceil}\left[\frac{H}{h}(O+1)\right]$. Then the restriction operator can be written similarly to (8.10) as

$$\mathscr{R}_M[u_N]^{IJ} = \frac{h^2}{H^2} \sum_{k=0}^{R} A_{Q_I^k}^{q_I+k} \sum_{l=0}^{R} A_{Q_J^l}^{q_J+l} u_N^{(q_I+k)(q_J+l)} \tag{8.16}$$

with the coefficients as per (8.12) and indices

$$q_I = \text{floor}(\frac{H}{h} I) - \text{floor}(\frac{R}{2} - 1), \tag{8.17a}$$

$$Q_I^k = I - P_{q_I+k}. \tag{8.17b}$$

As for prolongation, we compute the indices and the coefficients once during initialization and then reuse them repeatedly.

8.5.1 Restricting non-smooth functions

Since the equation coefficients are provided to the solver only on the finest grid, we need to restrict them to the coarser grids. This process is somewhat complicated by the fact that the coefficients are not always smooth on the boundary. Our solver allows two kinds of non-smoothness: a discontinuity or a first-order pole.

In the case of a discontinuity the function in question is written as

$$w = w_0 + d, \tag{8.18}$$

where w_0 is smooth, and d is smooth along the boundary and zero outside of it. The solver accepts the non-zero values of d as a 1D array sampled on the finest grid, in addition to the 2D values of w. Restriction then consists of restricting w_0 normally, then applying 1D restriction on d and adding the two results.

In the case of a pole, we multiply the coefficient in question by the corresponding coordinate x_i before restriction, then divide the result by it (except on the boundary itself, where the value is set to zero; see discussion in Chapter 9.1).

8.6 MPI

Beside shared-memory multithreading, MG2D also supports multi-process parallelization through MPI. In MPI mode each process instantiates its local solver that takes ownership of some sub-rectangle of each grid. That sub-rectangle may be empty for some levels — then the process in question does not participate in the solve on that level. The finest grid is partitioned externally by the code invoking MG2D (in our use cases the partitioning is chosen by CACTUS). The coarser grids are partitioned by the solver — for sufficiently large grids ($N > 2^7 + 1$) we reuse the finer grid's partitioning rounded to the nearest grid point. For small grids ($N \leq 2^7 + 1$) we judge the communication overhead to be too expensive and aggregate the entire domain in one process.

This partitioning is rather naive and likely sub-optimal — we did not spend much effort on investigating efficient grid partitioning techniques, since the vast majority of our simulations run in just one or two processes. Despite that, even in the case of a dual-socket system with processor-local memory we found it advantageous to divide the solve into two processes, each bound to one processor.

Synchronization between the individual processes needs to be performed during relaxation, prolongation, and restriction. We extend the notion of ghost zones, placing extra ghost point layers on inter-component boundaries. These ghost zones are then synchronized after each relaxation step. When restricting, we first perform restriction in each process into a temporary buffer, then distribute the results with the MPI all-to-all operation. The reasoning behind this approach is that coarser grids have fewer points, so less data needs to be synchronized. For the same reason, prolongation first gathers the necessary source points at each component, then computes the interpolated values from local data.

Chapter 9

Slicing solvers

MaximalSlicingAxiMG and QuasiMaximalSlicingMG [1] are the Cactus thorns we wrote to implement maximal and quasi-maximal slicing described in Chapter 3. The two slicing methods involve solving elliptic equations (3.1) and (3.6), respectively, which differ only in the right-hand side. The two codes are thus very similar and we describe them both in tandem. Certain features are specific to QuasiMaximalSlicingMG, those are described in section 9.3. We refer to the unknown function as u — it stands for $\alpha - 1$ in the maximal slicing case and directly for the source term W in the QMS case.

In contrast to the initial data thorns BrillData and TeukolskyData, the code in question is not merely trivial "glue" between the libmg2d solver library and Cactus. The main reason behind these thorns' complexity (over 1500 lines of C code) is that elliptic conditions cannot be incorporated into Berger-Oliger-style mesh refinement schemes in a straightforward manner.

This follows from the fact that mesh refinement assumes "locality" — one typically does a coarse-grid evolution step and then corrects it on a part of the domain with the result of multiple fine-grid steps. Clearly this only works if the errors from the under-resolved parts of the domain do not propagate too far and can be overwritten by the correction. Elliptic equations, however, are global: errors from a badly resolved area will spread over the entire domain. Naively performing elliptic solves during each refinement level's evolution stepping (using only information from that level, with boundary conditions provided by time interpolation from the next coarser level) would lead to non-convergent artifacts on the refinement boundaries. Therefore a more sophisticated approach is needed.

Our code follows the method suggested in [46], where the basic idea is to delay solving the elliptic equation on each level until all information from the enclosed finer levels is available. During time evolution of the hyperbolic equations the variable u is approximated with linear extrapolation in time from past elliptic solves, plus an extra correction term.

[1] The MG suffix refers to the fact that the solvers are using a multigrid method, and was added to distinguish them from our previous pseudo-spectral codes that proved to be non-viable. Cf. also section A.3

9.1 Elliptic solve

The elliptic solve code is executed on every refinement level l after recursive evolution on finer levels has finished and the "final" values of the evolved quantities were restricted onto the current level. All the levels that exist at the given time t are processed in the order of increasing l, i.e. from the coarsest level l_{c_t} to the finest l_f. This corresponds to the schedule group CCTK_POSTSTEP in Cactus (which also runs at the beginning $t = 0$).

A separate instance of the libmg2d solver is maintained for each level. The elliptic solve domain on the coarsest level $l = 0$ covers the entire evolved interior, exluding the ghost points. On refined levels $l > 0$ the domain excludes most of the buffer points, so that the outer boundary coincides with the first coarser-level point outside of the evolved interior. Beside the solver, the thorns maintain two past solutions u_0 and u_1 at times t_0 and t_1. At the beginning they are initialized to $u_0 = u_1 = 0$; the more complex suggestion from [46] involving alternating forwards and backwards stepping does not seem to be necessary for our use cases.

The PDE coefficients and the right-hand sides are computed from the BSSN quantities on the given level and provided to the solver. Applying the Cartoon method to the Laplacian in the elliptic equations results in the term $(\gamma_{yy}/x)\partial_x u$, which is singular on the symmetry axis $x = 0$. This singularity is easily treated by applying the l'Hôpital's rule, which transforms the term into $\gamma_{yy}\partial_{xx}u$. For this reason, the ∂_{xx} PDE coefficient is marked as having a discontinuous jump of γ_{yy} on the axis, while the ∂_x coefficient is marked as having a pole there (note that the pole designation applies only to the restriction process, the actual value directly on the axis is set to zero).

The outer boundary conditions and the initial guess are set depending on the level. On the coarsest level we use C/r lapse decay for maximal slicing and fixed value $W = 0$ for QMS (though typically we do not solve QMS on the coarsest level at all, as described in more detail below). For $l > 0, l = l_{c_t}$ (i.e. the coarsest level that exists at the time t), the boundary conditions and the initial guess are extrapolated in time from u_0 and u_1. For finer levels at the same time t, we prolongate the already obtained solution from the next coarser level and use it as the initial guess and the fixed-value boundary condition.

With the setup done, we invoke the libmg2d solver. The residual tolerance is set through the parameter file either directly or as a "base" value that is then scaled to each refinement level by dividing it with the square of the grid spacing (which corresponds to the scaling of the roundoff error).

The resulting solution u is used depending on the level l. When $l > 0, l = l_{c_t}$, the result is *only* used to set the boundary conditions for the next finer level. Otherwise (i.e. when this is the coarsest level, or this level is in sync with some coarser level), the solution history u_0, u_1 is updated as follows:

$$u_0 \leftarrow u_1 + \Delta u - \frac{1}{2^{l-l_c}}\Delta u, \tag{9.1a}$$

$$t_0 \leftarrow t_1, \tag{9.1b}$$

$$u_1 \leftarrow u, \tag{9.1c}$$

$$t_1 \leftarrow t, \tag{9.1d}$$

where $\Delta = u - \tilde{u}$ and $\tilde{u}$ is the extrapolated value at the time t that was used for evolution. At the refined levels, the history used for extrapolation is thus updated

once per every two evolution steps.

9.2 Elliptic variable evaluation

As u is used in the right-hand sides of the evolved hyperbolic equations, we need to update it before each right-hand side computation. That is achieved by simple linear extrapolation from the solution history u_0, u_1. On the finest level we can also optionally (depending on a thorn parameter) solve the elliptic equation at each evolution step. That requires maintaining a separate solver intance for each time integrator sub-step in the two steps between the successive coarse-level prolongations — each one "peeling off" an outer buffer layer. The boundary conditions in this case are set by the same extrapolation in time as above.

Since solving the elliptic equation on each integrator sub-step is very resource-intensive, and typically the leading sources of error are elsewhere, we usually do not treat the finest level specially.

9.3 QMS-specific features

The idea behind QMS is reducing the computational cost of maximal slicing, at the price of our slices no longer being exactly maximal. QuasiMaximalSlicingMG has several extra features beyond MaximalSlicingAxiMG to achieve that goal.

The main one is *temporal subsampling* — skipping the elliptic solve on some time levels. The solve_level option supplied through the parameter file restricts solving the elliptic equation only to those times at which the given level l_s exists. The denominator in the "velocity factor" in equation (9.1a) then needs to be changed to $2^{\min(l_s,l)}$.

There is also the option solve_level_max, which specifies the coarsest level l_m on which the elliptic equation is solved. Towards the boundaries of this level we send W_a smoothly to zero by multiplying it with the function $\min(1, \exp(-36R^6))$ (inspired by [32]), where $R = (r - r_0)/|L_m - r_0|$, $r = (x^2 + z^2)^{1/2}$, L_m the linear size (in σ) of the l_m-th refinement level. On the coarser levels we simply set W_a to zero.

Finally, a simple way to improve performance at the cost of accuracy is just raising the residual tolerance requested from the solver.

Chapter 10

Horizon location

During the course of our work we implemented two different methods for locating AHs in axisymmetry, each with its strengths and weaknesses. Both are implemented as standalone Python code that is invoked as "postprocessing" — either interactively or as a script — on data files produced as simulation output.

The finders share the AHCalc class, which encapsulates the physics. Its instances are constructed from given Cartesian components of the metric functions γ_{ij}, $\boldsymbol{K}_{ij}$, sampled on an equidistant grid in the $x - z$ plane. An AHCalc instance can then be used to evalute the right-hand sides of (4.14) and (4.19), handling interpolation and transforming from Cartesian to spherical coordinates as necessary.

For interpolating the metric functions we use either splines (in earlier versions of the code; implemented as RectBivariateSpline in the scipy.interpolate package [34]) or Lagrange interpolation (implemented in our own C code for performance reasons; using 6-point polynomials by default). Besides performance, we did not see significant differences in behavior between these two options.

10.1 Pseudospectral AH finder

Our first AH finder uses a pseudospectral method to solve the equation (4.14), i.e. it uses spherical parametrization. The obvious basis choice for representing origin-centered horizons in our spacetimes (i.e. axial and equator-reflection symmetries) are even cosines $\cos(2i\theta)$, which we also use in the initial data solvers. As was reported in [33] and confirmed in our work, supercritical spacetimes close to the critical point may contain "off-center" AHs that are centered on $\{x = 0, z = z_0 \neq 0\}$. To search for such horizons we expand the basis to include odd cosines $\cos(i\theta)$, in addition to instantiating AHCalc with an offset z coordinate. In both cases the bases automatically satisfy the boundary conditions, so we do not need to impose them explicitly.

To handle non-linearity we use the Newton-Kantorovich method described in section 7.2. A substantial difference here is that we have just one 1D equation and performance is much less of a pressing concern. We can thus implement the method in ~ 100 lines of Python code. It requires an initial guess — we typically try a sequence of circles of different radii, or the previously found horizon when searching for AHs on multiple time slices. The origin for off-center horizons is typically taken to be the location of the maximum value of the Kretschmann scalar on the axis of symmetry.

The main advantages of this approach are conceptual simplicity, ease of implementation, and computational efficiency. It also allows us to avoid the extra work of

regularizing equation (4.14) on the axis by not placing the collocation points there (though we eventually did that anyway, in order to experiment with the shooting method). Its primary flaw, other than the ray-body restriction, is lack of control over the solution: the Newton-Kantorovich iteration either converges to a solution, converges to origin (all-zero spectral coefficients), or diverges (a trial surface no longer lies entirely inside the grid). In the latter two cases we get no information on whether the solution exists at all or how the initial guess should be modified. When multiple solutions are present, the method does not allow us to choose which one is found.

These issues could be resolved by using a shooting method, which is the standard literature treatment of axisymmetric horizons. It involves repeatedly integrating the ODE (4.14) from $\theta = 0$ to $\theta = \theta_{\max}$ (π or $\pi/2$) with varying $h(0)$ to find the value that satisfies the boundary condition $h'(\theta_{\max}) = 0$. However we found that the shooting method in this form is unworkable for off-center horizons. The reason is that any trial surface that is not exactly the solution will stop conforming to parametrization (4.11) before reaching $\theta = \pi$ and integration will fail. Similar issues were reported in [32] who suggest shooting from both $\theta = 0$ and $\theta = \pi$ and tuning the mismatch at $\theta = \pi/2$. Our alternative approach, which also addresses the ray-body restriction, is using the Cartesian parametrization.

10.2 Cartesian shooting AH finder

The Cartesian AH finder uses a version of the shooting method with the ODEs (4.19). An important property of these equations is that the axis of symmetry is *repulsive* for any $X(\lambda)$, $Z(\lambda)$ curves that do not approach it at the right angle (i.e. which do not satisfy the boundary condition). More specifically, the right-hand side for $\mathrm{d}u^x/\mathrm{d}\lambda$ diverges as the curve approaches the axis, unless u^z happens to vanish at the same time. This is also the reason for the abovementioned failures of the shooting method with the spherical parametrization (left panel of Figure 10.1).

A MOTS (or more generally a constant-expansion surface) is thus the curve which, when integrated from some z-axis point $z = z_0$ with $s^z > 0$, hits the axis again at some $z_1 < z_0$. Curves integrated from other points above or below z_0 will then loop away from the axis in different directions. This observation forms the basis of our search strategy.

We construct a "weight" function $\Omega(z_0)$ that integrates the $X(\lambda)$, $Z(\lambda)$ curve from $X(0) = 0$, $Z(0) = z_0$ until

- The axis is hit (to within numerical tolerance, which defaults to 10^{-8}), then $\Omega(z_0) = 0$.
- The integrated curve is "repelled" by the z-axis, meaning that we reach a point $\lambda = \lambda_r$ where
  - the sign of $u^x(\lambda_r)$ changes from negative to positive;
  - either $u^z(\lambda_r) > 0$ or $X(\lambda_r)$ is "close enough" to the axis of symmetry (which defaults to 5% of the maximum value of X seen so far).

  In this case we evaluate $\Omega(z_0) = X^2(\lambda_r)\mathrm{sign}\left[(Z(\lambda_r) - Z(0))u^z(\lambda_r)\right]$.
- The integrated curve hits the boundary of the numerical slice at $\lambda = \lambda_b$, then we evaluate $\Omega(z_0) = -X^2(\lambda_b)$.

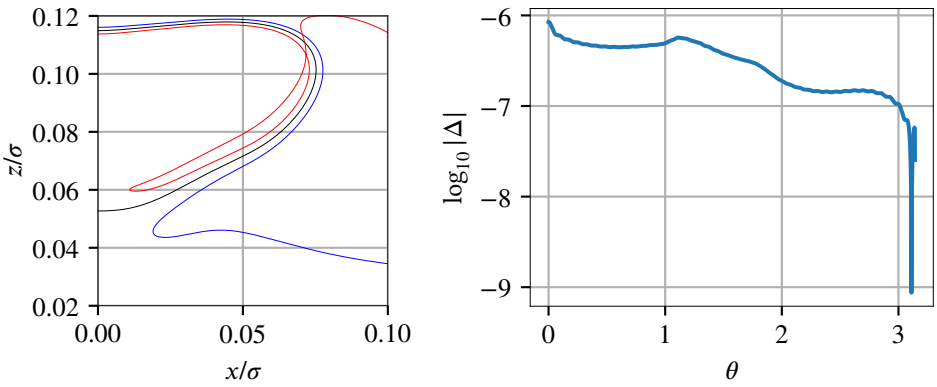

FIGURE 10.1: Locating an off-center MOTS for $A = \overline{1.300805}$ TA waves at $t = 20.25$.
Left: the MOTS itself (black), along with curves generated by integrating (4.19) from points on the z-axis slightly above (blue) and below (red) the MOTS.
Right: Difference between the MOTS found by the pseudospectral finder of section 10.1 and the shooting finder of section 10.2.

$\Omega(z_0)$ should be a continuous function of the starting point z_0 and MOTS are found by integrating (4.19) from the roots of Ω. We locate those roots by using the Brent's method, implemented as BRENTQ in the SCIPY.OPTIMIZE package. Initial bracketing is done by integrating a sequence of trial curves between the largest z_0 where the lapse $\alpha < 0.3$ (a common heuristic used to estimate horizon location) and $z_0 = 0$. If no positive values of Ω are found during this process, then we assume no horizon is present on the slice.

Once the root is found to sufficient precision, we also pass the resulting curve as an initial guess to the pseudospectral finder of the previous section. The results produced by both solvers are very close even for very "wild" data, as shown in the right panel of Figure 10.1.

Part III

Simulations

Chapter 11

Code validation

In this Chapter we validate our code and simulation setup using self-consistency and convergence tests and comparing to existing literature.

11.1 Initial data

Convergence plots in Chapters 6 and 7 are based on residuals computed internally by the solvers. In this section we also look at the constraints computed by McLachlan from initial data on a Cactus grid. We use a setup similar to that used for near-critical evolutions, with 11 levels of refinement. The constraints (1.14) are computed on each level independently, using the BSSN quantities and 8th order finite differences.

As seen in Fig. 11.1, the constraint violations converge away as refinement levels are added. At very high resolutions the roundoff error starts to dominate, growing quadratically with resolution. This shows that the equations solved by our initial data solvers are consistent with the constraint evaluation code in McLachlan.

Since Brill waves have been used in previously published studies, we can also validate our solvers by comparing with existing literature. Table 1 in [33] lists ADM masses for near-critical initial slices. Assuming the conformal factor ψ behaves as $1 + M_{\mathrm{ADM}}/2r$ towards infinity, we can easily approximate the ADM mass by evaluating ψ for a large value of r. The values we obtain this way are fully consistent with the cited table; specifically for the near-critical $A = 4.697$ we get $M_{\mathrm{ADM}} = 0.62122$.

The final test we do with initial data is searching for an apparent horizon on the initial slice. The weakest data where our AH finders can locate a horizon are $A = 11.8163$ and $A = -5.2928$, which is consistent with $A = 11.82$ and $A = -5.3$ given in [33]. To validate horizon mass calculation we use the $A = 12$ Brill data with $M_{\mathrm{ADM}} \approx 4.669$. We find two nested MOTS'es in this data, with the outer one having the mass $M_{\mathrm{AH}} \approx 4.658$, consistent with [3, 32]. For TA waves we find a horizon for $A = \overline{0.3055}$.

The above tests give us confidence that our Brill wave initial data are, up to tiny elliptic solver error, identical to those used by Hilditch et al.

11.2 1+log slicing failure

In [31] Hilditch et al. tried to use the moving puncture coordinates to study Brill wave collapse. They discovered that for near-critical values of A the simulations will

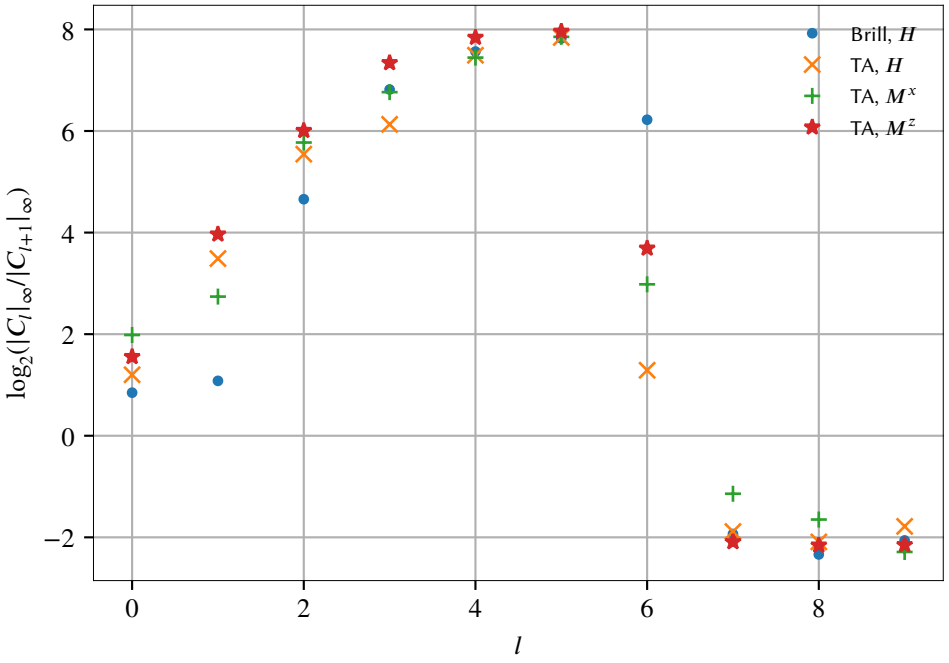

FIGURE 11.1: Constraints convergence on the initial slice for $A = 5$ Brill waves and $A = \overline{1.3}$ TA waves. Plotted is the base-2 logarithm of the ratio of the maximum constraint violation between two successive refinement levels l and $l + 1$. Only the Hamiltonian constraint is considered for Brill waves, as the momentum constraints are identically zero; the M^y momentum constraint is omitted for TA waves for the same reason.

fail, apparently because of a coordinate singularity. In this section we validate our evolution setup by reproducing this failure.

As in the cited study, we evolve $A = 5$ Brill data with the 1+log lapse condition. We use 6 levels of refinement, 8th order finite differences and 9th order dissipation. Around simulation time $t = 4.5$, a sharp feature resembling a discontinuous jump starts developing on the x axis in α and K. As the evolution continues, it expands, gets progressively steeper and leaks into other evolved quantities, with the code eventually failing at $t \sim 5.5$. This happens independently of the spatial resolution and the shift condition.

The problem is illustrated in Figure 11.2. The square of the circumferential radius $\overline{\rho}^2$ should be a smooth spacetime invariant, so for a regular hypersurface $t = \text{const.}$ its derivative along the unit normal n^μ should also be smooth. As can be seen in the figure, the directional derivative develops a sharply pronounced feature that gets sharper as spatial resolution is increased. When testing with a number of different resolutions, we found that the maximum of $\mathscr{L}_n\overline{\rho}^2$ exhibits non-convergent behavior consistent with computing finite differences across a step function. This indicates that the spatial slices themselves become non-smooth, which leads to simulation failure. This pathology seems to be generic, since we also see it with near-critical TA waves.

11.3 Quasi-maximal slicing

We conduct a number of tests validate the behavior of the QMS solver. The first one involves evolving weak ($A = 0.1$) Brill waves with a single refinement level (but still

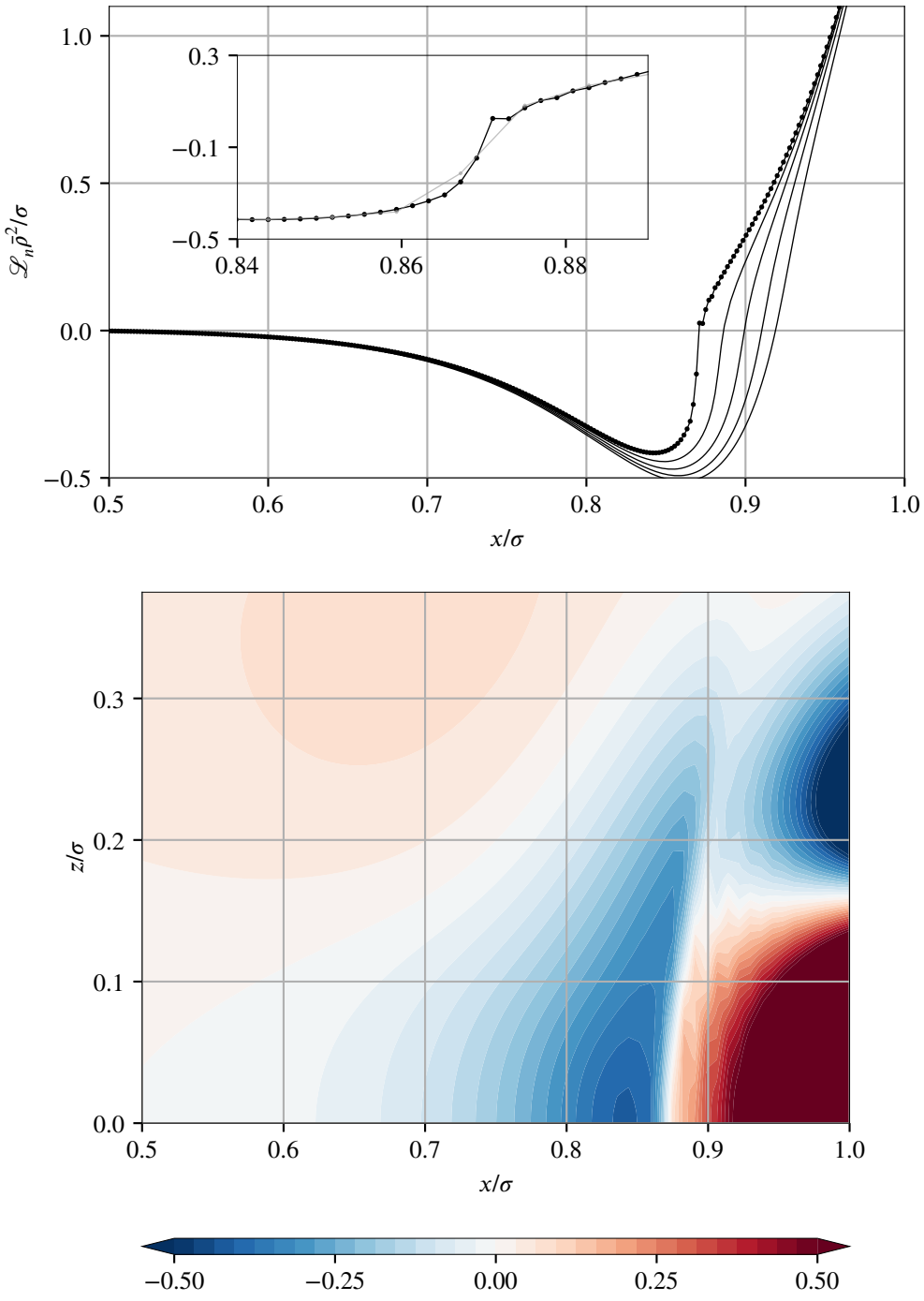

FIGURE 11.2: Failure of the 1+log slicing for $A = 5$ Brill waves.
Top: Lie derivative of $\overline{\rho}^2$ along the normal to the spatial surfaces, computed on the x-axis at times $\{4.625, 4.6875, 4.75, 4.8125, 4.875\}$ (bottom to top). The inset compares the data on the $t = 4.875$ slice for runs with 2048 (black) and 512 (gray) grid points. Circles indicate the position of the grid points.
Bottom: the same quantity in the $x - z$ plane at the time $t = 4.875$.

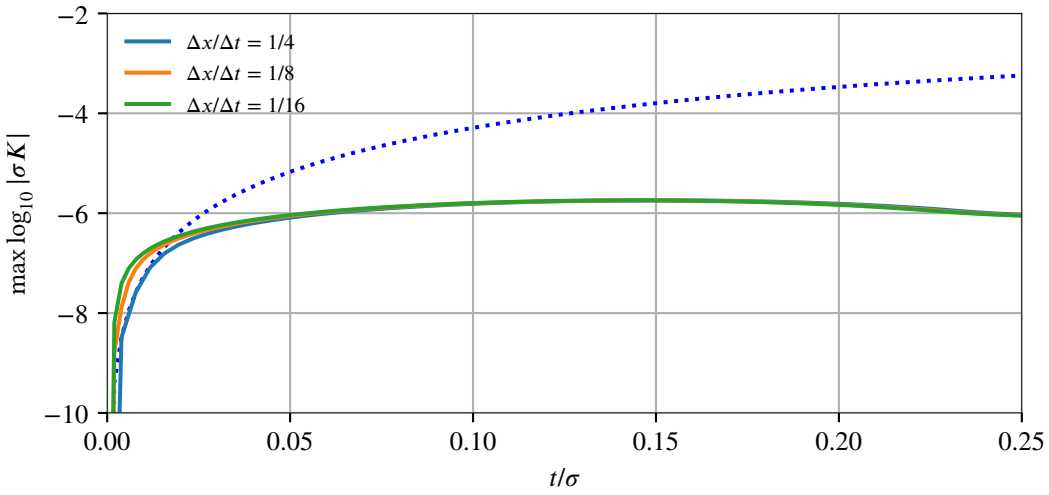

FIGURE 11.3: Convergence of the QMS solver for $A = 0.1$ Brill waves with 1 refinement level. Plotted is the maximum of K in the central area as a function of time, scaled for 2nd order convergence (solid lines). For comparison, the dotted line shows the unscaled maximum of K for 1+log slicing.

using the delayed-solve+extrapolation scheme described in Chapter 9) and vary the CFL factor $\Delta x/\Delta t$. For the purposes of these tests we disable the smoothing scheme that makes W_a gradually vanish towards the boundaries of the solve_level_max-th refinement level. Then we monitor the deviation of K from zero in the central region (i.e. disregarding the values close to the outer boundary). As can be seen in Figure 11.3, K converges to zero at 2nd order as the time step is shortened. The convergence rate corresponds to the linear time extrapolation process. This test establishes that the QMS solver on a single refinement level does indeed converge to maximal slicing as solver precision is increased.

However in a practical scenario we do not tune the solver precision towards the continuum limit; rather we keep it constant to obtain some slicing that is close to, but distinct from, maximal. Therefore we also want to check that increasing the simulation resolution while keeping the QMS accuracy constant converges to a well-defined continuum limit. For that purpose we run a sequence of simulations with stronger $A = 1$ Brill waves, using 5 refinement levels and a constant time step Δt, while varying the spatial resolution. The results are shown in Figure 11.4: we see roughly fourth order convergence, consistent with the order of the FDOs used in the QMS code.

The final test involves evolving the $A = 5$ Brill waves, for which the 1+log slicing fails before the outcome of the collapse becomes clear. When using QMS the coordinate singularity does not form and we are eventually able to find an apparent horizon at coordinate time $t = 9$. Its mass is $M_{\mathrm{AH}} = 0.54$, which is close to the value 0.56 from [33]. Their coordinates are very different from ours, so there is no expectation of observing the same horizon parameters, especially during the "active" phase early after horizon formation. Eventually we would expect the spacetime to evolve into a central region that rapidly relaxes into a Schwarzschild black hole, plus an outward-travelling wave packet. The apparent horizon mass M_{AH} can then be expected to approach the final black hole mass.

As can be seen from Figure 11.5, QMS does indeed reduce the values of K compared

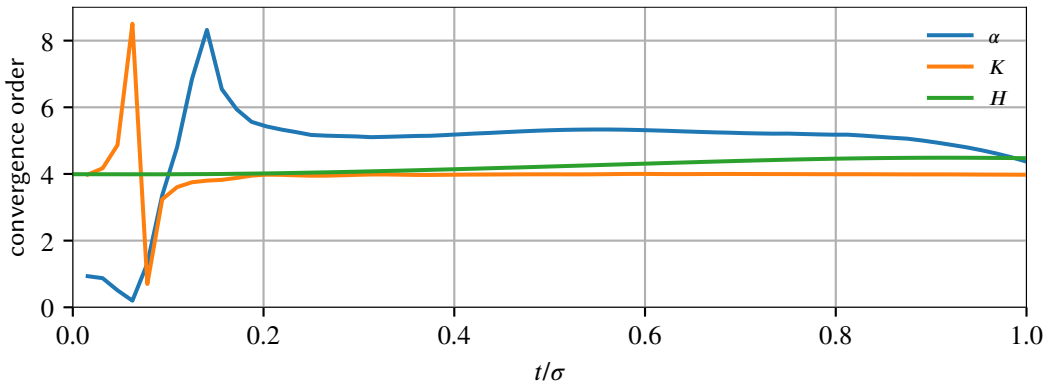

FIGURE 11.4: Convergence of $A = 1$ Brill waves with increasing spatial resolution. For α and K the convergence order is derived from comparing the results of three simulations with spatial step size h, $2h$ and $4h$. The Hamiltonian constraint H should converge to zero, so we compare only h and $2h$.

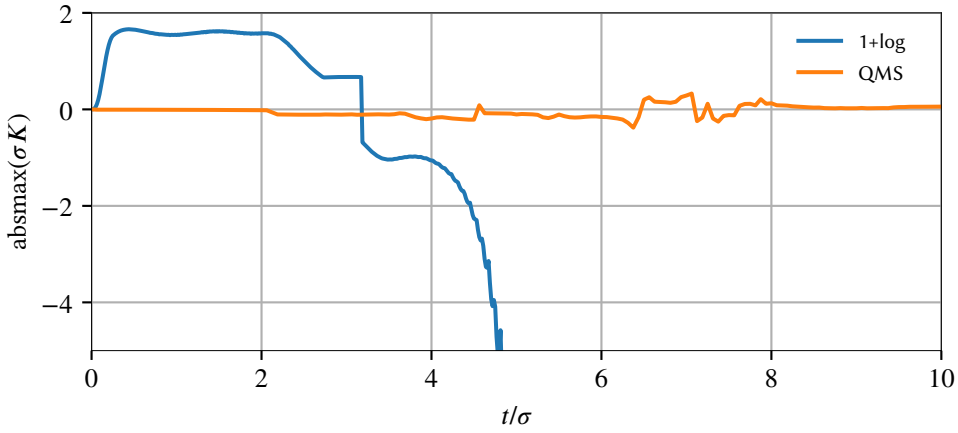

FIGURE 11.5: Comparison of peaks in K between the 1+log slicing and QMS, for $A = 5$ Brill waves. The plotted function absmax(K) is the value of K (on a given spatial slice) that has the largest magnitude, irrespective of its sign.

to the 1+log slicing even in the non-linear regime, bringing the slicing closer to maximal.

11.4 Convergence & accuracy

Finally we look at the convergence of the entire setup in a near-critical case. We run two simulations for TA waves with $A = \overline{1.30114}$ (subcritical, $\log|A - A_*| \approx -8$), one with twice the spatial resolution. As can be seen from the Figure 11.6, the constraint violations are at the roundoff level in the initial part of the simulation, but grow later on, as stronger curvature peaks appear. The violations converge away as the step size is made smaller. The precise convergence order is hard to estimate, as it is a product of several different codes, but it stays above 4. After the waves have dispersed, a rather high (but convergent) level of constraint violations remains on the grid, due to so-called "zero-speed" modes of BSSN.

Our setup also seems robust against changing the number of refinement levels, as illustrated in Figure 11.7. We found no indication that adding refinement levels would change the outcome (sub- versus supercritical) of the simulation. When the resolution provided by the configured number of refinement levels is not sufficient for the spacetime being evolved, the simulation will typically crash due to a coordinate singularity forming.

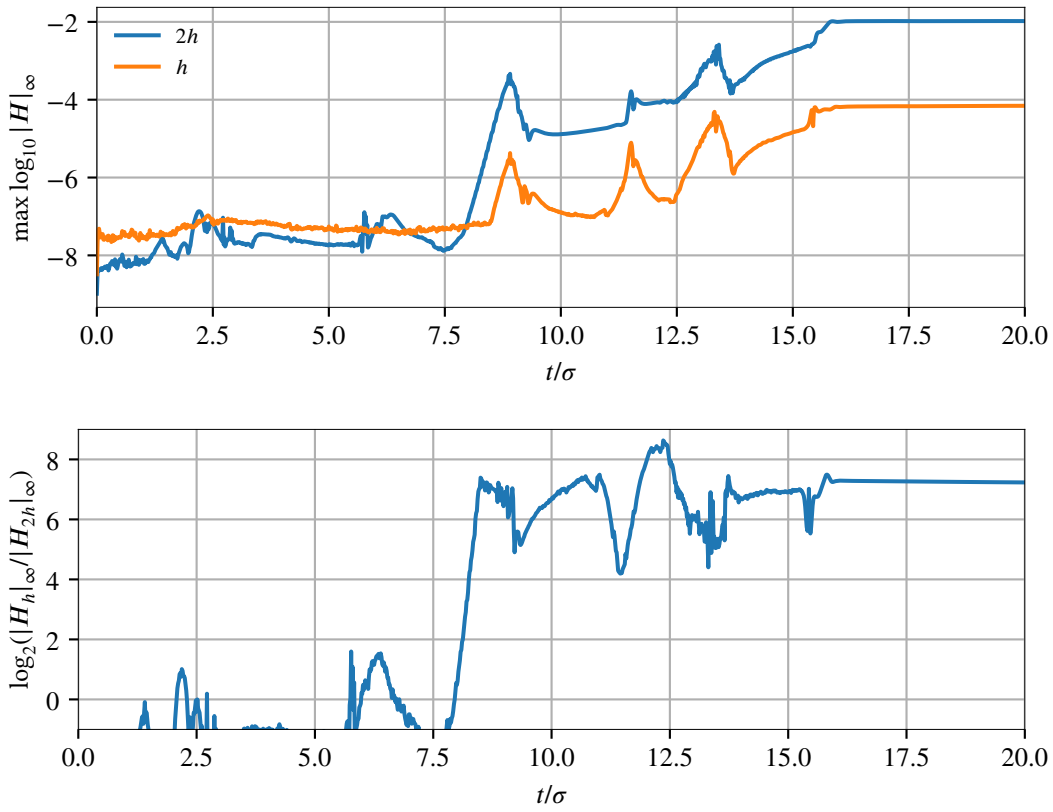

FIGURE 11.6: Convergence of the Hamiltonian constraint with increasing spatial resolution for TA waves with $A = \overline{1.30114}$ (subcritical). Note the relatively high level of constraint violations that remain on the grid after the waves have dispersed due to so-called "zero-speed modes" in BSSN.
Top: maximum of the constraint violation as a function of time for two simulations with spatial stepsizes h and $2h$.
Bottom: convergence order estimated from those two simulations. Note that in the initial part of the simulation the roundoff error dominates, so the convergence order is low.

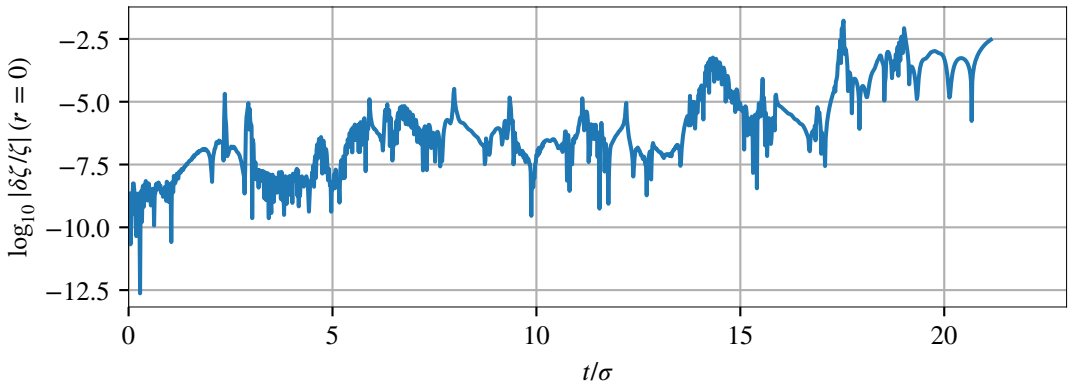

FIGURE 11.7: Relative change in ζ from adding one refinement level, for TA waves with $A = \overline{1.3008}$ (supercritical). $\delta\zeta$ is the difference between the values in the 10-level and the 11-level simulation.

Chapter 12

The search for the critical point

With the code tested and validated, we run a series of simulations with the intent to approach the critical point(s) as closely as possible. We use four one-parameter families of initial data:

Brill+: Brill waves with $A > 0$ (section 2.1);

Brill-: Brill waves with $A < 0$;

TA+: TA waves with $A > 0$ (section 2.2);

TA-: TA waves with $A < 0$, equivalent to positive-A TA waves evolved backwards in time.

12.1 Hardware

We ran the majority of our simulations on six Intel Xeon Phi 7210 machines [1], which were available to us through the MetaCentrum National Grid Infrastructure system [20]. Every such machine contains 64 execution cores, each clocked at 1.3 GHz and carrying a 512-bit AVX-512 vector unit. As the general-purpose (non-vector) performance of these cores is very low (comparable to contemporary low-end laptops), the use of these machines is only worthwhile with heavily vectorized code. Thankfully the BSSN right-hand side computation code implemented in McLachlan can be easily compiled into AVX-512 instructions; we also wrote AVX2 (256-bit) paths for the critical parts of libmg2d. The vectorized numerical code was then comparable in runtime to the non-vectorized "overhead" code, which usually takes negligible amounts of time on common CPUs. At that point we did not pursue optimizing our code further for this architecture.

As the machines lacked fast interconnects, we ran most of our simulations without the use of MPI parallelization — as a single multithreaded process using all the 64 cores in the package. [2] Our simulations were thus mainly constrained by what could be computed on a single such system in "reasonable time", where we took "reasonable"

[1] The primary reason we chose those machines was that they were almost entirely unutilized (likely due to their unusual architecture), which allowed us effectively unlimited exclusive access to them.

[2] Each core is also capable of 4-way hardware multithreading (making for 256 "virtual" cores), but it is not useful to us as the threads would be competing for the same floating-point execution units. We thus bind one process thread to each hardware core.

to be about 10 days (i.e. $\sim 15 \times 10^3$ CPU-hours) for the simulations closest to A_* (and up to several weeks to verify convergence of chosen runs using higher resolution).

12.2 Simulation setup

The Cactus parameter files are set up as per the following table:

numerical domain extent	square from $x = 0, z = 0$ to $x = 128, z = 128$
number of points on the coarsest grid	128×128 (i.e. the step size is 1)
inner boundary conditions	reflection (along both z axes)
outer boundary conditions	fixed (along both axes)
refinement level count	varies, up to 12
refinement prolongation order	9
ML_BSSN FDO order	8
dissipation order	9
ghost zone size	5 points
time integrator	RK4
CFL factor ($\Delta x/\Delta t$)	$\frac{1}{8}$
2D data output interval	$\delta t = 0.5$ (typically; corresponds to once per 4 coarsest-level time steps)
1D data output interval	$\delta t = \frac{1}{64}$ (typically; corresponds to 8× per one coarsest-level time step)
lapse condition	QMS
shift condition	zero

The following table summarizes the main QMS solver parameters:

FDO order	4
base residual tolerance	10^{-8} (corresponds to 0.04 on 11-th refinement level)
F_r ("CFL factor")	0.17
K_{pre}	1
K_{post}	24
solve_level	varies, from 4 to 10
solve_level_max	varies, from 2 to 6

The precise parameter files we used are provided as an attachment to the electronic version of this work.

12.3 Slicing

The quasi-maximal slicing condition has largely met or exceeded our expectations. We found that it can indeed be used to avoid coordinate singularities that appear with 1+log slicing, even very close to the critical point(s). Meanwhile it increased the computational cost only moderately, with typically $\lesssim 10\%$ of the run time spent in QMS code.

The rough "rule of thumb" we observed was that coordinate singularities appear at the boundaries between large positive and negative peaks in K and can be avoided by not letting $|K|$ grow too large. That can be achieved by increasing the accuracy

of the QMS solver, thus bringing W_a closer to the exact solution W. This, of course, increases the computational cost of QMS.

Furthermore, we found that the different ID families we experimented with are susceptible to these coordinate singularities to different degrees, with positive-A Brill waves proving the hardest to evolve. Within the computation constraints stated above, we were able to obtain five clear echoes for TA+, four (with less clear outlines of another one or two) for TA-, four for Brill-, and two for Brill+. Even at highest resolutions we can afford, the near-critical Brill+ are polluted by numerical noise and constraint violations to such an extent that we would not put much confidence in them, were it not for comparisons with [33] (who were able to access significantly larger amounts of computing resources) and the universal form of the echoes observed (discussed in section 12.5). It would thus seem that the convergence in literature towards using positive-A Brill waves to study critical collapse was rather unfortunate.

While we found no fundamental reasons why our approach could not be pushed closer to the critical point, the growth of simulation runtime with increasing grid resolution makes this impractical. This obstacle might be removable by switching to *adaptive* mesh refinement, since extremely high resolutions are only needed for short periods of coordinate time. However, the estimated effort required to implement adaptive refinement within our setup placed it beyond the scope of this work. It is also conceivable that roundoff error, growing as resolution increases, would make 64-bit floating point numbers insufficient for accurately resolving the collapse, necessitating a switch to some other numerical representation.

12.4 Near-critical spacetime properties

Our primary way of classifying the data as super- or subcritical is by monitoring the minimum of the lapse at each time slice. Since the lapse tends to "collapse" in regions of high curvature, its minimum is a simple dimensionless indicator for imminent horizon formation. Typical behavior of the minimum is shown in Figure 12.1: it goes through several oscillations before eventually growing towards unity (subcritical) or starting to vanish rapidly (supercritical). While some earlier studies only monitor the central value $\alpha(r = 0)$ (compare e.g. FIG. 2 in [2]), we confirm the finding of [33] that the strongest dynamics happens on the z-axis away from the origin, so the central value is not always a good indicator. Sometimes the simulation will fail after a horizon can be found, but before a definitive trend can be seen in the central value of the lapse.

As A is tuned closer towards A_*, the lapse remains visually indistinguishable initially, deviating only in the final part of the collapse as new oscillations are added. An infinite train of oscillations is conjectured in the limit $A \to A_*$. Analogous behavior can be seen (Figure 12.2) in the extrema of curvature invariant I_K, except their magnitude gets progressively larger. Again, it has been conjectured that arbitrarily high curvatures appear close enough to A_*, with a naked singularity as the limit [28].

A subcritical spacetime can then also be characterized by the global maximum of I_K that appears before the waves disperse — Figure 12.3 plots these maxima as a function of the distance from the critical point. The graph for each ID family is composed of arched segments. This structure can be understood from the behavior sketched above: for a given value of A the evolution is a series of progressively stronger "echoes". Typically the last echo is the strongest one (though not always). As A is further tuned towards A_*:

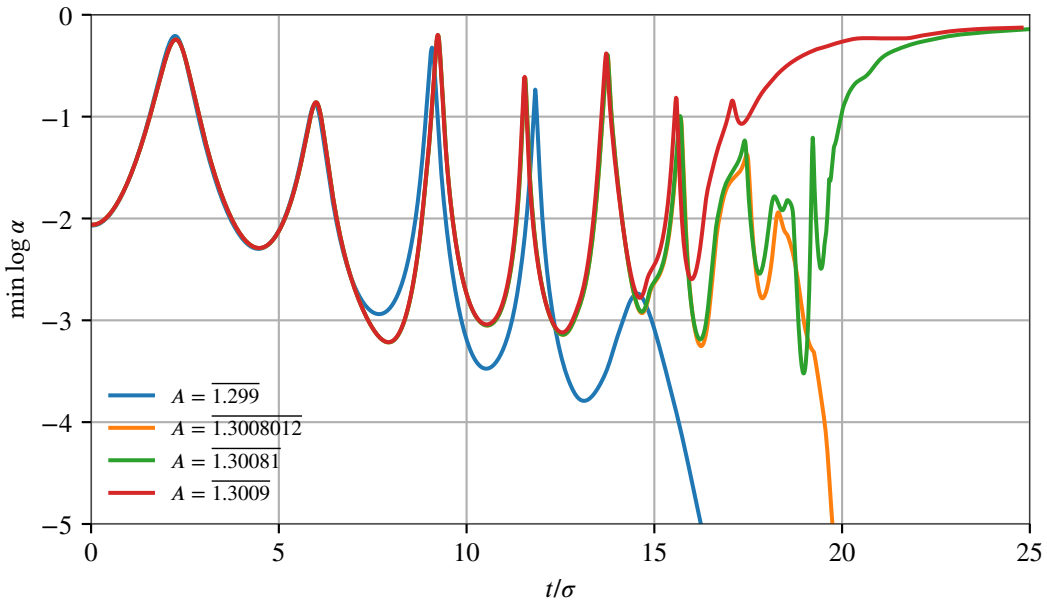

FIGURE 12.1: Minimum value of the lapse at each time slice, for several sub- and supercritical TA+ evolutions.

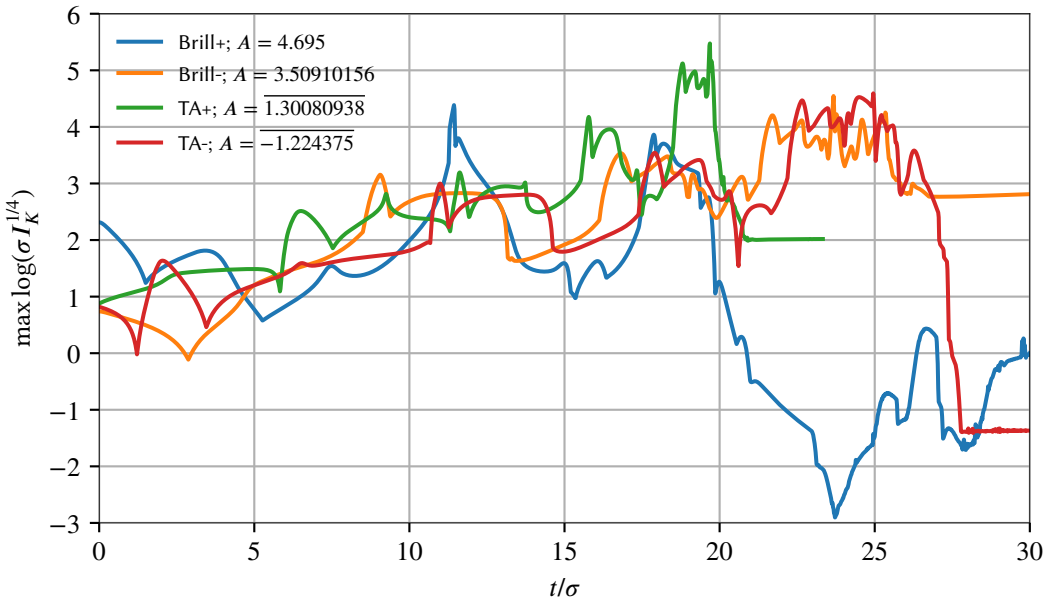

FIGURE 12.2: Maximum value of the Kretschmann scalar I_K at each time slice, for subcritical simulation of different ID families. It is noticeable that I_K does not vanish completely towards the end of the simulation, after the waves have dispersed. This is associated with the so-called "standing modes" in BSSN, which cause constraint violations to linger on rather than disperse or decay. In future work it would be desirable to use a formulation with constraint damping, such as CCZ4 [7].

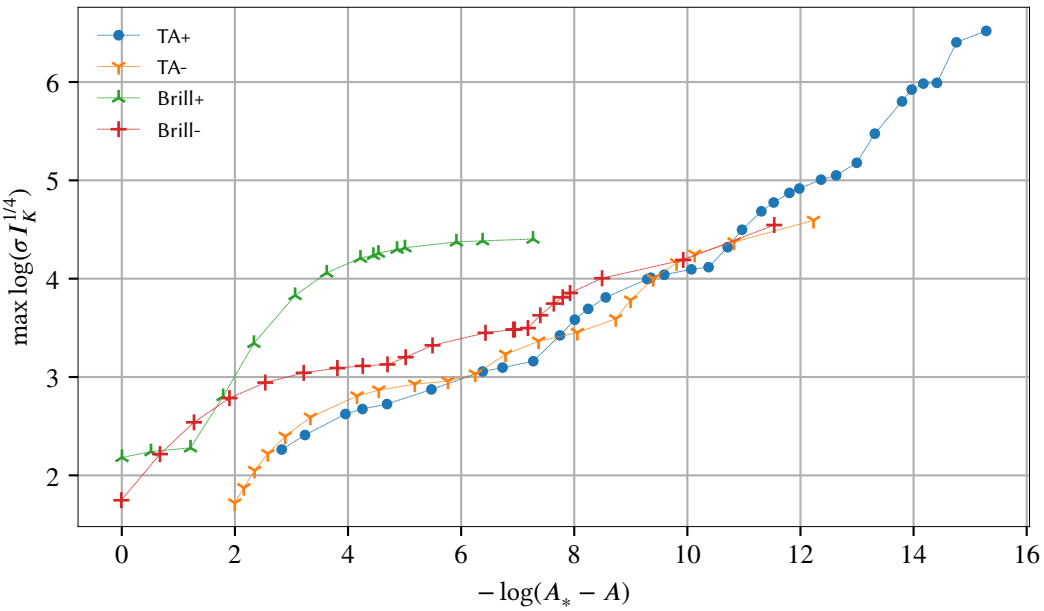

FIGURE 12.3: Global maxima of the Kretschmann scalar I_K for different ID families, as a function of the distance from the critical point. A_* is estimated to be the midpoint between the bounds in Table 12.1, except for Brill+ waves where we take the much more accurate value from [33]. Every point represents the maximum of I_K over an entire simulation with that value of A. Compare Figure 4 in [33].

ID family	subcritical	supercritical
TA+	1.30080796	1.3008075
TA-	−1.224375	−1.22436524
Brill+	4.696	4.698
Brill-	−3.50910156	−3.5091211

TABLE 12.1: Our best bounds on the critical point A_* for each ID family.

- early parts of the evolution remain almost unchanged;
- the strongest peak (for the current range of A) grows towards its "critical" value (i.e. the value it has in the exactly critical evolution);
- the final dispersal process following the strongest peak gradually coalesces into a new peak, which eventually overcomes the previous one and becomes the new global maximum.

Each segment then corresponds to the range of A where a specific local maximum is the globally strongest one.

12.5 Universality & echoing

To investigate the presence of self-similarity in our collapse simulations, we look at the scalar ζ (Figures 12.4-12.11). While its detailed behavior is very complex, it can be seen from the figures that it attains a series of negative and positive extrema — initially at the origin, later the peaks appear at non-zero values of z. This "bifurcation"

ID family	$\lambda^{(1)}/\sigma$	$\lambda^{(2)}/\sigma$	$\lambda^{(3)}/\sigma$	$\lambda^{(4)}/\sigma$	$\lambda^{(5)}/\sigma$	$\lambda^{(6)}/\sigma$
TA+	0.15	0.076	0.028	0.011	0.0078	
TA-	0.36	0.093	0.054	0.021	0.020	0.019
Brill+	0.30	0.0022	0.032			
Brill-	0.079	0.054	0.028	0.020		

TABLE 12.2: Scaling factors $\lambda^{(i)}$ used to equalize the echoes in Figures 12.12 and 12.13

happens for all the studied ID families, which suggests it is generic. While there are many local extrema, the most conspicuous ones are those that coincide with the global maxima of I_K for some range of A (remembering that on the axis of symmetry $I_K \sim \zeta^2$). These peaks always correspond to a negative local *minimum* of ζ, which is then shortly followed by a positive local maximum.

For each such minimum-maximum pair $\zeta^{(i)}_{\min}, \zeta^{(i)}_{\max}$ we consider the timelike geodesic that connects the two extrema. For simplicity, and to minimize the effects of interpolation that would otherwise be necessary, we approximate this geodesic with the worldline $z = z^{(i)}_0$ that passes through $\zeta^{(i)}_{\min}$ and consider the values

$$\zeta^{(i)}(t) = \zeta\left(t, z = z^{(i)}_0\right). \tag{12.1}$$

In order to compare invariant quantities, we construct the proper time

$$\tau^{(i)}(t) = \tau^{(i)}\left(t, z = z^{(i)}_0\right) \tag{12.2}$$

along this worldline, offsetting it so that $\tau^{(i)} = 0$ at the minimum $\zeta^{(i)}_{\min}$. Finally, since the individual peaks have vastly different magnitudes, we rescale them as

$$\zeta^{(i)}_0 = \left(\lambda^{(i)}\right)^2 \zeta^{(i)}, \tag{12.3a}$$

$$\tau^{(i)}_0 = \tau^{(i)}/\lambda^{(i)}, \tag{12.3b}$$

where $\lambda^{(i)} = (-\zeta^{(i)}_{\min})^{-1/2}$. The values of $\lambda^{(i)}$ we obtain are listed in Table 12.2.

We plot these rescaled profiles for the best subcritical simulation in each ID family in Figures 12.12. We believe that the striking match between the profiles, obtained by rescaling with a *single* parameter λ and despite gross simplifications, justifies calling these profiles *echoes*. Furthermore, as seen from Figure 12.13, the echo profile is *universal* across the four very different ID families we use.

The first echo for Brill+ is very different from all the others and is thus not a universal echo. The second echo is closer to the universal one, but still considerably different. Interestingly enough, the third echo for the $A = 4.696$ evolution is *weaker* than the second one, but conforms well to the universal one. The first echo for TA+ and TA- is close to the universal one, but is still markedly different in its trailing part. The fifth and sixth echoes for TA- are very close to each other in space, time, and magnitude, so they could rather be parts of a single echo.

This behavior is similar to what is known about spherically symmetric critical collapse, where during the initial few non-universal echoes the system "sheds" all imprint of the initial deformation and approaches the universal critical solution. The final echo we observe also tends to deviate slightly — this could be caused by it not

yet reaching its final "critical" form (as discussed previously) or limited temporal resolution of stored simulation data (since the echo duration in simulation coordinates is very short). We also find significant differences from the DSS spherical collapse:

- The collapse "bifurcates" into two centers above and below the origin on the axis of symmetry.
- As can be glimpsed from the contour plots of ζ, even locating the echoes is a nontrivial process. E.g. for our best subcritical Brill+ data, the last echo is actually weaker than the preceding one.
- In a DSS scenario, any dimensionless quantity is exactly periodic in the logarithmic proper time with constant period Δ. Dimensionful quantities scale with the appropriate power of e^{Δ}. In our axially symmetric vacuum collapse, we could construct $\Delta^{(i)} = \log(\lambda^{(i+1)}/\lambda^{(i)})$, but we did not find any regularity in this quantity. The scale ratios also seem to differ between then ID families, so they are not universal.
- It is not clear how to measure delays between the echoes, as they depend on the worldline chosen. Connecting every two consecutive minima $\zeta^{(i)}_{\text{min}}$ with a timelike geodesic (different for each pair), we again did not observe any regularity.

12.6 Apparent horizons

In all supercritical evolutions we are always able to find a MOTS (Figure 12.15), further confirming the formation of a black hole. In agreement with [33], we find two disjoint "off-center" horizons for some Brill+ and TA+ evolutions. We only find centered horizons for Brill- and TA- waves, but their oblong shapes suggest that off-center horizons either are present at an earlier time in the respective simulation (but we do not store the corresponding slices for postprocessing) or will appear after tuning the initial data closer to A_*.

While the apparent horizon mass was used as an indicator of critical behavior in [2], there is a number of problems with this approach, as pointed out in [33]:

- apparent horizons are foliation-dependent and may appear with very different initial masses, or not at all, depending on the chosen slicing;
- obtaining the final black hole mass requires evolving the data with very high curvatures for a long time, which is computationally demanding;
- there is a question of how to count the masses from multiple apparent horizons;
- since our horizon finder code runs during "postprocessing", it requires stored simulation data; since this data is stored with low temporal resolution (to avoid overwhelming storage requirements), we can only locate the initial horizon mass with a large "sampling" error, even when disregarding the abovementioned slicing effects.

While we do present the horizon masses in Figure 12.14 and the tables in Appendix A.1, we do this mainly for completeness and make no claims as to what, if any, significance should be ascribed to them.

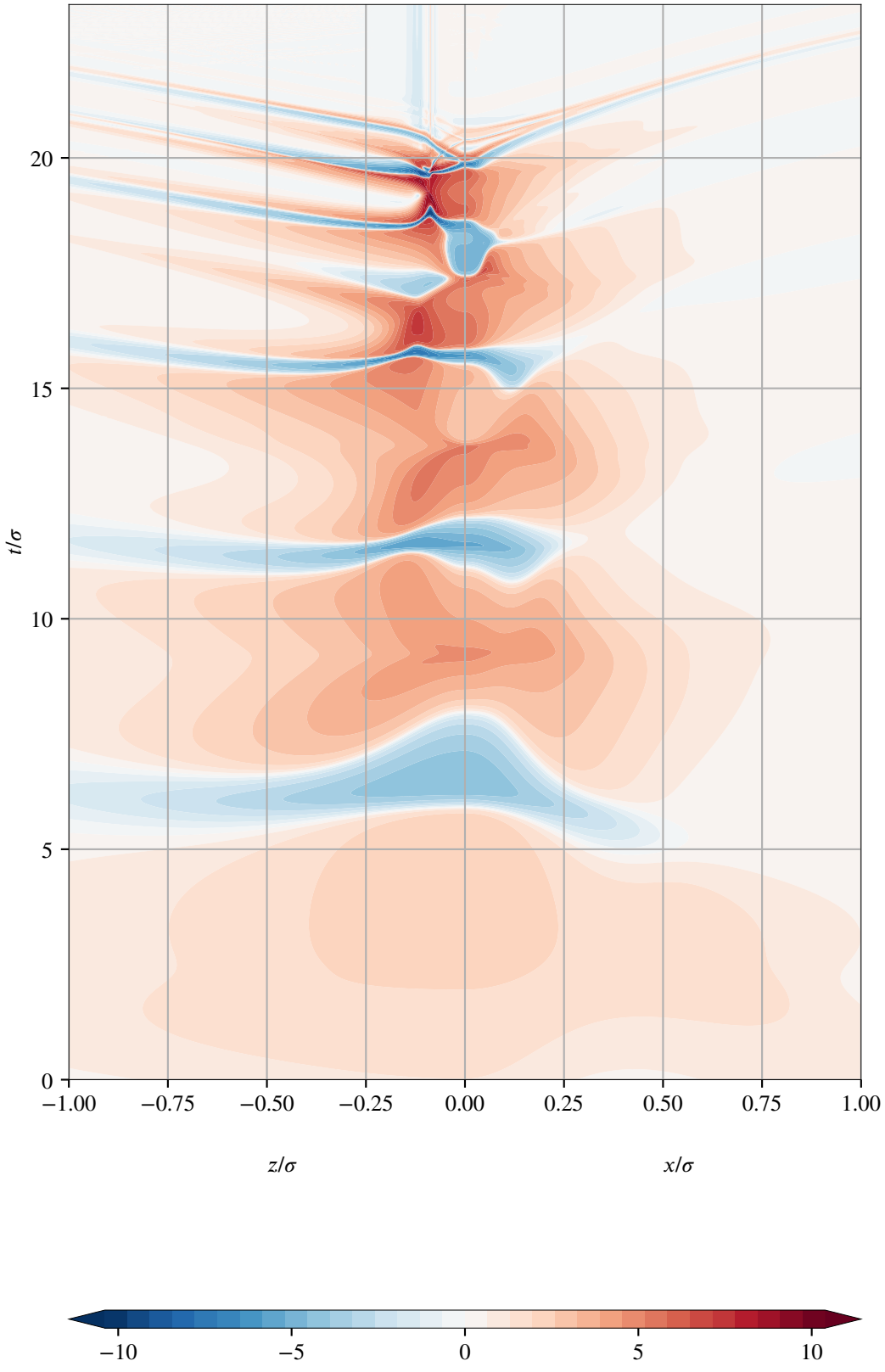

FIGURE 12.4: Contours of $\operatorname{arcsinh}(\sigma^2\zeta)$ for TA+ waves with $A = \overline{1.30080938}$ (subcritical). The right half of the plot (positive horizontal axis) shows the values along the x axis, the left half shows the values along the z axis.

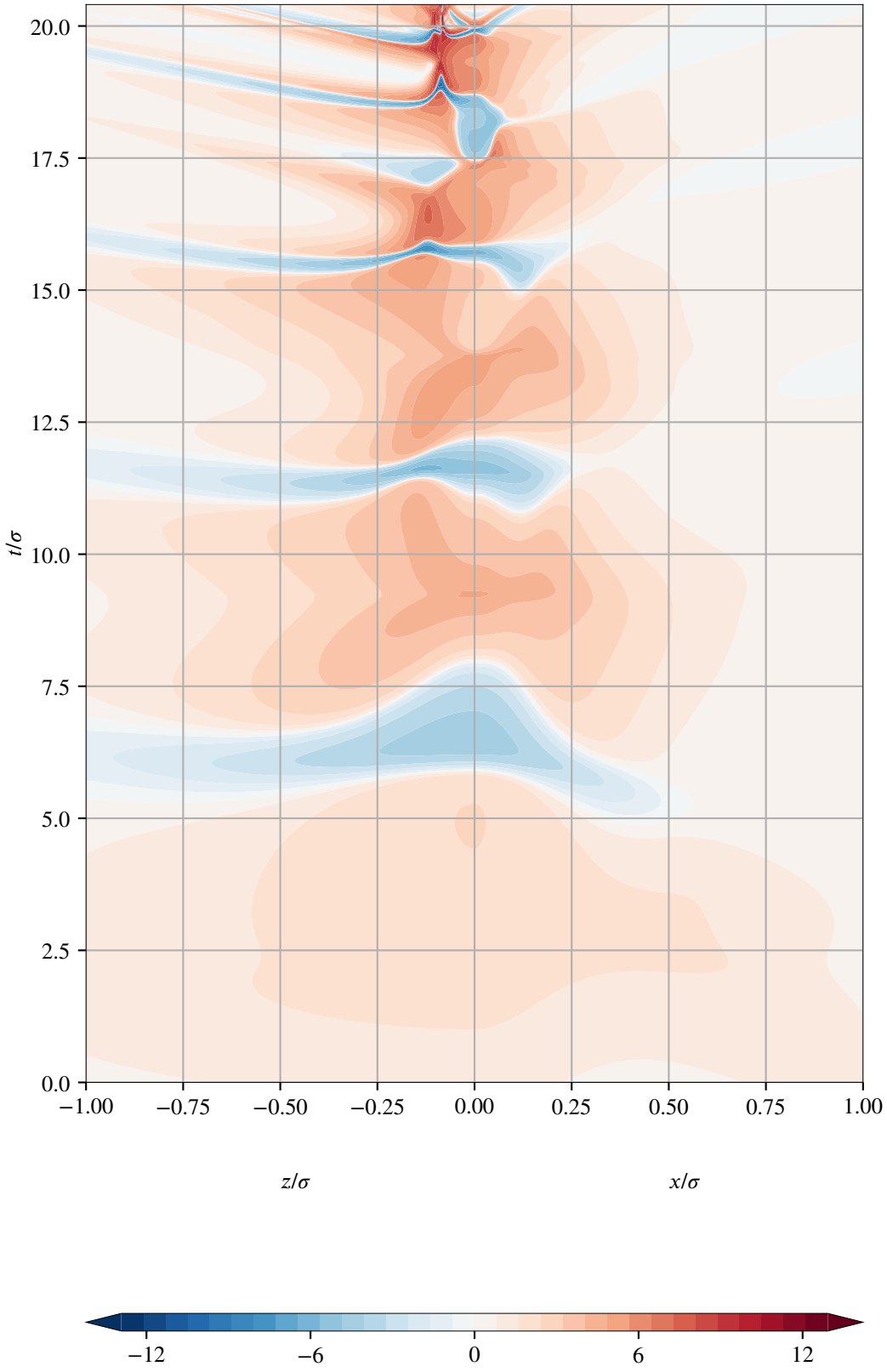

FIGURE 12.5: Contours of $\operatorname{arcsinh}(\sigma^2\zeta)$ for TA+ waves with $A = \overline{1.3008075}$ (supercritical). The right half of the plot (positive horizontal axis) shows the values along the x axis, the left half shows the values along the z axis.

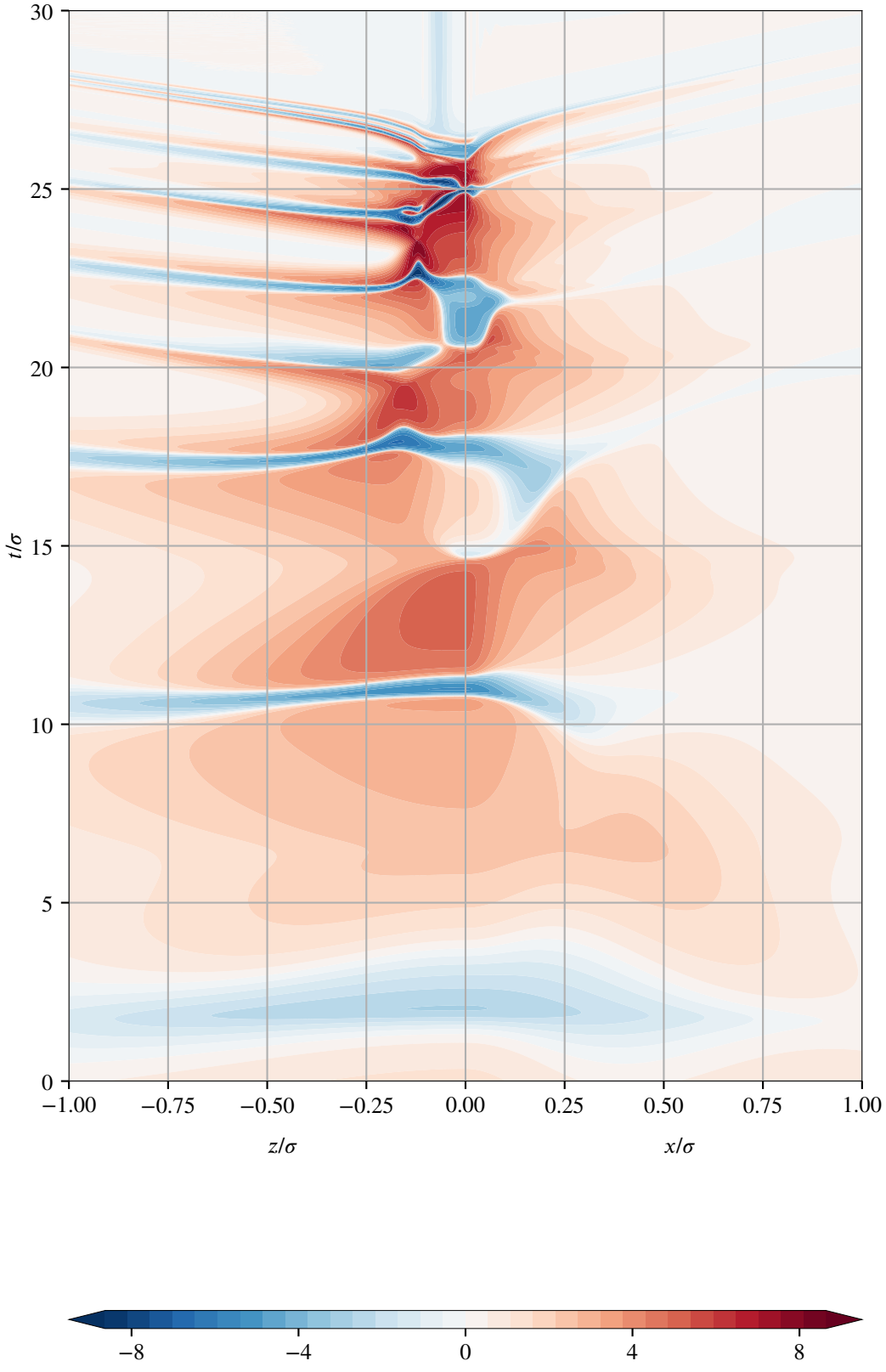

FIGURE 12.6: Contours of $\operatorname{arcsinh}(\sigma^2\zeta)$ for TA- waves with $A = -\overline{1.224375}$ (subcritical). The right half of the plot (positive horizontal axis) shows the values along the x axis, the left half shows the values along the z axis.

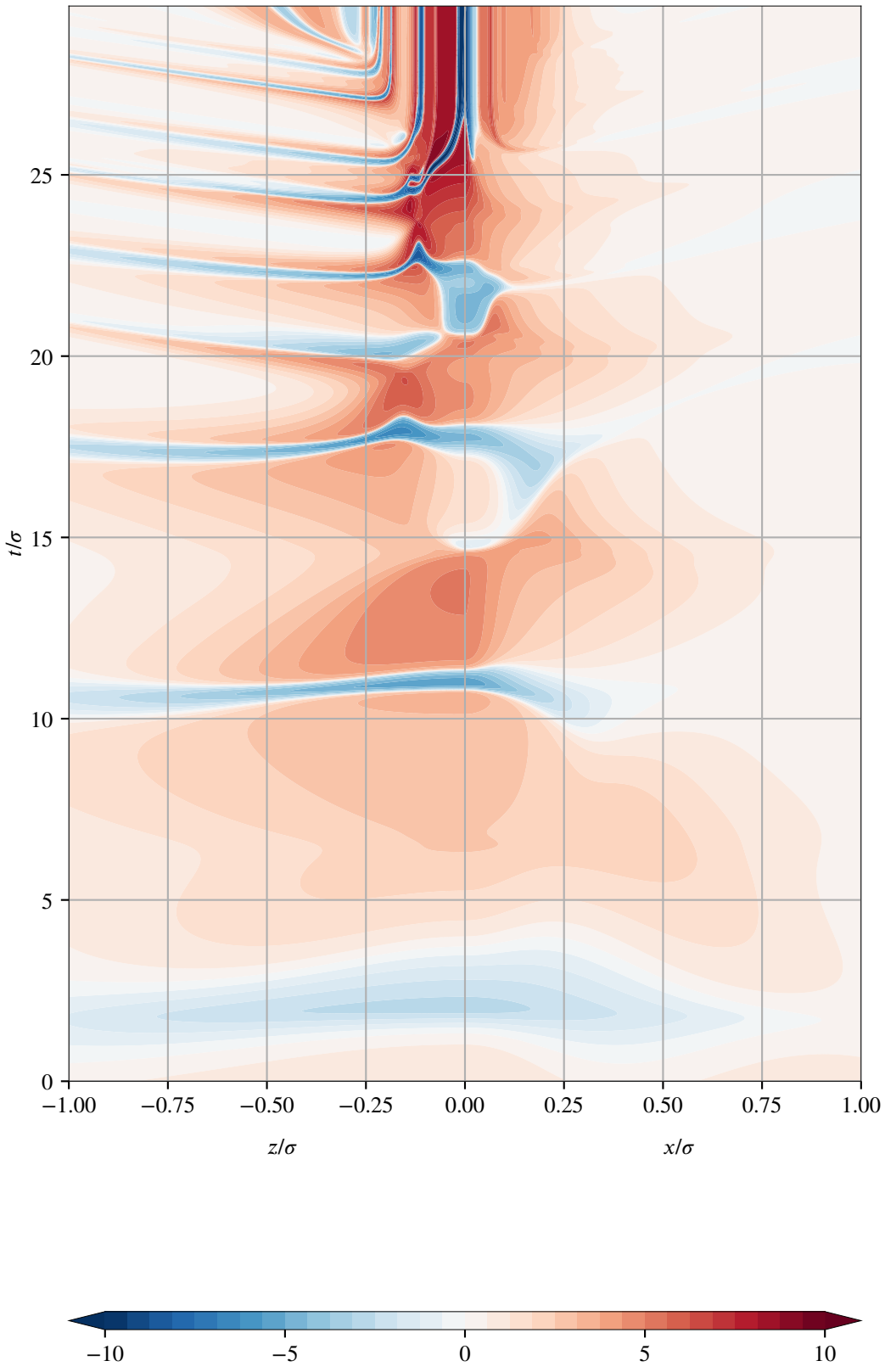

FIGURE 12.7: Contours of $\operatorname{arcsinh}(\sigma^2\zeta)$ for TA- waves with $A = -\overline{1.22436524}$ (supercritical). The right half of the plot (positive horizontal axis) shows the values along the x axis, the left half shows the values along the z axis.

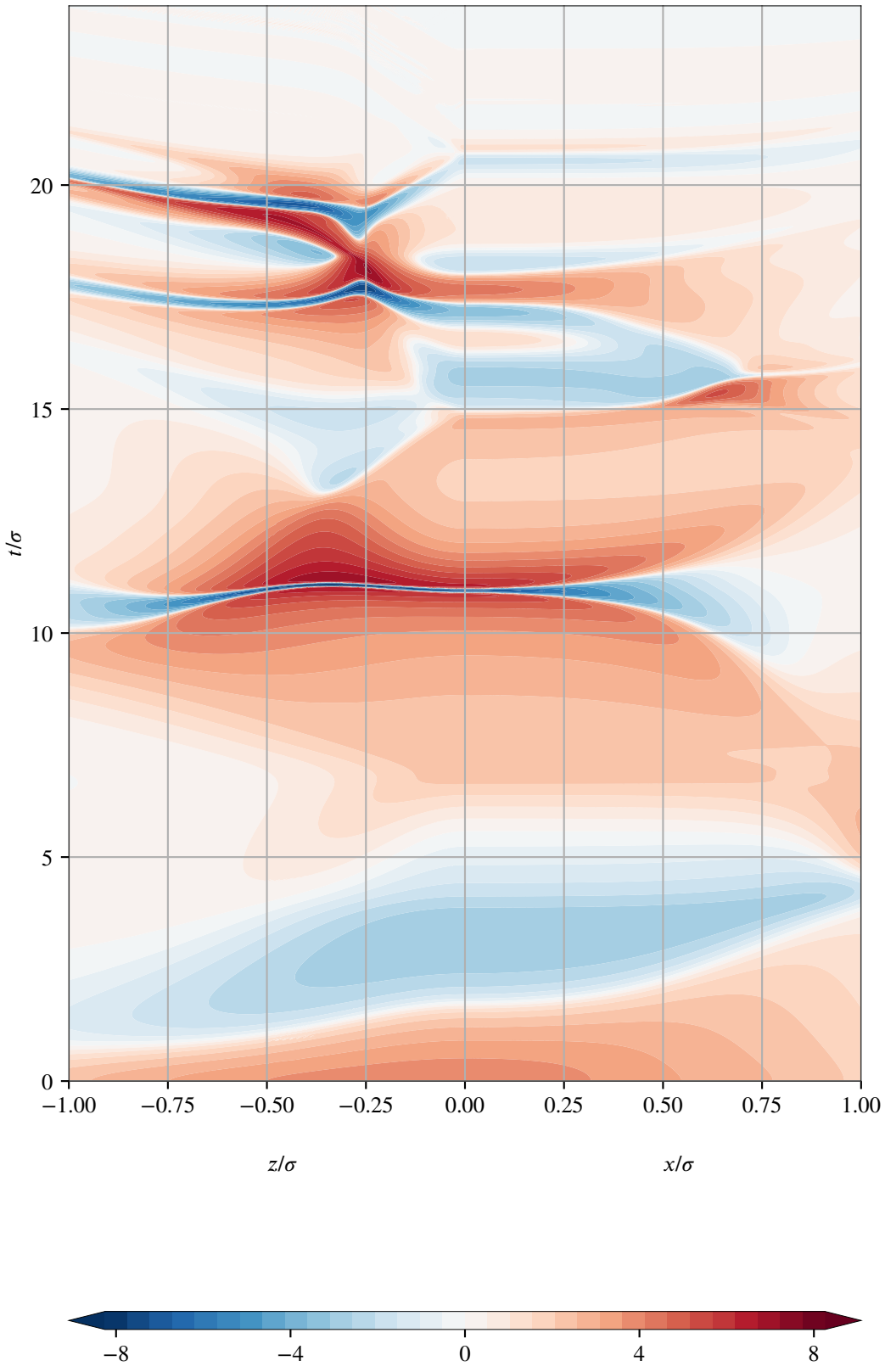

FIGURE 12.8: Contours of $\operatorname{arcsinh}(\sigma^2\zeta)$ for Brill+ waves with $A = 4.696$ (subcritical). The right half of the plot (positive horizontal axis) shows the values along the x axis, the left half shows the values along the z axis.

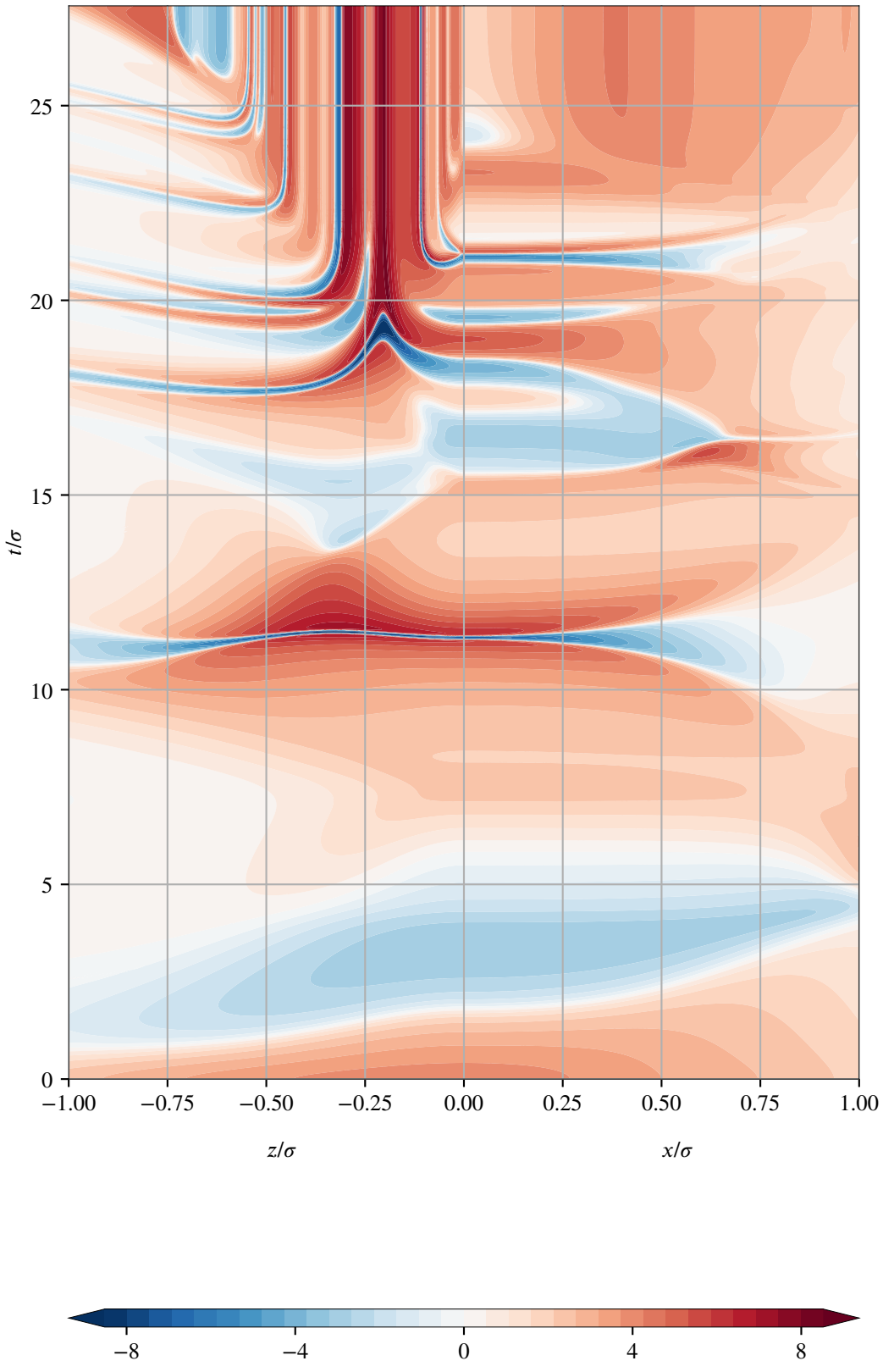

FIGURE 12.9: Contours of $\operatorname{arcsinh}(\sigma^2\zeta)$ for Brill+ waves with $A = 4.698$ (supercritical). The right half of the plot (positive horizontal axis) shows the values along the x axis, the left half shows the values along the z axis.

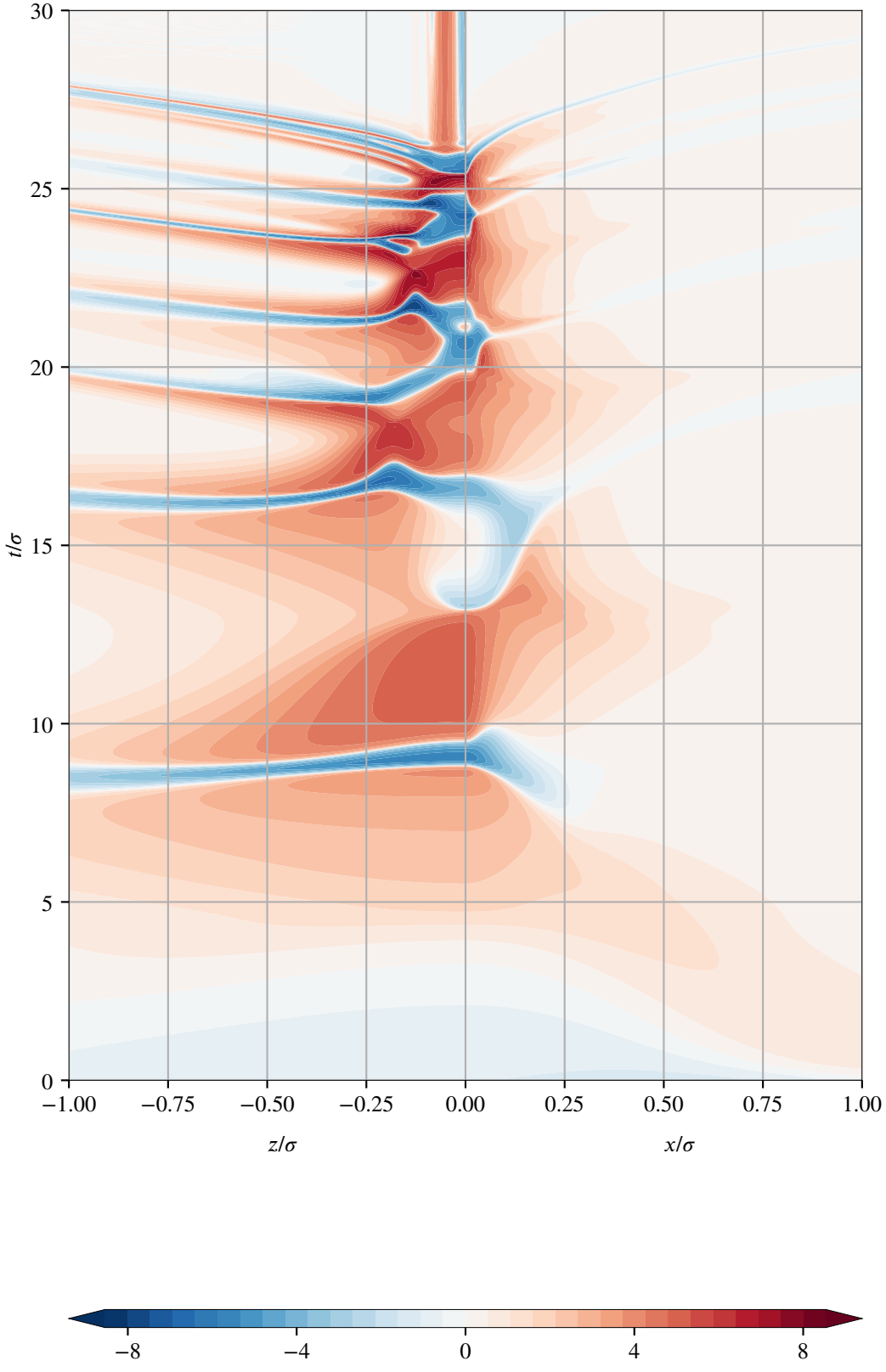

FIGURE 12.10: Contours of $\operatorname{arcsinh}(\sigma^2\zeta)$ for Brill- waves with $A = -3.50910156$ (subcritical). The right half of the plot (positive horizontal axis) shows the values along the x axis, the left half shows the values along the z axis.

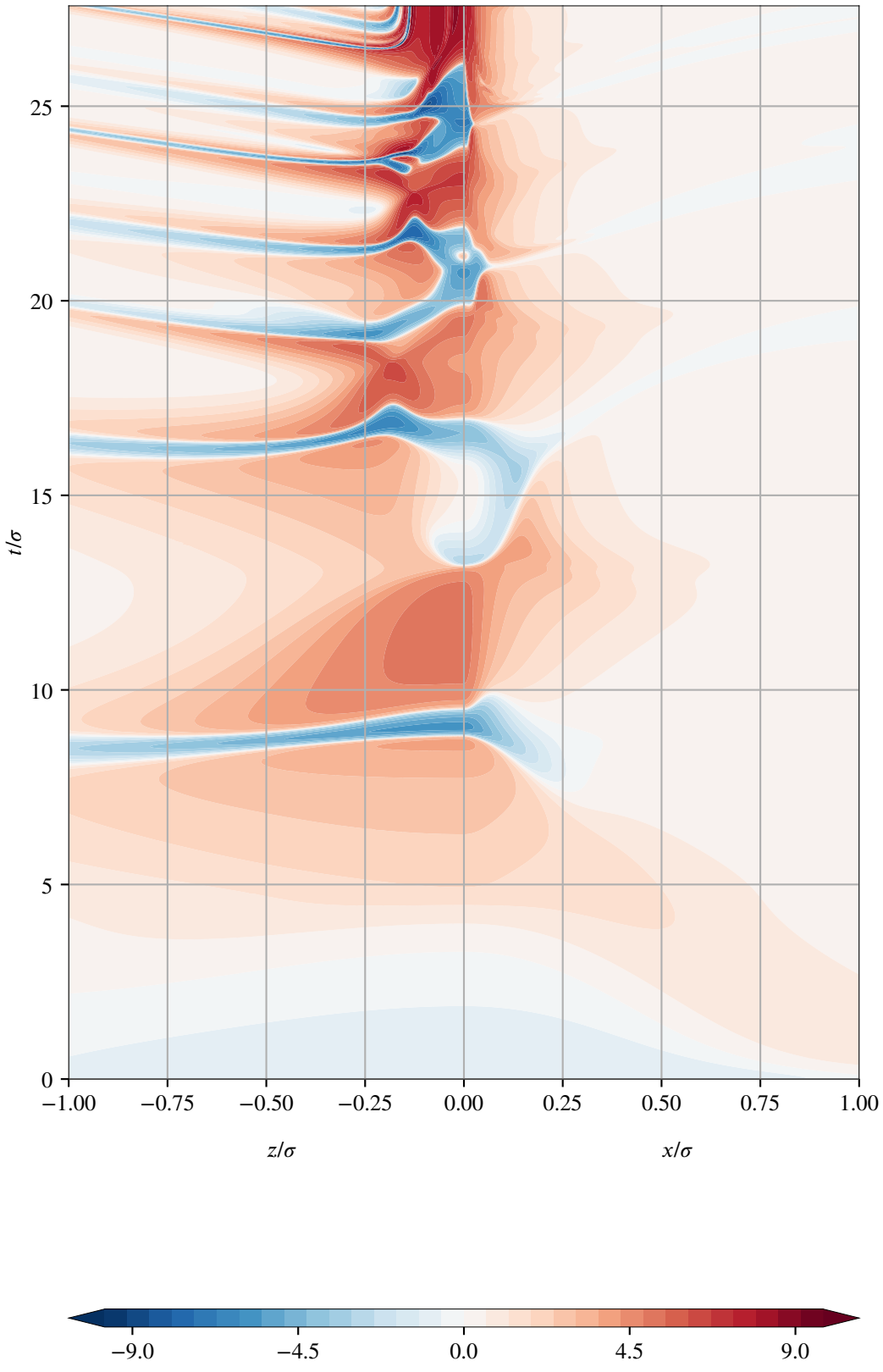

FIGURE 12.11: Contours of $\operatorname{arcsinh}(\sigma^2\zeta)$ for Brill- waves with $A = -3.5091211$ (supercritical). The right half of the plot (positive horizontal axis) shows the values along the x axis, the left half shows the values along the z axis.

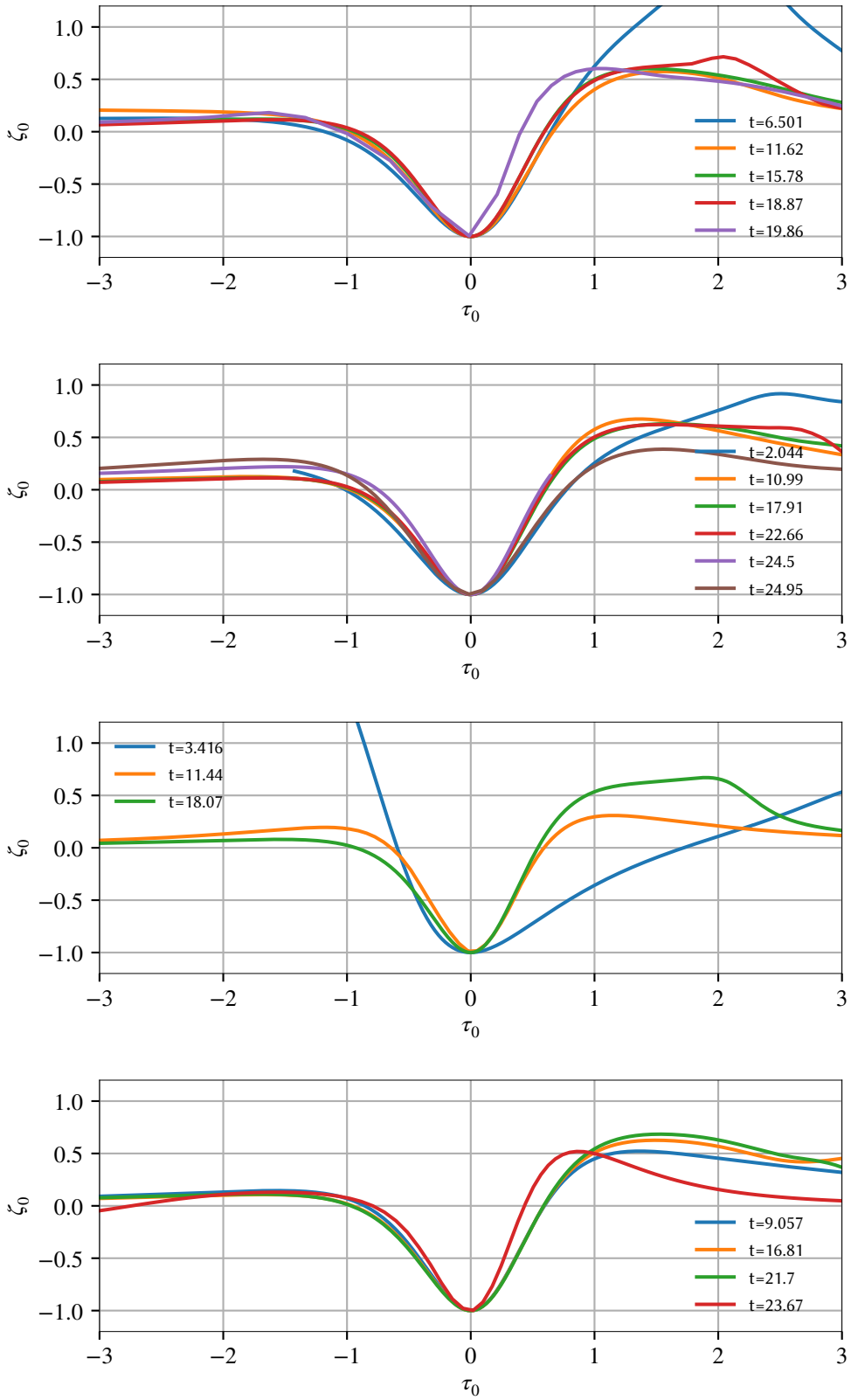

FIGURE 12.12: Echoing in the subcritical simulations from Table 12.1: top to bottom TA+, TA-, Brill+, Brill-. Plotted are rescaled values of the invariant ζ as functions of rescaled proper time τ on a worldline through the echo. The legend indicates the coordinate time t at which the corresponding minimum $\zeta^{(i)}_{\text{min}}$ is attained. See main text for details. Cf. also FIG. 3 in Appendix A.4, where spatial interpolation is used to further improve the match.

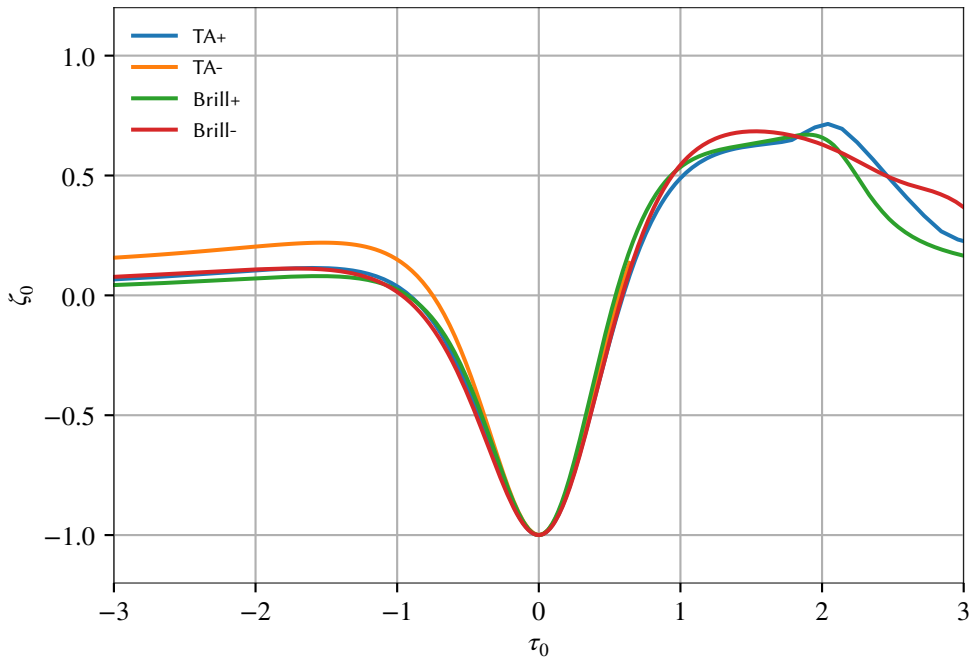

FIGURE 12.13: Universality of the echoes across different ID family. For each family we plot the second-to-last echo from Figure 12.12, except for Brill+ where we take the last. See main text for details.

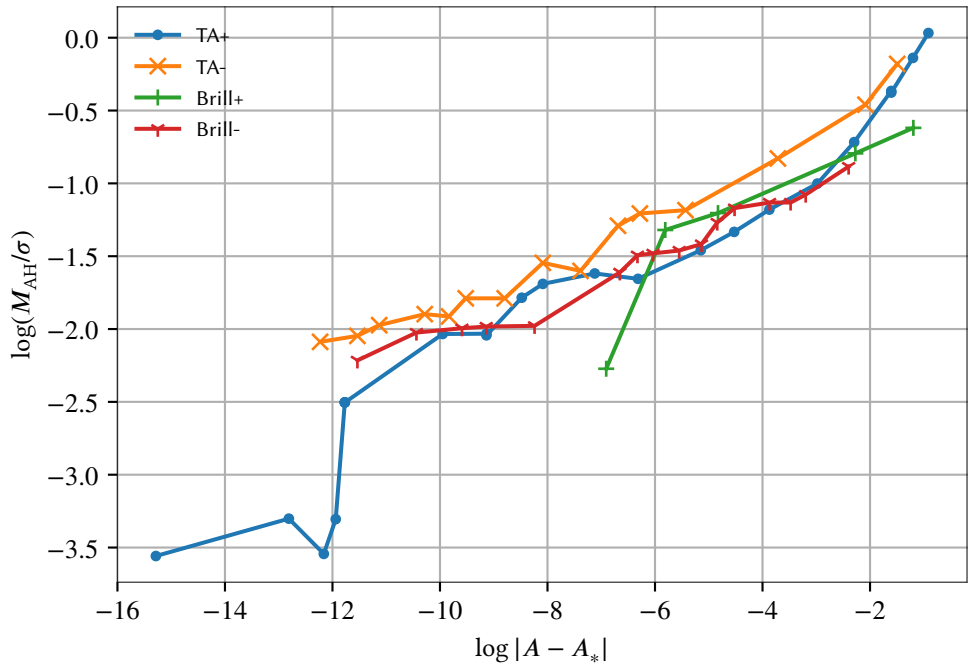

FIGURE 12.14: Apparent horizon mass scaling for different ID families. Plotted is the mass $M_{\rm AH}$ of the earliest AH located in each simulation. The sharp drop in the TA+ and Brill+ curves corresponds to the appearance of pairs of "off-center" horizons, where we show the mass of one horizon in the pair.

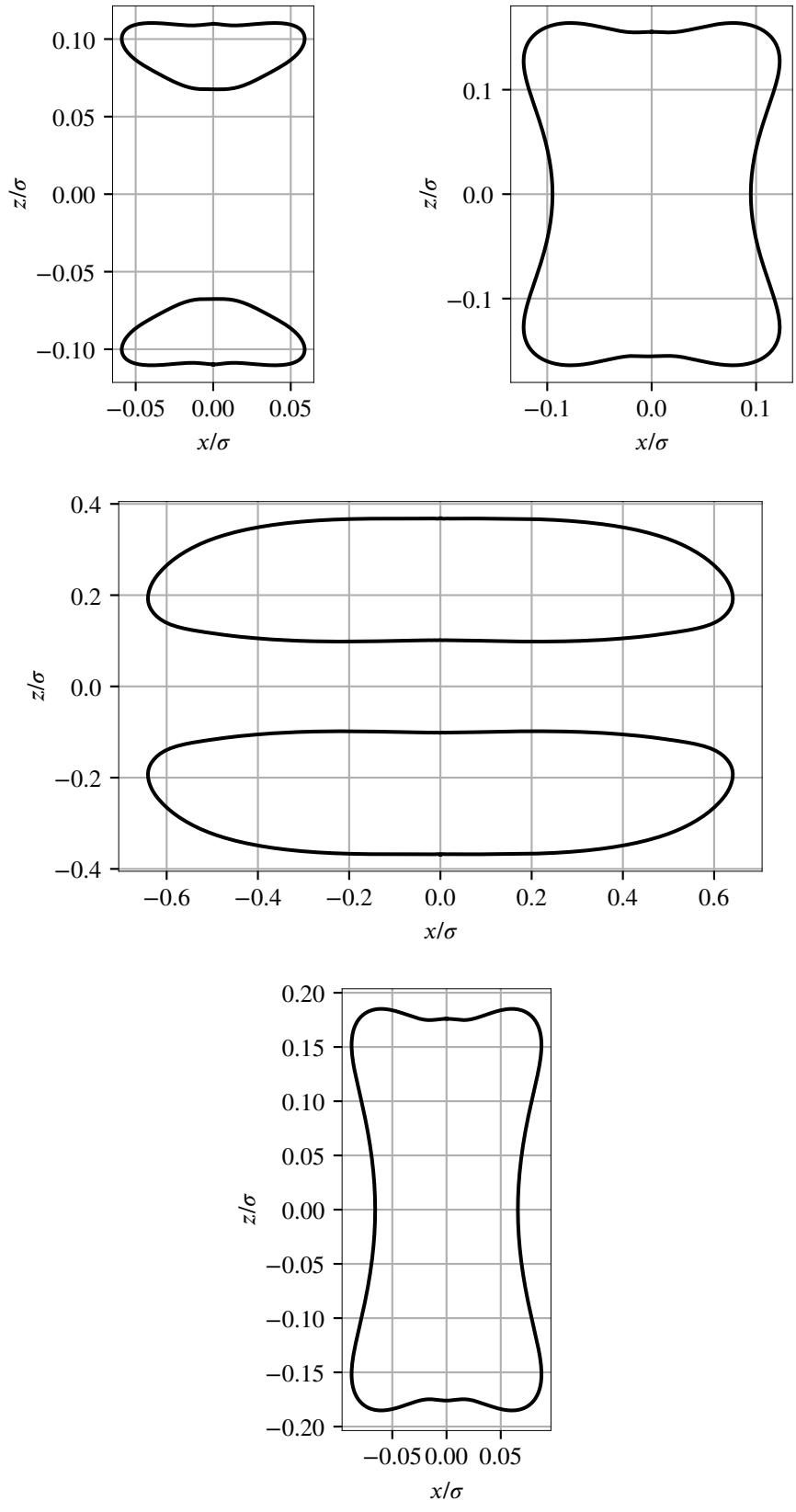

FIGURE 12.15: MOTS for different ID families. Top left: TA+ $A = \overline{1.3008075}, t = 20.75$; top right: TA- $A = -\overline{1.22436524}, t = 26$; middle: Brill+ $A = 4.698, t = 20.5$; bottom: Brill- $A = -3.5091211, t = 27$.

Summary & conclusion

We used numerical simulations to study the gravitational collapse of gravitational waves in axial symmetry. Under consideration were four one-parameter initial data families: of those two were time symmetric Brill waves, also treated in other recent research; the other two were time asymmetric waves inspired by the pioneering papers of Abrahams and Evans. All four families also had reflection symmetry with respect to the equatorial plane. We developed two new pseudospectral elliptic solvers to construct the initial data: one linear and scalar, the other non-linear and vector.

Inasmuch as possible, we tried to use tried and robust evolution techniques for black hole spacetimes. Since the standard 1+log time coordinate condition is known to break down for near-critical spacetimes, we developed a modified version, which we named quasi-maximal slicing (QMS). To implement QMS we developed a scalar linear elliptic solver based on the multigrid method, in order to match the discretization used by the evolution code.

Herein lies, in our view, the main moral of our work: sophisticated methods, arcane codes, and vast supercomputer clusters are — while certainly useful and good to have — *not* strictly necessary to make new discoveries in this field. One can obtain nontrivial new results by modest refinements of tried and true techniques and some care put into tuning the code for performance.

QMS allowed us to explore critical collapse very close to the critical point, even with our limited computational resources. We confirmed the recently reported "bifurcation" of the evolution into two collapse centers above and below the equatorial plane, resulting in two disjoint apparent horizons in the supercritical spacetimes. We also discovered that some of the families are more susceptible to coordinate singularities and are thus harder (in terms of required numerical resources) to evolve. Specifically the positive-A Brill waves turned out to be the hardest, while the positive-A time-asymmetric waves were easiest.

Focusing our attention on the curvature peaks appearing during the collapse, we discovered a form of self-similarity. The invariant ζ along a timelike wordline through each peak turned out to have the same profile after a one-parameter rescaling, justifying the label *echoes*. Furthermore, this profile was *universal* among the different initial data families we investigated. We did not, however, find any self-similarity or universality in the delays between the echoes or their scales.

These results stand in contrast to the spherically symmetric scalar-field collapse of Choptuik, which was confirmed and reproduced many times and with various matter models. They are also markedly different from the original reports of Abrahams and Evans, who claim behavior very close to that of Choptuik. As their code is not accessible ot us, we can say nothing specific about the reasons for this discrepancy. Comparison with other concurrent research gives us confidence in our results.

Limitations & constraints

The main factor currently limiting our simulations closest to the critical point is their massive demand for computational time, which grows quickly as we get closer to A_*. While we could, perhaps, push a little further, it does not seem to be an efficient use of computer time. Adaptive mesh refinement then appears to be the most important missing optimization. Beyond that, switching from BSSN to a formulation with constraint damping should be straightforward and could improve stability.

In light of the mentioned differences from spherical symmetry, it also seems important to explore a wider variety of diverse initial data families. Specifically removing the equatorial symmetry could yield interesting results.

List of abbreviations & symbols

ADM	Arnowitt-Deser-Misner
AH	Apparent horizon
AVX	Advanced vector extensions, a set of CPU instructions for vector operations
BSSN	Baumgarte-Shapiro-Shibata-Nakamura formulation of the GR evolution equations
DSS	Discrete self-symmetry
FDO	Finite difference operator
FLOP	Floating-point operation
GR	General relativity
GW	Gravitational waves
ID	Initial data
I/O	Input/output
MOTS	Marginally outer trapped surface
MPI	Message-passing interface, a parallelization method
NR	Numerical relativity
QMS	Quasi-maximal slicing
TA	Time-asymmetric wave initial data
∂_i	partial derivative
∇_μ	4-dimensional covariant derivative
D_i	3-dimensional covariant derivative
$\mathscr{L}_n$	Lie derivative along the vector field n^μ
$\det(\cdots)$	matrix determinant
$\lvert\cdots\rvert$	absolute value
$\lvert\cdots\rvert_\infty$	infinity norm (maximum of the absolute value)

$\prec$ relation operator ordering initial data from the same family by increasing mass

$\text{floor}(x)$ the floor function, returning the largest integer not larger than x

$\%$ the modulo operator, returning the remainder after integer division

List of publications

Anton Khirnov and Tomáš Ledvinka. “Slicing conditions for axisymmetric gravitational collapse of Brill waves.” In: *Classical and Quantum Gravity* 35.21 (2018-10), p. 215003. DOI: 10.1088/1361-6382/aae1bc. arXiv: 1908.06034 [gr-qc]. [35]

Tomáš Ledvinka and Anton Khirnov. “Universality of Curvature Invariants in Critical Vacuum Gravitational Collapse.” In: *Phys. Rev. Lett.* 127.1 (2021), p. 011104. DOI: 10.1103/PhysRevLett.127.011104. arXiv: 2102.09579 [gr-qc]. [38]

Appendix A

Attachments

A.1 Tables of near-critical simulations

The following tables contain global properties of our sub- and supercritical simulations for all the ID families. The subcritical tables contain the global minima and maxima of ζ, along with the coordinate times at which they appear. The supercritical tables contain the AH masses at the first two slices at which we can locate a horizon, along with the z component of the horizon's origin (defined as the average between the two locations where the AH intersects the axis).

A.1.1 TA+ waves, subcritical

A	$t_{\zeta_{\min}}$	$\zeta_{\min}$	$t_{\zeta_{\max}}$	$\zeta_{\max}$
0.7	1.344	−3.357	0.5	3.183
0.8	1.422	−4.145	0.5469	3.57
0.9	1.516	−5.102	0.6094	3.944
1	1.625	−6.296	0.6875	4.306
1.1	1.797	−7.878	0.8125	4.666
1.2	2.047	−10.15	0.9844	5.035
1.3	2.516	−14.18	3.188	6.36
1.32	2.672	−15.56	3.375	7.404
1.34	2.906	−17.51	3.672	9.053
1.36	3.406	−21.57	4.344	13.27
$\overline{1.36}$	4.047	−26.64	5.328	20.44
$\overline{1.34}$	4.969	−33.57	6.828	35.81
$\overline{1.32}$	5.688	−38.86	7.922	54.87
$\overline{1.315}$	5.875	−40.18	8.188	60.74
$\overline{1.31}$	10.42	−50.36	8.469	67.19
$\overline{1.305}$	11.03	−90.46	8.75	74.01
$\overline{1.3025}$	11.34	−130.3	8.891	77.93
$\overline{1.302}$	11.42	−141.1	13.34	82.72
$\overline{1.3015}$	11.48	−152.9	13.5	96.42
$\overline{1.30124}$	15.34	−271.5	15.59	153.6
$\overline{1.30114}$	15.42	−373.6	15.68	218.9
$\overline{1.30107}$	15.48	−465.9	15.77	282.3
$\overline{1.301}$	15.55	−588	15.88	367.8
$\overline{1.3009}$	15.66	−851.9	16.09	544
$\overline{1.3008944}$	15.66	−877.6	16.11	556.2
$\overline{1.300876}$	15.69	−932.9	16.17	599.1
$\overline{1.30085}$	15.72	−1041	16.27	666.8
$\overline{1.3008389}$	15.73	−1088	16.31	698.1
$\overline{1.30083}$	18.44	−1633	18.58	1181
$\overline{1.300825}$	18.48	−2329	18.67	1697
$\overline{1.30082}$	18.55	−3387	18.78	2490
$\overline{1.3008176}$	18.59	−4053	18.84	3021
$\overline{1.3008152}$	18.64	−4927	18.94	3700
$\overline{1.300814}$	18.66	−5381	18.97	4025
$\overline{1.300812}$	18.72	−6450	19.08	4792
$\overline{1.300811}$	18.75	−7025	19.11	5238
$\overline{1.30081}$	19.62	−9086	19.16	5768
$\overline{1.30080938}$	19.69	$-1.644 \cdot 10^4$	19.73	8854
$\overline{1.30080875}$	19.75	$-3.165 \cdot 10^4$	19.8	$1.744 \cdot 10^4$
$\overline{1.30080859}$	19.77	$-4.027 \cdot 10^4$	19.81	$2.071 \cdot 10^4$
$\overline{1.30080843}$	19.78	$-4.546 \cdot 10^4$	19.84	$2.527 \cdot 10^4$
$\overline{1.30080828}$	19.8	$-4.615 \cdot 10^4$	19.86	$3.036 \cdot 10^4$
$\overline{1.30080812}$	19.83	$-7.342 \cdot 10^4$	19.89	$3.782 \cdot 10^4$
$\overline{1.30080796}$	19.85	$-9.126 \cdot 10^4$	19.92	$4.67 \cdot 10^4$

A.1.2 TA+ waves, supercritical

A	t_1	O_1^z	M_{AH_1}	t_2	O_2^z	M_{AH}^2
$\overline{1.3008075}$	20.75	0.0888	0.0285			
$\overline{1.300805}$	20.25	0.0832	0.0381	20.5	0.0823	0.0394
$\overline{1.3008025}$	20.25	0.0758	0.0395	20.5	0	0.0913
$\overline{1.3008012}$	20	0.0797	0.0367	20.25	0	0.0894
$\overline{1.3008}$	20	0	0.0818	20.25	0	0.0943
$\overline{1.30076}$	19	0	0.131	19.5	0	0.137
$\overline{1.3007}$	17.75	0	0.13	18	0	0.134
$\overline{1.3006}$	17	0	0.168	17.25	0	0.174
$\overline{1.3005}$	16.5	0	0.172	16.75	0	0.181
$\overline{1.3}$	15.75	0	0.198	16	0	0.203
$\overline{1.299}$	14.5	0	0.191	15	0	0.207
$\overline{1.295}$	13	0	0.232	13.25	0	0.243
$\overline{1.29}$	12.5	0	0.264	13	0	0.27
$\overline{1.28}$	12.5	0	0.307	13	0	0.316
$\overline{1.25}$	11	0	0.367	11.5	0	0.377
$\overline{1.2}$	10.5	0	0.488	10.75	0	0.508
$\overline{1.1}$	9.75	0	0.686	10	0	0.715
$\overline{1}$	10.25	0	0.871	10.5	0	0.893
$\overline{0.9}$	10.5	0	1.03	10.75	0	1.06

A.1.3 TA- waves, subcritical

A	$t_{\zeta_{\min}}$	$\zeta_{\min}$	$t_{\zeta_{\max}}$	$\zeta_{\max}$
-1.1	0.5156	-6.888	1.547	3.171
-1.2	0.5938	-7.672	1.734	3.583
-1.3	0.7188	-8.491	2.125	4.177
-1.36	0.9531	-8.998	2.891	5.018
$\overline{-1.36}$	1.094	-8.986	3.422	5.479
$\overline{-1.34}$	6.234	-12.25	4.094	5.971
$\overline{-1.32}$	6.969	-17.4	7.562	7.424
$\overline{-1.3}$	7.688	-24.5	8.281	11.79
$\overline{-1.28}$	8.438	-34.9	9.062	18.7
$\overline{-1.26}$	9.234	-51.37	9.984	30.16
$\overline{-1.24}$	10.14	-79.31	11.33	50.19
$\overline{-1.235}$	10.41	-89.26	11.82	57.35
$\overline{-1.23}$	10.69	-100.9	12.46	65.67
$\overline{-1.2275}$	10.88	-107.1	12.88	71.6
$\overline{-1.2263}$	17.16	-123.6	17.77	86.55
$\overline{-1.2255}$	17.38	-184.9	18.21	135
$\overline{-1.225}$	17.56	-241.3	18.62	180.8
$\overline{-1.2246875}$	17.69	-287.9	18.97	218.9
$\overline{-1.22453125}$	22.05	-380.8	22.31	272.1
$\overline{-1.224494}$	22.13	-556.1	22.44	404.6
$\overline{-1.22445313}$	22.2	-862.3	22.61	636.5
$\overline{-1.224425}$	22.32	-1189	22.85	882.7
$\overline{-1.22441}$	22.4	-1420	23.02	1059
$\overline{-1.22439}$	22.53	-1816	23.88	1363
$\overline{-1.224375}$	24.95	-3800	23.95	1668

A.1.4 TA- waves, supercritical

A	t_1	O_1^z	M_{AH_1}	t_2	O_2^z	M_{AH}^2
$\overline{-1.22436524}$	26	0	0.124	26.5	0	0.129
$\overline{-1.22436035}$	25.5	0	0.129	26	0	0.135
$\overline{-1.22435547}$	25.5	0	0.139	26	0	0.143
$\overline{-1.22433594}$	25	0	0.15	25.5	0	0.161
$\overline{-1.2243164}$	24	0	0.148	24.5	0	0.156
$\overline{-1.22429687}$	24	0	0.167	24.5	0	0.17
$\overline{-1.22421875}$	23	0	0.167	23.5	0	0.191
$\overline{-1.2240625}$	22.5	0	0.213	23	0	0.219
$\overline{-1.22375}$	21.5	0	0.202	22	0	0.242
$\overline{-1.223125}$	20	0	0.275	20.5	0	0.284
$\overline{-1.2225}$	19.5	0	0.299	20	0	0.307
$\overline{-1.22}$	17.75	0	0.306	18	0	0.323
$\overline{-1.2}$	14.75	0	0.437	15	0	0.45
$\overline{-1.1}$	12.5	0	0.631	12.75	0	0.658
$\overline{-1.0}$	10	0	0.835	10.25	0	0.843

A.1.5 Brill+ waves, subcritical

A	$t_{\zeta_{\mathrm{min}}}$	ζ_{min}	$t_{\zeta_{\mathrm{max}}}$	ζ_{max}
3.7	1.875	−9.08	0	22.71
4.1	2.344	−9.855	0	25.55
4.4	8.844	−17.59	0	27.59
4.53	9.625	−78.38	0	28.43
4.6	10.19	−231.1	10.34	61.17
4.65	10.72	−611.6	10.84	174.1
4.67	10.98	−968.5	11.11	285.2
4.682	11.17	−1308	11.3	394.2
4.685	11.16	−1384	11.28	428.8
4.686	11.17	−1448	11.3	441.6
4.689	11.22	−1566	11.36	480.6
4.69	11.34	−1616	11.47	495.8
4.694	11.42	−1818	11.55	557.5
4.695	11.44	−1859	11.58	575.7
4.696	11.08	−1926	11.2	592.5

A.1.6 Brill+ waves, supercritical

A	t_1	O_1^z	M_{AH_1}	t_2	O_2^z	M_{AH}^2
4.698	20	0.226	0.0987	20.25	0.228	0.103
4.7	19	0	0.267	19.25	0	0.271
4.705	16.75	0	0.3			
4.8	12.25	0	0.452	12.5	0	0.466
5	8	0	0.528	8.125	0	0.523

A.1.7 Brill- waves, subcritical

A	$t_{\zeta_{\min}}$	$\zeta_{\min}$	$t_{\zeta_{\max}}$	$\zeta_{\max}$
−2.5	3.703	−9.497	2.656	5.057
−3	5.391	−24.3	5.797	8.806
−3.23	6.547	−46.37	6.969	20.2
−3.36	7.406	−75.89	8	36.09
−3.43	8.016	−104	8.969	51.86
−3.469	8.453	−126.9	9.922	64.74
−3.487	8.703	−139.8	10.58	72.11
−3.495	8.812	−146.1	10.95	75.75
−3.5	10.05	−150.5	17.2	110.7
−3.5025	16.2	−174.3	17.14	142.9
−3.505	16.41	−221.8	17.52	186.8
−3.5075	16.64	−286.2	17.97	249.2
−3.508125	16.7	−305.6	18.09	268.9
−3.50815	16.7	−306.4	18.11	269.4
−3.50835	21.06	−315.7	18.14	275.8
−3.5085	21.17	−409.3	21.55	326.5
−3.50863	21.28	−520	21.7	418.1
−3.5087	21.34	−591	21.8	477.9
−3.50875	21.41	−643	21.92	524.8
−3.50890625	21.42	−868.8	22.14	716.1
−3.5090625	23.52	−1260	22.53	995.2
−3.50910156	23.67	−2563	25.34	1407

A.1.8 Brill- waves, supercritical

A	t_1	O_1^z	M_{AH_1}	t_2	O_2^z	M^2_{AH}
−3.5091211	27	0	0.109	27.5	0	0.115
−3.50914063	25.5	0	0.132	26	0	0.138
−3.50917969	24.5	0	0.136	25	0	0.145
−3.50921875	24.25	0	0.138	24.5	0	0.149
−3.509375	23.25	0	0.138	23.5	0	0.148
−3.5104	21.5	0	0.2	22	0	0.21
−3.5109	21.5	0	0.225	22	0	0.228
−3.5115	21	0	0.227	21.5	0	0.242
−3.513	20.5	0	0.232	21	0	0.262
−3.515	19	0	0.242	19.5	0	0.271
−3.517	18.5	0	0.281	19	0	0.299
−3.52	18.5	0	0.31	19	0	0.318
−3.53	17.5	0	0.323	17.75	0	0.325
−3.54	17	0	0.323	17.25	0	0.341
−3.55	16.88	0	0.341	17	0	0.35
−3.6	15.38	0	0.413	15.5	0	0.417

A.2 Source codes

The data package distributed with this thesis contains the source codes necessary to reproduce our simulations, along with the corresponding parameter files. The README file in the root directory of the package describes the contents in detail. The package may also be obtained through the GIT protocol from `git://git.khirnov.net/phd_thesis_sources`.

A.3 Published paper: Slicing conditions for axisymmetric gravitational collapse of Brill waves

This section contains the author version of our paper "Slicing conditions for axisymmetric gravitational collapse of Brill waves" [35], published in *Classical and Quantum Gravity* (© 2018 IOP Publishing Ltd).

Note that in this paper we describe our initial QMS implementation based on a pseudospectral method. Afterwards we concluded that the code does not scale well enough to be viable for spacetimes very close to the critical point and replaced it with a new implementation based on the multigrid method. This new implementation is described in this thesis (Chapters 8 and 9).

Slicing conditions for axisymmetric gravitational collapse of Brill waves

Anton Khirnov, Tomáš Ledvinka

Institute of Theoretical Physics, Faculty of Mathematics and Physics, Charles University, V Holešovičkách 2, Prague, 18000, Czech Republic

E-mail: anton@khirnov.net, Tomas.Ledvinka@mff.cuni.cz

Abstract. In numerical relativity, spacetimes involving compact strongly gravitating objects are constructed as numerical solutions of Einstein's equations. Success of such a process strongly depends on the availability of appropriate coordinates, which are typically constructed dynamically. A very robust coordinate choice is a so-called moving puncture gauge, commonly used for numerical simulations of black hole spacetimes. Nevertheless it is known to fail for evolving near-critical Brill wave data. We construct a new "quasi-maximal" slicing condition and demonstrate that it exhibits better behavior for such data. This condition is based on the 1+log slicing with an additional source term derived from maximal slicing. It is relatively simple to implement in existing moving puncture codes and computationally inexpensive. We also illustrate the properties of constructed spacetimes based on gauge-independent quantities in compactified spacetime diagrams. These invariants are also used to show how created black holes settle down to a Schwarzschild black hole.

Keywords: Numerical relativity, gravitational collapse, axial symmetry

1. Introduction

During the last decade numerical relativity became tremendously successful at simulating many astrophysically important processes in which general relativity plays a crucial role. A significant portion of this progress can be attributed to the discovery of coordinates suitable for such wildly dynamic spacetime geometries. Present codes can handle mergers both of purely vacuum black hole binaries and neutron star binaries (followed by a collapse). It may thus seem surprising that a certain theoretically important situation — gravitational wave packet collapse — still defies these codes.

Numerical treatment of gravitational waves collapsing to form a black hole dates back to the early age of numerical relativity. Eppley in his pioneering work [1] successfully constructed what is known as the Brill waves [2] — a family of vacuum initial data describing axisymmetric wave packets at the moment of time symmetry — and located apparent horizons in them. Another breakthrough was the paper of Abrahams and Evans [3] which used another family of axisymmetric data called the Teukolsky waves [4]. They found evidence of critical behavior — discrete self-similarity and power-law scaling of various quantities close to the threshold of black hole formation — analogous to the results of Choptuik for a scalar field [5, 6].

Since then, as computer performance and numerical methods improved, several attempts were made to investigate the nonlinear regime of the Brill data. Alcubierre

et al. [7] used a combination of the BSSN evolution system and maximal slicing to put rough bounds on the critical value of the amplitude parameter. That result was confirmed in [8] using a mixed elliptic-hyperbolic reduction of the evolution equations, again with maximal slicing. Ultimately, insufficient performance of contemporary computers constrained what could be achieved with these attempts.

As the moving puncture gauge [9, 10] showed its strength for black hole mergers, Hilditch et al. [11] attempted to use it also for gravitational wave collapse. One of their discoveries was the pathological behavior of this gauge for near-critical Brill waves, producing what they conjectured to be coordinate singularities. Recently [12, 13] they managed to avoid these problems with a new pseudospectral code using a generalized harmonic formulation.

In this paper, we follow up on the work of [11], using the same initial data and similar numerical techniques (although a completely different code). We analyze the way in which the moving puncture gauge — a combination of the "1+log" slicing and the "Γ-driver" shift condition — breaks, and modify the slicing condition. Our modification takes the form of an extra source term pushing the constant-time hypersurfaces closer towards maximal slicing, which is known to guarantee smooth solutions and has shown promising results in earlier work [7, 8, 14]. The result is a new slicing condition — which we call "quasi-maximal slicing" — that no longer exhibits the problems reported in [11]. While quasi-maximal slicing involves solving an elliptic equation, solver accuracy only affects the distance from the preferred gauge and does not give rise to ADM constraint violations — in contrast to the usual implementations of maximal slicing.

Using this slicing, we illustrate features of gravitational wave collapse that are present even farther away from the critical amplitude, such as formation of trapped surfaces and the event horizon, and construct compactified diagrams of considered spacetimes. These were previously not available, since earlier techniques are not able to handle long-term simulations easily.

This paper is laid out as follows. In Section 2 we briefly summarize the continuum equations on which our numerics is based. In Section 3 we introduce our modified slicing condition. Section 4 describes the key points of the numerical codes used and why we trust their output. Finally in Section 5 we present the results for long-term evolution of the considered class of initial data. We use geometrized units $G = c = 1$ in this paper.

2. Field equations

2.1. Brill wave initial data

The Brill waves [2] are a family of vacuum axially symmetric initial data defined by the spatial metric in cylindrical coordinates $\{\rho, z, \varphi\}$

$$\gamma_{ij} = \psi^4 \left[e^{2q} \left(\mathrm{d}\rho^2 + \mathrm{d}z^2 \right) + \rho^2 \mathrm{d}\varphi^2 \right], \tag{1}$$

and the condition of time symmetry, which implies initially vanishing extrinsic curvature. Here ψ is the conformal factor and $q = q(\rho, z)$ is the so-called "seed function", which needs to have certain regularity and decay properties [2], but can otherwise be chosen arbitrarily. For ease of comparison, we use the same form of q as was used in [11], specifically

$$q(\rho, z) = A \left(\frac{\rho}{\sigma} \right)^2 e^{-(\rho^2 + z^2)/\sigma^2}. \tag{2}$$

The parameter A determines the amplitude of the waves, with $A = 0$ being flat space. We only consider non-negative values of A in this paper, though it is worth noting that $A < 0$ also produces valid initial data. The critical point A^* is the smallest value of A for which a gravitational singularity is formed. For the above choice of q it is bracketed by our simulations to $A^* \in [4.69, 4.7]$, which is compatible with tighter bounds given in [13].

The scale σ (with dimension of length) fixes the units of all physical quantities. In our code we set $\sigma = 1$, so all numerical values of time, length or mass are assumed to be in units of σ. This allows us to compare our results directly with [11], where the same choice is made. For quantities that do not clearly have dimension of length we print the appropriate power of σ explicitly.

Due to time symmetry the momentum constraints are trivial, so to construct the data we only need to solve the Hamiltonian constraint. It reduces to

$$\Delta\psi + \frac{1}{4}\left(\partial_{\rho\rho}q + \partial_{zz}q\right)\psi = 0, \tag{3}$$

a linear elliptic equation for the conformal factor ψ, where Δ is the flat space Laplacian. Our code for solving this equation is described in Section 4.1.

2.2. Evolution system

We follow the standard 3+1 splitting procedure, with the 4-dimensional metric $g_{\mu\nu}$ decomposed into the spatial metric γ_{ij}, the lapse α and the shift vector β^i

$$\mathrm{d}s^2 = \left(-\alpha^2 + \beta_i\beta^i\right)\mathrm{d}t^2 + 2\beta_i \mathrm{d}t\mathrm{d}x^i + \gamma_{ij}\mathrm{d}x^i\mathrm{d}x^j. \tag{4}$$

The vacuum Einstein equations can then be written as a set of evolution equations

$$\left(\partial_t - \mathcal{L}_\beta\right)\gamma_{ij} = -2\alpha K_{ij}, \tag{5a}$$

$$\left(\partial_t - \mathcal{L}_\beta\right)K_{ij} = -\mathrm{D}_i\mathrm{D}_j\alpha + \alpha\left[R_{ij} + KK_{ij} - 2K_{ik}K^k_j\right], \tag{5b}$$

and a set of constraints

$${}^{(3)}R + K^2 - K_{ij}K^{ij} = 0, \tag{6a}$$

$$\mathrm{D}_j K^{ij} - \mathrm{D}^i K = 0. \tag{6b}$$

Here K_{ij}, defined by (5a), is the extrinsic curvature of the spatial slices, $\mathcal{L}_\beta$ is the Lie derivative along β^i, D_i, R_{ij} and ${}^{(3)}R$ are respectively the covariant derivative, the Ricci tensor and the scalar curvature associated with γ_{ij}.

Introducing new evolved variables

$$\varphi = \det\left(\gamma_{ij}\right)^{-\frac{1}{6}}, \tag{7a}$$

$$K = \gamma_{ij}K^{ij}, \tag{7b}$$

$$\bar{\gamma}_{ij} = \varphi^2\gamma_{ij}, \tag{7c}$$

$$\bar{A}_{ij} = \varphi^2\left(K_{ij} - \frac{1}{3}K\gamma_{ij}\right), \tag{7d}$$

$$\bar{\Gamma}^i = \bar{\gamma}^{jk}\bar{\Gamma}^i_{jk}, \tag{7e}$$

where $\bar{\Gamma}^i_{jk}$ are the Christoffel symbols associated with $\bar{\gamma}_{ij}$, and using the constraints, we can write the evolution equations as

$$\left(\partial_t - \mathcal{L}_\beta\right)\varphi = \frac{1}{3}\varphi\alpha K, \tag{8a}$$

$$\left(\partial_t - \mathcal{L}_\beta\right)\bar{\gamma}_{ij} = -2\alpha\bar{A}_{ij}, \tag{8b}$$

$$(\partial_t - \mathcal{L}_\beta)\, K \;= -\mathrm{D}^i \mathrm{D}_i \alpha + \alpha \left(A_{ij} A^{ij} + \frac{1}{3} K^2 \right), \tag{8c}$$

$$(\partial_t - \mathcal{L}_\beta)\, \bar{A}_{ij} = \varphi^2 \left[-\mathrm{D}_i \mathrm{D}_j \alpha + \alpha R_{ij} \right]^{\mathrm{TF}} + \alpha \left(K \bar{A}_{ij} - 2 \bar{A}_{ik} \bar{A}^k_j \right), \tag{8d}$$

$$\begin{aligned}(\partial_t - \mathcal{L}_\beta)\, \bar{\Gamma}^i \;&= \bar{\gamma}^{jk} \partial_j \partial_k \beta^i + \frac{1}{3} \bar{\gamma}^{ij} \partial_j \partial_k \beta^k - 2 \bar{A}^{ij} \partial_j \alpha \\ &\quad + 2\alpha \left(\bar{\Gamma}^i_{jk} \bar{A}^{jk} + 6 \bar{A}^{ij} \partial_j \varphi - \frac{2}{3} \bar{\gamma}^{ij} \partial_j K \right),\end{aligned} \tag{8e}$$

where TF denotes the trace-free part of the bracketed expression. The conformally related metric $\bar{\gamma}_{ij}$ and its inverse $\bar{\gamma}^{ij}$ are used to raise and lower indices on all the quantities with a bar. Terms involving D_i, R_{ij} and ${}^{(3)}R$ are now assumed to be computed from $\bar{\gamma}_{ij}$ and φ and their derivatives in a straightforward manner.

This is known as the BSSN [15, 16] formulation, which (given appropriate gauge choice) is known to be strongly hyperbolic, and has a solid track record in black hole simulations.

One of the slicing conditions we consider is the maximal slicing, defined by requiring the volume elements associated with normal observers to remain constant. This implies that the trace of the extrinsic curvature K is identically zero at all times. Provided that the initial slice satisfies $K = 0$, equation (8*c*) immediately gives us

$$\partial_t K = -\mathrm{D}^i \mathrm{D}_i \alpha + K_{ij} K^{ij} \alpha = 0, \tag{9}$$

a linear elliptic equation for the lapse α that needs to be solved at each step of the evolution. For its many desirable properties [17], this slicing has been used since early times of numerical relativity, and specifically for Brill waves e.g. in [7, 8]. Its main disadvantage is the fact that solving an elliptic equation at each time step is usually impractical in 3D and tends to be very resource-intensive even in 2D.

According to [18], maximal slicing is only well-posed when the constraint $K = 0$ is enforced, which is easily achieved with BSSN by just not evolving K.

The other common lapse choice we make use of is the 1+log slicing [9], given by a hyperbolic evolution equation for the lapse

$$(\partial_t - \mathcal{L}_\beta)\, \alpha = -2\alpha K. \tag{10}$$

This condition is intended to approximate the main advantages of the maximal slicing at a lower computational cost.

For the shift we mainly use the Γ-driver condition (original version introduced in [10], the flavor we use is from [19])

$$(\partial_t - \mathcal{L}_\beta)\, \beta^i = \xi \bar{\Gamma}^i - \eta \beta^i, \tag{11}$$

a hyperbolic equation related to elliptic minimal-distortion conditions. Here the dimensionless parameter ξ determines the velocities of the longitudinal and transversal gauge waves, while η is a damping parameter (with a dimension of inverse length) that seems to be necessary to avoid shock formation in the shift. In some cases we also set the shift to be identically zero.

Together 1+log slicing and Γ-driver shift are known as the moving puncture gauge and are widely used in numerical relativity, among other thing for black-hole spacetimes.

2.3. Invariants

To extract coordinate-independent information from our simulations we look at spacetime invariants. Since the end-state of the super-critical runs has to be a Schwarzschild black hole, the invariants' behavior in Schwarzschild is of special interest to us.

The most prominent scalars are those constructed from curvature, e.g. the Kretschmann scalar $\mathcal{K} = R_{\mu\nu\alpha\beta}R^{\mu\nu\alpha\beta}$ (where $R_{\mu\nu\alpha\beta}$ is the four-dimensional Riemann tensor). For a Schwarzschild black hole with mass M we have

$$\mathcal{K}_{\text{Schwarzschild}} = \frac{48M^2}{R^6}, \tag{12}$$

where R is the usual areal radius. This allows us to define the "Kretschmann mass" as

$$M_{\mathcal{K}}(R) = \sqrt{\frac{\mathcal{K}R^6}{48}}, \tag{13}$$

where $\mathcal{K}$ and R are evaluated at some point in the equatorial plane $z = 0$.

One practical issue with $\mathcal{K}$ is that it contains second derivatives of the evolved quantities and its stationary limit (on which the mass estimate (13) is based) has a rather fast $1/R^6$ falloff. When the evolved metric contains noise — e.g. caused by reflections on the grid refinement boundaries — the spurious terms decay less steeply and can become of comparable size to the stationary value.

Other invariants, which are simpler and easier to calculate, can be obtained from the fact that our spacetimes are axially symmetric. That implies the existence of an angular Killing vector $\eta^\mu = \frac{\partial}{\partial\varphi}^\mu$. When appropriately scaled, its norm is the circumferential radius

$$\bar{\rho}^2 = \eta_\mu\eta^\mu = \gamma_{yy}x^2, \tag{14}$$

where the second equality holds in the $y = 0$ plane. Clearly in spherical symmetry we have $\bar{\rho}^2 = R^2$ in the equatorial plane. Another related scalar we can construct is the norm of the gradient of $\bar{\rho}^2$

$$4\bar{\rho}'^2 = \left|\nabla\bar{\rho}^2\right|^2 = (\mathrm{D}_i\bar{\rho}^2)(\mathrm{D}^i\bar{\rho}^2) - 4(K_{ij}\eta^i\eta^j)^2. \tag{15}$$

Note that despite the notation, $\bar{\rho}'^2$ can be negative — e.g. in Schwarzschild the $R = \text{const.}$ surfaces are spacelike above the horizon and timelike below it.

From the above two scalars we can construct a dimensionless quantity

$$\chi = \frac{\bar{\rho}'^2}{\bar{\rho}^2}. \tag{16}$$

In our simulations we can use χ as a measure of spacetime deformation in the equatorial plane. For the Schwarzschild solution $\chi = 1 - \frac{2M}{R}\sin^2\theta$, so in super-critical spacetimes the black hole horizon will settle down to the $\chi = 0$ hypersurface. Additionally, in stationary regions we can form another mass estimate in the equatorial plane

$$M_{\bar{\rho}} = \frac{\bar{\rho}^2 - \bar{\rho}'^2}{2\bar{\rho}}. \tag{17}$$

In future studies of the critical collapse of axisymmetric gravitational waves, the quantity χ may be of interest, since for a spacetime with discrete self-similarity such a dimensionless quantity should repeat with the same range of values on each echoing period [6].

3. Modified 1+log slicing

Though maximal slicing appears to be well-behaved for near-critical Brill waves, implementing it in a numerical code entails practical difficulties — solving elliptic equations at each time step is computationally demanding and errors involved immediately lead to constraint violations. However, since the lapse is a gauge function, we do not actually insist on it being exactly the maximal one. A "good-enough" approximation that still avoids the problems of the 1+log slicing works just as well for our purposes.

In obtaining such an approximation, we start with the maximal slicing condition (9) and take its time derivative

$$0 = (\partial_t - \mathcal{L}_\beta) \left[\gamma^{ij} \mathrm{D}_i \mathrm{D}_j \alpha - K_{ij} K^{ij} \alpha\right], \tag{18}$$

which with some straightforward manipulations transforms into

$$\begin{aligned} &\gamma^{ij} \mathrm{D}_i \mathrm{D}_j \left[(\partial_t - \mathcal{L}_\beta)\alpha\right] - K_{ij} K^{ij} \left[(\partial_t - \mathcal{L}_\beta)\alpha\right] = \\ &- \left[(\partial_t - \mathcal{L}_\beta)\, \gamma^{ij}\right] \mathrm{D}_i \mathrm{D}_j \alpha + \gamma^{ij} \left(\partial_t \Gamma^k_{ij}\right) \partial_k \alpha \\ &- \left(\gamma^{ij} \mathrm{D}_i \mathrm{D}_j \beta^k\right) \mathrm{D}_k \alpha - \beta^j R^i_j \mathrm{D}_i \alpha + \alpha \left(\partial_t - \mathcal{L}_\beta\right) \left(K_{ij} K^{ij}\right). \end{aligned} \tag{19}$$

Now we replace the time derivative of the lapse with a new function W

$$W = (\partial_t - \mathcal{L}_\beta)\alpha \tag{20}$$

and use the evolution equations to replace the time derivatives of γ_{ij} and K_{ij} with purely spatial quantities known on one slice. We obtain

$$\begin{aligned} &\mathrm{D}^i \mathrm{D}_i W - K_{ij} K^{ij} W = \\ &- 2\alpha K^{ij} \mathrm{D}_i \mathrm{D}_j \alpha + \gamma^{ij} \left(\partial_t \Gamma^k_{ij}\right) \partial_k \alpha - \left(\gamma^{ij} \mathrm{D}_i \mathrm{D}_j \beta^k\right) \mathrm{D}_k \alpha \\ &- \beta^j R^i_j \mathrm{D}_i \alpha + \alpha \left(2 \dot{K}_{ij} K^{ij} + 4\alpha K^i_j K^k_i K^j_k\right), \end{aligned} \tag{21}$$

where $\dot{K}_{ij}$ is a shorthand for the right-hand side of (5*b*). This is an elliptic equation for the new function W, with the same structure as the maximal slicing condition (9), but a more complicated right-hand side.

Our starting point (18) is nothing else than the demand that the acceleration of K vanishes. So if we now solved (21) exactly at each time step and used its solution W_e to evolve the lapse according to (20), then — assuming that initially $K = 0$ and the lapse is maximal — we would get precisely the maximal slicing (leaving aside the question of stability of such a system). Of course in a numerical simulation we will not have an exact solution and good approximations to it do not come cheaply. One can hope, however, that even a rough approximation to the solution of (21) can be exploited to get closer to maximal slicing.

The basic idea is then as follows — at each time step we compute W_a, a (very approximate) solution to (21). Then we use this quantity as an extra source term in the 1+log slicing

$$(\partial_t - \mathcal{L}_\beta)\alpha = -2\alpha K + \kappa W_a, \tag{22}$$

where κ is a function of coordinate time used to switch between slicing conditions — we set it equal to one when the extra term is to be active and smoothly send it to zero when we want to recover the original 1+log slicing.

The motivation for this extra term is as follows. In the weak-field regime, the Bona-Massó class of slicing conditions yields a hyperbolic (K, α) subsystem — a scalar

sector corresponding to so-called lapse gauge waves. These exhibit the usual wave behavior and naturally drive weak-field configurations to $K = 0$ as waves (assuming their finite "energy") disperse outwards. The aim of our modification (22) is to push the slicing towards $K = 0$ even in a non-linear regime.

It is known that for PDE systems such as BSSN we are not free to modify even the gauge conditions arbitrarily — adding combinations of evolved functions and their derivatives may lead to loss of well-posedness of the system. The source term added in (22) does not modify the principal part of the PDEs linearized around flat space, so it does not affect the hyperbolicity of the PDE system. This follows from the fact that (21) has a trivial principal part — $\Delta W = 0$. Also — as described later — in our approach we do not assume that the elliptic solver should provide a solution close to the continuum limit.

Further in this paper we show numerical evidence that slicing condition (22) is not only stable for small-amplitude Brill waves, but also that for larger amplitudes the extra term acts as a driver that pushes the slicing closer to the maximal one, curing the problems of 1+log slicing. We call this condition the "quasi-maximal" slicing.

4. Numerics

4.1. Implementation

Initial data Due to axial symmetry, the equation (3) only needs to be solved in two dimensions. We use a pseudo-spectral method, writing the conformal factor as a series

$$\psi\left(x^0, x^1\right) = 1 + \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} C_{kl} B_k^0\left(x^0\right) B_l^1\left(x^1\right). \tag{23}$$

Using (23) in (3) and demanding that the equation be satisfied exactly at NM collocation points gives us NM linear equations for the coefficients C_{kl}. We solve those using LU decomposition, which then allows us to reconstruct ψ at arbitrary points through (23).

Specifically, we choose polar coordinates $\left\{x^0, x^1\right\} = \{r, \theta\}$ and basis functions [20]

$$B_k^0(r) = \mathrm{SB}_{2k}(r) = \sin\left((2k+1)\operatorname{arccot}\frac{r}{L}\right), \tag{24a}$$

$$B_l^1(\theta) = \cos(2l\theta). \tag{24b}$$

Here L is a constant that determines the compactification scale and needs to be tuned to the problem being solved. In our simulations we always use empirically determined value $L = 3$ (as usual, in the units of σ) for the initial data.

Functions $\mathrm{SB}_k(x)$ decay as $\frac{1}{x}$ towards infinity, which is the behavior we expect from ψ. Taken together this combination of basis functions automatically satisfies the symmetry and decay properties of the solution, so we need not impose any explicit boundary conditions. One property this basis set does not guarantee is regularity of the solution at origin — that would further require that the sub-series corresponding to each B_l^1 has a $2l$-th order root at $r = 0$ (this is sometimes called the parity theorem [21]). However it turns out in practice that it is not necessary to impose these conditions explicitly and the solution is regular anyway.

We implemented the above procedure as a stand-alone library written in C/x86 assembler, which is then called from the evolution code to construct the initial data. The linear system is solved using LAPACK [22].

Evolution Our evolution code is based on the Einstein Toolkit [23, 24], which bundles together the Cactus framework [25, 26] and other packages relevant to numerical relativity. The Carpet [27] code provides Berger-Oliger-style [28] fixed mesh refinement. The BSSN evolution equations are implemented by the McLachlan project [29, 30, 31].

To exploit the axial symmetry of our systems, we use Cartesian coordinates $\{x, y, z\}$ with the analytic Cartoon method, as described in detail in [12]. Its main idea is replacing the y derivatives of arbitrary tensors with analytically derived combinations of x and z derivatives. Compared to the "classical" Cartoon method [32] implemented in the Einstein Toolkit, which uses a thin layer of points in the y-direction filled by interpolation, the analytic method has significantly reduced memory requirements and is much faster. It should also be more accurate, though this was not our main motivation for using it. As our initial data is also symmetric with respect to reflection through the $z = 0$ plane, we evolve just the $z \geq 0$ region.

The simulation domain is thus composed of a set of nested squares. Each one encloses an equidistant grid, with the step size Δx halving per each nesting (refinement) level. We use the method of lines for time evolution, with 4th order Runge-Kutta as the time integrator. Spatial derivatives in the BSSN equations are approximated with 8th order finite differences — upwind in the advection terms, centered elsewhere. Kreiss-Oliger dissipation of the 9th order is applied to damp high-frequency noise. When (quasi-)maximal slicing is used, there is also a separate pseudo-spectral grid present, as described later.

Maximal slicing As the maximal slicing condition (9) is a linear elliptic PDE, same as the equation (3) we solve for initial data construction, we reuse the basic methods (and some of the code) from the initial data solver, with a number of changes.

The most obvious of those stem from the fact that the equation is not stand-alone, but a part of a system — it needs to be solved at each intermediate step of the time integrator and requires the evolved metric variables as input. Since the evolution happens on the equidistant finite difference grid, we need to interpolate the metric variables onto the pseudo-spectral grid. Additionally, due to mesh refinement, not all data will always be available at the required time level, which means we have to interpolate in time as well. We use fourth order Lagrange interpolation in space and linear in time.

Another change concerns the coordinates. When using polar coordinates we observed that the regularity conditions at origin stop being satisfied automatically during evolution and the lapse develops a discontinuity at $r = 0$. We believe this happens because the data injected from the finite difference grid is not sufficiently smooth for the spectral solver. To avoid this issue we use Cartesian coordinates $\{x, z\}$ and $B_k^0 = B_k^1 = \mathrm{SB}_{2k}$. The scaling constant L is chosen in such a way that the outermost collocation point remains causally disconnected from the outer boundary during evolution.

Since the equation is now solved many times, performance of the solver becomes important. For this reason we augment LU decomposition with the BiCGSTAB iterative method [33]. Since the pseudo-spectral matrix changes slowly with time, we run the LU decomposition once per S solves and use it to compute the exact inverse. This inverse is then used as the preconditioner for the next S BiCGSTAB iterative solves. Typically we take $S = 1024$, then on average around 5 iterations are needed for the solution to converge to within 10^{-15} absolute error, which takes about

$20\times$ less time to compute than the LU decomposition.

A significant weakness of this approach is that it involves solving a dense linear system of N^2 equations, where N is the number of the basis functions in one dimension. Our memory requirements thus grow as N^4, so the maximum practically achievable number of collocation points is much lower than the number of points on the finite differencing grid.

With our chosen basis, the collocation points are closely clustered near the origin and become very sparse further out. So the way lack of resolution manifests in practice is that as the initial wave pulse travels away from the origin, it enters the area where the distance between collocation points is too large for the waves to be resolved. The higher frequencies then become invisible to the pseudospectral solver, strongly reducing the accuracy of the solution. Since we enforce $K = 0$ for maximal slicing, the errors manifest as large violations of the ADM constraints. Similar issues have also been reported in [8].

We might be able to mitigate this issue somewhat with a better choice of the basis, or even resolve it fully through some sort of a multi-domain method. However that would require a significant amount of additional programming and, since maximal slicing is only an intermediate step of our work, we do not pursue such efforts any further.

Quasi-maximal slicing The obvious approach to computing W_a is reusing the method we used for maximal slicing — solve on each time step, using time interpolation to fill in missing data points. That, however, consumes a large amount of computational time and, as previously mentioned, we have considerable freedom to sacrifice accuracy for efficiency in solving (21).

So we use another method, which requires far less resources and turns out to work well in practice. At its core is the pseudo-spectral elliptic solver we used for maximal slicing, but it is not run at each time step. It works as follows:

- We pick the finest refinement level L that encloses all the collocation points used by the pseudo-spectral solver.
- At simulation time t^L_n, before we start the recursive Berger-Oliger evolution to time t^L_{n+1}, we have the final BSSN quantities for level L and all the finer levels. After interpolating this data onto the pseudo-spectral grid, we execute the pseudo-spectral solver to solve (21), obtaining a set of spectral coefficients C_n.
- During the following recursive time stepping we need the values of W_a at times between t^L_n and t^L_{n+1}, i.e. in the future from the solutions we already have. So we perform *linear extrapolation* from the last two solutions — C_{n-1} and C_n — to predict the coefficients at time t^L_{n+1}. We denote this prediction C'_{n+1}.
- To evaluate W_a at times in the interval $\left(t^L_n, t^L_{n+1}\right)$, we *linearly interpolate* between the two most recent sets of predicted coefficients — C'_n and C'_{n+1}. This ensures that W_a is piecewise linear in time.

This process is sketched in Figure 1.

Finally, we discovered empirically that simply evaluating W_a as a sum of the spectral series tends to introduce strong high-frequency components to the finite difference grid. It seems that those consequently propagate to α and from there to the other evolved functions and due to non-linearities get aliased as spurious low-frequency noise, manifesting as ADM constraint violations.

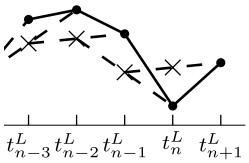

Figure 1. A diagram of the interpolation-extrapolation scheme for evaluating W_a. Crosses are the C_ns — the solutions obtained from the pseudo-spectral solver. Circles are the C'_ns — the predicted coefficients obtained from C_ns by extrapolation (dashed lines). Solid lines denote the evaluation of $W_a(t)$ by linear interpolation.

We deal with this issue by applying a low-pass filter to the basis functions. In our current multi-grid setup, the Nyquist frequency at a given grid point depends on the distance from origin r. For each basis function $B_k(x)$ we estimate its "base" frequency from the location of its first zero x^0_k, which implies it is no longer accurately resolved at the radius r^0_k where $\Delta x \approx x^0_k$. Therefore when evaluating W_a, we damp each basis function such that [34]

$$B'_k = B_k \exp\left(-J\left(\frac{r}{r^0_k}\right)^d\right). \tag{25}$$

$J = 36$ ensures that for $r = r_0$ the exponential evaluates to the double-precision ε, while the empirically determined choice of $d = 6$ makes sure the transition area is never smaller than the roughest step size (the results do not seem to depend on the precise value of d very strongly). Altogether the filter ensures that W_a remains non-zero only close to the origin and does not affect the grid further out, where it is not needed.

At late simulation times we usually switch from quasi-maximal back to 1+log slicing, since the added term no longer seems to be necessary for stability and for super-critical spacetimes the spectral solver can be ill-behaved in the presence of punctures. The switch is implemented by using a damping function similar to (25) for κ in (22), specifically

$$\kappa = \min\left[\exp\left(-J\left(t - t_0\right)^d\right), 1\right], \tag{26}$$

where t_0 is the time at which the switch starts, determined empirically for a given simulation. The exact choice of parameters did not make a qualitative difference in our runs, t_0 is rougly the time when the apparent horizon forms for super-critical runs or the time when the largest peak in $\mathcal{K}$ forms and dissolves.

4.2. Code validation

Super-critical Brill waves in moving puncture gauge We validate our moving puncture gauge setup by reproducing the slicing failure for $A = 5$ Brill waves from [11]. We use 6 levels of mesh refinement, with 2048×2048 grid points on each, the coarsest level using a spatial step size of $\Delta x = \frac{1}{4}$. The Γ-driver parameters are set to $\xi = 1; \eta = 11.4/\sigma$. Our results are fully consistent with the cited paper.

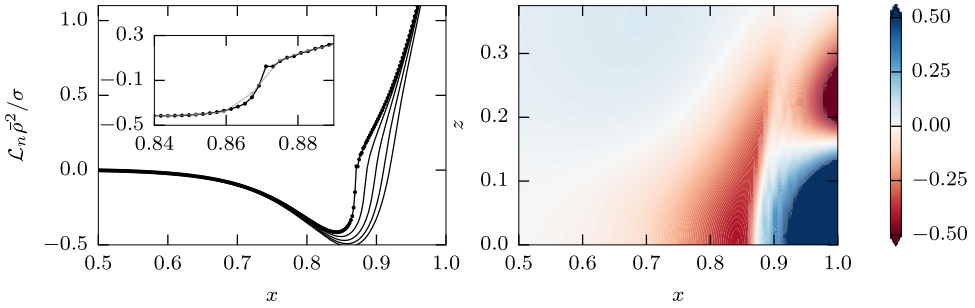

Figure 2. Failure of the 1+log slicing for $A = 5$ Brill waves. The shift is set to zero. Left: plotted is the derivative of $\bar{\rho}^2$ along the normal to the spatial surfaces on the x-axis at times $\{4.625, 4.6875, 4.75, 4.8125, 4.875\}$ (bottom to top). The inset compares the data on the $t = 4.875$ slice for runs with 2048 (black) and 512 (gray) grid points. Circles/crosses indicate the positions of the grid points. Right: the same quantity in the $x - z$ plane at time $t = 4.875$.

The slicing issues are illustrated in Fig. 2. The circumferential radius $\bar{\rho}$ should be a smooth spacetime invariant, so for a regular hypersurface $t = \text{const.}$ the derivative of $\bar{\rho}$ along the unit normal n^μ should also be smooth. In the left panel we show that at times $t \lesssim 4.9$ — just before the simulations fail (independently of the spatial resolution) — this function develops a feature resembling a discontinuity. As can be seen in the inset, the gradient becomes higher with increasing number of grid points. When testing with a number of different resolutions from 384 to 2048 grid points, we found that the maximum of the gradient of the plotted function exhibits non-convergent behavior consistent with computing finite differences across a step function. We thus conjecture that the spatial slices themselves become non-smooth, which leads to simulation failure. The right panel then shows the "break" in the same quantity in the $x - z$ plane.

Quasi-maximal slicing First, we verify our solver for the equation (21) does indeed converge to $\partial_{tt} K = 0$. To that end, we run a series of simulations with weak waves ($A = 0.1$) and no mesh refinement, where W is evaluated at each full step of the time integrator (i.e. not using the extrapolation/interpolation procedure from Section 4.1). The simulations differ only in the time step Δt, which halves for each finer run — the coarsest run makes just one full time step, while the finest one makes 128. As the left panel of Figure 3 shows, the maximum norm of K goes linearly to zero as the time step decreases.

Next, we run a series of simulations for stronger waves ($A = 1$), with the grid setup as for the previously described 1+log runs. The reference run uses the 1+log slicing, and we compare it against several quasi-maximally sliced runs (now as described in Section 4.1) with varying CFL factor. As can be seen from the top panel of Figure 3, even when W_a is a very crude approximation to the solution of (21), K is significantly lower for quasi-maximal slicing than for 1+log, and further decreases with smaller time step.

The vanishing of K as the time step goes down is slow and hard to quantify, so we find it more useful to regard quasi-maximal slicing merely as a procedure that gives us

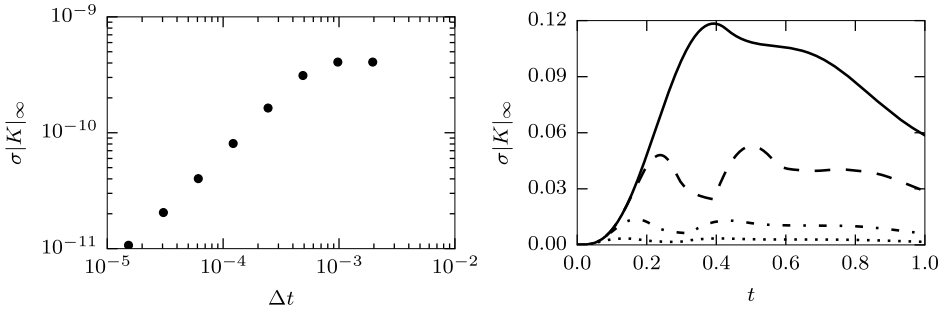

Figure 3. Convergence of the quasi-maximal slicing with the time step Δt. Left: $A = 0.1$ Brill waves, plotted is the maximum norm of K at simulation time $t = \frac{1}{512}$ (one full time step for the coarsest run) as a function of the time step Δt. Right: $A = 1$ Brill waves, time evolution of the maximum norm of K over the finest refinement level for (top to bottom): 1+log slicing and quasi-maximal slicing with $\Delta x/\Delta t = \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}\right\}$ (changing the CFL factor for the 1+log run has no visible effect on the plot within the bounds of stability). See main text for more details.

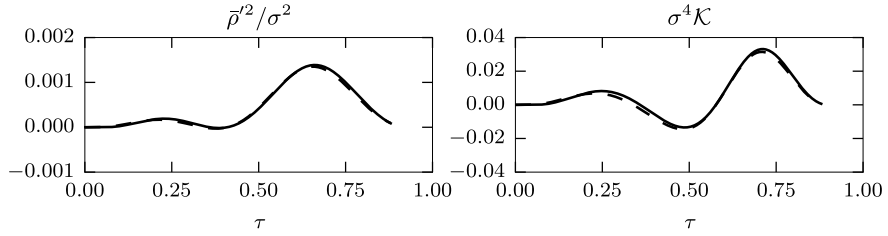

Figure 4. Convergence of spacetime invariants with quasi-maximal slicing for $A = 1$ Brill waves. Invariant quantities at $\bar{\rho} = 1$ — left: $\bar{\rho}'^2$, right: $\mathcal{K}$ — are computed as functions of the proper time τ at a given location for simulations with $\{256, 384, 512\}$ grid points per refinement level. Plotted is the difference between the $256 - 384$-point run (solid line) and the $384 - 512$-point run (dashed line), scaled for 4th order convergence.

some slicing which is "close" to the maximal one, but distinct from it. What we want to verify then is that when we hold the parameters of the quasi-maximal solver fixed and increase the resolution of the finite difference grid, the simulations converge to the same underlying spacetime, though possibly in different slicing. For that purpose, we look at spacetime invariants.

We run three simulations, again with $A = 1$, and $\{256, 385, 512\}$ grid points per refinement level. For this data, the circumferential radius $\bar{\rho}$ in the equatorial plane $z = 0$ remains a monotonous function of x throughout the simulation, so it can be used as a radial coordinate (this is not the case for stronger Brill waves). For each simulation we calculate the value of $\bar{\rho}'^2$ and $\mathcal{K}$ seen by the observer at $\bar{\rho} = 1$. To work with purely invariant quantities, we also compute that observer's proper time τ by integrating $\mathrm{d}\tau = -(g_{\mu\nu}\mathrm{d}x^\mu \mathrm{d}x^\nu)^{\frac{1}{2}}$ and plot the aforementioned invariants as functions of τ. As we can see from Figure 4, those values converge to each other with the

fourth order, which is the order of the time integrator. We can thus conclude that quasi-maximally sliced simulations (with fixed pseudo-spectral solver order) do indeed converge to the same physical spacetime. As previously mentioned, it would be hard to state precisely the measure of convergence towards the maximal slicing.

5. Evolution of Brill waves with quasi-maximal slicing

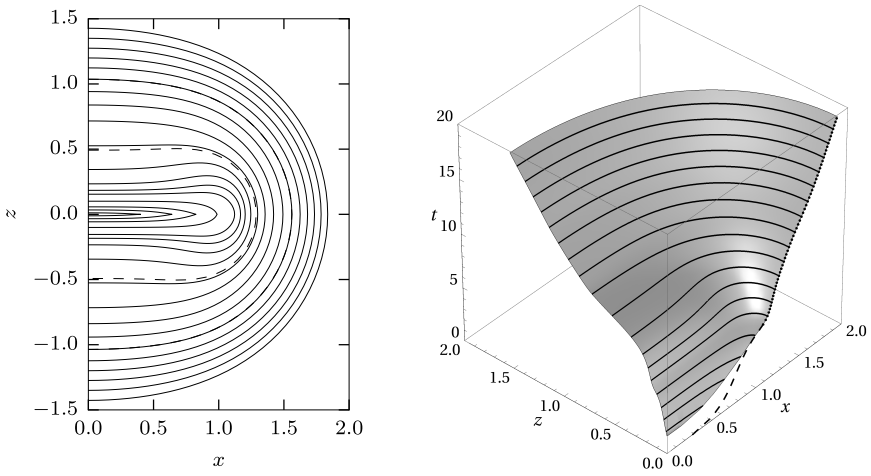

Figure 5. Formation of the event horizon for $A = 5$ Brill waves, zero shift. It is constructed by evolving the apparent horizon section from the $t = 25$ slice as a null surface back in time. Left: solid lines show cross-sections of the event horizon in the $x - z$ plane at simulation times $t = \{2, 3, \cdots, 14\}$ (inner to outer). Dashed lines show the apparent horizon at time $t = 10$ (when it first appears) and $t = 14$. At later times both horizons cannot be distinguished in this plot. Right: the event horizon as a surface in $x - z - t$ coordinates. When it first appears it is not smooth at the equator, which is represented by the x coordinate in this plot. It remains non-smooth until the radial null geodesic (dashed line in the $z = 0$ plane) enters the event horizon as its generator (dotted line).

Near-critical case: $A = 5$ Our main result is demonstrating that quasi-maximal slicing cures the singularities arising with 1+log slicing for $A = 5$ Brill waves. This near-critical initial data, with ADM mass 0.70, was used as the failing case in [11]. Though quasi-maximal slicing resolves the singularities from the α/K sector, we have encountered additional issues with the shift. When using the Γ-driver, the shift seems to develop a shock profile around $t \approx 10$. For that reason, we use zero shift in the simulations described here. While that allows us to run the simulation further, it is well known that a non-zero shift vector is required in black-hole spacetimes to avoid "slice stretching" — we are thus still unable to evolve this data indefinitely.

Around simulation time $t \approx 10.25$ we first find an apparent horizon with mass $M_{AH} = 0.56$, which grows to $M_{AH} = 0.58$ by $t = 20$. The masses agree with values reported in [13]. By integrating the apparent horizon to the past as a null surface we construct the event horizon, seen in Figure 5. Its shape at early times turns out to be that of a disk with a non-smooth rim. This can be understood from the curious fact that if we shoot photons from the origin at $t = 0$, those along the z-axis will escape

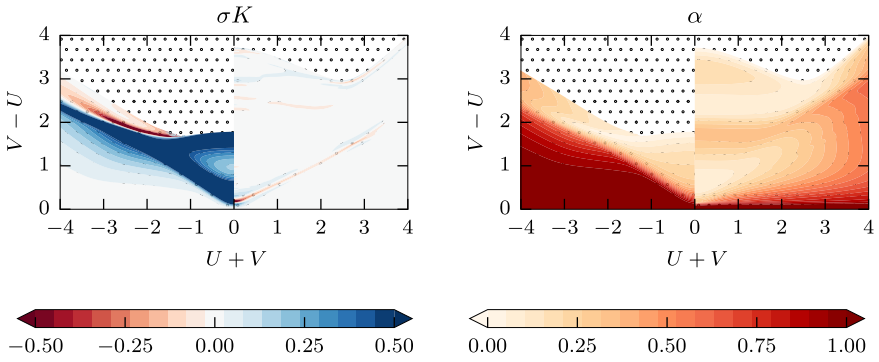

Figure 6. Comparison of the $A = 5$ runs with 1+log / quasi-maximal slicings — respectively the left and right half of each panel — in the double-null coordinates $\{U, V\}$ in the equatorial plane. Color denotes the value of K (left panel) and α (right panel), dotted regions are not covered by respective simulations.

to infinity, while those along the x-axis will end up below the horizon. The event horizon thus does not intersect the initial slice and appears around $t = 1.7$, when the first photon in the z direction fails to escape. Its worldline along the x-axis then has a spatial character until it connects with its generating null geodesic, as is sketched in the right panel of Figure 5. This means the event horizon can briefly exist as a torus, which we observed for a run with quasi-maximal slicing and $A = 4.8$.

There are various sources of uncertainty in determination of the event horizon. The most important one is the error of the simulation, which can be judged by relative variation of late-time MOTS areas between various simulations and appears to be of order 10^{-3}. Of similar order is the error with which we can trace null geodesics forming the event horizon (this is more accurate along the x and z axes, where the data available to the tracing code is much better resolved in time). Then there is the difference between the event horizon and MOTS and the uncertainty of future evolution of the spacetime, but due to unstable nature of future-oriented null geodesics near the event horizon, we find that at $t = 10$ a spatial variation of $\Delta x \approx 10^{-5}$ will — during the runtime of the simulation — either direct the null ray inside the black hole or put it on an escape trajectory.

To demonstrate that the quasi-maximal run actually does cover a larger chunk of spacetime than the crashing 1+log run (rather than merely collapsing the lapse in the critical area), we compare the two slicings in double-null coordinates $\{U, V\}$. Those are constructed by integrating null geodesics along the x-axis in, respectively, the positive and negative x directions. The values of U and V in the initial slice $t = 0$ are both set equal to half the proper distance from the origin. Figure 6 clearly shows that quasi-maximal slicing advances farther towards the physical singularity in the central area.

Super-critical case: $A = 5.5$ The ADM mass of this data is 0.84. Unlike the previous case, we are able to use the Γ-driver without any pathologies, which allows the simulations to run much longer. At $t \approx 7.5$ we find an apparent horizon with mass $M_{AH} = 0.63$, which quickly increases to its final value of 0.73.

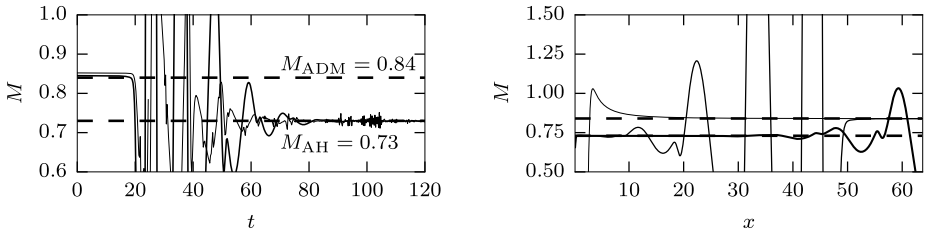

Figure 7. Evolution of mass estimates for $A = 5.5$ Brill waves. Left: evolution of $M_{\mathcal{K}}$ (thin line) and $M_{\bar{\rho}}$ (thick line) at constant spatial coordinate $\{x = 20, z = 0\}$ with the simulation time t. Right: $M_{\bar{\rho}}$ along the x-axis at times $t = \{0, 50, 100\}$ (thin to thick). In both pictures the dashed horizontal lines indicate the ADM mass of the initial slice (upper) and the final apparent horizon mass M_{AH} (lower). See the main text for details.

At later times, the spacetime should evolve into a central region that rapidly relaxes into a Schwarzschild black hole, plus an outward-travelling wave packet. We illustrate this fact in Figure 7 by plotting mass estimates (13) and (17). These, as mentioned, have the meaning of mass only in the static end-state, so we can see them oscillate wildly as the waves radiate away. At late simulation times they closely approach the value of M_{AH}, which shows that we really obtain a Schwarzschild black hole with expected properties.

The Γ-driver behaves in a manner similar to that known for the evolutions of Schwarzschild starting with "wormhole" slices, effectively excising the central region from the numerical grid. At late times the numerical slices terminate at $\bar{\rho} \approx 1.31$.

Other near-critical amplitudes We run a number of additional simulations with both sub- and super-critical initial data until, respectively, the waves disperse in the central region or an apparent horizon forms. We observe no slicing singularities in any of them.

For analyzing the spacetime dynamics we rely on the invariants introduced in Section 2.3. Figure 8 shows values of the scalar χ, defined by (16), for long-term evolutions of sub- and super-critical waves. As previously mentioned, χ is negative when the gradient of the circumferential radius is timelike. In Schwarzschild this is true inside the event horizon, with $\nabla^\mu R$ pointing to the future in the "white hole" region and to the past in the "black hole" region. It is interesting that for Brill waves, the orientation is to the future in the first negative-χ region (i.e. "the wrong way") and then switches direction, possibly multiple times, which shows how wildly dynamic those spacetimes are.

A drawback of χ is that it only has a clear mapping to Schwarzschild in the equatorial plane, since physically meaningful values of θ are not available. One way around that is the scalar $\frac{1-\chi}{\bar{\rho}^2}$, which is equal to $\frac{2M}{R^3}$ in Schwarzschild. It is, however, no longer dimensionless. This quantity is shown alongside $\mathcal{K}$ in Figure 9 for a sub-critical run at the time when those scalars attain its highest values. Interestingly this happens on the z axis some distance away from the origin (cf. the results of [13]).

To compare the spacetime geometries with differing amplitude parameter A side-by-side, we mimic the usual compactification of the Schwarzschild manifold by a transformation that maps the positive part of the x-axis through $X = \arctan\left[\frac{3}{2}\sinh\left(\frac{x}{6}\right)\right]$ into the interval $(0, \frac{\pi}{2})$ and then maps the null geodesics in the

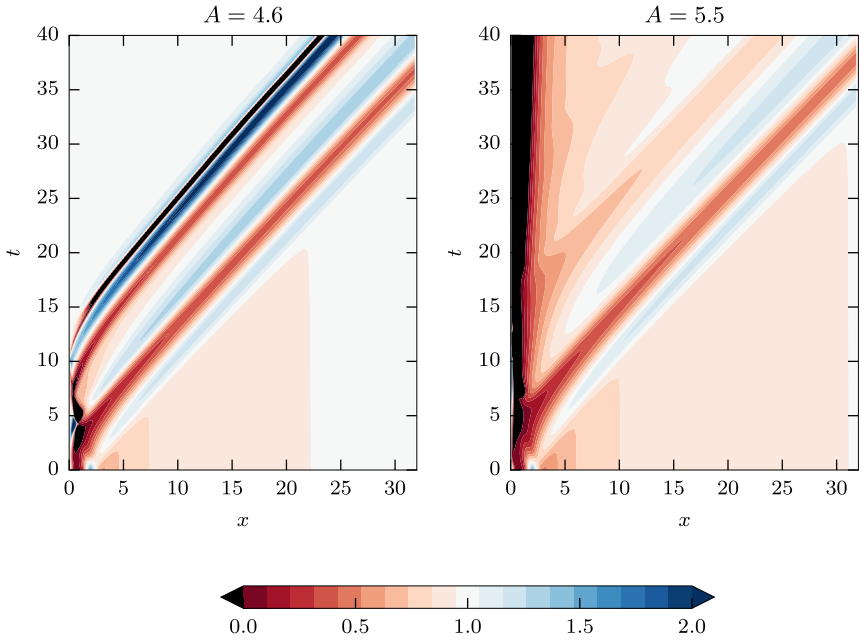

Figure 8. Evolution of χ in the equatorial plane for Brill waves with $A = 4.6$ (left) and $A = 5.5$ (right).

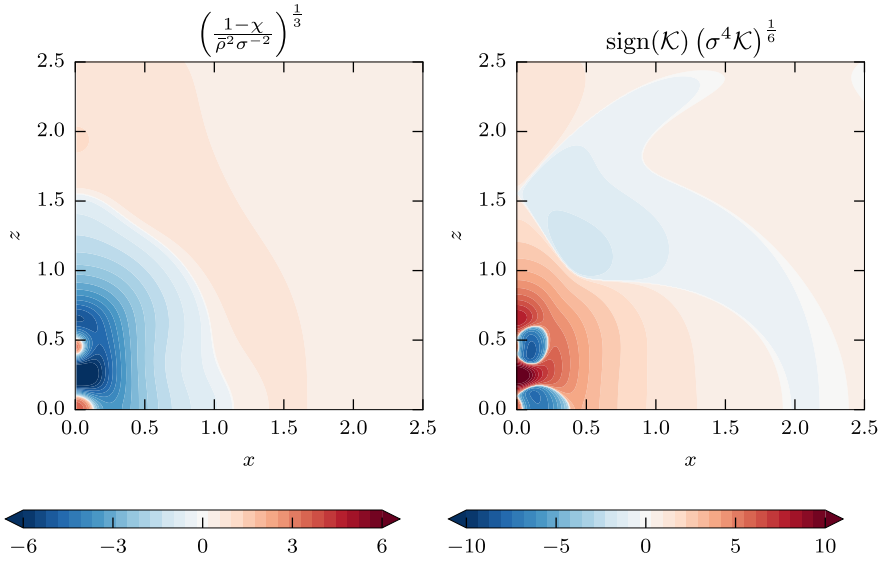

Figure 9. Invariants in the $x - z$ plane at coordinate time when they reach their highest values ($t = 11$); $A = 4.65$ data, zero shift.

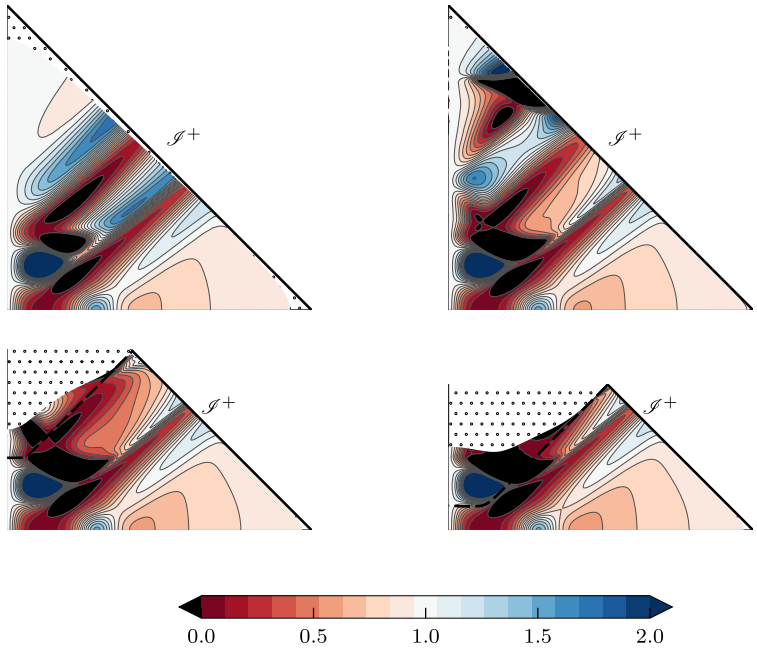

Figure 10. Conformal diagrams for the equatorial plane of the spacetimes produced by evolving the Brill waves. The values of A are $\{4.0, 4.65, 4.8, 5.0\}$ for, respectively, top left, top right, bottom left, and bottom right. The top two spacetimes are sub-critical, while the bottom two are super-critical, as seen from the presence of an event horizon (dashed lines). The spacelike portion of the event horizon is non-smooth, as described in more detail in the main text. The color denotes the value of χ, defined by (16). Note the range the quantity χ spans — for the spherically symmetric sub-critical spacetimes of [5] it stays between 0.4 and 1.

$x - t$ plane into straight 45-degree lines. This produces the conformal diagrams in Figure 10, which illustrate propagation of waves towards future null infinity.

6. Summary

We run numerical simulations of sub- and super-critical Brill wave initial data. We manage to verify previously described slicing problems for strong data with a different code and illustrate them with figures showing that the t = const. slices develop a cusp-like shape. These problems turn out to be removable by amending the 1+log slicing with a source term derived from the maximal slicing condition.

This new slicing condition, which we call the "quasi-maximal" slicing, improves the regularity of evolved spatial slices by pushing them closer towards the maximal ones. We describe long-term evolution of the spacetimes arising from near-critical waves and construct their conformal diagrams. Of special note is the non-smooth shape of the event horizon for weaker data. This data is not obtainable with the 1+log slicing because of the aforementioned pathologies.

As mentioned in the introduction, Hilditch et al. are currently working on

the same problem with a new pseudospectral code using a generalized harmonic formulation [12, 13], so a brief comparison is in order. Since our code is based on finite differences and the moving puncture approach, it inherits their advantages, such as simplicity, robustness and the ability to run long-term simulations containing black holes without the need for excision. The price to pay is mainly lower efficiency, i.e. higher resolution required for the same accuracy. Of course, the fact that both these approaches produce the same results (as confirmed in private communication), is greatly encouraging.

The code used for running the simulations described in this paper is available for download through the *git* protocol at `git://git.khirnov.net/qms_source_2017`.

Acknowledgments

We would like to thank David Hilditch for very productive discussions. This work is supported by the Charles University in Prague, project GA UK No 2000314, GA UK No 1176217 and SVV-260211. T.L. acknowledges the support from the Czech Science Foundation grant No 14-37086G (A. Einstein Center). Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures". We would also like to thank the developers of the Einstein Toolkit for making open-source numerical relativity possible.

References

[1] Kenneth Eppley. Evolution of time-symmetric gravitational waves: Initial data and apparent horizons. *Phys. Rev. D*, 16:1609–1614, Sep 1977.

[2] Dieter R. Brill. On the positive definite mass of the bondi-weber-wheeler time-symmetric gravitational waves. *Annals of Physics*, 7(4):466 – 483, 1959.

[3] Andrew M. Abrahams and Charles R. Evans. Critical behavior and scaling in vacuum axisymmetric gravitational collapse. *Phys. Rev. Lett.*, 70:2980–2983, 5 1993.

[4] Saul A. Teukolsky. Linearized quadrupole waves in general relativity and the motion of test particles. *Phys. Rev. D*, 26:745–750, 8 1982.

[5] Matthew W. Choptuik. Universality and scaling in gravitational collapse of a massless scalar field. *Phys. Rev. Lett.*, 70:9–12, 1 1993.

[6] Carsten Gundlach and José M. Martín-García. Critical phenomena in gravitational collapse. *Living Reviews in Relativity*, 10(1):5, 01 2007.

[7] Miguel Alcubierre, Gabrielle Allen, Bernd Brügmann, Gerd Lanfermann, Edward Seidel, Wai-Mo Suen, and Malcolm Tobias. Gravitational collapse of gravitational waves in 3d numerical relativity. *Phys. Rev. D*, 61:041501, 1 2000.

[8] David Garfinkle and G. Comer Duncan. Numerical evolution of brill waves. *Phys. Rev. D*, 63:044011, 01 2001.

[9] Carles Bona, Joan Massó, Edward Seidel, and Joan Stela. New formalism for numerical relativity. *Phys. Rev. Lett.*, 75:600–603, Jul 1995.

[10] Miguel Alcubierre, Bernd Brügmann, Peter Diener, Michael Koppitz, Denis Pollney, Edward Seidel, and Ryoji Takahashi. Gauge conditions for long-term numerical black hole evolutions without excision. *Phys. Rev. D*, 67:084023, Apr 2003.

[11] David Hilditch, Thomas W. Baumgarte, Andreas Weyhausen, Tim Dietrich, Bernd Brügmann, Pedro J. Montero, and Ewald Müller. Collapse of nonlinear gravitational waves in moving-puncture coordinates. *Phys. Rev. D*, 88:103009, 11 2013.

[12] David Hilditch, Andreas Weyhausen, and Bernd Brügmann. Pseudospectral method for gravitational wave collapse. *Phys. Rev. D*, 93:063006, 3 2016.

[13] David Hilditch, Andreas Weyhausen, and Bernd Brügmann. Evolutions of centered brill waves with a pseudospectral method. *Phys. Rev. D*, 96:104051, Nov 2017.

[14] Oliver Rinne. Constrained evolution in axisymmetry and the gravitational collapse of prolate brill waves. *Classical and Quantum Gravity*, 25(13):135009, 2008.

[15] Masaru Shibata and Takashi Nakamura. Evolution of three-dimensional gravitational waves: Harmonic slicing case. *Phys. Rev. D*, 52:5428–5444, 11 1995.
[16] Thomas W. Baumgarte and Stuart L. Shapiro. On the numerical integration of einstein's field equations. *Phys. Rev. D*, 59:024007, 1999.
[17] M. Alcubierre. *Introduction to 3+1 Numerical Relativity*. Oxford University Press, UK, 2008.
[18] Carsten Gundlach and José M. Martín-García. Well-posedness of formulations of the einstein equations with dynamical lapse and shift conditions. *Phys. Rev. D*, 74:024016, 7 2006.
[19] James R. van Meter, John G. Baker, Michael Koppitz, and Dae-Il Choi. How to move a black hole without excision: Gauge conditions for the numerical evolution of a moving puncture. *Phys. Rev. D*, 73:124011, Jun 2006.
[20] John P. Boyd. Spectral methods using rational basis functions on an infinite interval. *Journal of Computational Physics*, 69:112–142, 1987.
[21] John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications Inc., New York, 2001.
[22] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.
[23] Einstein Toolkit: Open software for relativistic astrophysics.
[24] Frank Löffler, Joshua Faber, Eloisa Bentivegna, Tanja Bode, Peter Diener, Roland Haas, Ian Hinder, Bruno C. Mundim, Christian D. Ott, Erik Schnetter, Gabrielle Allen, Manuela Campanelli, and Pablo Laguna. The Einstein Toolkit: A Community Computational Infrastructure for Relativistic Astrophysics. *Class. Quantum Grav.*, 29(11):115001, 2012.
[25] Cactus Computational Toolkit.
[26] Tom Goodale, Gabrielle Allen, Gerd Lanfermann, Joan Massó, Thomas Radke, Edward Seidel, and John Shalf. The Cactus framework and toolkit: Design and applications. In *Vector and Parallel Processing – VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science*, Berlin, 2003. Springer.
[27] Erik Schnetter, Scott H. Hawley, and Ian Hawke. Evolutions in 3-D numerical relativity using fixed mesh refinement. *Class. Quantum Grav.*, 21:1465–1488, 2004.
[28] Marsha J. Berger and Joseph Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *J. Comput. Phys.*, 53:484, 1984.
[29] McLachlan, a public BSSN code.
[30] Kranc: Kranc assembles numerical code.
[31] J. David Brown, Peter Diener, Olivier Sarbach, Erik Schnetter, and Manuel Tiglio. Turduckening black holes: an analytical and computational study. *Phys. Rev. D*, 79:044023, 2009.
[32] Miguel Alcubierre, Steven Brandt, Bernd Brügmann, Daniel Holz, Edward Seidel, Ryoji Takahashi, and Jonathan Thornburg. Symmetry without symmetry: Numerical simulation of axisymmetric systems using Cartesian grids. *Int. J. Mod. Phys.*, D10:273–290, 2001.
[33] H. A. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.
[34] Béla Szilágyi, Lee Lindblom, and Mark A. Scheel. Simulations of binary black hole mergers using spectral methods. *Phys. Rev. D*, 80:124010, Dec 2009.

A.4 Published paper: Universality of Curvature Invariants in Critical Vacuum Gravitational Collapse

This section contains the author version of our paper "Universality of Curvature Invariants in Critical Vacuum Gravitational Collapse" [38], published in *Physical Review Letters* (© 2021 American Physical Society).

Universality of curvature invariants in critical vacuum gravitational collapse

Tomáš Ledvinka* and Anton Khirnov†
Institute of Theoretical Physics, Faculty of Mathematics and Physics, Charles University, CZ-180 00 Prague, Czech Republic

We report on a numerical study of gravitational waves undergoing gravitational collapse due to their self-interaction. We consider several families of asymptotically flat initial data which, similarly to the well-known Choptuik's discovery, can be fine-tuned between dispersal into empty space and collapse into a black hole. We find that near-critical spacetimes exhibit behavior similar to scalar-field collapse: For different families of initial data we observe universal "echoes" in the form of irregularly repeating, approximate, scaled copies of the same piece of spacetime.

Introduction.—Choptuik's surprising discovery of critical behavior in gravitational collapse [1] showed that numerical simulations of the Einstein equations may reveal unforeseen features of the theory. It turned out that evolution of a spherically symmetric massless scalar field minimally coupled to general relativity with initial data (ID) near the threshold between field dispersal and collapse into a black hole takes the form of "echoes"—repeated concentrations of the field appearing on progressively smaller scales. While the first few echoes retain some imprint of the ID family, later on the metric and the field look like scaled-down copies of earlier moments (discrete self-similarity) and approach the same profile for any ID fine-tuned towards the black-hole threshold (universality). Extensive follow-up research (see Ref. [2] for a detailed review) then showed that similar behavior occurs for many other fields.

If the initial data form a one-parameter family, with A being the parameter, then quantities such as the mass of the black hole created during the collapse m_{BH}, the number of echoes observed, or the maximal field strength depend on A in a characteristic way as it approaches the critical value A_*; e.g., $m_{\mathrm{BH}} \sim (A - A_*)^\gamma$ with $\gamma \doteq 0.373$ for spherically symmetric minimally-coupled massless scalar field. Later on, more detailed features such as a periodic modulation [2] and a lower bound of the scaling law for m_{BH} [3] were described.

Gravitational waves (GWs), an appealing alternative to a massless scalar field, have also been studied in the context of critical collapse [4–12] (see Ref. [12] for a detailed description of these attempts).

In general relativity (in 3+1 dimensions, cf. Ref. [13]), there are no spherically symmetric gravitational waves, which makes numerical simulations much more computationally demanding. Even with modern powerful computers and advanced numerical techniques, no universal and across scales repeating profile of the gravitational field analogous to that in the seminal paper [1] has been reported. Moreover, Hilditch et al. [12] brought a strong argument against a simple analogy with spherically symmetric collapse—for the initial data closest to the critical amplitude, a pair of apparent horizons appeared. Recently, critical collapse away from spherical symmetry has also been probed using models of combined gravitational and electromagnetic fields [14] and a semilinear scalar wave [15].

Methods.—Our numerical simulations use an unconstrained evolution scheme, the so-called Baumgarte-Shapiro-Shibata-Nakamura version of the Einstein equations [16, 17]. We use the Einstein Toolkit framework [18] modified to analytically assume axial symmetry, and extended by a code solving elliptic equations for initial data and slicing.

Coordinate choice: The standard 1+log slicing condition has been shown to break down in the numerical evolution of collapsing gravitational waves [10]. Instead, we use the quasi-maximal slicing (QMS) that we introduced recently [19] to handle such highly dynamic spacetime geometries. We changed its implementation to use a multigrid method together with a scheme based on Ref. [20] in order to be compatible with the Berger-Oliger mesh refinement, which we use to evolve hyperbolic equations.

Initial data: In the $3+1$ approach to general relativity, initial data are specified as 3-tensor fields γ_{ij} and K_{ij}—the intrinsic metric and the extrinsic curvature—on the initial Cauchy hypersurface. These fields must satisfy a set of coupled nonlinear elliptic equations (Hamilton and momentum constraints) implied by the Einstein equations. We study two axi- and plane-symmetric ID families with trace $K = \gamma^{ij}K_{ij} = 0$ (i.e., compatible with the maximal slicing). The first family is the Brill data, first studied numerically in Ref. [21]. Their main feature is time-symmetry due to $K_{ij}(t = 0) = 0$. Gravitational waves are then encoded as a deformation of the initial slice intrinsic metric $\gamma_{ij}(t = 0)$ written in standard spherical coordinates,

$$\gamma_{ij}dx^i\,dx^j = \psi^4\left[e^{2q}(dr^2 + r^2d\theta^2) + r^2\sin^2\theta d\phi^2\right]. \quad (1)$$

We choose the "seed function" thoroughly studied in Ref. [12]

$$q(x^i) = A\,\sigma^{-2}r^2e^{-r^2/\sigma^2}\sin^2\theta, \quad (2)$$

where we introduce a scale parameter σ, leaving A dimensionless. The conformal factor ψ must be found by solving the Hamilton constraint.

The second family of initial data is inspired by Ref. [5] where the initial 3-metric is taken to be conformally flat, and one component of K_{ij}—$K^r{}_\theta$—is chosen to be the deformation seed. This leads to three coupled constraint

equations to solve for ψ, $K^r{}_r$ and $K^\phi{}_\phi$. However, despite following Ref. [5] to the best of our ability, we were not able to reproduce their data exactly (as is clear, e.g., from very different critical amplitudes), and so we use it merely as inspiration, choosing a visually similar profile

$$K^r_\theta(x^i, t=0) = A\sigma^{-3} r^2 (\sigma - r) e^{-r^2/\sigma^2} \sin 2\theta. \quad (3)$$

Solutions of the constraints for $A > 0$ then turn out to be non-unique in a way very similar to Ref. [22]; there exists a value $A_{\max} \approx 1.36247$ such that there are two solutions for $0 < A < A_{\max}$ and none for $A > A_{\max}$. On the "lower" branch, the data behave as expected; they approach flat space as $A \to 0$ and their Arnowitt-Deser-Misner (ADM) mass $M_{\rm ADM}$ grows with increasing A. By contrast, on the "upper" branch the mass grows with *decreasing* A, apparently diverging as $A \to 0$. As $A \to A_{\max}$, both branches approach the same solution, so we can consider them together as a single ID family with continuously growing ADM mass. We mark the upper-branch solutions with a bar; e.g., $A = \overline{1.0}$ is an upper-branch solution with $M_{\rm ADM} \doteq 1.06\sigma$, while $A = 1.0$ is a lower-branch solution with $M_{\rm ADM} \doteq 0.104\sigma$.

One can also consider negative values of A—for ID (1) this leads to a different initial data family [10]. However, for ID (3) replacing $A \to -A$ merely flips the sign of K_{ij}; i.e., we get the same initial slice evolved backward in time. Since the data are time asymmetric (TA), critical collapse can be studied for $A < 0$ as for a new ID family.

Coordinate-independent analysis: Even though the Kretschmann scalar $I_K \equiv R_{\alpha\beta\gamma\delta} R^{\alpha\beta\gamma\delta} (\alpha, \beta = 0..3)$, built from the Riemann tensor $R_{\alpha\beta\gamma\delta}$ is not a direct measure of spacetime curvature due to Lorentzian signature of spacetime metric, it is still an obvious coordinate-independent scalar quantity providing an invariant indicator of the gravitational field strength. In an axisymmetric vacuum spacetime, there are additional coordinate-independent scalars available. The circumferential radius ρ and the norm of its gradient $\rho_{,\alpha}\rho^{,\alpha}$ are the simplest ones, but their values are trivial at the axis of symmetry $\rho = 0$. We thus propose to use their combination $\zeta \equiv (1 - \rho_{,\alpha}\rho^{,\alpha})/\rho^2$ (completed by an appropriate limit at the axis, see also [19]) as a coordinate-independent indicator of the spacetime geometry. It can be shown that $\Psi_{2\,|\rho=0} = \frac{1}{2}\zeta_{|\rho=0}$ is the only non-vanishing projection of the Weyl tensor (as defined, e.g., in Ref. [23]) onto an axis-aligned null tetrad, so at the axis, we also have $I_{K\,|\rho=0} = 12\zeta^2_{|\rho=0}$.

Results.—Critical amplitudes: We observed behavior compatible with the existence of critical amplitudes separating dispersal and black-hole formation. The limiting factor in near-critical simulations is the sufficient resolution of the QMS solver, without which the coordinate singularities known from Ref. [10] appear. Importantly, it turned out that some ID families are less prone to those pathologies than others and so require less computational effort. Among several attempts, the initial data (3) appeared to be least demanding. We concentrated the available resources ($\sim 10^4$ CPU-hours per run) here and obtained five echoes and $A_*^{\rm TA+} \doteq \overline{1.3008079^{\pm 4}}$. For negative values of the parameter, we get $A_*^{\rm TA-} \doteq \overline{-1.22434^{\pm 5}}$. The Brill initial data (1) defied our bisection attempts most and we got an interval $A_*^{\rm Brill+} \doteq 4.697^{\pm 1}$, compatible with the much better result $A_*^{\rm Brill+} \doteq 4.6966953^{\pm 78}$ in [12]. With negative A and less effort, we found $A_*^{\rm Brill-} \doteq -3.509106^{\pm 5}$.

Scaling: To relate our work to existing results, we start by discussing how the extrema of the Kretschmann scalar, $I_K^{\max}$, depend on the amplitude parameter A. Plots showing $I_K^{\max}(A)$, which reduce the entire evolution of the initial data to a single number, are inspired by the typical behavior of a scale-invariant spherically-symmetric critical collapse. It admits critical solutions exhibiting a discrete self-symmetry (DSS) [2], i.e., containing a geometric sequence with a quotient $e^{-\Delta}$ of scaled-down copies of the same field configuration. The evolution of near-critical initial data in the central region (in the past null cone of the accumulation point) first approaches this solution, then exhibits several almost-DSS cycles, and finally, either the scalar field undergoes dispersion or forms a black hole. Then, for subcritical spacetimes, the quantity $(I_K^{\max})^{1/4}$, with the dimension of inverse length indicates the smallest scale up to which the evolution in the central region stays close to the critical solution. The approximate relation between this scale and the parameter A again has the form $(I_K^{\max})^{-1/4} \sim |A - A_*|^\gamma$.

For subcritical GW collapse, we observed that near A_* the Kretschmann invariant I_K attains its most pronounced extrema at the axis of symmetry coinciding with the minima of the invariant ζ. As $A \to A_*$, echoes with ever higher amplitudes appear and the strongest echo for a given A determines the value $I_K^{\max}$. These are shown in Fig. 1, where individual simulations are shown as data points for four families of initial data. (The overlapping markers at $|A - A_*| \approx e^{-7.3}$ show the simulation for which $I_K^{\max}$ appears off the z axis.) Each ID family can be approximated by a power law, with the critical exponent estimates $\gamma_{\rm TA^+} = 0.35^{\pm 3}$, $\gamma_{\rm TA^-} = 0.37^{\pm 8}$, and $\gamma_{\rm Brill-} = 0.19^{\pm 3}$. According to [12], $\gamma_{\rm Brill+} \doteq 0.37$. In our fits, we excluded data points corresponding to the first echo so that a direct influence of the initial data form is suppressed. A similar approach applied to the results of Ref. [12] seems to yield $\gamma_{\rm Brill+} > 0.5$.

These differences in the exponent γ appear to be significant, but we cannot decide if the slopes in Fig. 1 really settle towards a specific value for a given family or whether they continue fluctuating significantly with further echoes appearing without apparent order. One could claim that our simulations are merely not close enough to A_*, but we will show that individual echoes have a universal form, so this argument does not seem convincing.

We categorize the ID as supercritical if we find an apparent horizon (AH). Because some AHs grow rapidly at first, and it is impractical to store all the data for postprocessing, we determine the initial AH mass $M_{\rm AH}$

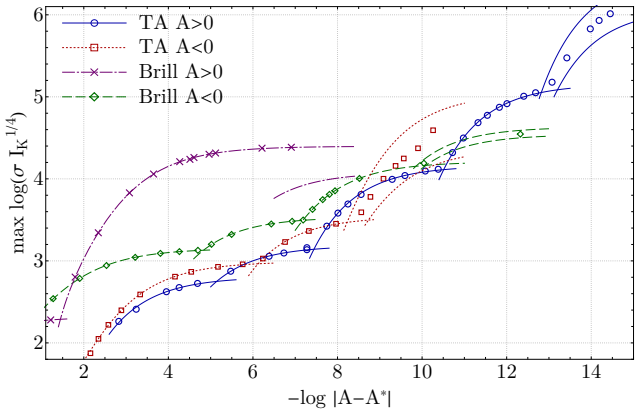

FIG. 1. Global maxima of the Kretschmann invariant in subcritical spacetimes with four families of initial data depending on a parameter A. As A approaches the critical value A_*, the maxima get ever larger, as newly appearing local extrema overtake earlier ones. To illustrate the smooth dependence of these local extrema (echoes) on the parameter A, we fit the simulation results shown as points with a polynomial—typically a simple linear dependence $\log I_K^{\max} = pA + q$. The plotted curves are thus composed of segments, each corresponding to a specific local maximum being the strongest one. An effect of the uncertainty of A_* within the final bisection interval is indicated in the rightmost segments; $A_*^{\text{Brill}+}$ is taken from Ref. [12].

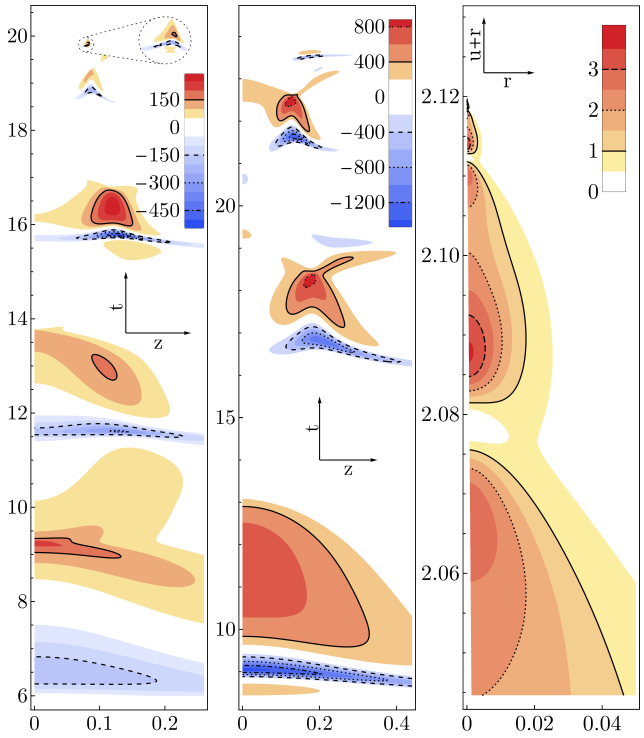

FIG. 2. Comparison of the echoes in the curvature invariant ζ between different collapse scenarios. Since its extrema span several orders of magnitude, shown are the contours of a dimensionless (but coordinate-dependent; see text) quantity $(\tau_* - \tau)^2\zeta$ in the $t - z$ plane. In a DSS setup, this quantity would repeatedly acquire the same extremal values on ever smaller scales as $\tau \to \tau_*$ (near-critical massless scalar-field collapse [3] in the right panel). Left: $A_{\text{TA}^+} = \overline{1.30080828}, z_0 = 0.08415\sigma, \tau_* = 3.88\sigma$; center: $A_{\text{Brill}-} = -3.5090625, z_0 = 0.161\sigma, \tau_* = 5.9\sigma$

with a considerable "sampling" error. We estimate it to be $\lesssim 20\%$ for $-9.5 < \epsilon_A < -2$, where $\epsilon_A \equiv \log|A - A_*|$. In this interval $\mu \equiv \log(M_{\text{AH}}/\sigma)$ satisfies $\mu_{\text{TA}^+} = -0.44 + 0.17\epsilon_A \pm 0.15$, $\mu_{\text{TA}^-} = 0.07 + 0.21\epsilon_A \pm 0.13$, and $\mu_{\text{Brill}^-} = -0.49 + 0.17\epsilon_A \pm 0.10$. For ID (1) with $A > 0$ we get $\gamma \approx 0.16$ for $-6 < \epsilon_A < -1$. Below these intervals, we observe bifurcated horizons: For ID (1) with $A_{\text{Brill}^+} = 4.698$, in agreement with Ref. [12], we find a pair of AHs, each with $M_{\text{AH}} \doteq 0.10\sigma$. In addition, for ID (3) with $A_{\text{TA}^+} = \overline{1.3008012}$, we get a pair with $M_{\text{AH}} \doteq 0.037\sigma$. By contrast, the $I_K^{\max}$ slopes $\gamma_{\text{TA}^\pm}$ and $\gamma_{\text{Brill}-}$ given above include only the bifurcated curvature extrema (as seen in Fig. 2). Thus we assume they describe the behavior close to A_* more faithfully. Further complications associated with the use of apparent horizons for critical behavior investigation are discussed in Ref. [12].

Self-similarity and universality: Typically, the scale-invariant collapse in spherical symmetry permits a DSS critical solution. An approximation of this critical solution then appears inside a so-called self-similarity horizon for an arbitrary spherically symmetric one-parameter ID family if the parameter is fine-tuned between dispersal and collapse. In this Letter we argue that for GW, we find analogous yet more complicated behavior: The spacetime regions near the extrema of $I_K^{\max}$ appear repeatedly as scaled approximate copies (a limited and irregular analog of self-similarity) of the same piece of a spacetime and independent of the ID family (universality).

In a spherically symmetric DSS spacetime and assuming adapted coordinates, all dimensionless quantities are periodic functions of the logarithmic time $\log|\tau_* - \tau|$ [2], with τ being the central proper time, which takes the value of τ_* at the accumulation event. Because the curvature extrema in critical GW collapse appear at $z \neq 0$, to define τ we choose the worldline of constant $z = z_0$ on which the global maximum of the Kretschmann invariant appears. Because of our choice of shift $\beta^i = 0$, this worldline is timelike. We then distribute τ over our (approximately maximal) slices $t = \text{const}$ and construct a dimensionless quantity $(\tau_* - \tau)^2\zeta$. Despite its coordinate dependence, it is remarkably efficient in "equalizing" the echoes to a common scale. The spacetime diagrams in Fig. 2 demonstrate this by showing the similarity of the echoes across very different families (1) and (3).

The right panel shows the same quantity for a scalar-field collapse computed according to Ref. [3], where we take $\tau \equiv u + r$ outside of the center. It illustrates significant differences between scalar-field and GW critical collapse, with the latter having spacetime curvature con-

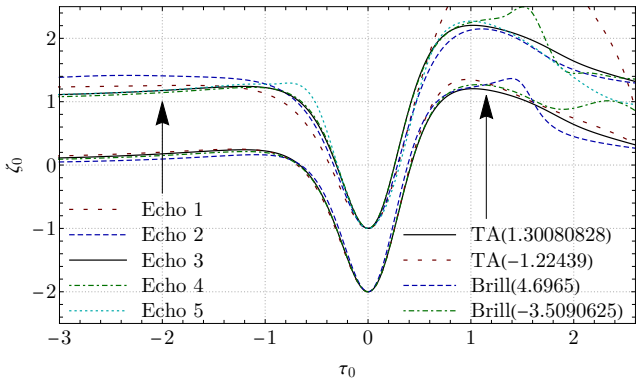

FIG. 3. Profiles of invariant ζ as a function of proper time τ along a timelike worldline through "echo". To compare the profiles, rescaled quantities are used: $\zeta_0 = \lambda^2\zeta$ and $\tau_0 = \tau/\lambda$, where the same scale λ is chosen so that $\min(\zeta_0) = -2$. *Top curves* show shifted value $\zeta_0 + 1$ of five successive "echoes" evolved from initial data (3) with $A_{\mathrm{TA}^+} = \overline{1.30080828}$. *Bottom curves* relate observed profiles of ζ_0 for indicated amplitudes of four different families of initial data.

centrated into irregularly appearing spikes with $\approx 200\times$ larger values of the same *dimensionless* quantity.

To assess the tendency towards DSS behavior, we consider consecutive local spacetime minima ζ_{n-1}, ζ_n separated by the geodesic proper time interval $\tau_n - \tau_{n-1}$. We define $\Delta_n^{(\zeta)} \equiv \log[(\zeta_n/\zeta_{n-1})^{1/2}]$ for the curvature-scale ratio and $\Delta_n^{(\tau)} \equiv \log[(\tau_n - \tau_{n-1})/(\tau_{n+1} - \tau_n)]$ for the time-scale ratio. In a DSS spacetime $\Delta_n^{(\zeta)} = \Delta_n^{(\tau)} = \Delta$. From ζ_n in sub- and super-critical simulations closest to A_*, we obtain $\Delta^{(\zeta)}_{\mathrm{TA}^+} = \{0.68, 0.98, 1.02^{\pm 2}, 1.2^{\pm 3}\}$, $\Delta^{(\zeta)}_{\mathrm{Brill}^-} = \{0.38, 0.68^{\pm 1}, 0.5^{\pm 1}\}$, and $\Delta^{(\zeta)}_{\mathrm{TA}^-} = \{0.55, 0.96^{\pm 4}\}$ for five, four, and three echoes seen for respective ID in Fig. 1. For the time-scales, we get $\Delta^{(\tau)}_{\mathrm{TA}^+} = \{-0.1, 0.4, 2.1^{\pm 1}\}$, $\Delta^{(\tau)}_{\mathrm{Brill}^-} = \{0.3, 1.1^{\pm 1}\}$, and $\Delta^{(\tau)}_{\mathrm{TA}^-} = \{0.1\}$.

Although these numbers seem incompatible with DSS behavior, it is remarkable that while ζ spans the ratio > 400, using the quantity $(\tau_* - \tau)^2\zeta$ devised on self-similarity arguments, this ratio reduces to a factor ~ 3.

To demonstrate the universal shape of the echoes, we notice that they consist of characteristic pairs of negative and positive extrema of the invariant ζ appearing on the z axis separated by a time-like spacetime interval, where the negative peak of ζ of the strongest echo determines $I_K^{\max}$ in Fig. 1. Then we can consider the dependence of ζ on the proper time τ along the geodesic $x^\alpha(\tau)$ connecting these two nearby spacetime extrema. We multiply both τ and ζ by an appropriate power of the same scale factor λ which we fix so that the minimal values of the dimensionless function $\zeta_0(\tau_0) \equiv \lambda^2\zeta(x^\alpha(\lambda\tau_0))$ match.

In Fig. 3, we compare such rescaled profiles of the invariant ζ for different extrema of the same evolution to demonstrate their mutual similarity, and for various initial data families to demonstrate universality. We approximate the geodesic connecting the extrema by the worldline $z = \text{const}$, and to draw the curves, we interpolate the grid values by third-order polynomials. For a generic ID family, the first extreme(s) of ζ will have a different profile. For (1) with $A > 0$, it not only has a different shape, but its amplitude is so high that the next echo does not surpass the already established $I_K^{\max}$. As we see in Fig. 3 the profile of this weaker echo already agrees well with that of a "universal" one. Its segment appears in Fig. 1 at $\log(\sigma I_K^{1/4}) \approx 4$.

A single scalar invariant is not enough to determine the spacetime geometry unambiguously, but because we know that ζ is the only nonvanishing component of the Riemann tensor at the axis, the echoes also represent approximate scaled copies of the same patch of spacetime. Because near its maximum ζ changes only slowly in the z direction, it is interesting that a similar but time-symmetric profile of ζ appears at the axis for the Weber-Wheeler-Bonnor cylindrical GW pulse [24].

Conclusions.—Critical collapse of gravitational waves has been studied for a long time with the hope that a clear, universal, discretely self-symmetric structure will appear. We showed that the first echoes in a near-critical collapse exhibit only a partial similarity to the DSS behavior of a massless scalar field. While we observed a universal profile of the echo forming patches of strongest spacetime curvature as approximate copies of a universal template, these appear with apparently irregular delays and scales. Thus, we did not observe a universal and regularly self-similar solution in the $A \to A_*$ limit, and the dimensionless characteristics of the near-critical behavior seem to depend on the ID family.

We think the observed critical behavior, so distinct from that of spherically symmetric fields, requires further attention. It is natural to focus on the closest neighborhood of the critical amplitudes, but it is possible that even then the nonuniversal aspects of the critical GW collapse will remain, because without the spherical symmetry, ID may leave behind a curved spacetime arena in the vicinity of the accumulation event.

As $A \to A_*$ the numerical simulations become increasingly expensive. It seems important to study the critical behavior of more diverse or more dimensional families of initial data. This may be a computationally cheaper way to understand certain phenomena, e.g., the origins of the apparently irregular echoing structure.

We thank D. Hilditch for interesting discussions, e.g., spotlighting Ref. [22]. This work is supported by the Charles University Project GA UK No. 1176217 and Czech Science Foundation Project GACR 21-11268S. Computational resources were supplied by the project "e-Infrastruktura CZ" (No. e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

* tomas.ledvinka@mff.cuni.cz
† anton@khirnov.net

[1] M. W. Choptuik, Universality and scaling in gravitational collapse of a massless scalar field, Phys. Rev. Lett. **70**, 9 (1993).
[2] C. Gundlach, Critical phenomena in gravitational collapse, Living Rev. Relativity **2**, 4 (1999).
[3] M. Pürrer, S. Husa, and P. C. Aichelburg, News from critical collapse: Bondi mass, tails, and quasinormal modes, Phys. Rev. D **71**, 104005 (2005).
[4] A. M. Abrahams and C. R. Evans, Critical behavior and scaling in vacuum axisymmetric gravitational collapse, Phys. Rev. Lett. **70**, 2980 (1993).
[5] A. M. Abrahams and C. R. Evans, Universality in axisymmetric vacuum collapse, Phys. Rev. D **49**, 3998 (1994).
[6] M. Alcubierre, G. Allen, B. Brügmann, G. Lanfermann, E. Seidel, W.-M. Suen, and M. Tobias, Gravitational collapse of gravitational waves in 3D numerical relativity, Phys. Rev. D **61**, 041501(R) (2000).
[7] D. Garfinkle and G. C. Duncan, Numerical evolution of Brill waves, Phys. Rev. D **63**, 044011 (2001).
[8] O. Rinne, Constrained evolution in axisymmetry and the gravitational collapse of prolate Brill waves, Classical and Quantum Gravity **25**, 135009 (2008).
[9] E. Sorkin, On critical collapse of gravitational waves, Classical and Quantum Gravity **28**, 025011 (2011).
[10] D. Hilditch, T. W. Baumgarte, A. Weyhausen, T. Dietrich, B. Brügmann, P. J. Montero, and E. Müller, Collapse of nonlinear gravitational waves in moving-puncture coordinates, Phys. Rev. D **88**, 103009 (2013).
[11] D. Hilditch, A. Weyhausen, and B. Brügmann, Pseudospectral method for gravitational wave collapse, Phys. Rev. D **93**, 063006 (2016).
[12] D. Hilditch, A. Weyhausen, and B. Brügmann, Evolutions of centered Brill waves with a pseudospectral method, Phys. Rev. D **96**, 104051 (2017).
[13] P. Bizoń, T. Chmaj, and B. G. Schmidt, Critical Behavior in Vacuum Gravitational Collapse in 4+1 Dimensions, Phys. Rev. Lett. **95**, 071102 (2005).
[14] T. W. Baumgarte, C. Gundlach, and D. Hilditch, Critical Phenomena in the Gravitational Collapse of Electromagnetic Waves, Phys. Rev. Lett. **123**, 171103 (2019).
[15] I. Suárez Fernández, R. Vicente, and D. Hilditch, A semilinear wave model for critical collapse, Phys. Rev. D **103**, 044016 (2021).
[16] M. Shibata and T. Nakamura, Evolution of three-dimensional gravitational waves: Harmonic slicing case, Phys. Rev. D **52**, 5428 (1995).
[17] T. W. Baumgarte and S. L. Shapiro, On the numerical integration of einstein's field equations, Phys. Rev. D **59**, 024007 (1998).
[18] F. Löffler, J. Faber, E. Bentivegna, T. Bode, P. Diener, R. Haas, I. Hinder, B. C. Mundim, C. D. Ott, E. Schnetter, G. Allen, M. Campanelli, and P. Laguna, The Einstein Toolkit: a community computational infrastructure for relativistic astrophysics, Classical and Quantum Gravity **29**, 115001 (2012).
[19] A. Khirnov and T. Ledvinka, Slicing conditions for axisymmetric gravitational collapse of Brill waves, Classical and Quantum Gravity **35**, 215003 (2018).
[20] F. Pretorius and M. W. Choptuik, Adaptive mesh refinement for coupled elliptic-hyperbolic systems, Journal of Computational Physics **218**, 246 (2006).
[21] K. Eppley, Evolution of time-symmetric gravitational waves: Initial data and apparent horizons, Phys. Rev. D **16**, 1609 (1977).
[22] H. P. Pfeiffer and J. W. York, Uniqueness and Nonuniqueness in the Einstein Constraints, Phys. Rev. Lett. **95**, 091101 (2005).
[23] H. Stephani, D. Kramer, M. MacCallum, C. Hoenselaers, and E. Herlt, *Exact solutions of Einstein's field equations; 2nd ed.* (Cambridge Univ. Press, Cambridge, 2003).
[24] J. Weber and J. A. Wheeler, Reality of the Cylindrical Gravitational Waves of Einstein and Rosen, Reviews of Modern Physics **29**, 509 (1957).

Bibliography

[1] Andrew M. Abrahams and Charles R. Evans. "Critical behavior and scaling in vacuum axisymmetric gravitational collapse." In: *Phys. Rev. Lett.* 70.20 (1993-05), pp. 2980–2983. DOI: 10.1103/PhysRevLett.70.2980. URL: http://link.aps.org/doi/10.1103/PhysRevLett.70.2980.

[2] Andrew M. Abrahams and Charles R. Evans. "Universality in axisymmetric vacuum collapse." In: *Phys. Rev. D* 49.8 (1994-04), pp. 3998–4003. DOI: 10.1103/PhysRevD.49.3998. URL: https://link.aps.org/doi/10.1103/PhysRevD.49.3998.

[3] M Alcubierre et al. "Test-beds and applications for apparent horizon finders in numerical relativity." In: *Classical and Quantum Gravity* 17.11 (2000-05), pp. 2159–2190. DOI: 10.1088/0264-9381/17/11/301. URL: https://doi.org/10.1088/0264-9381/17/11/301.

[4] M. Alcubierre. *Introduction to 3+1 Numerical Relativity*. Oxford University Press, UK, 2008. ISBN: 978-0-19-920567-7.

[5] Miguel Alcubierre et al. "Gravitational collapse of gravitational waves in 3D numerical relativity." In: *Phys. Rev. D* 61.4 (2000-01), p. 041501. DOI: 10.1103/PhysRevD.61.041501. URL: http://link.aps.org/doi/10.1103/PhysRevD.61.041501.

[6] Miguel Alcubierre et al. "Symmetry without symmetry: Numerical simulation of axisymmetric systems using Cartesian grids." In: *Int. J. Mod. Phys.* D10 (2001), pp. 273–290. DOI: 10.1142/S0218271801000834. arXiv: gr-qc/9908012 [gr-qc].

[7] Daniela Alic et al. "Conformal and covariant formulation of the Z4 system with constraint-violation damping." In: *Phys. Rev. D* 85.6 (2012-03), p. 064040. DOI: 10.1103/PhysRevD.85.064040. URL: http://link.aps.org/doi/10.1103/PhysRevD.85.064040.

[8] E. Anderson et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8 (paperback).

[9] Thomas W. Baumgarte and Stuart L. Shapiro. "On the numerical integration of Einstein's field equations." In: *Phys. Rev. D* 59 (1999), p. 024007. DOI: 10.1103/PhysRevD.59.024007. arXiv: gr-qc/9810065 [gr-qc].

[10] Thomas W. Baumgarte and Stuart L. Shapiro. *Numerical Relativity*. Cambridge University Press, 2010. ISBN: 978-0-521-51407-1.

[11] Marsha J. Berger and Joseph Oliger. "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations." In: *J. Comput. Phys.* 53 (1984), p. 484.

[12] Carles Bona et al. “New Formalism for Numerical Relativity.” In: *Phys. Rev. Lett.* 75.4 (1995-07), pp. 600–603. DOI: 10.1103/PhysRevLett.75.600. URL: https://link.aps.org/doi/10.1103/PhysRevLett.75.600.

[13] John P. Boyd. “Spectral methods using rational basis functions on an infinite interval.” In: *Journal of Computational Physics* 69 (1987), pp. 112–142.

[14] John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications Inc., New York, 2001.

[15] Dieter R. Brill. “On the positive definite mass of the Bondi-Weber-Wheeler time-symmetric gravitational waves.” In: *Annals of Physics* 7.4 (1959), pp. 466–483. ISSN: 0003-4916. DOI: http://dx.doi.org/10.1016/0003-4916(59)90055-7. URL: //www.sciencedirect.com/science/article/pii/0003491659900557.

[16] J. David Brown et al. “Turduckening black holes: an analytical and computational study.” In: *Phys. Rev. D* 79 (2009), p. 044023. DOI: 10.1103/PhysRevD.79.044023. eprint: arXiv:0809.3533[gr-qc].

[17] *Cactus Computational Toolkit*. URL: http://www.cactuscode.org/.

[18] *Cactus Computational Toolkit Prizes*. URL: http://cactuscode.org/media/prizes/.

[19] Carpet: Adaptive Mesh Refinement for the Cactus Framework. URL: http://www.carpetcode.org/.

[20] CESNET. *MetaCentrum NGI*. URL: https://www.metacentrum.cz.

[21] Matthew W. Choptuik. “Universality and scaling in gravitational collapse of a massless scalar field.” In: *Phys. Rev. Lett.* 70.1 (1993-01), pp. 9–12. DOI: 10.1103/PhysRevLett.70.9. URL: http://link.aps.org/doi/10.1103/PhysRevLett.70.9.

[22] Collaborative Effort. *Einstein Toolkit for Relativistic Astrophysics*. Astrophysics Source Code Library. 2011-02. eprint: ascl:1102.014.

[23] *Einstein Toolkit: Open software for relativistic astrophysics*. URL: http://www.einsteintoolkit.org.

[24] Kenneth Eppley. “Evolution of time-symmetric gravitational waves: Initial data and apparent horizons.” In: *Phys. Rev. D* 16.6 (1977-09), pp. 1609–1614. DOI: 10.1103/PhysRevD.16.1609. URL: https://link.aps.org/doi/10.1103/PhysRevD.16.1609.

[25] David Garfinkle and G. Comer Duncan. “Numerical evolution of Brill waves.” In: *Phys. Rev. D* 63.4 (2001-01), p. 044011. DOI: 10.1103/PhysRevD.63.044011. URL: https://link.aps.org/doi/10.1103/PhysRevD.63.044011.

[26] Tom Goodale et al. “The Cactus Framework and Toolkit: Design and Applications.” In: *Vector and Parallel Processing – VECPAR’2002, 5th International Conference, Lecture Notes in Computer Science*. Berlin: Springer, 2003. URL: http://edoc.mpg.de/3341.

[27] Carsten Gundlach and José M. Martín-García. “Well-posedness of formulations of the Einstein equations with dynamical lapse and shift conditions.” In: *Phys. Rev. D* 74.2 (2006-07), p. 024016. DOI: 10.1103/PhysRevD.74.024016. URL: http://link.aps.org/doi/10.1103/PhysRevD.74.024016.

[28] Carsten Gundlach and José M. Martín-García. "Critical Phenomena in Gravitational Collapse." In: *Living Reviews in Relativity* 10.1 (2007-01), p. 5. ISSN: 1433-8351. DOI: 10.12942/lrr-2007-5. URL: http://dx.doi.org/10.12942/lrr-2007-5.

[29] S. W. Hawking and G. F. R. Ellis. *The Large Scale Structure of Space-Time*. Cambridge Monographs on Mathematical Physics. Cambridge University Press, 1973. DOI: 10.1017/CBO9780511524646.

[30] HDF Group. *HDF5 File Format Specification Version 2.0*. 2012. URL: http://www.hdfgroup.org/HDF5/doc/H5.format.html.

[31] David Hilditch et al. "Collapse of nonlinear gravitational waves in moving-puncture coordinates." In: *Phys. Rev. D* 88.10 (2013-11), p. 103009. DOI: 10.1103/PhysRevD.88.103009. URL: http://link.aps.org/doi/10.1103/PhysRevD.88.103009.

[32] David Hilditch, Andreas Weyhausen, and Bernd Brügmann. "Pseudospectral method for gravitational wave collapse." In: *Phys. Rev. D* 93.6 (2016-03), p. 063006. DOI: 10.1103/PhysRevD.93.063006. URL: http://link.aps.org/doi/10.1103/PhysRevD.93.063006.

[33] David Hilditch, Andreas Weyhausen, and Bernd Brügmann. "Evolutions of centered Brill waves with a pseudospectral method." In: *Phys. Rev. D* 96.10 (2017-11), p. 104051. DOI: 10.1103/PhysRevD.96.104051. URL: https://link.aps.org/doi/10.1103/PhysRevD.96.104051.

[34] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–. URL: http://www.scipy.org/.

[35] Anton Khirnov and Tomáš Ledvinka. "Slicing conditions for axisymmetric gravitational collapse of Brill waves." In: *Classical and Quantum Gravity* 35.21 (2018-10), p. 215003. DOI: 10.1088/1361-6382/aae1bc. arXiv: 1908.06034 [gr-qc]. URL: https://doi.org/10.1088/1361-6382/aae1bc.

[36] *Kranc: Kranc Assembles Numerical Code*. URL: http://kranccode.org/.

[37] H.O. Kreiss and J. Oliger. *Methods for the approximate solution of time dependent problems*. GARP publications series. International Council of Scientific Unions, World Meteorological Organization, 1973.

[38] Tomáš Ledvinka and Anton Khirnov. "Universality of Curvature Invariants in Critical Vacuum Gravitational Collapse." In: *Phys. Rev. Lett.* 127.1 (2021), p. 011104. DOI: 10.1103/PhysRevLett.127.011104. arXiv: 2102.09579 [gr-qc].

[39] *McLachlan, a Public BSSN Code*. URL: http://www.cct.lsu.edu/~eschnett/McLachlan/.

[40] *MPI Forum*. URL: http://www.mpi-forum.org/.

[41] *OpenMP®*. URL: http://openmp.org.

[42] OpenMP Architecture Review Board. *OpenMP Application Programming Interface Version 5.1*. 2020. URL: https://www.openmp.org/spec-html/5.1/openmp.html.

[43] Harald P. Pfeiffer and James W. York. "Uniqueness and Nonuniqueness in the Einstein Constraints." In: *Phys. Rev. Lett.* 95.9 (2005-08), p. 091101. DOI: 10.1103/PhysRevLett.95.091101. URL: https://link.aps.org/doi/10.1103/PhysRevLett.95.091101.

[44] William H. Press et al. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

[45] Frans Pretorius. "Evolution of Binary Black-Hole Spacetimes." In: *Phys. Rev. Lett.* 95.12 (2005-09), p. 121101. DOI: 10.1103/PhysRevLett.95.121101. URL: http://link.aps.org/doi/10.1103/PhysRevLett.95.121101.

[46] Frans Pretorius and Matthew W. Choptuik. "Adaptive mesh refinement for coupled elliptic-hyperbolic systems." In: *Journal of Computational Physics* 218.1 (2006), pp. 246–274. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2006.02.011. URL: https://www.sciencedirect.com/science/article/pii/S0021999106000805.

[47] Oliver Rinne. "Constrained evolution in axisymmetry and the gravitational collapse of prolate Brill waves." In: *Classical and Quantum Gravity* 25.13 (2008), p. 135009. URL: http://stacks.iop.org/0264-9381/25/i=13/a=135009.

[48] Erik Schnetter, Scott H. Hawley, and Ian Hawke. "Evolutions in 3-D numerical relativity using fixed mesh refinement." In: *Class. Quantum Grav.* 21 (2004), pp. 1465–1488. DOI: 10.1088/0264-9381/21/6/014. eprint: arXiv:gr-qc/0310042.

[49] Masaru Shibata and Takashi Nakamura. "Evolution of three-dimensional gravitational waves: Harmonic slicing case." In: *Phys. Rev. D* 52.10 (1995-11), pp. 5428–5444. DOI: 10.1103/PhysRevD.52.5428. URL: http://link.aps.org/doi/10.1103/PhysRevD.52.5428.

[50] Evgeny Sorkin. "On critical collapse of gravitational waves." In: *Classical and Quantum Gravity* 28.2 (2011-01), p. 025011. DOI: 10.1088/0264-9381/28/2/025011. URL: https://doi.org/10.1088/0264-9381/28/2/025011.

[51] Saul A. Teukolsky. "Linearized quadrupole waves in general relativity and the motion of test particles." In: *Phys. Rev. D* 26.4 (1982-08), pp. 745–750. DOI: 10.1103/PhysRevD.26.745. URL: http://link.aps.org/doi/10.1103/PhysRevD.26.745.

[52] H. A. van der Vorst. "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems." In: *SIAM Journal on Scientific and Statistical Computing* 13.2 (1992), pp. 631–644. DOI: 10.1137/0913035. eprint: http://dx.doi.org/10.1137/0913035. URL: http://dx.doi.org/10.1137/0913035.