# Event and Apparent Horizon Finders for $3 + 1$ Numerical Relativity

Jonathan Thornburg
Max-Planck-Institut für Gravitationsphysik
Albert-Einstein-Institut
email: jthorn@aei.mpg.de
http//www.aei.mpg.de/~jthorn

Version of 1 January 2006

**Abstract**

Event and apparent horizons are key diagnostics for the presence and properties of black holes. In this article I review numerical algorithms and codes for finding event and apparent horizons in numerically-computed spacetimes, focusing on calculations done using the $3+1$ ADM formalism.

The event horizon of an asymptotically-flat spacetime is the boundary between those events from which a future-pointing null geodesic can reach future null infinity, and those events from which no such geodesic exists. The event horizon is a (continuous) null surface in spacetime. The event horizon is defined *nonlocally in time*: it's a global property of the entire spacetime, and must be found in a separate post-processing phase *after* (part of) the spacetime has been numerically computed.

There are 3 basic algorithms for finding event horizons, based respectively on integrating null geodesics *forwards* in time, integrating null geodesics *backwards* in time, and integrating null *surfaces* backwards in time. The last of these is generally the most efficient and accurate.

In contrast to an event horizon, an apparent horizon is defined locally in time in a spacelike slice, and so can be (and usually is) found "on the fly" during the numerical computation of a spacetime. A marginally outer trapped surface (MOTS) in a slice is a smooth closed 2-surface whose future-pointing outgoing null geodesics have zero expansion $\Theta$. An apparent horizon is then defined as a MOTS not contained in any other MOTS. The MOTS condition is a nonlinear elliptic partial differential equation (PDE) for the surface shape, containing the ADM 3-metric, its spatial derivatives, and the extrinsic curvature as coefficients. Most "apparent-horizon" finders actually find MOTSs.

There are a large number of apparent-horizon finding algorithms, with differing trade-offs between speed, robustness, accuracy, and ease of programming. In axisymmetry, shooting algorithms work well and are fairly easy to program. In slices with no continuous symmetries, Nakamura *et al.*'s algorithm and elliptic-PDE algorithms are fast and accurate, but require good initial guesses to converge. In many cases, Schnetter's "pretracking" algorithm can greatly improve an elliptic-PDE algorithm's robustness. Flow algorithms are generally quite slow, but can be very robust in their convergence.

# Contents

# Chapter 1

# Introduction

Compact objects – ones which may contain event horizons and/or apparent horizons – are a major focus of numerical relativity. The usual output of a numerical relativity simulation is some (approximate, discrete) representation of the spacetime geometry (the 4-metric and possibly its derivatives) and any matter fields, but *not* any explicit information about the existence, precise location, or other properties of any event/apparent horizons. To gain this information, we must explicitly *find* the horizons from the numerically-computed spacetime geometry. The subject of this review is numerical algorithms and codes for doing this, focusing on calculations done using the $3+1$ ADM formalism ([12, 149]).[1] Baumgarte and Shapiro [23, section 6] have also recently reviewed event and apparent-horizon finding algorithms.

In this review I distinguish between a numerical *algorithm* (an abstract description of a mathematical computation; also often known as a "method" or "scheme"), and a computer *code* (a "horizon finder", a specific piece of computer software which implements a horizon-finding algorithm or algorithms). My main focus is on the algorithms, but I also mention specific codes where they are freely available to other researchers.

In this review I have tried to cover all the major horizon-finding algorithms and codes, and to accurately credit the earliest publication of important ideas. However, in a field as large and active as numerical relativity, it's inevitable that I have overlooked and/or misdescribed some important research. I apologise to anyone whose work I've slighted, and I ask readers to help make this a truly "living" review by sending me corrections, updates, and/or pointers to additional work (either their own or others) which I should discuss in future revisions of this review.

The general outline of this review is as follows: In the remainder of this chapter I define notation and terminology (section 1.1), discuss how 2-surfaces should be

[1] There are many interesting uses of event and/or apparent horizons in gaining physical understanding of numerically-computed spacetimes. However, a discussion of these applications would encompass much of strong-field numerical relativity, and would be be far beyond the scope of this review.

parameterized (section 1.2), and outline some of the software-engineering issues that arise in modern numerical relativity codes (section 1.3). I then discuss numerical algorithms and codes for finding event horizons (chapter 2) and apparent horizons (chapter 3). Finally, in the appendices I briefly outline some of the excellent numerical algorithms/codes available for two standard problems in numerical analysis, the solution of a single nonlinear algebraic equation (appendix A) and the time integration of a system of ordinary differential equations (appendix B).

## 1.1 Notation and Terminology

I generally follow the sign and notation conventions of Wald [145]. I assume that all spacetimes are globally hyperbolic, and for event-horizon finding I further assume asymptotic flatness; in this latter context $\mathcal{J}^+$ is future null infinity. I use the Penrose abstract-index notation, with summation over all repeated indices. 4-indices $abc$ range over all spacetime coordinates $\{x^a\}$, and 3-indices $ijk$ range over the spatial coordinates $\{x^i\}$ in a spacelike slice $t =$ constant. The spacetime coordinates are thus $x^a = (t, x^i)$.

$g_{ab}$ is the spacetime 4-metric, and $g^{ab}$ the inverse spacetime 4-metric; these are used to raise and lower 4-indices. $\Gamma^c_{ab}$ are the 4-Christoffel symbols. $\mathcal{L}_v$ is the Lie derivative along the 4-vector field $v^a$.

$g_{ij}$ is the 3-metric defined in a slice, and $g^{ij}$ is the inverse 3-metric; these are used to raise and lower 3-indices. $\nabla_i$ is the associated 3-covariant derivative operator, and $\Gamma^k_{ij}$ are the 3-Christoffel symbols. $\alpha$ and $\beta^i$ are the $3+1$ lapse function and shift vector respectively,[2] so the spacetime line element is

$$\begin{aligned} ds^2 &= g_{ab}\, dx^a\, dx^b & (1.1)\\ &= -(\alpha^2 - \beta_i\beta^i)\, dt^2 + 2\beta_i\, dx^i\, dt + g_{ij}\, dx^i\, dx^j & (1.2) \end{aligned}$$

As is common in $3+1$ numerical relativity, I follow the sign convention of Misner, Thorne, and Wheeler [101] and York [149] in defining the extrinsic curvature of the slice as $K_{ij} = -\frac{1}{2}\mathcal{L}_n g_{ij} = -\nabla_i n_j$, where $n^a$ is the future-pointing unit normal to the slice. (In contrast, Wald [145] omits the minus signs from this definition.) $K \equiv K_i{}^i$ is the trace of the extrinsic curvature $K_{ij}$. $m_{\rm ADM}$ is the ADM mass of an (asymptotically flat) slice.

Indices $uvw$ range over generic angular coordinates $(\theta, \phi)$ on $S^2$ or on a horizon surface. Note that these coordinates are conceptually completely distinct from the 3-dimensional spatial coordinates $x^i$. Depending on the context, $(\theta, \phi)$ may or may not have the usual polar-spherical topology.

Indices IJK label angular grid points on $S^2$ or on a horizon surface. Notice that these are *2-dimensional* indices: a *single* such index uniquely specifies an angular

[2] See York [149] for a general overview of the $3+1$ formalism as it's used in numerical relativity.

grid point. $\delta_{\mathrm{IJ}}$ is the Kronecker delta on the space of these indices, or equivalently on surface grid points.

I often write a differential operator as $F = F(y, \partial_u y, \partial_{uv} y; g_{ij}, \partial_k g_{ij}, K_{ij})$. The ";" notation means that $F$ is a (generally nonlinear) algebraic function of the variable $y$ and its 1st and 2nd angular derivatives, and that $F$ also depends on the coefficients $g_{ij}$, $\partial_k g_{ij}$, and $K_{ij}$ at the apparent horizon position.

There are 3 common types of spacetimes/slices where numerical event or apparent horizon finding is of interest: spherically-symmetric spacetimes/slices, axisymmetric spacetimes/slices, and spacetimes/slices with no continuous spatial symmetries (no spacelike Killing vectors). I refer to the latter as "fully generic" spacetimes/slices.

In this review I use the abbreviations "ODE" for ordinary differential equation, "PDE" for partial differential equation, "CE surface" for constant-expansion surface, and "MOTS" for marginally outer trapped surface. Names in Small Capitals refer to horizon finders and other computer software.

When discussing iterative numerical algorithms, it's often useful to use the concept of an algorithm's "radius of convergence": Suppose the solution space within which the algorithm is iterating is $\mathsf{S}$. Then given some norm $\|\cdot\|$ on $\mathsf{S}$, the algorithm's radius of convergence about a solution $\mathsf{s} \in \mathsf{S}$ is defined as the smallest $r > 0$ such that the algorithm will converge to the correct solution $\mathsf{s}$ for any initial guess $\mathsf{g}$ with $\|\mathsf{g} - \mathsf{s}\| \leq r$. We only rarely know the exact radius of convergence of an algorithm, but practical experience often provides a rough estimate.[3]

## 1.2 2-Surface Parameterizations

### 1.2.1 Level-Set–Function Parameterizations

The most general way to parameterize a 2-surface in a slice is to define a scalar "level-set function" $F$ on some neighborhood of the surface, with the surface itself then being defined as the level set

$$F = 0 \qquad \text{on the surface} \tag{1.3}$$

Assuming the surface to be orientable, it's conventional to choose $F$ so that $F > 0$ ($F < 0$) outside (inside) the surface.

This parameterization is valid for any surface topology, including time-dependent topologies. The 2-surface itself can then be found by a standard isosurface-finding algorithm such as the marching-cubes algorithm [94]. (This algorithm is widely used

[3] An algorithm's actual "convergence region" (the set of all initial guesses for which the algorithm converges to the correct solution) may even be fractal in shape. For example, the Julia set is the convergence region of Newton's method on a simple nonlinear algebraic equation.

in computer graphics, and is implemented in a number of widely-available software libraries.)

### 1.2.2 Strahlkörper Parameterizations

Most apparent-horizon finders, and many event-horizon finders, assume that each connected component of the apparent (event) horizon has $S^2$ topology. With the exception of toroidal event horizons (discussed in section 2.1), this is generally a reasonable assumption.

To parameterize an $S^2$ surface's shape, it's common to further assume that we are given (or can compute) some "local coordinate origin" point inside the surface such that the surface's 3-coordinate shape relative to that point is a "Strahlkörper", (literally "ray body", or more commonly "star-shaped region") defined by Minkowski ([123, p. 108]) as

> a region in $n$-D Euclidean space containing the origin and whose surface, as seen from the origin, exhibits only one point in any direction.

The Strahlkörper assumption is a significant restriction on the horizon's coordinate shape (and the choice of the local coordinate origin). For example, it rules out the coordinate shape and local coordinate origin illustrated in figure 1.1: a horizon with such a coordinate shape about the local coordinate origin couldn't be found by any horizon finder which assumes a Strahlkörper surface parameterization.

For event-horizon finding, algorithms and codes are now available which allow an arbitrary horizon topology, with no Strahlkörper assumption (see the discussion in section 2.2.3.3 for details). For apparent-horizon finding, the flow algorithms discussed in section 3.2.7 theoretically allow any surface shape, although many implementations still make the Strahlkörper assumption. Removing this assumption for other apparent-horizon finding algorithms might be a fruitful area for further research.

Given the Strahlkörper assumption, the surface can be explicitly parameterized as

$$r = h(\theta, \phi) \tag{1.4}$$

where $r$ is the Euclidean distance from the local coordinate origin to a surface point, $(\theta, \phi)$ are generic angular coordinates on the horizon surface (or equivalently on $S^2$), and the "horizon shape function" $h : S^2 \to \Re^+$ is a positive real-valued function on the domain of angular coordinates defining the surface shape.

There are two common discretizations of this surface representation:

**Spectral representation**

Here we expand the horizon shape function $h$ in an infinite series in some

(typically orthonormal) set of basis functions such as spherical harmonics $Y_{\ell m}$ or symmetric trace-free tensors,[4]

$$h(\theta, \phi) = \sum_{\ell,m} a_{\ell m} Y_{\ell m}(\theta, \phi) \tag{1.5}$$

This series can then be truncated at some finite order $\ell_{\max}$, and the $N_{\text{coeff}} = \ell_{\max}(\ell_{\max}+1)$ coefficients $\{a_{\ell m}\}$ used to represent (discretely approximate) the horizon shape. For reasonable accuracy, $\ell_{\max}$ is typically on the order of 8 to 12.

**Finite difference representation**

Here we choose some finite grid of angular coordinates $\{(\theta_{\text{K}}, \phi_{\text{K}})\}$, $\text{K} = 1, 2, 3, \ldots, N_{\text{ang}}$ on $S^2$ (or equivalently on the surface),[5] and represent (discretely approximate) the surface shape by the $N_{\text{ang}}$ values

$$\left\{h(\theta_{\text{K}}, \phi_{\text{K}})\right\} \qquad \text{K} = 1, 2, 3, \ldots, N_{\text{ang}} \tag{1.6}$$

For reasonable accuracy, $N_{\text{ang}}$ is typically on the order of a few thousand.

[4]For convenience of exposition I use spherical harmonics here, but there are no essential differences if other basis sets are used.

[5]I discuss the choice of this angular grid in more detail in section 3.2.5.1.

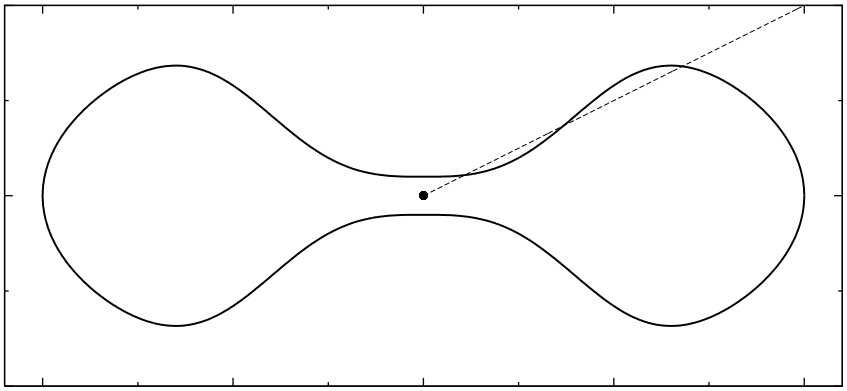

Figure 1.1: This figure shows a cross-section of a coordinate shape (the thick curve) which isn't a Strahlkörper about the local coordinate origin shown (the large dot). The dashed line shows a ray from the local coordinate origin, which intersects the surface in more than one point.

It's sometimes useful to explicitly construct a level-set function describing a given Strahlkörper. A common choice here is

$$F \equiv r - h(\theta, \phi) \tag{1.7}$$

### 1.2.3 Finite-Element Parameterizations

Another way to parameterize a 2-surface is via finite elements, where the surface is modelled as a triangulated mesh, i.e. as a set of interlinked "vertices" (points in the slice, represented by their spatial coordinates $\{x^i\}$), "edges" (represented by ordered pairs of vertices), and faces. Typically only triangular faces are used (represented as oriented triples of vertices).

A key benefit of this representation is that it allows an arbitrary topology for the surface. However, determining the actual surface topology (e.g. testing for whether or not the surface self-intersects) is somewhat complicated.

This representation is similar to that of Regge calculus [114, 64],[6] and can similarly be expected to show 2nd order convergence with the surface resolution.

Finite element surface representations have been used for apparent-horizon finding by Metzger [98].

## 1.3 Software-Engineering Issues

Historically, numerical relativists wrote their own codes from scratch. As these became more complex, many researchers changed to working on "group codes" with multiple contributors.

### 1.3.1 Software Libraries and Toolkits

More recently, particularly in work on fully generic spacetimes, where all 3 spatial dimensions must be treated numerically, there has been a strong trend towards the use of higher-level software libraries and modular "computational toolkits" such as Cactus [66] (`http://www.cactuscode.org`). These have a substantial learning overhead, but can allow researchers to work much more productively by focusing more on numerical relativity, instead of computer-science and software-engineering issues such as parameter-file parsing, parallelization, I/O, etc.

[6] There has been some controversy over whether, and if so how quickly, Regge calculus converges to the continuum Einstein equations. (See, for example, the debate between Brewin [36] and Miller [99], and the explicit convergence demonstration of Gentle and Miller [65].) However, Brewin and Gentle [37] seem to have resolved this: Regge calculus does in fact converge to the continuum solution, and this convergence is generically 2nd order in the resolution.

A particularly important area for such software infrastructure is mesh refinement.[7] This is essential to much current numerical-relativity research, but is moderately difficult to implement even in only one spatial dimension, and much harder in multiple spatial dimensions. There are now a number of software libraries providing multi-dimensional mesh-refinement infrastructure (sometimes combined with parallelization), such as DAGH/GrACE [107] (`http://www.caip.rutgers.edu/~parashar/DAGH/`) and ParaMesh [95] (`http://ct.gsfc.nasa.gov/paramesh/Users_manual/amr.html`). The Cactus toolkit can be used in either unigrid or mesh-refinement modes, the latter using a "mesh-refinement driver" such as PAGH or Carpet [120, 117] (`http://www.carpetcode.org`).

In this review I point out event and apparent-horizon finders which have been written in particular frameworks, and comment on whether they work with mesh refinement.

### 1.3.2 Code Reuse and Sharing

Another issue is that of code reuse and sharing. It's common for codes to be shared within a research group, but relatively uncommon for them to be shared between different (competing) research groups. Even apart from concerns about competitive advantage, without a modular structure and clear documentation it's difficult to reuse another group's code. The use of a common computational toolkit can greatly simplify such reuse.

If such reuse *can* be accomplished, it becomes much easier for other researchers to build on existing work, rather than having to "reinvent the wheel". As well as the obvious ease of reusing existing code that (hopefully!) already works and has been thoroughly debugged and tested, there's another – less obvious – benefit of code sharing: It greatly eases the replication of past work, which is essential as a foundation for new development. That is, without access to another researcher's code, it can be surprisingly difficult to replicate her results, because the success or failure of a numerical algorithm frequently depends on subtle implementation details not described in even the most complete of published papers.

Event and apparent-horizon finders are excellent candidates for software reuse: Many numerical-relativity researchers can benefit from using them, and they have a relatively simple interface to an underlying numerical-relativity simulation. Even if a standard computational toolkit isn't used, this makes it relatively easy to port an event or apparent-horizon finder to a different code.

Throughout this review I note event and apparent-horizon finders which are freely available to other researchers.

[7] See, for example, Choptuik [44], Pretorius [113], and Pretorius and Choptuik [112] for general surveys of the uses of, and methods for, mesh refinement in numerical relativity.

### 1.3.3 Using Multiple Event/Apparent Horizon Finders

It's often useful to have multiple event or apparent-horizon finders available: their strengths and weaknesses may complement each other, and the extent of (dis)agreement between their results can give a good measure of the numerical accuracy. For example, figure 3.3 shows a comparison between the irreducible masses of apparent horizons in a binary black hole coalescence simulation (Alcubierre *et al.* [5, figure 4b]), as computed by two different apparent-horizon finders in the Cactus toolkit, AHFinder and AHFinderDirect. In this case the two agree to within about 2% for the individual horizons, and 0.5% for the common horizon.

# Chapter 2

# Finding Event Horizons

## 2.1 Introduction

The black hole region of an asymptotically-flat spacetime is defined ([73, 74]) as the set of events from which no future-point null geodesic can reach future null infinity ($\mathcal{J}^+$). The event horizon is defined as the boundary of the black hole region. The event horizon is a null surface in spacetime with (in the words of Hawking and Ellis [74, p. 319]) "a number of nice properties" for studying the causal stucture of spacetime.

The event horizon is a *global* property of an entire spacetime, and is defined *nonlocally in time*: the event horizon in a slice is defined in terms of (and can't be computed without knowing) the full *future* development of that slice.

In practice, to find an event horizon in a numerically-computed spacetime, we typically instrument a numerical evolution code to write out data files of the 4-metric. After the evolution has reached an approximately-stationary final state, we then compute a numerical approximation to the event horizon in a separate post-processing pass, using the 4-metric data files as inputs.

As a null surface, the event horizon is necessarily continuous. In theory it need not be *anywhere* differentiable,[1] but in practice this behavior rarely occurs:[2] The event horizon is generally smooth except for possibly a finite set of "cusps" where new generators join the surface; the surface normal has a jump discontinuity across each cusp. (The classic example of such a cusp is the "inseam" of the "pair of pants" event horizon illustrated in figures 2.3 and 2.4.)

A black hole is defined as a connected component of the black hole region in a $3 + 1$ slice. The boundary of a black hole (the event horizon) in a slice is a

[1] Chruściel and Galloway [45] showed that if a "cloud of sand" falls into a large black hole, each "sand grain" generates a non-differentiable caustic in the event horizon.

[2] This is a statement about the types of spacetimes usually studied by numerical relativists, not a statement about the mathematical properties of the event horizon itself.

2-dimensional set of events. Usually this has 2-sphere ($S^2$) topology. However, numerically simulating rotating dust collapse, Abrahams *et al.* [1] found that in some cases the event horizon in a slice may be *toroidal* in topology. Lehner *et al.* [89] and Husa and Winicour [81] have used null (characteristic) algorithms to give a general analysis of the event horizon's topology in black hole collisions; they find that there is generically a (possibly brief) toroidal phase before the final 2-spherical state is reached. Lehner *et al.* [90] later calculated movies showing this behavior for several asymmetric black hole collisions.

## 2.2 Algorithms and Codes for Finding Event Horizons

There are 3 basic event-horizon finding algorithms:

- Integrate null geodesics *forwards* in time (section 2.2.1).
- Integrate null geodesics *backwards* in time (section 2.2.2).
- Integrate null *surfaces* backwards in time (section 2.2.3).

I describe these in detail in the following subsections.

### 2.2.1 Integrating Null Geodesics Forwards in Time

The first generation of event-horizon finders were based directly on Hawking's original definition of an event horizon: an event $\mathcal{P}$ is within the black hole region of spacetime if and only if there is no future-pointing "escape route" null geodesic from $\mathcal{P}$ to $\mathcal{J}^+$; the event horizon is the boundary of the black hole region.

That is, as described by Hughes *et al.* [78], we numerically integrate the null geodesic equation

$$\frac{d^2x^a}{d\lambda^2} + \Gamma^a_{bc}\frac{dx^a}{d\lambda}\frac{dx^b}{d\lambda} = 0 \tag{2.1}$$

(where $\lambda$ is an affine parameter) forwards in time from a set of starting events and check which events have "escaping" geodesics. For analytical or semi-analytical studies like that of Bishop [27], this is an excellent algorithm.

For numerical work it's straightforward to rewrite the null geodesic equation (2.1) as a coupled system of two first-order equations, giving the time evolution of photon positions and 3-momenta in terms of the $3+1$ geometry variables $\alpha$, $\beta^i$, $g^{ij}$, and their spatial derivatives. These can then be time-integrated by standard numerical algorithms.[3] However, in practice several factors complicate this algorithm:

[3] I briefly review ODE integration algorithms and codes in appendix B.

We typically only know the $3+1$ geometry variables on a discrete lattice of spacetime grid points, and we only know the $3+1$ geometry variables themselves, not their spatial derivatives. Therefore we must numerically differentiate the field variables, and interpolate the field variables and their spacetime derivatives to each integration point along each null geodesic. This is straightforward to implement,[4] but the numerical differentiation tends to amplify any numerical noise that may be present in the field variables.

Another complicating factor is that the numerically computations generally only span a finite region of spacetime, so it's not entirely obvious whether or not a given geodesic will eventually reach $\mathcal{J}^+$. However, if the final numerically-generated slice contains an apparent horizon, we can use this as an approximation: any geodesic which is inside this apparent horizon will definitely *not* reach $\mathcal{J}^+$, while any other geodesic may be assumed to eventually reach $\mathcal{J}^+$ if its momentum is directed away from the apparent horizon. If the final slice is approximately stationary, the error from this approximation should be small. (I discuss the "final slice is approximately stationary" assumption further in section 2.2.3.1.)

#### 2.2.1.1 Spherically-Symmetric Spacetimes

In spherical symmetry this algorithm works well, and has been used by a number of researchers. For example, Shapiro and Teukolsky [126, 127, 128, 129] used it to study event horizons in a variety of dynamical evolutions of spherically symmetric collapse systems. Figure 2.1 shows an example of the event and apparent horizons in one of these simulations.

#### 2.2.1.2 Non–Spherically-Symmetric Spacetimes

In a non–spherically-symmetric spacetime, several factors make this algorithm very inefficient:

- Many trial events must be tried to accurately resolve the event horizon's shape. (Hughes *et al.* [78] describe a 2-stage adaptive numerical algorithm for choosing the trial events so as to accurately locate the event horizon as efficiently as possible.)

- At each trial event we must try many different trial-geodesic starting directions to see if any of the geodesics escape to $\mathcal{J}^+$ (or our numerical approximation to it). Hughes *et al.* [78] report needing only 48 geodesics per trial event in several nonrotating axisymmetric spacetimes, but about 750 geodesics per trial event in rotating axisymmetric spacetimes, with up to 3000 geodesics per trial event in some regions of the spacetimes.

[4] In practice the differentiation can usefully be combined with the interpolation; I outline how this can be done in section 3.1.5.

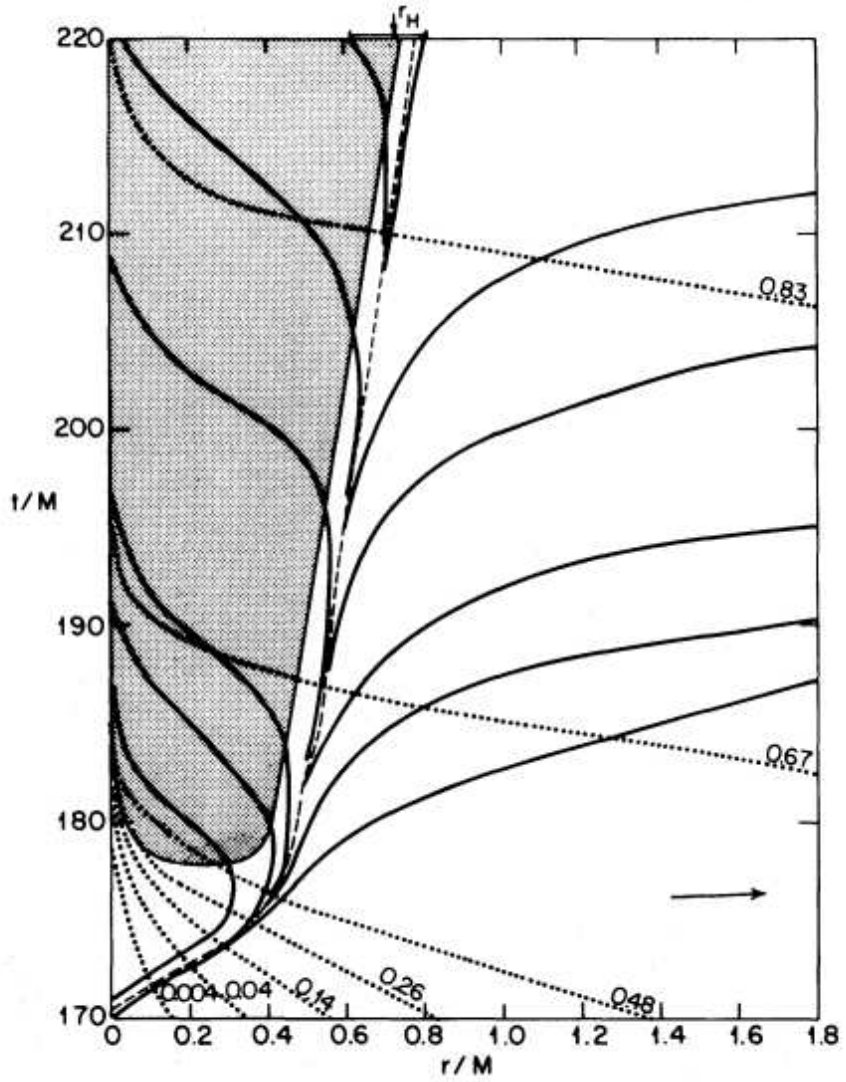


Figure 2.1: This figure shows part of a simulation of the spherically symmetric collapse of a model stellar core (a $\Gamma = \frac{5}{3}$ polytrope) to a black hole. The event horizon (shown by the dashed line) was computed using the "integrate null geodesics forwards" algorithm described in section 2.2.1; solid lines show outgoing null geodesics. The apparent horizon (the boundary of the trapped region, shown shaded) was computed using the zero-finding algorithm discussed in section 3.2.1. The dotted lines show the world lines of Lagrangian matter tracers, and are labeled by the fraction of baryons interior to them. Figure reprinted with permission from Shapiro and Teukolsky, *The Astrophysical Journal* **235**, 199–215 (1980). Copyright 1980 by the American Astronomical Society.

- Finally, each individual geodesic integration requires many (short) time steps for an accurate integration, particularly in the strong-field region near the event horizon.

Because of these limitations, for non–spherically-symmetric spacetimes the "integrate null geodesics forwards" algorithm has generally been supplanted by the more efficient algorithms I describe in the following sections.

### 2.2.2 Integrating Null Geodesics Backwards in Time

It's well-known that future-pointing outgoing null geodesics near the event horizon tend to diverge exponentially in time away from the event horizon. Figure 2.2 illustrates this behavior for Schwarzschild spacetime, but the behavior is actually quite generic.

Anninos *et al.* [6] and Libson *et al.* [93] observed that while this instability is a problem for the "integrate null geodesics forwards in time" algorithm (it forces that algorithm to take quite short time steps when integrating the geodesics), we can turn it to our advantage by integrating the geodesics *backwards* in time: the geodesics will now *converge* on to the horizon.[5]

This event-horizon finding algorithm is thus to integrate a large number of such (future-pointing outgoing) null geodesics backwards in time, starting on the final numerically-generated slice. As the backwards integration proceeds, even geodesics which started far from the event horizon will quickly converge to it. This can be seen, for example, in figures 2.1 and 2.2.

Unfortunately, this convergence property holds only for *outgoing* geodesics. In spherical symmetry the distinction between outgoing and ingoing geodesics is trivial, but as described by Libson *et al.* [93],

> [...] for the general 3D case, when the two tangential directions of the EH are also considered, the situation becomes more complicated. Here normal and tangential are meant in the 3D spatial, not spacetime, sense. Whether or not a trajectory can eventually be "attracted" to the EH, and how long it takes for it to become "attracted," depends on the photon's starting direction of motion. We note that even for a photon which is already exactly on the EH at a certain instant, if its velocity at that point has some component tangential to the EH surface as generated by, say, numerical inaccuracy in integration, the photon will move outside of the EH when traced backward in time. For a small tangential velocity,

[5] This convergence is only true in a global sense: locally the event horizon has no special geometric properties, and the Riemann tensor components which govern geodesic convergence/divergence may have either sign.

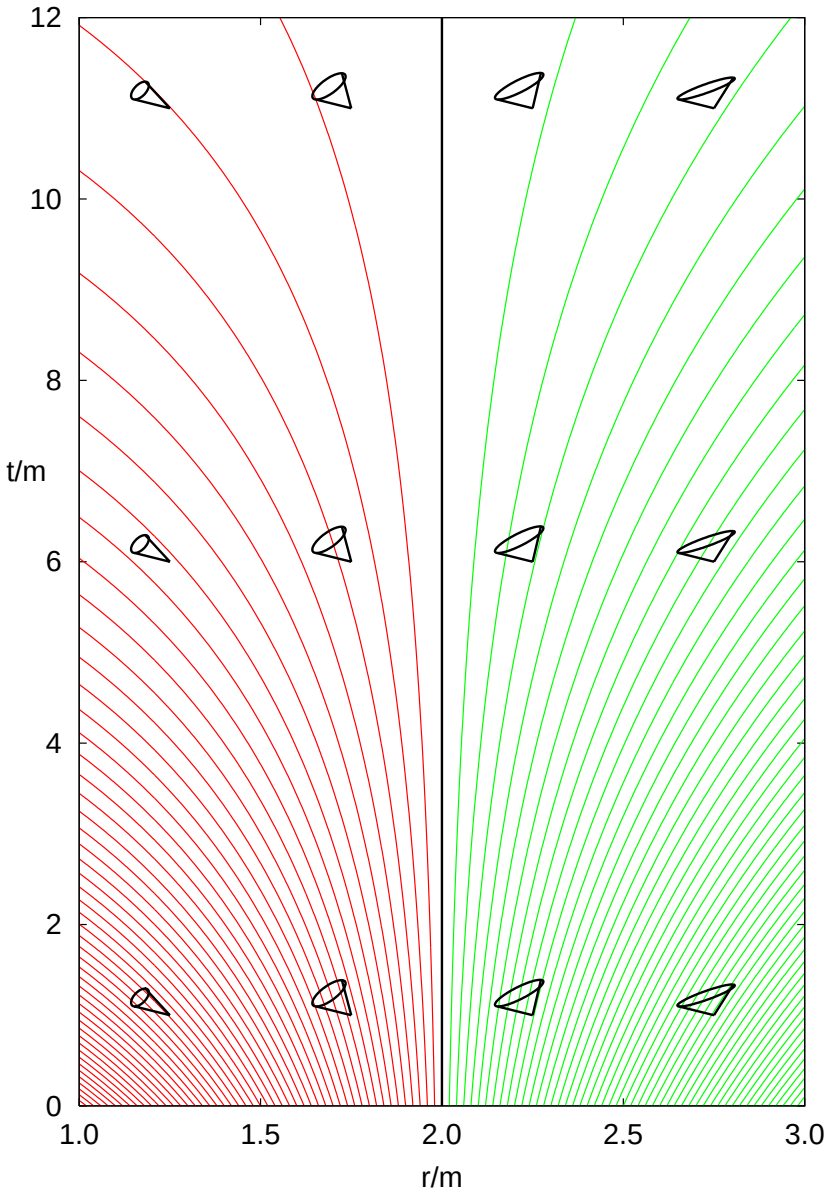


Figure 2.2: This figure shows a number of light cones and future-pointing outgoing null geodesics in a neighborhood of the event horizon in Schwarzschild spacetime, plotted in ingoing Eddington-Finkelstein coordinates $(t, r)$. (These coordinates are defined by the conditions that $t + r$ is an ingoing null coordinate, while $r$ is an areal radial coordinate.) The geodesics outside the event horizon are shown in green; those inside the event horizon are shown in red. All the geodesics start out close together near the event horizon; they diverge away from each other exponentially in time (here with an $e$-folding time of $4m$ near the horizon). Equivalently, they *converge* towards each other if integrated *backwards* in time (downwards on the page).

> the photon will eventually return to the EH [... but] the position to which it returns will not be the original position.
>
> This kind of tangential drifting is undesirable not just because it introduces inaccuracy in the location of the EH, but more importantly, because it can lead to spurious dynamics of the "EH" thus found. Neighboring generators may cross, leading to numerically artificial caustic points [...].

Libson *et al.* [93] also observe that

> Another consequence of the second order nature of the geodesic equation is that not just the positions but also the directions must be specified in starting the backward integration. Neighboring photons must have their starting direction well correlated in order to avoid tangential drifting across one another.

Libson *et al.* [93] give examples of the numerical difficulties that can result from these difficulties, and conclude that this event-horizon finding algorithm

> [...] is still quite demanding in finding an accurate history of the EH, although the difficulties are much milder than those arising from the instability of integrating forward in time.

Because of this difficulty, this algorithm has generally been supplanted by the "backwards surface" algorithm I describe in the next section.

### 2.2.3 Integrating Null Surfaces Backwards in Time

Anninos *et al.* [6], Libson *et al.* [93], and Walker [147] introduced the important concept of explicitly (numerically) finding the event horizon as a null *surface* in spacetime. They observed that if we parameterize the event horizon with a (any) level-set function $F$ satisfying (1.3), then the condition for the surface $F = 0$ to be null is just

$$g^{ab}\partial_a F \partial_b F = 0 \tag{2.2}$$

Applying a $3+1$ decomposition to this then gives a quadratic equation which can be solved to find the time evolution of the level-set function,

$$\partial_t F = \frac{-g^{ti}\partial_i F + \sqrt{(g^{ti}\partial_i F)^2 - g^{tt}g^{ij}\partial_i f \partial_j f}}{g^{tt}} \tag{2.3}$$

Alternatively, assuming the event horizon in each slice to be a Strahlkörper in the manner of section 1.2.2, we can define a suitable level-set function $F$ by (1.7).

Substituting this definition into (2.3) then gives an explicit evolution equation for the horizon shape function,

$$\partial_t h = \frac{-g^{tr} + g^{ru}\partial_u h + \sqrt{(g^{tr} - g^{tu}\partial_u h)^2 - g^{tt}\left(g^{rr} - 2g^{ru}\partial_u h + g^{uv}\partial_u h \partial_v h\right)}}{g^{tt}} \quad (2.4)$$

Surfaces near the event horizon share the same "attraction" property discussed in section 2.2.2 for geodesics near the event horizon. Thus by integrating either surface representation (2.3) or (2.4) backwards in time, we can refine an initial guess into a very accurate approximation to the event horizon.

Notice that in contrast to the null geodesic equation (2.1), neither (2.3) nor (2.4) contain any derivatives of the 4-metric (or equivalently the 3+1 geometry variables). This makes it much easier to integrate these latter equations accurately.[6]

This formulation of the event-horizon finding problem also completely eliminates the tangential-drifting problem discussed in section 2.2.2, since the level-set function only parameterizes motion normal to the surface.

#### 2.2.3.1 Error Bounds: Integrating a Pair of Surfaces

For a practical algorithm, it's useful to integrate a *pair* of trial null surfaces backwards, an "inner-bound" one which starts (and thus always is) inside the event horizon, and an "outer-bound" one which starts (and thus always is) outside the event horizon. If the final slice contains an apparent horizon, then any 2-surface inside this can serve as our inner-bound surface. However, choosing an outer-bound surface is more difficult.

It's this desire for a reliable outer bound on the event horizon position that motivates our requirement for the final slice to be approximately stationary, since (in the absence of time-dependent equations of state or external perturbations entering the system) this ensures that, for example, any surface substantially outside the apparent horizon can serve as an outer-bound surface.

Assuming we have an inner- and an outer-bound surface on the final slice, the spacing between these two surfaces after some period of backwards integration then gives a reliable error bound for the computed event horizon position. Equivalently, a necessary (and, if there are no other numerical problems, sufficient) condition for the event-horizon finding algorithm to be accurate is that the backwards integration must have proceeded far enough for the spacing between the two trial surfaces to be "small". For a reasonable definition of "small", this typically takes at least $15m_{\mathrm{ADM}}$ of backwards integration, with $20m_{\mathrm{ADM}}$ or more providing much higher accuracy.

In some cases it's difficult to obtain a long enough span of numerical data for this backwards integration. For example, in many simulations of binary black hole

[6] Diener [54] describes how the algorithm can be enhanced to also determine (integrate) individual null generators of the event horizon. This requires interpolating the 4-metric to the generator positions, but (still) not taking any derivatives of the 4-metric.

collisions, the evolution becomes unstable and crashes soon after a common apparent horizon forms. This means that we can't compute an accurate event horizon for the most interesting region of the spacetime, that which is close to the black-hole merger. There's no good solution to this problem except for the obvious one of developing a stable (or less-unstable) simulation that can be continued for a longer time.

#### 2.2.3.2 Explicit Strahlkörper Surface Representation

The initial implementations of the "integrate null surface backwards" algorithm by Anninos *et al.* [6], Libson *et al.* [93], and Walker [147] were based on the explicit Strahlkörper surface integration formula (2.4), further restricted to axisymmetry.[7]

For a single black hole the coordinate choice is straightforward. For the two–black-hole case, the authors used topologically cylindrical coordinates $(\rho, z, \phi)$, where the two black holes collide along the axisymmetry ($z$) axis. Based on the symmetry of the problem, they then assumed that the event horizon shape could be written in the form

$$\rho = h(z) \tag{2.5}$$

in each $t =$ constant slice.

This spacetime's event horizon has the now-classic "pair of pants" shape, with a non-differentiable cusp along the "inseam" (the $z$ axis $\rho = 0$) where new generators join the surface. The authors tried two ways of treating this cusp numerically:

- Since the cusp's location is known *a priori*, it can be treated as a special case in the angular finite differencing, using one-sided numerical derivatives as necessary.

- Alternatively, Thorne [141] suggested calculating the *union* of the event horizon and all its null generators (including those which haven't yet joined the surface). This "surface" has a complicated topology (it self-intersects along the cusp), but it's smooth everywhere. This is illustrated by figure 2.3, which shows a cross-section of this surface in a single slice, for a head-on binary black hole collision. For comparison, figure 2.4 shows a perspective view of part of the event horizon and some of its generators, for a similar head-on binary black hole collision.

Caveny *et al.* [40, 42] implemented the "integrate null surfaces backwards" algorithm for fully generic numerically-computed spacetimes, using the explicit Strahlkörper surface integration formula (2.4). To handle moving black holes, they recentered each black hole's Strahlkörper parameterization (1.4) on the black hole's coordinate centroid at each time step.

[7] Walker [147] mentions an implementation for fully generic slices, but only presents results for the axisymmetric case.

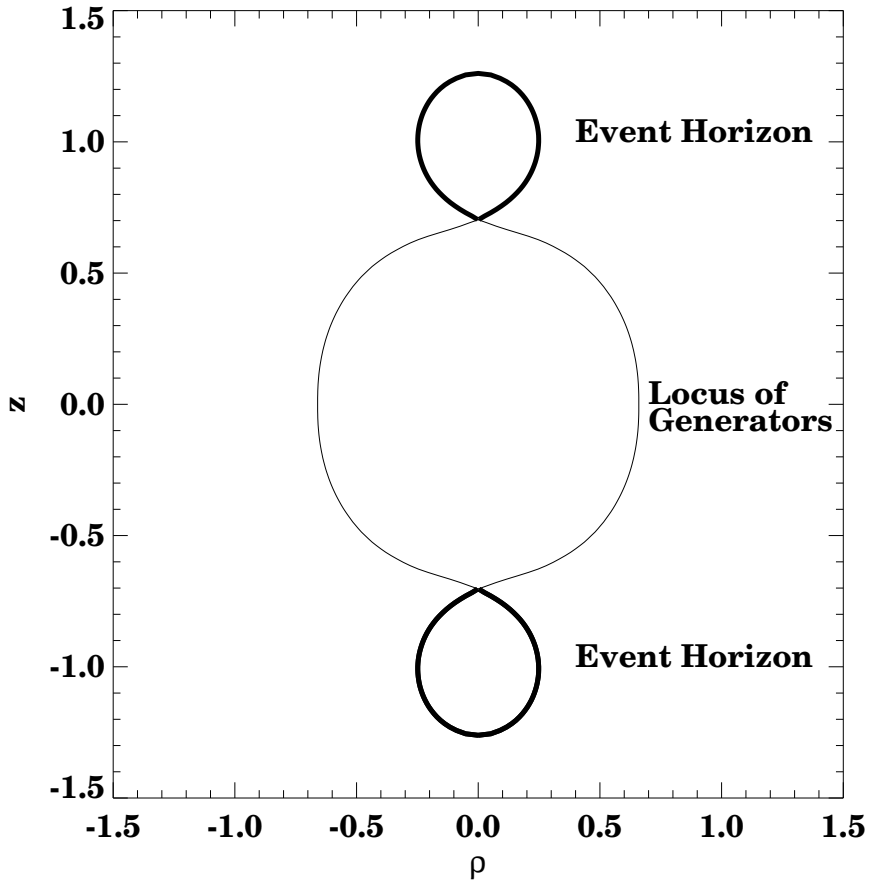


Figure 2.3: This figure shows a view of the numerically-computed event horizon in a single slice, together with the locus of the event horizon's generators that haven't yet joined the event horizon in this slice, for a head-on binary black hole collision. Notice how the event horizon is non-differentiable at the cusp where the new generators join it. Figure reprinted with permission from Libson *et al.*, *Physical Review D* **53**, 4335–4350 (1996). Copyright 1996 by the American Physical Society.

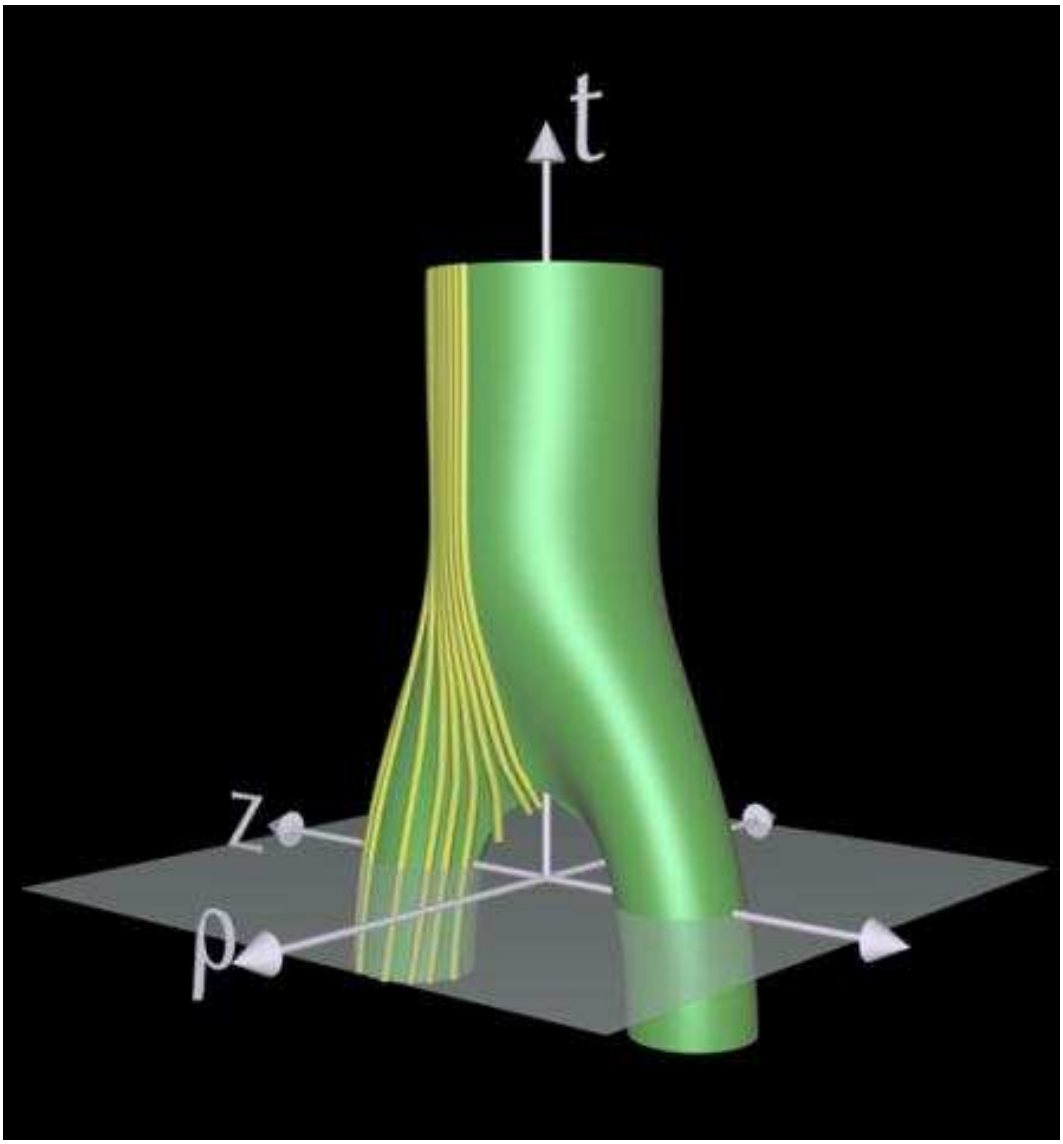


Figure 2.4: This figure shows a perspective view of the numerically-computed event horizon, together with some of its generators, for the head-on binary black hole collision discussed in detail by Matzner *et al.* [97]. Figure courtesy of Edward Seidel.

For single-black-hole test cases (Kerr spacetime in various coordinates), they report typical accuracies of a few percent in determining the event horizon position and area.

For binary-black-hole test cases (the Kastor-Traschen extremal-charge black hole coalescence with a cosmological constant), they detect black hole coalescence (which appears as a bifurcation in the backwards time integration) by the "necking off" of the surface. Figure 2.5 shows an example of their results.

#### 2.2.3.3 Level-Set Parameterization

Caveny *et al.* [40, 41] and Diener [54] (independently) implemented the "integrate null surfaces backwards" algorithm for fully generic numerically-computed spacetimes, using the level-set function integration formula (2.3). Here the level-set function $F$ is initialized on the final slice of the evolution, and evolved backwards in time using (2.3) on (conceptually) the entire numerical grid. (In practice, only a smaller box containing the event horizon need be evolved.)

This surface parameterization has the advantage that the event-horizon topology and (non-)smoothness are completely unconstrained, allowing the numerical study of configurations such as toroidal event horizons (discussed in section 2.1). It's also convenient that the level-set function $F$ is defined on the same numerical grid as the spacetime geometry, so that no interpolation is needed for the evolution.

The major problem with this algorithm is that during the backwards evolution, spatial gradients in $F$ tend to steepen into a jump discontinuity at the event horizon,[8] eventually causing numerical difficulty.

Caveny *et al.* [40, 41] deal with this problem by adding an artificial viscosity term to the level-set function evolution equation, smoothing out the jump discontinuity in $F$. That is, instead of (2.3), they actually evolve $F$ via

$$\partial_t F = \varepsilon^2 \nabla^2 F + \text{RHS of (2.3)} \tag{2.6}$$

where $\nabla^2$ is a generic 2nd order linear spatial differential operator, and $\varepsilon > 0$ is a (small) dissipation constant. This scheme works, but the numerical viscosity does seem to lead to significant errors (several percent) in their computed event-horizon positions and areas,[9] and even failure to converge to the correct solution for some test cases (e.g. rapidly-spinning Kerr black holes).

Alternatively, Diener [54] developed a technique of periodically reinitializing the level-set function to approximately the signed distance from the event horizon. To

[8] Equivalently, Diener [54] observed that the locus of any given nonzero value of the level-set function $F$ is itself a null surface, and tends to move (exponentially) closer and closer to the event horizon as the backwards evolution proceeds.

[9] They describe how Richardson extrapolation can be used to improve the accuracy of the solutions from $O(\varepsilon)$ to $O(\varepsilon^2)$, but it appears that this hasn't been done for their published results.

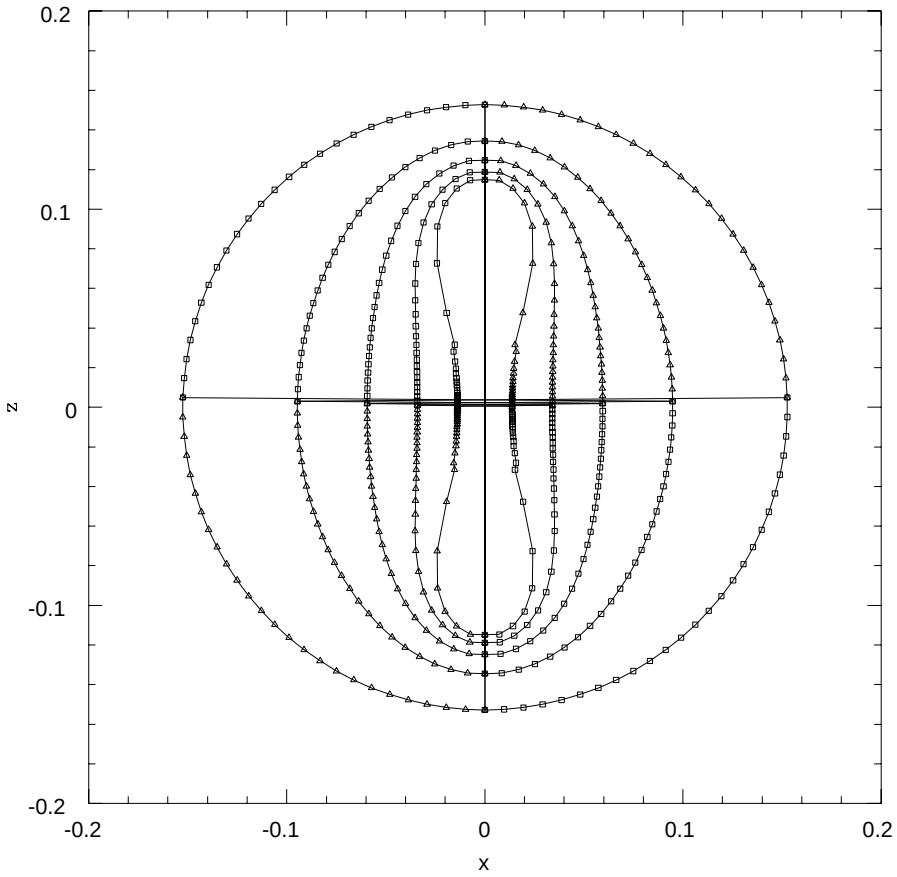


Figure 2.5: This figure shows the cross-section of the numerically-computed event horizon in each of 5 different slices, for the head-on collision of two extremal Kastor-Traschen black holes. Figure reprinted with permission from Caveny and Matzner, *Physical Review D* **68**, 104003 (2003). Copyright 2003 by the American Physical Society.

do this, he periodically evolves

$$\partial_\lambda F = -\frac{F}{\sqrt{F^2+1}}\big(|\nabla F| - 1\big) \tag{2.7}$$

in an unphysical "pseudo-time" $\lambda$ until an approximate steady state has been achieved. He reports that this works well in most circumstances, but can significantly distort the computed event horizon if the $F = 0$ isosurface (the current approximation to the event horizon) is only a few grid points thick in any direction, as typically occurs just around the time of a topology change in the isosurface. He avoids this problem by estimating the minimum thickness of this isosurface and, if it's below a threshold, deferring the reinitialization.

In various tests on analytical data, Diener [54] found this event-horizon finder, EHFinder, to be robust and highly accurate, typically locating the event horizon to much less than 1% of the 3-dimensional grid spacing. Even with only 10 grid points across the event horizon, this already corresponds to accuracies on the order of 0.1%, and this accuracy improves as expected (2nd order convergence) with increasing resolution.

As an example of the results obtained with EHFinder, figure 2.6 shows two views of the numerically-computed event horizon for a spiraling binary black hole collision. As another example, figure 2.7 shows the numerically-computed event and apparent horizons in the collapse of a rapidly rotating neutron star to a Kerr black hole. (The apparent horizons were computed using the AHFinderDirect code described in section 3.2.5.7.)

EHFinder is implemented as a module ("thorn") in the Cactus computational toolkit. It originally worked only with the PUGH unigrid driver, but Diener [55] is currently enhancing it to work with the Carpet mesh-refinement driver. EHFinder is freely available by anonymous CVS, and is now used by several research groups.

## 2.3 Summary of Algorithms/Codes for Finding Event Horizons

In spherical symmetry the "integrate null geodesics forwards" algorithm (section 2.2.1) is reasonable, though the "integrate null geodesics backwards" algorithm (section 2.2.2) is more efficient.

In non–spherically-symmetric spacetimes the "integrate null surfaces backwards" algorithm (section 2.2.3) is clearly the best algorithm known: it's efficient, accurate, and fairly easy to implement. For generic spacetimes, Diener's event-horizon finder EHFinder [54] is particularly notable as a freely available implementation of this algorithm as a module ("thorn") in the widely-used Cactus computational toolkit.

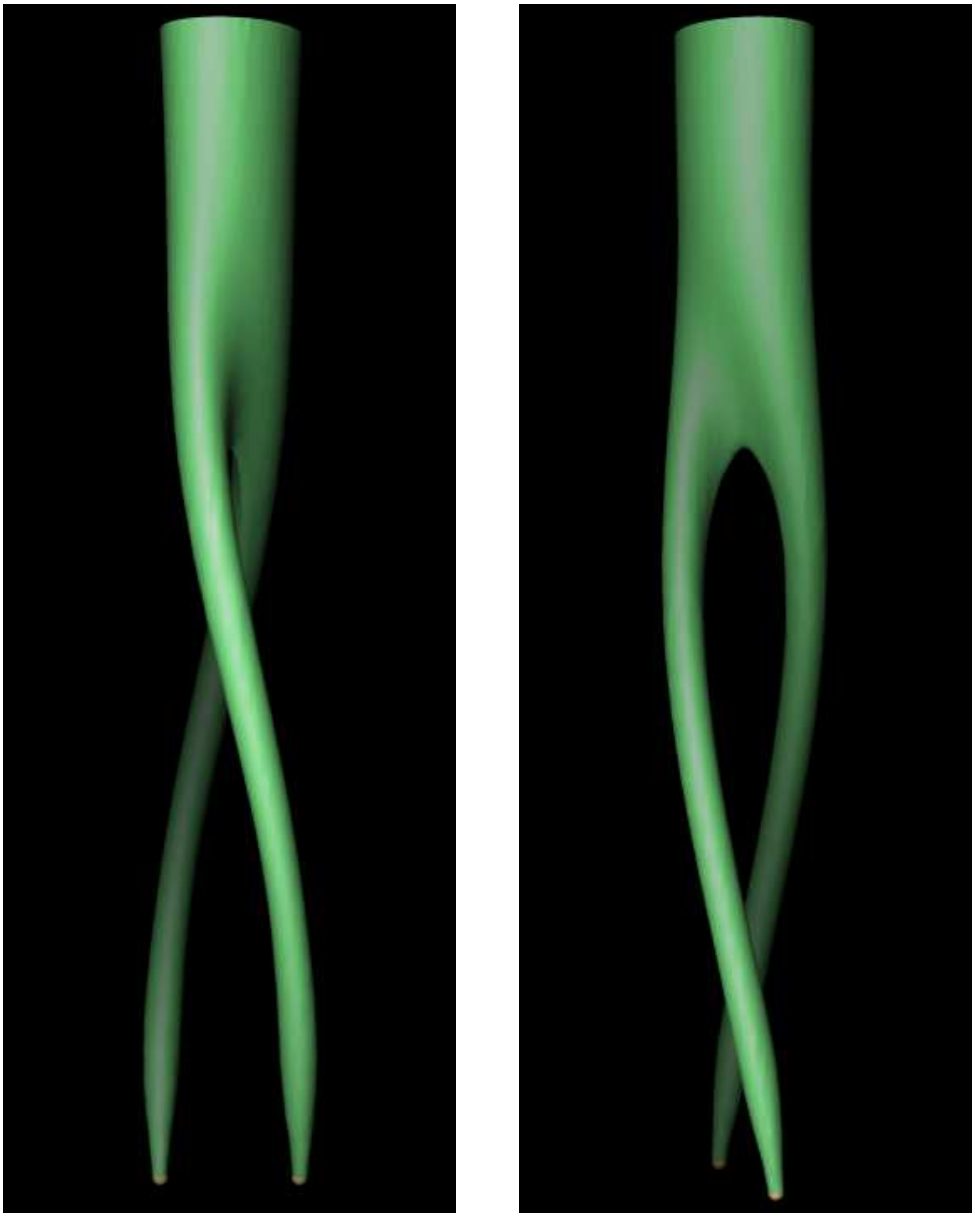

Figure 2.6: This figure shows two views of the numerically-computed event horizon for a spiraling binary black hole collision. The initial data was constructed to have an approximate helical Killing vector, corresponding to black holes in approximately circular orbits (the $D = 18$ case of Grandclément *et al.* [70]), with a proper separation of the apparent horizons of $6.9m$. Figure courtesy of Peter Diener, visualization by Werner Benger.

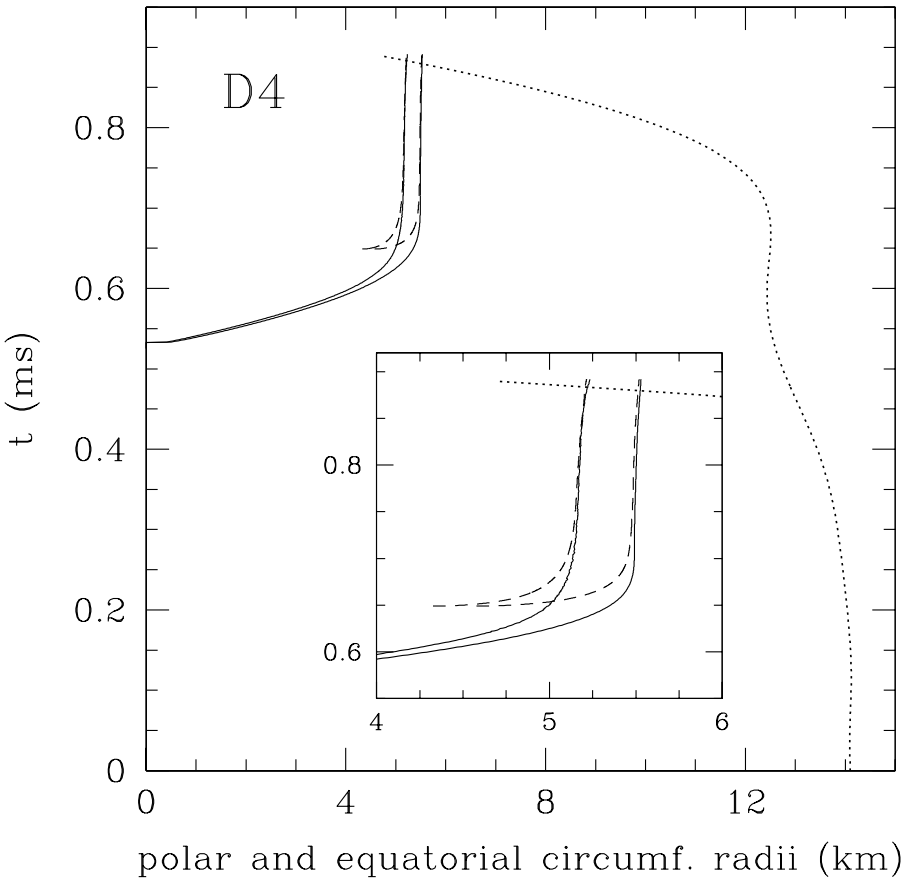


Figure 2.7: This figure shows the polar and equatorial radia of the event horizon (solid lines) and apparent horizon (dashed lines) in a numerical simulation of the collapse of a rapidly rotating neutron star to form a Kerr black hole. The dotted line shows the equatorial radius of the stellar surface. These results are from the D4 simulation of Baiotti *et al.* [18]. Notice how the event horizon grows from zero size, while the apparent horizon first appears at a finite size, and grows in a spacelike manner. Notice also that both surfaces are flattened due to the rotation. Figure reprinted with permission from Baiotti *et al.*, *Physical Review D* **71**, 024035 (2005).

# Chapter 3

# Finding Apparent Horizons

## 3.1 Introduction

### 3.1.1 Definition

Given a (spacelike) $3+1$ slice, a "marginally outer trapped surface" (MOTS) is defined as a smooth (differentiable) closed orientable 2-surface in the slice whose future-pointing outgoing null geodesics have zero expansion $\Theta$. There may be multiple MOTSs in a slice. MOTSs may nest within each other, but they can't cross. An "apparent horizon" is then defined as an outermost MOTS in a slice, i.e., a MOTS not contained in any other MOTS.

Equivalently, a "trapped surface" is defined as a smooth closed 2-surface in the slice whose future-pointing outgoing null geodesics have *negative* expansion. The "trapped region" in the slice is then defined as the union of all trapped surfaces, and the apparent horizon is defined as the outer boundary of the trapped region.

Notice that the apparent horizon is defined *locally in time* (it can be computed using only Cauchy data on a spacelike slice), but (because of the requirement that it be closed) *non-locally in space*.[1] Hawking and Ellis [74] discuss the general properties of MOTSs and apparent horizons in more detail.

### 3.1.2 General Properties

Given certain technical assumptions (including energy conditions), the existence of any MOTS (and hence any apparent horizon) implies that the slice contains a black hole.[2] Moreover, the apparent horizon necessarily coincides with, or is

[1] As an indication of the importance of the "closed" requirement, Hawking [73] has observed that if we consider two spacelike-separated events in Minkowski spacetime, the intersection of their backwards light cones satisfies all the conditions of the MOTS definition, except that it's not closed.

[2] Note that the converse of this latter statement is *not* true: An arbitrary (spacelike) slice through a black hole need not contain any apparent horizon. Notably, Wald and Iyer [146] have

contained in, an event horizon. In a stationary spacetime the event and apparent horizons coincide.

It's this relation to the event horizon which makes apparent horizons valuable for numerical computation: an apparent horizon provides a useful approximation to the event horizon in a slice, but unlike the event horizon, an apparent horizon is defined locally in time and so can be computed "on the fly" during a numerical evolution.

Given a family of spacelike $3 + 1$ slices which foliate part of spacetime, the union of the slices' apparent horizons (assuming they exist) forms a world-tube. This world-tube is necessarily either null or spacelike. If it's null this world-tube is slicing-independent (choosing a different family of slices gives the same world-tube, at least so long as each slice still intersects the world-tube in a surface with 2-sphere topology). However, if the world-tube is spacelike, it's *slicing-dependent*: choosing a different family of slices will in general give a different world-tube.[3]

Except for flow algorithms (section 3.2.7), all numerical "apparent horizon" finding algorithms and codes actually find MOTSs, and hereinafter I generally follow the common (albeit sloppy) practice in numerical relativity of blurring the distinction between an MOTS and an apparent horizon.

### 3.1.3 Trapping, Isolated, and Dynamical Horizons

Hayward [75] introduced the important concept of a "trapping horizon", roughly speaking an apparent-horizon world-tube where the expansion becomes negative if the surface is deformed in the inward null direction, along with several useful variants. Ashtekar, Beetle, and Fairhurst [13] and Ashtekar and Krishnan [15] later defined the related concepts of an "isolated horizon", essentially an apparent-horizon world-tube which is null, and a "dynamical horizon", essentially an apparent-horizon world-tube which is spacelike.

These world-tubes obey a variety of local and global conservation laws, and have many applications in analyzing numerically-computed spacetimes. See the references cited above, and also Dreyer *et al.* [57], Ashtekar and Krishnan [16, 17], Gourgoulhon and Jaramillo [68], and Booth [32] for further discussions, including applications to numerical relativity.

---

constructed a family of angularly anisotropic slices in Schwarzschild spacetime which approach arbitrarily close to $r = 0$ yet contain no apparent horizons. However, Schnetter and Krishnan [122] have recently studied the behavior of apparent horizons in various anisotropic slices in Schwarzschild and Vaidya spacetimes, finding that the Wald and Iyer behavior seems to be rare.

[3] Ashtekar and Galloway [14] have recently proved "a number of physically interesting constraints" on this slicing-dependence.

### 3.1.4 Description in Terms of the $3+1$ Variables

In terms of the $3+1$ variables, a marginally outer trapped surface (and thus an apparent horizon) satisfies the condition ([148], [72, section IIA])

$$\Theta \equiv \nabla_i s^i + K_{ij} s^i s^j - K = 0 \tag{3.1}$$

where $s^i$ is the outward-pointing unit 3-vector normal to the surface.[4]

Assuming the Strahlkörper surface parameterization (1.4), (3.1) can be rewritten in terms of angular 1st and 2nd derivatives of the horizon shape function $h$,

$$\Theta \equiv \Theta(h, \partial_u h, \partial_{uv} h; g_{ij}, \partial_k g_{ij}, K_{ij}) = 0 \tag{3.2}$$

where $\Theta$ is a complicated nonlinear algebraic function of the arguments shown. (Shibata [131] and Thornburg [137, 140] give the $\Theta(h, \partial_u h, \partial_{uv} h)$ function explicitly.)

### 3.1.5 Geometry Interpolation

$\Theta$ depends on the slice geometry variables $g_{ij}$, $\partial_k g_{ij}$, and $K_{ij}$ at the horizon position.[5] In practice these variables are usually only known on the (3-dimensional) numerical grid of the underlying numerical-relativity simulation,[6] so they must be interpolated to the horizon position, and more generally, to the position of each intermediate-iterate trial shape the apparent-horizon finding algorithm tries in the process of (hopefully) converging to the horizon position.

Moreover, usually the underlying simulation gives only $g_{ij}$ and $K_{ij}$, so $g_{ij}$ must be numerically differentiated to obtain $\partial_k g_{ij}$. As discussed by Thornburg [140, section 6.1], it's somewhat more efficient to combine the numerical differentiation and interpolation operations, essentially doing the differentiation inside the interpolator.[7]

Thornburg [140, section 6.1] argues that for an elliptic-PDE algorithm (section 3.2.5), for best convergence of the nonlinear elliptic solver, the interpolated

[4] Notice that in order for the 3-divergence in (3.1) to be meaningful, $s^i$ (defined only as a field on the marginally outer trapped surface) must be smoothly continued off the surface, and extended to a field in some 3-dimensional neighborhood of the surface. The off-surface continuation is non-unique, but it's easy to see that this doesn't affect the value of $\Theta$ on the surface.

[5] Or, in the Huq *et al.* [79, 80] algorithm described in section 3.2.5.2, at the local Cartesian grid point positions.

[6] If the underlying simulation uses spectral methods (see Gottlieb and Orszag [67] and Boyd [33] for general discussions of spectral methods, and (for example) Ansorg *et al.* [9, 10, 11], Bonazzola *et al.* [29, 30, 31], Grandclément *et al.* [69], Kidder *et al.* [86, 87, 88], and Pfeiffer *et al.* [110] for applications to numerical relativity) then the spectral series can be evaluated anywhere, so no actual interpolation need be done.

[7] An interpolator generally works by (conceptually) locally fitting a fitting function (usually a low-degree polynomial) to the data points in a neighborhood of the interpolation point, then evaluating the fitting function at the interpolation point. By evaluating the *derivative* of the fitting function, the $\partial_k g_{ij}$ values can be obtained very cheaply at the same time as the $g_{ij}$ values.

geometry variables should be smooth (differentiable) functions of the trial horizon surface position. He argues that that the usual Lagrange polynomial interpolation doesn't suffice here (in some cases his Newton's-method iteration failed to converge), because this interpolation gives results which are only piecewise differentiable.[8] To avoid this problem, Thornburg [140, section 6.1] uses Hermite polynomial interpolation; Cook and Abrahams [47] use bicubic spline interpolation. Most other researchers either don't describe their interpolation scheme, or use Lagrange polynomial interpolation, and don't report serious non-convergence problems.

### 3.1.6 Criteria for Assessing Algorithms

Ideally, an apparent-horizon finder should have several attributes:

**Robust:** The algorithm/code should find an (the) apparent horizon in a wide range of numerically-computed slices, without requiring extensive tuning of initial guesses, iteration parameters, etc. This is often relatively easy to achieve for "tracking" the time evolution of an existing apparent horizon (where the most recent previously-found apparent horizon provides an excellent initial guess for the new apparent-horizon position), but may be difficult for detecting the appearance of a new (outermost) apparent horizon in an evolution, or for initial-data or other studies where there is no "previous time step".

**Accurate:** The algorithm/code should find an (the) apparent horizon to high accuracy and shouldn't report spurious "solutions" ("solutions" which aren't actually apparent horizons or, at least, marginally outer trapped surfaces).

**Efficient:** The algorithm/code should be efficient in terms of its memory use and CPU time; in practice CPU time is generally the major constraint. It's often desirable to find apparent horizons at each time step (or, at least, at frequent intervals) during a numerical evolution. For this to be practical the apparent-horizon finder must be very fast.

In practice, no apparent-horizon finder is perfect in all these dimensions, so trade-offs are inevitable, particularly when ease of programming is considered.

As discussed in section 1.3, there are also significant advantages to having an apparent-horizon finder that's freely available to other research groups, particularly if it's designed and documented in such a way as to be relatively portable.

### 3.1.7 Local versus Global Algorithms

Apparent-horizon finding algorithms can usefully be divided into two broad classes:

[8] Thornburg [138, appendix F] gives a more detailed discussion of this non-smoothness of Lagrange-polynomial interpolation errors.

**Local** algorithms are those whose convergence is only guaranteed in some (functional) neighborhood of a solution. These algorithms require a "good" initial guess in order to find the apparent horizon. Most apparent-horizon finding algorithms are local.

**Global** algorithms are those which can (in theory, ignoring finite-step-size and other numerical effects) converge to the apparent horizon independent of any initial guess. Flow algorithms (section 3.2.7) are the only truely global algorithms. Zero-finding in spherical symmetry (section 3.2.1) and shooting in axisymmetry (section 3.2.2) are "almost global" algorithms: they require only 1-dimensional searches, which (as discussed in appendix A) can be programmed to be very robust and efficient. In many cases horizon pretracking (section 3.2.6) can semi-automatically find an initial guess for a local algorithm, essentially making the local algorithm behave like an "almost global" one.

One might wonder why local algorithms are ever used, given the apparently superior robustness (guaranteed convergence independent of any initial guess) of global algorithms. There are two basic reasons:

- In practice, local algorithms are much faster than global ones, particularly when "tracking" the time evolution of an existing apparent horizon.
- Due to finite-step-size and other numerical effects, in practice even "global" algorithms may fail to converge to an apparent horizon (that is, the algorithms may sometimes fail to find an apparent horizon even when one exists in the slice).

## 3.2 Algorithms and Codes for Finding Apparent Horizons

Many researchers have studied the apparent-horizon–finding problem, and there are a large number of different apparent-horizon finding algorithms and codes. Almost all of these require (assume) that any apparent horizon to be found is a Strahlkörper (section 1.2) about some local coordinate origin; both finite-difference and spectral parameterizations of the Strahlkörper are common.

For slices with continuous symmetries, special algorithms are sometimes used:

**Zero-Finding in Spherical Symmetry** (section 3.2.1)
In spherical symmetry the apparent horizon equation (3.2) becomes a 1-dimensional nonlinear algebraic equation, which can be solved by zero-finding.

**The Shooting Algorithm in Axisymmetry** (section 3.2.2)

In axisymmetry the apparent horizon equation (3.2) becomes a nonlinear 2-point boundary value ODE, which can be solved by a shooting algorithm.

Alternatively, all the algorithms described below for generic slices are also applicable to axisymmetric slices, and can take advantage of the axisymmetry to simplify the implementation and boost performance.

For fully generic slices, there are several broad categories of apparent-horizon finding algorithms and codes:

**Minimization Algorithms** (section 3.2.3)

These algorithms define a scalar norm on $\Theta$ over the space of possible trial surfaces. A general-purpose scalar-function-minimization routine is then used to search trial-surface-shape space for a minimum of this norm (which should give $\Theta = 0$).

**Nakamura *et al.*'s Spectral Integral-Iteration Algorithm** (section 3.2.4)

This algorithm expands the (Strahlkörper) apparent-horizon shape function in a spherical-harmonic basis, uses the orthogonality of spherical harmonics to write the apparent horizon equation as a set of integral equations for the spectral coefficients, and solves these equations using a functional-iteration algorithm.

**Elliptic-PDE Algorithms** (section 3.2.5)

These algorithms write the apparent horizon equation (3.2) as a nonlinear elliptic (boundary-value) PDE for the horizon shape, and solve this PDE using (typically) standard elliptic-PDE numerical algorithms.

**Horizon Pretracking** (section 3.2.6)

Horizon pretracking solves a slightly more general problem than apparent-horizon finding: roughly speaking, the determination of the smallest $E \geq 0$ such that the equation $\Theta = E$ has a solution, and the determination of that solution. By monitoring the time evolution of $E$ and of the surfaces satisfying this condition, we can determine – *before* it appears – approximately where (in space) and when (in time) a new marginally outer trapped surface *will* appear in a dynamic numerically-evolving spacetime. Horizon pretracking is implemented as a 1-dimensional (binary) search using a slightly-modified elliptic-PDE apparent-horizon finding algorithm as a "subroutine".

**Flow Algorithms** (section 3.2.7)

These algorithms start with a large 2-surface (larger than any possible apparent horizon in the slice), and shrink it inwards using an algorithm which ensures that the surface will stop shrinking when it coincides with the apparent horizon.

I describe the major algorithms and codes in these categories in detail in the following subsections.

### 3.2.1 Zero-Finding in Spherical Symmetry

In a spherically symmetric slice, any apparent horizon must also be spherically symmetric, so the apparent horizon equation (3.2) becomes a 1-dimensional nonlinear algebraic equation $\Theta(h) = 0$ for the horizon radius $h$. For example, assuming the usual polar-spherical spatial coordinates $x^i = (r, \theta, \phi)$, we have ([138, equation (B7)])

$$\Theta \equiv \frac{\partial_r g_{\theta\theta}}{g_{\theta\theta}\sqrt{g_{rr}}} - 2\frac{K_{\theta\theta}}{g_{\theta\theta}} = 0 \tag{3.3}$$

Given the geometry variables $g_{rr}$, $g_{\theta\theta}$, $\partial_r g_{\theta\theta}$, and $K_{\theta\theta}$, this equation may be easily and accurately solved using one of the zero-finding algorithms discussed in appendix A.[9]

Zero-finding has been used by many researchers, including [126, 127, 128, 129, 109, 43, 124, 8, 138, 139].[10] For example, the apparent horizons shown in figure 2.1 were obtained using this algorithm. As another example, figure 3.1 shows $\Theta(r)$ and $h$ at various times in a (different) spherically symmetric collapse simulation.

### 3.2.2 The Shooting Algorithm in Axisymmetry

In an axisymmetric spacetime, the space of angular coordinates $(\theta, \phi)$ is effectively 1-dimensional, and given the Strahlkörper assumption, without further loss of generality we can write the horizon shape function as $h = h(\theta)$, where $\theta$ is the single nontrivial angular coordinate. The apparent horizon equation (3.2) then becomes a nonlinear 2-point boundary-value ODE for the horizon shape function $h$ ([131, equation (1.1)])

$$\Theta \equiv \Theta(h, \partial_\theta h, \partial_{\theta\theta} h; g_{ij}, \partial_k g_{ij}, K_{ij}) = 0 \tag{3.4}$$

where $\Theta(h)$ is a nonlinear 2nd order (ordinary) differential operator in $h$ as shown.

Taking the angular coordinate $\theta$ to have the usual polar-spherical topology, local smoothness of the apparent horizon gives the boundary conditions

$$\partial_\theta h = 0 \qquad \text{at } \theta{=}0 \text{ and } \theta{=}\theta_{\max} \tag{3.5}$$

where $\theta_{\max}$ is $\pi/2$ if there is "bitant" reflection symmetry about the $z = 0$ plane, or $\pi$ otherwise.

[9] Note that $\partial_r g_{\theta\theta}$ is a known coefficient field here, not an unknown (if necessary, it can be obtained by numerically differentiating $g_{\theta\theta}$). Therefore, despite the appearance of the derivative, (3.3) is still an *algebraic* equation for the horizon radius $h$, not a differential equation.

[10] See also the work of Bizoń, Malec, and Ó Murchadha [28] for an interesting analytical study giving necessary and sufficient conditions for apparent horizons to form in non-vacuum spherically symmetric spacetimes.

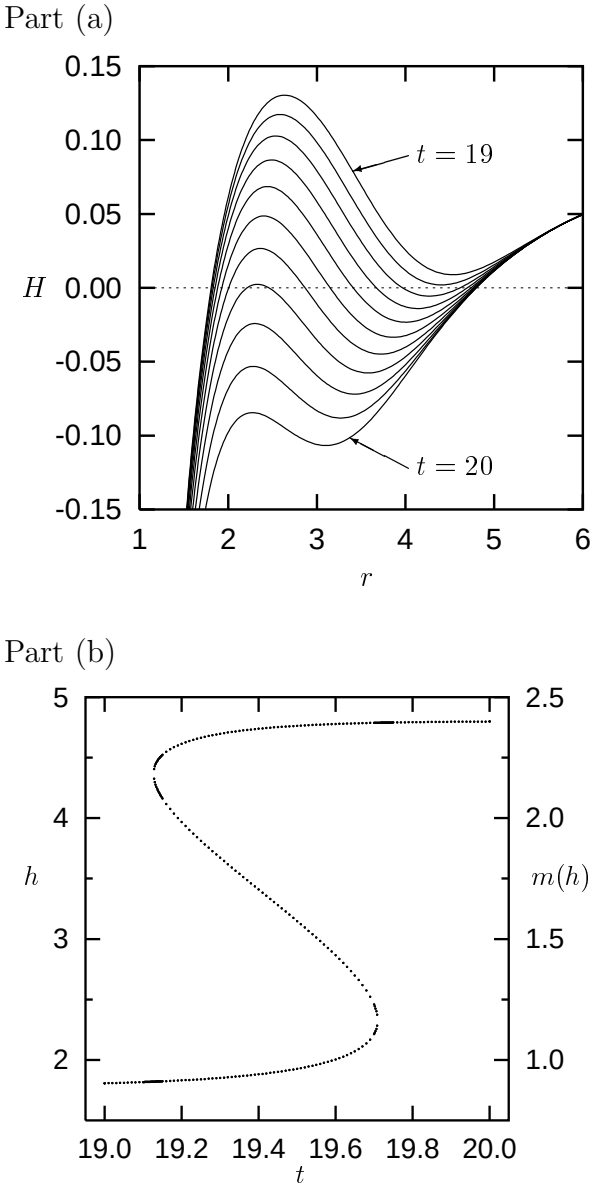


Figure 3.1: This figure shows results for a spherically symmetric numerical evolution of a black hole accreting a narrow shell of scalar field, the 800.pqw1 evolution of Thornburg [139]. Part (a) of this figure shows $\Theta(r)$ (here labelled $H$) for a set of equally-spaced times between $t$=19 and $t$=20, while part (b) shows the corresponding horizon radius $h(t)$ and the Misner-Sharp [100] mass $m(h)$ internal to each marginally outer trapped surface (MOTS). Notice how two new MOTSs appear when the local minimum in $\Theta(r)$ touches the $\Theta$=0 line, and two existing MOTS disappear when the local maximum in $\Theta(r)$ touches the $\Theta$=0 line.

As well as the more general algorithms described in the following subsections, this may be solved by a shooting algorithm:

1. Guess the value of $h$ at one endpoint, say $h(\theta{=}0) \equiv h_*$.

2. Use this guessed value of $h(\theta{=}0)$ together with the boundary condition there (3.5) as initial data to integrate ("shoot") the ODE (3.4) from $\theta{=}0$ to the other endpoint $\theta{=}\theta_{\max}$.[11]

3. If the numerically computed solution satisfies the other boundary condition (3.5) at $\theta{=}\theta_{\max}$ to within some tolerance, then the just-computed $h(\theta)$ describes the (an) apparent horizon, and the algorithm is finished.

4. Otherwise, adjust the guessed value $h(\theta{=}0) \equiv h_*$ and try again. Because there's only a single parameter ($h_*$) to be adjusted, this can be done easily and efficiently using one of the 1-dimensional zero-finding algorithms discussed in appendix A.

This algorithm is fairly efficient and easy to program. By trying a sufficiently wide range of initial guesses $h_*$ this algorithm can give a high degree of confidence that all apparent horizons have been located, although this, of course, increases the cost.

Shooting algorithms of this type have been used by many researchers, for example [144, 60, 2, 25, 26, 130, 3, 4].

### 3.2.3 Minimization Algorithms

This class of algorithms defines a scalar norm $\|\cdot\|$ on the expansion $\Theta$ over the space of possible trial surfaces, typically the mean-squared norm

$$\|\Theta\| \equiv \int \Theta^2 \, d\Omega \tag{3.6}$$

where the integral is over all solid angles on a trial surface.

Assuming the horizon surface to be a Strahlkörper and adopting the spectral representation (1.5) for the horizon surface, we can view the norm (3.6) as being defined on the space of spectral coefficients $\{a_{\ell m}\}$.

This norm clearly has a global minimum $\|\Theta\| = 0$ for each solution of the apparent horizon equation (3.2). To find the apparent horizon we numerically search the spectral-coefficient space for this (a) minimum, using a general-purpose "function-minimization" algorithm (code) such as Powell's algorithm.[12]

[11] I briefly review ODE integration algorithms and codes in appendix B.

[12] See, for example, Dennis and Schnabel [82] or Brent [35] for general surveys of general-purposes function-minimization algorithms and codes.

Evaluating the norm (3.6) requires a numerical integration over the horizon surface: We choose some grid of $N_{\text{ang}}$ points on the surface, interpolate the slice geometry fields ($g_{ij}$, $\partial_k g_{ij}$, and $K_{ij}$) to this grid (see section 3.1.5), and use numerical quadrature to approximate the integral. In practice this must be done for many different trial surface shapes (see section 3.2.3.2), so it's important that it be as efficient as possible. Anninos *et al.* [7] and Baumgarte *et al.* [22] discuss various ways to optimize and/or parallelize this calculation.

Unfortunately, minimization algorithms have two serious disadvantages for apparent-horizon finding: they are susceptible to spurious local minima, and they're very slow. I discuss these disadvantages further in the following two subsections.

#### 3.2.3.1 Spurious Local Minima

While the norm (3.6) clearly has a single *global* minimum $\|\Theta\| = 0$ for each marginally outer trapped surface $\Theta = 0$, it typically also has a large number of other *local* minima with $\Theta \neq 0$, which are "spurious" in the sense that they don't correspond (even approximately) to marginally outer trapped surfaces.[13] Unfortunately, general-purpose "function-minimization" routines only locate local minima, and thus may easily converge to one of the spurious $\Theta \neq 0$ minima.

What this problem means in practice is that a minimization algorithm needs quite a good (accurate) initial guess for the horizon shape in order to ensure that the algorithm converges to the true global minimum $\Theta = 0$ rather than to one of the spurious $\Theta \neq 0$ local minima.

To view this problem from a different perspective, once the function-minimization algorithm does converge, we must somehow determine whether the "solution" found is the true one $\Theta = 0$ or a spurious one $\Theta \neq 0$. Due to numerical errors in the geometry interpolation and the evaluation of the integral (3.6), $\|\Theta\|$ will almost never evaluate to *exactly* zero; rather, we must set a tolerance level for how large $\|\Theta\|$ may be. Unfortunately, in practice it's hard to choose this tolerance: if it's too small, the genuine solution may be falsely rejected, while if it's too large, we may accept a spurious solution (which may be very different from any of the true solutions).

Anninos *et al.* [7] and Baumgarte *et al.* [22] suggest screening out spurious solutions by repeating the algorithm with varying resolutions of the horizon-surface grid, and checking that $\|\Theta\|$ shows the proper convergence towards zero. This seems like a good strategy, but it's tricky to automate and, again, it may be difficult to

[13] There's a simple heuristic argument (see, for example, Press *et al.* [111, section 9.6]) that at least some spurious local minima should be expected: We are trying to solve a system of $N_{\text{ang}}$ nonlinear equations $\Theta_{\text{I}} = 0$ (one equation for each horizon-surface grid point). Equivalently, we are trying to find the intersection of the $N_{\text{ang}}$ codimension-one hypersurfaces $\Theta_{\text{I}} = 0$ in surface-shape space. The problem is that anywhere two or more of these hypersurfaces closely approach, but don't actually touch, there is a spurious local minimum in $\|\Theta\|$.

choose the necessary error tolerances in advance.

#### 3.2.3.2 Performance

For convenience of exposition, suppose the spectral representation (1.5) of the horizon-shape function $h$ uses spherical harmonics $Y_{\ell m}$. (Symmetric trace-free tensors or other basis sets don't change the argument in any important way.) Then if we keep harmonics up to some maximum degree $\ell_{\max}$, the number of coefficients is then $N_{\text{coeff}} = (\ell_{\max}+1)^2$. $\ell_{\max}$ is set by the desired accuracy (angular resolution) of the algorithm, and is typically on the order of 6 to 12.

To find a minimum in an $N_{\text{coeff}}$-dimensional space (here the space of surface-shape coefficients $\{a_{\ell m}\}$), a general-purpose function-minimization algorithm typically needs on the order of $5N_{\text{coeff}}^2$ to $10N_{\text{coeff}}^2$ iterations.[14] Thus the number of iterations grows as $\ell_{\max}^4$.

Each iteration requires an evaluation of the norm (3.6) for some trial set of surface-shape coefficients $\{a_{\ell m}\}$, which requires $O(N_{\text{coeff}}) = O(\ell_{\max}^2)$ work to compute the surface positions, together with $O(N_{\text{ang}})$ work to interpolate the geometry fields to the surface points and compute the numerical quadrature of the integral (3.6).

The result is that minimization horizon-finders tend to be quite slow, particularly if high accuracy is required (large $\ell_{\max}$ and $N_{\text{ang}}$). The one exception is in axisymmetry, where only spherical harmonics $Y_{\ell m}$ with $m$=0 need be considered. In this case minimization algorithms are much faster, though probably still slower than shooting or elliptic-PDE algorithms.

[14] A simple counting argument suffices to show that any general-purpose function-minimization algorithm in $n$ dimensions must involve at least $O(n^2)$ function evaluations (see, for example, Press *et al.* [111, section 10.6]): Suppose the function to be minimized is $f : \Re^n \to \Re$, and suppose $f$ has a local minimum near some point $\mathsf{x}_0 \in \Re^n$. Taylor-expanding $f$ in a neighborhood of $x_0$ gives $f(\mathsf{x}) = f(\mathsf{x}_0) + \mathsf{a}^T(\mathsf{x}-\mathsf{x}_0) + (\mathsf{x}-\mathsf{x}_0)^T\mathsf{B}(\mathsf{x}-\mathsf{x}_0) + O(\|\mathsf{x}-\mathsf{x}_0\|^3)$, where $\mathsf{a} \in \Re^n$, $\mathsf{B} \in \Re^{n\times n}$ is symmetric, and $\mathsf{v}^T$ denotes the transpose of the column vector $\mathsf{v} \in \Re^n$.

Neglecting the higher order terms (i.e., approximating $f$ as a quadratic form in $\mathsf{x}$ in a neighborhood of $x_0$), and ignoring $f(\mathsf{x}_0)$ (which doesn't affect the position of the minimum), there are a total of $N = n + \frac{1}{2}n(n+1)$ coefficients in this expression. Changing any of these coefficients may change the position of the minimum, and at each function evaluation the algorithm "learns" only a single number (the value of $f$ at the selected evaluation point), so the algorithm must make at least $N = O(n^2)$ function evaluations to (implicitly) determine all the coefficients.

Actual functions aren't exact quadratic forms, so in practice there are additional $O(1)$ multiplicative factors in the number of function evaluations. Minimization algorithms may also make additional performance and/or space-versus-time trade-offs to improve numerical robustness or to avoid explicitly manipulating $n \times n$ Jacobian matrices.

#### 3.2.3.3 Summary of Minimization Algorithms/Codes

Minimization algorithms are fairly easy to program and have been used by many researchers, for example [39, 62, 92, 7, 22, 4]. However, they're susceptible to spurious local minima, have relatively poor accuracy, and tend to be quite slow. I believe that the other algorithms discussed in the following sections are generally preferable.

Alcubierre's apparent-horizon finder AHFinder [4] includes a minimization algorithm based on the work of Anninos *et al.* [7].[15] It's implemented as a module ("thorn") in the Cactus computational toolkit, and is freely available by anonymous CVS (it's part of the CactusEinstein set of thorns included with the standard Cactus distribution). It has been used by a number of research groups.

### 3.2.4 Nakamura *et al.*'s Spectral Integral-Iteration Algorithm

Nakamura, Kojima, and Oohara [102] developed a functional-iteration algorithm for solving the apparent horizon equation (3.2).

This algorithm begins by choosing the usual polar-spherical topology for the angular coordinates $(\theta, \phi)$, and rewriting the apparent horizon equation (3.2) in the form

$$L \equiv \partial_{\theta\theta} h + \frac{\partial_\theta h}{\tan\theta} + \frac{\partial_{\phi\phi} h}{\sin^2\theta} = F(\partial_{\theta\phi} h, \partial_{\phi\phi} h, \partial_\theta h, \partial_\phi h; g_{ij}, K_{ij}, \Gamma^k_{ij}) \qquad (3.7)$$

where $F$ is a complicated nonlinear algebraic function of the arguments shown, which remains regular even at $\theta$=0 and $\theta$=$\pi$, and where for future use we define $L$ to be the left hand side of (3.7).

Next we expand $h$ in spherical harmonics (1.5). Because the left hand side $L$ of (3.7) is just the flat-space angular Laplacian of $h$, which has the $Y_{\ell m}$ as orthogonal eigenfunctions, multiplying both sides of (3.7) by $Y^*_{\ell m}$ (the complex conjugate of $Y_{\ell m}$) and integrating over all solid angles gives

$$a_{\ell m} = -\frac{1}{\ell(\ell+1)} \int Y^*_{\ell m} F \, d\Omega \qquad (3.8)$$

for each $\ell$ and $m$ except $\ell = m = 0$.

Based on this, Nakamura *et al.* [102] propose the following functional-iteration algorithm for solving (3.7):

1. Start with some (initial-guess) set of horizon-shape coefficients $\{a_{\ell m}\}$. These determine a surface shape via (1.5).

2. Interpolate the geometry variables to this surface shape (see section 3.1.5).

[15] AHFinder also includes a "fast flow" algorithm (section 3.2.7).

3. For each $\ell$ and $m$ except $\ell = m = 0$, evaluate the integral (3.8) by numerical quadrature to obtain a next-iteration coefficient $a_{\ell m}$.

4. Determine a next-iteration coefficient $a_{00}$ by numerically solving (finding a root of) the equation

$$\int Y_{00}^* F \, d\Omega = 0 \tag{3.9}$$

   This can be done using any of the 1-dimensional zero-finding algorithms discussed in appendix A.

5. Iterate until all the coefficients $\{a_{\ell m}\}$ converge.

Gundlach [72] observed that the subtraction and inversion of the flat-space angular Laplacian operator in this algorithm is actually a standard technique for solving nonlinear elliptic PDEs by spectral methods. I discuss this observation and its implications further in section 3.2.7.4.

Nakamura *et al.* [102] report that their algorithm works well, but Nakao [103] has argued that it tends to become inefficient (and possibly inaccurate) for large $\ell$ (high angular resolution) because the $Y_{\ell m}$ fail to be numerically orthogonal due to the finite resolution of the numerical grid. I know of no other published work addressing Nakao's criticism.

Kemball and Bishop [84] investigated the behavior of Nakamura *et al.*'s algorithm, and found that its (only) major weakness seems to be that the $a_{00}$-update equation (3.9) "may have multiple roots or minima even in the presence of a single marginally outer trapped surface, and all should be tried for convergence".

Kemball and Bishop [84] suggested and tested several modifications to improve the algorithm's convergence behavior. They verified that (either in its original form or with their modifications) the algorithm's convergence speed (number of iterations to a given error level) is roughly independent of the degree $\ell_{\max}$ of spherical-harmonic expansion used. They also give an analysis that the algorithm's cost is $O(\ell_{\max}^4)$, and its accuracy $\varepsilon = O(1/\ell_{\max})$, i.e. the cost is $O(1/\varepsilon^4)$.

Despite what appears to be fairly good numerical behavior and reasonable ease of implementation, this algorithm has not been widely used apart from later work by its original developers (see, for example, [105, 104]).

### 3.2.5 Elliptic-PDE Algorithms

The basic concept of elliptic-PDE algorithms is simple: we view the apparent horizon equation (3.2) as a nonlinear elliptic PDE for the horizon shape function $h$ on the angular-coordinate space and solve this equation by standard finite-differencing techniques,[16] generally using Newton's method to solve the resulting set of nonlinear

[16] In theory this equation could also be solved by a spectral method on $S^2$, using spectral differentiation to evaluate the angular derivatives. (See the references cited in footnote 6 on page 29

algebraic (finite-difference) equations. Algorithms of this type have been widely used both in axisymmetry and in fully generic slices.

#### 3.2.5.1 Angular Coordinates, Grid, and Boundary Conditions

In more detail, elliptic-PDE algorithms assume that the horizon is a Strahlkörper about some local coordinate origin, and choose an angular coordinate system and a finite-difference grid of $N_{\text{ang}}$ points on $S^2$ in the manner discussed in section 1.2.2.

The most common choices are the usual polar-spherical coordinates $(\theta, \phi)$ and a uniform "latitude/longitude" grid in these coordinates. Since these coordinates are "unwrapped" relative to the actual $S^2$ trial-horizon-surface topology, the horizon shape function $h$ satisfies periodic boundary conditions across the artificial grid boundary at $\phi = 0$ and $\phi = 2\pi$. The north and south poles $\theta = 0$ and $\theta = \pi$ are trickier, but Huq *et al.* [79, 80], Shibata and Uryū [132], and Schnetter [118, 119] all describe suitable "reaching across the pole" boundary conditions for these artificial grid boundaries.

Alternatively, Thornburg [140] avoids the $z$ axis coordinate singularity of polar-spherical coordinates by using an "inflated-cube" system of 6 angular patches to cover $S^2$. Here each patch's nominal grid is surrounded by a "ghost zone" of additional grid points where $h$ is determined by interpolation from the neighboring patches. The interpatch interpolation thus serves to tie the patches together, enforcing the continuity and differentiability of $h$ across patch boundaries. Thornburg reports that this scheme works well but was quite complicated to program.

Overall, the latitude/longitude grid seems to be the superior choice: it works well, is simple to program, and eases interoperation with other software.

#### 3.2.5.2 Evaluating the Expansion $\Theta$

The next step in the algorithm is to evaluate the expansion $\Theta$ given by (3.2) on the angular grid given a trial horizon surface shape function $h$ on this same grid (1.6).

Most researchers compute $\Theta$ via 2-dimensional angular finite differencing of (3.2) on the trial horizon surface. 2nd order angular finite differencing is most common, but Thornburg [140] uses 4th order angular finite differencing for increased accuracy.

With a $(\theta, \phi)$ latitude/longitude grid the $\Theta(h, \partial_u h, \partial_{uv} h)$ function in (3.2) is singular on the $z$ axis (at the north and south poles $\theta = 0$ and $\theta = \pi$), but can be regularized by applying L'Hopital's rule. Schnetter [118, 119] observes that using a *Cartesian* basis for all tensors greatly aids in this regularization.

Huq *et al.* [79, 80] choose, instead, to use a completely different computation technique for $\Theta$, based on *3-dimensional Cartesian* finite differencing:

for further discussion of spectral methods.) This should yield a highly efficient apparent-horizon finder. However, I know of no published work taking this approach.

1. They observe that the scalar field $F$ defined by (1.7) can be evaluated at any (3-dimensional) position in the slice by computing the corresponding $(r, \theta, \phi)$ using the usual flat-space formulas, then interpolating $h$ in the 2-dimensional $(\theta, \phi)$ surface grid.

2. Rewrite the apparent horizon condition (3.1) in terms of $F$ and it's (3-dimensional) *Cartesian* derivatives,

$$\Theta \equiv \Theta(F, \partial_i F, \partial_{ij} F; g_{ij}, \partial_k g_{ij}, K_{ij}) = 0 \tag{3.10}$$

   Huq *et al.* [79, 80] give the $\Theta(F, \partial_i F, \partial_{ij} F)$ function explicitly.

3. For each (latitude/longitude) grid point on the trial horizon surface, define a 3×3×3-point *local Cartesian grid* centered at that point. The spacing of this grid should be such as to allow accurate finite differencing, i.e. in practice it should probably be roughly comparable to that of the underlying numerical-relativity simulation's grid.

4. Evaluate $F$ on the local Cartesian grid as described in step 1 above.

5. Evaluate the Cartesian derivatives in (3.10) by centered 2nd order Cartesian finite differencing of the $F$ values on the local Cartesian grid.

Comparing the different ways of evaluating $\Theta$, 2-dimensional angular finite differencing of (3.2) seems to me to be both simpler (easier to program) and likely more efficient than 3-dimensional Cartesian finite differencing of (3.10).

#### 3.2.5.3 Solving the Nonlinear Elliptic PDE

A variety of algorithms are possible for actually solving the nonlinear elliptic PDE (3.2) (or (3.10) for the Huq *et al.* [79, 80] horizon finder).

The most common choice is to use some variant of Newton's method. That is, finite differencing (3.2) or (3.10) (as appropriate) gives a system of $N_{\text{ang}}$ nonlinear algebraic equations for the horizon shape function $h$ at the $N_{\text{ang}}$ angular grid points; these can be solved by Newton's method in $N_{\text{ang}}$ dimensions. (As explained by Thornburg [137, section VIII.C], this is usually equivalent to applying the Newton-Kantorovich algorithm ([33, appendix C]) to the original nonlinear elliptic PDE (3.2) or (3.10).)

Newton's method converges very quickly once the trial horizon surface is sufficiently close to a solution (a marginally outer trapped surface). However, for a less accurate initial guess, Newton's method may converge very slowly or even fail to converge at all. There's no usable way of determining *a priori* just how large the

radius of convergence of the iteration will be, but in practice $\frac{1}{4}$ to $\frac{1}{3}$ of the horizon radius is often a reasonable estimate.[17]

Thornburg [137] described the use of various "line search" modifications to Newton's method to improve its radius and robustness of convergence, and reported that even fairly simple modifications of this sort roughly doubled the radius of convergence.

Schnetter [118, 119] used the PETSc general-purpose elliptic-solver library [19, 20, 21] to solve the equations. This offers a wide variety of Newton-like algorithms already implemented in a highly optimized form.

Rather than Newton's method or one of its variants, Shibata *et al.* [131, 132] use a functional iteration algorithm directly on the nonlinear elliptic PDE (3.2). This seems likely to be less efficient than Newton's method but avoids having to compute and manipulate the Jacobian matrix.

#### 3.2.5.4 The Jacobian Matrix

Newton's method, and all its variants, require an explicit computation of the Jacobian matrix

$$\mathbf{J}_{\mathrm{IJ}} = \frac{\partial \Theta_{\mathrm{I}}}{\partial h_{\mathrm{J}}} \tag{3.11}$$

where the indices I and J label angular grid points on the horizon surface (or equivalently on $S^2$).

Notice that $\mathbf{J}$ includes contributions both from the direct dependence of $\Theta$ on $h$, $\partial_u h$, and $\partial_{uv} h$, and also from the indirect dependence of $\Theta$ on $h$ through the position-dependence of the geometry variables $g_{ij}$, $\partial_k g_{ij}$, and $K_{ij}$ (since $\Theta$ depends on the geometry variables *at the horizon surface position*, and this position is determined by $h$). Thornburg [137] discusses this indirect dependence in detail.

There are two basic ways to compute the Jacobian matrix.

**Numerical Perturbation:** The simplest way to determine the Jacobian matrix is by "numerical perturbation", where for each horizon-surface grid point J, $h$ is perturbed by some (small) amount $\varepsilon$ at the J th grid point (that is, $h_{\mathrm{I}} \to h_{\mathrm{I}} + \varepsilon \delta_{\mathrm{IJ}}$), and the expansion $\Theta$ is recomputed.[18] The J th column of the Jacobian matrix (3.11) is then estimated as

$$\mathbf{J}_{\mathrm{IJ}} \approx \frac{\Theta_{\mathrm{I}}(h + \varepsilon \delta_{\mathrm{IJ}}) - \Theta_{\mathrm{I}}(h)}{\varepsilon} \tag{3.12}$$

[17] Thornburg [137] used a Monte-Carlo survey of horizon-shape perturbations to quantify the radius of convergence of Newton's method for apparent-horizon finding. He found that if strong high-spatial-frequency perturbations are present in the slice's geometry then the radius of convergence may be very small. Fortunately, this problem rarely occurs in practice.

[18] A very important optimization here is that $\Theta$ only needs to be recomputed within the finite difference domain of dependence of the J th grid point.

Curtis and Reid [49] and Stoer and Bulirsch [135, section 5.4.3] discuss the optimum choice of $\varepsilon$ in this algorithm.[19]

This algorithm is easy to program but somewhat inefficient. It's used by a number of researchers, including Schnetter [118, 119] and Huq *et al.* [79, 80].

**Symbolic Differentiation:** A more efficient, although somewhat more complicated, way to determine the Jacobian matrix is the "symbolic differentiation" algorithm described by Thornburg [137], and also used by Pasch [108], Shibata *et al.* [131, 132], and Thornburg [140]. Here the internal structure of the finite differenced $\Theta(h)$ function is used to directly determine the Jacobian matrix elements.

This algorithm is best illustrated by an example which is simpler than the full apparent horizon equation: Suppose we discretize the left hand side $L$ of the apparent horizon equation (3.7) with centered 2nd order finite differences in $\theta$ and $\phi$. Then neglecting finite-differencing trunation errors, and temporarily adopting the usual notation for 2-dimensional grid functions, $h_{i,j} \equiv h(\theta{=}\theta_i, \phi{=}\phi_j)$, $L$ is given by

$$\begin{aligned} L_{i,j} &= \frac{h_{i-1,j} - 2h_{i,j} + h_{i+1,j}}{(\Delta\theta)^2} + \frac{1}{\tan\theta}\frac{h_{i+1,j} - h_{i-1,j}}{2\,\Delta\theta} \\ &\quad + \frac{1}{\sin^2\theta}\frac{h_{i,j-1} - 2h_{i,j} + h_{i,j+1}}{(\Delta\phi)^2} \end{aligned} \tag{3.13}$$

The Jacobian of $L$ is thus given by

$$\frac{\partial L_{(i,j)}}{\partial h_{(k,\ell)}} = \begin{cases} \dfrac{1}{(\Delta\theta)^2} \pm \dfrac{1}{2\,\tan\theta\,\Delta\theta} & \text{if } (k,\ell) = (i{\pm}1, j) \\ \dfrac{1}{\sin^2\theta\,(\Delta\phi)^2} & \text{if } (k,\ell) = (i, j{\pm}1) \\ -\dfrac{2}{(\Delta\theta)^2} - \dfrac{2}{\sin^2\theta\,(\Delta\phi)^2} & \text{if } (k,\ell) = (i,j) \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

Thornburg [137] describes how to generalize this to nonlinear differential operators without having to explicitly manipulate the nonlinear finite difference equations.

#### 3.2.5.5 Solving the Linear Equations

All the algorithms described in section 3.2.5.3 for treating nonlinear elliptic PDEs require solving a sequence of linear systems of $N_{\text{ang}}$ equations in $N_{\text{ang}}$ unknowns. $N_{\text{ang}}$ is typically on the order of a few thousand, and the Jacobian matrices in

[19] Because of the one-sided finite differencing, the approximation (3.12) is only $O(\varepsilon)$ accurate. However, in practice this doesn't seriously impair the convergence of a horizon finder, and the extra cost of a centered–finite-differencing $O(\varepsilon^2)$ approximation isn't warranted.

question are sparse due to the locality of the angular finite differencing (see section 3.2.5.4). Thus, for reasonable efficiency, it's essential to use linear solvers that exploit this sparsity. Unfortunately, many such algorithms/codes only handle symmetric positive-definite matrices while, due to the angular boundary conditions[20] (see section 3.2.5.1), the Jacobian matrices that arise in apparent-horizon finding are generally neither of these.

The numerical solution of large sparse linear systems is a whole subfield of numerical analysis. See, for example, Duff, Erisman, and Reid [59] and Saad [116] for extensive discussions.[21] In practice, a numerical relativist is unlikely to write her own linear solver but, rather, will use an existing subroutine (library).

Kershaw's [85] ILUCG iterative solver is often used; this is only moderately efficient, but is quite easy to program.[22] Schnetter [118, 119] reports good results with an ILU-preconditioned GMRES solver from the PETSc library. Thornburg [140] experimented with both an ILUCG solver and a direct sparse LU decomposition solver (Davis's UMFPACK library [52, 53, 51, 50]), and found each to be more efficient in some situations; overall, he found the UMFPACK solver to be the best choice.

#### 3.2.5.6 Sample Results

As an example of the results obtained with this type of apparent-horizon finder, figure 3.2 shows the numerically-computed apparent horizons (actually, marginally outer trapped surfaces) at two times in a head-on binary black hole collision. (The physical system being simulated here is very similar to that simulated by Matzner *et al.* [97], a view of whose event horizon is shown in figure 2.4.)

As another example, figure 3.3 shows the time dependence of the irreducible masses of apparent horizons found in a (spiraling) binary black hole collision, simulated at several different grid resolutions, as found by both AHFinderDirect and another Cactus apparent-horizon finder, AHFinder.[23] For this evolution, the two apparent-horizon finders give irreducible masses which agree to within about 2% for the individual horizons and 0.5% for the common horizon.

As a final example, figure 2.7 shows the numerically-computed event and apparent horizons in the collapse of a rapidly rotating neutron star to a Kerr black hole. (The event horizons were computed using the EHFinder code described in

[20] Or the interpatch interpolation conditions in Thornburg's multiple-grid-patch scheme [140].

[21] Multigrid algorithms are also important here; these exploit the geometric structure of the underlying elliptic PDE. See Briggs, Henson, and McCormick [38] and Trottenberg, Oosterlee, and Schüller [143] for general introductions to multigrid algorithms.

[22] Madderom's Fortran subroutine DILUCG [96] has been used by a number of numerical relativists for both this and other purposes.

[23] AHFinder incorporates both a minimization algorithm (section 3.2.3) and a fast-flow algorithm (section 3.2.7.4); these tests used the fast-flow algorithm.

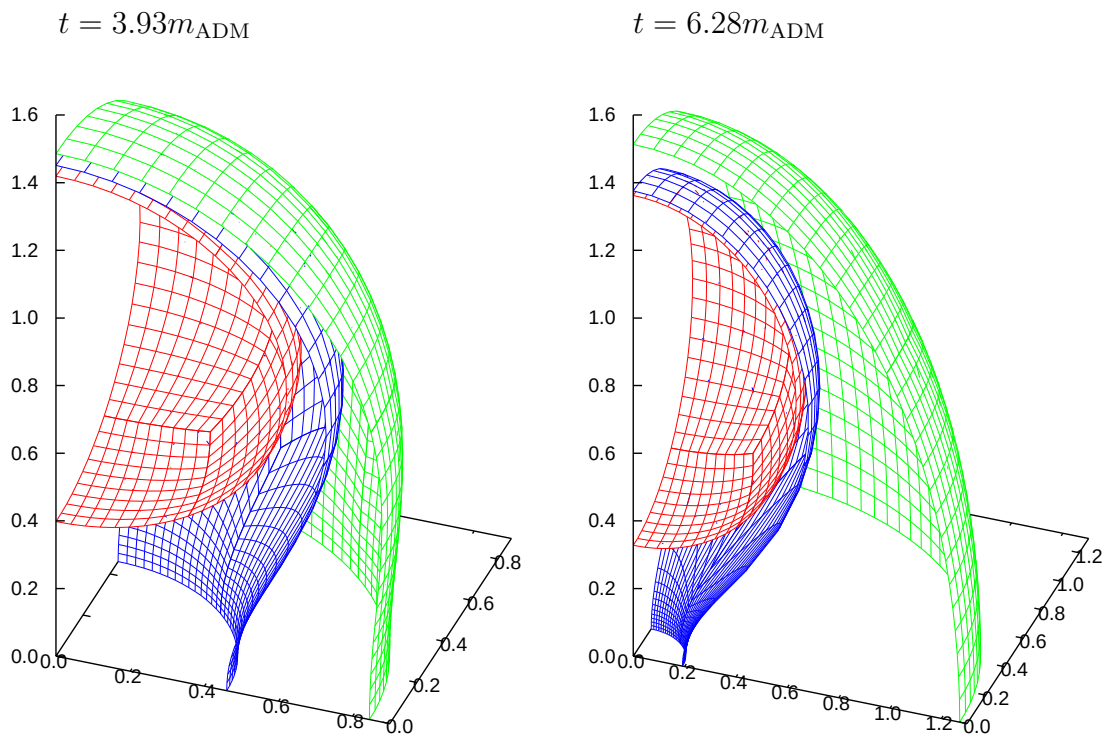


Figure 3.2: This figure shows the numerically-computed apparent horizons (actually marginally outer trapped surfaces) at two times in a head-on binary black hole collision. Figure reprinted with permission from Thornburg, *Classical and Quantum Gravity* **21**, 743–766. Copyright 2004 by IOP Publishing Ltd.

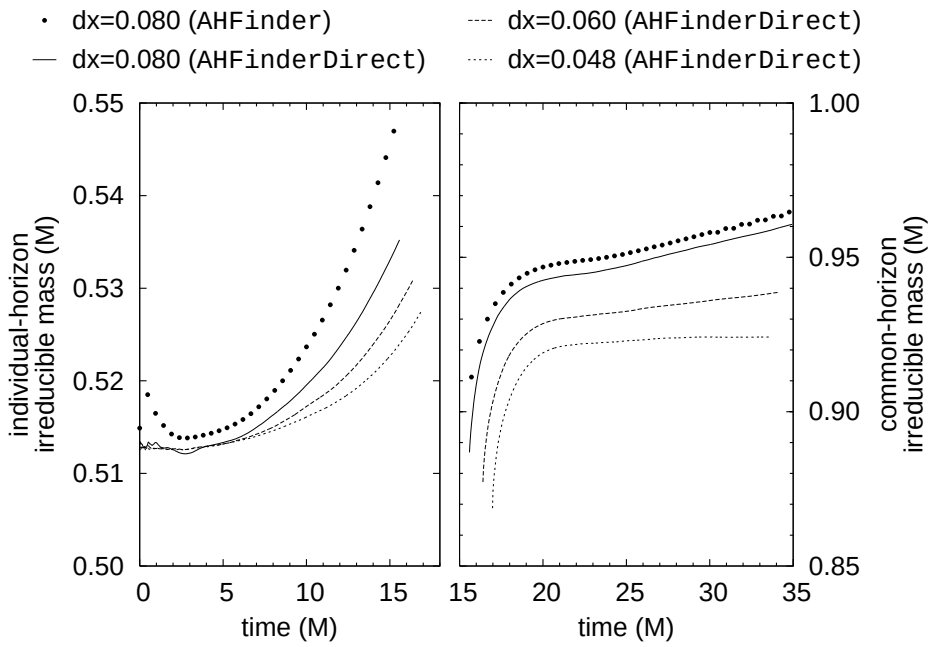


Figure 3.3: This figure shows the irreducible masses ($\sqrt{\text{area}}/16\pi$) of individual and common apparent horizons in a binary black hole collision, as calculated by two different apparent-horizon finders in the CACTUS toolkit, AHFINDER and AHFINDERDIRECT. (AHFINDERDIRECT was also run in simulations at several different resolutions.) Notice that when both apparent-horizon finders are run in the same simulation (resolution $dx$=0.080), there are only small differences between their results. Figure reprinted with permission from Alcubierre *et al.*, *Physical Review D* **72**, 044004 (2005). Copyright 2005 by the American Physical Society.

section 2.2.3.3.)

#### 3.2.5.7 Summary of Elliptic-PDE Algorithms/Codes

Elliptic-PDE apparent-horizon finders have been developed by many researchers, including Eardley [61], Cook [46, 48, 47], and Thornburg [137] in axisymmetry, and Shibata *et al.* [131, 132], Huq *et al.* [79, 80], Schnetter [118, 119], and Thornburg [140] in fully generic slices.

Elliptic-PDE algorithms are (or can be implemented to be) generally the fastest horizon-finding algorithms. For example, Thornburg [140] reports that the production version of his AHFinderDirect elliptic-PDE apparent-horizon finder, when run at each time step of a binary black hole evolution, averaged 1.7 seconds per time step, as compared with 61 seconds for an alternate "fast-flow" apparent-horizon finder AHFinder (discussed in more detail in section 3.2.7). However, achieving maximum performance comes at some cost in implementation effort (e.g. the "symbolic differentiation" Jacobian computation discussed in section 3.2.5.4).

Elliptic-PDE algorithms are probably somewhat more robust in their convergence (i.e. they have a slightly larger radius of convergence) than other types of local algorithms, particularly if the "line search" modifications of Newton's method described by Thornburg [137] are implemented.[24] Their typical radius of convergence is on the order of 30% of the horizon radius, but cases are known where it's much smaller. For example, Schnetter, Herrmann, and Pollney [121] report that (with no "line search" modifications) it's only about 10% for some slices in a binary black hole coalescence simulation.

Schnetter's TGRapparentHorizon2D [118, 119] and Thornburg's AHFinderDirect [140] are both elliptic-PDE apparent-horizon finders implemented as modules ("thorns") in the Cactus computational toolkit. Both are freely available by anonymous CVS, and work with either the PUGH unigrid driver or the Carpet mesh-refinement driver for Cactus. TGRapparentHorizon2D is no longer maintained, but AHFinderDirect is actively supported and is now used by many different research groups.[25]

### 3.2.6 Horizon Pretracking

Schnetter *et al.* [119, 121] introduced the important concept of "horizon pretracking". They focus on the case where we want to find a common apparent horizon as soon as it appears in a binary black-hole (or neutron-star) simulation. While a global (flow) algorithm (section 3.2.7) could be used to find this common apparent

[24] The convergence problems Thornburg [137] noted when high-spatial-frequency perturbations are present in the slice's geometry, seem to be rare in practice.

[25] In addition, at least two different research groups have now ported, or are in the process of porting, AHFinderDirect to their own (non-Cactus) numerical relativity codes.

horizon, these algorithms tend to be very slow. They observe that the use of a local (elliptic-PDE) algorithm for this purpose is somewhat problematic:

> The common [apparent] horizon [...] appears instantaneously at some late time and without a previous good guess for its location. In practice, an estimate of the surface location and shape can be put in by hand. The quality of this guess will determine the rate of convergence of the finder and, more seriously, also determines whether a horizon is found at all. Gauge effects in the strong field region can induce distortions that have a large influence on the shape of the common horizon, making them difficult to predict, particularly after a long evolution using dynamical coordinate conditions. As such, it can be a matter of some expensive trial and error to find the common apparent horizon at the earliest possible time. Further, if a common apparent horizon is not found, it is not clear whether this is because there is none, or whether there exists one which has only been missed due to unsuitable initial guesses – for a fast apparent horizon finder, a good initial guess is crucial.

Pretracking tries (usually successfully) to eliminate these difficulties by determining – *before* it appears – approximately where (in space) and when (in time) the common apparent horizon will appear.

#### 3.2.6.1 Constant-Expansion Surfaces

The basic idea of horizon pretracking is to consider surfaces of constant expansion ("CE surfaces"), i.e. smooth closed orientable 2-surfaces in a slice satisfying the condition

$$\Theta = E \tag{3.15}$$

where the expansion $E$ is a specified real number. Each marginally outer trapped surface (including the apparent horizon) is thus a CE surface with expansion $E = 0$; more generally (3.15) defines a 1-parameter family of 2-surfaces in the slice. As discussed by Schnetter *et al.* [119, 121], for asymptotically flat slices containing a compact strong-field region, some of the $E > 0$ members of this family typically foliate the weak-field region.

In the binary-coalescence context, for each $t =$ constant slice we define $E_*$ to be the smallest $E \geq 0$ for which a CE surface (containing both strong-field regions) exists with expansion $E$. If $E_* = 0$ then this "minimum-expansion CE surface" is the common apparent horizon, while if $E_* > 0$ this surface is an approximation to where the common apparent horizon *will* appear. We expect the minimum-expansion CE surface to change continuously during the evolution, and its expansion $E_*$ to decrease towards 0. Essentially, horizon pretracking follows the time evolution of the minimum-expansion CE surface and uses it as an initial guess for (searching for) the common apparent horizon.

#### 3.2.6.2 Generalized Constant-Expansion Surfaces

Schnetter [119] implemented an early form of horizon pretracking, which followed the evolution of the minimum-expansion constant-expansion surface, and found that it worked well for simple test problems. However, Schnetter *et al.* [121] found that for more realistic binary–black-hole coalescence systems the algorithm needs to be extended:

- While the expansion is zero for a common apparent horizon, it's also zero for a 2-sphere at spatial infinity. Figure 3.4 illustrates this for Schwarzschild spacetime. Notice that for small positive $E_*$ there will generally be two distinct CE surfaces with $E = E_*$, an inner surface just outside the horizon, and an outer one far out in the weak-field region. The inner CE surface converges to the common apparent horizon as $E_*$ decreases towards 0, and is the surface we would like the pretracking algorithm to follow. Unfortunately, without measures such as those described below, there's nothing to prevent the algorithm from following the outer surface, which does *not* converge to the common apparent horizon as $E_*$ decreases towards 0.

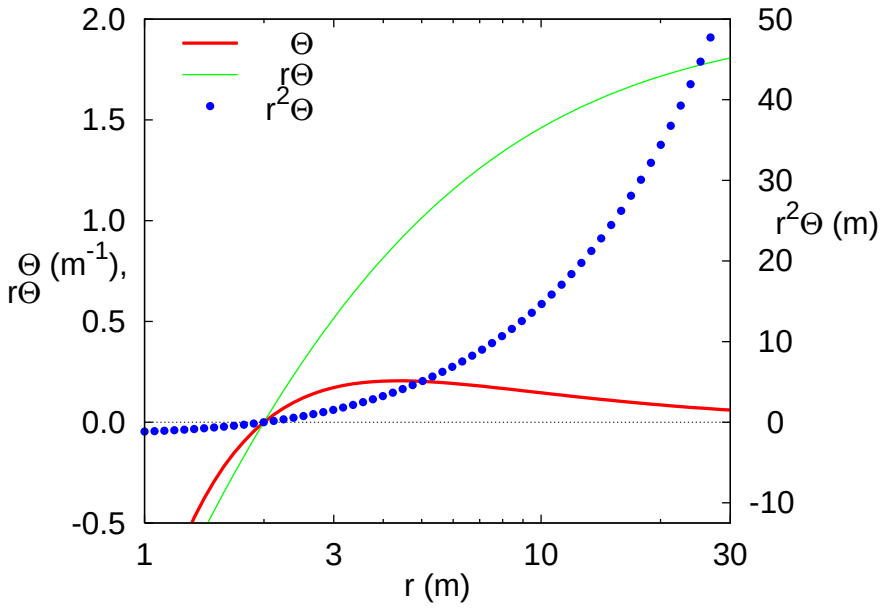


Figure 3.4: This figure shows the expansion $\Theta$ (left scale), and the "generalized expansions" $r\Theta$ (left scale) and $r^2\Theta$ (right scale), for various $r$ = constant surfaces in an Eddington-Finkelstein slice of Schwarzschild spacetime. Notice that all three functions have zeros at the horizon $r = 2m$, and that while $\Theta$ has a maximum at $r \approx 4.4m$, both $r\Theta$ and $r^2\Theta$ increase monotonically with $r$.

- In a realistic binary-coalescence simulation, the actual minimum-expansion CE surface may be highly distorted, which makes it hard to represent accurately with a finite-resolution angular grid.

Schnetter *et al.* [121] discuss these problems in more detail, arguing that to solve them, the expansion $\Theta$ should be generalized to a "shape function" $H$ given by one of

$$H_1 = \Theta \tag{3.16a}$$
$$H_r = h\Theta \tag{3.16b}$$
$$H_{r^2} = h^2\Theta \tag{3.16c}$$

CE surfaces are then generalized to surfaces satisfying

$$H = E \tag{3.17}$$

for some specified $E \geq 0$.

Note that unlike $H_1$, both $H_r$ and $H_{r^2}$ are typically monotonic with radius. Neither $H_r$ nor $H_{r^2}$ are 3-covariantly defined, but they both still have the property that $E = 0$ in (3.17) implies the surface is a marginally outer trapped surface, and in practice they work better for horizon pretracking.

#### 3.2.6.3 Goal Functions

To select a single "smallest" surface at each time, Schnetter *et al.* [121] introduce a second generalization, that of a "goal function" $G$, which maps surfaces to real numbers. The pretracking search then attempts, on each time slice, to find the surface (shape) satisfying $H = E$ with the minimum value of $G$. They experimented with several different goal functions,

$$G_H = \overline{H} \tag{3.18a}$$
$$G_{rH} = \overline{h}\,\overline{H} \tag{3.18b}$$
$$G_r = \overline{h} \tag{3.18c}$$

where in each case the overline ($\overline{\ }$) denotes an average over the surface.[26]

#### 3.2.6.4 The Pretracking Search

Schnetter's [119] original implementation of horizon pretracking (which followed the evolution of the minimum-expansion CE surface) used a binary search on the desired expansion $E$. Because $E$ appears only on the right hand side of the generalized

[26] Schnetter *et al.* [121] use a simple arithmetic mean over all surface grid points. In theory this average could be defined 3-covariantly by taking the induced metric on the surface into account, but in practice they found that this wasn't worth the added complexity.

CE condition (3.17), it's trivial to modify any apparent-horizon finder to search for a surface of specified expansion $E$. (Schnetter used his TGRapparentHorizon2D elliptic-PDE apparent-horizon finder described in section 3.2.5.7 for this.) A binary search on $E$ can then be used to find the minimum value $E_*$.[27]

Implementing a horizon-pretracking search on any of the generalized goal functions (3.18) is conceptually similar but somewhat more involved: As described by Schnetter *et al.* [121] for the case of an elliptic-PDE apparent-horizon finder,[28] we first write the equation defining a desired pretracking surface as

$$H - \overline{H} + G - p = 0 \tag{3.19}$$

where $p$ is the desired value of the goal function $G$. Since $H$ is the only term in (3.19) which varies over the surface, it must be constant for the equation to be satisfied. In this case $H - \overline{H}$ vanishes, so the equation just gives $G = p$, as desired.

Because $\overline{H}$ depends on $H$ at *all* surface points, directly finite differencing (3.19) would give a non-sparse Jacobian matrix, which would greatly slow the linear-solver phase of the elliptic-PDE apparent-horizon finder (section 3.2.5.5). Schnetter *et al.* [121, section III.B] show how this problem can be solved by introducing a single extra unknown into the discrete system. This gives a Jacobian which has a single non-sparse row and column, but is otherwise sparse, so the linear equations can still be solved efficiently.

When doing the pretracking search, the cost of a single binary-search iteration is approximately the same as that of finding an apparent horizon. Schnetter *et al.* [121, figure 5] report that their pretracking implementation (a modified version of Thornburg's AHFinderDirect [140] elliptic-PDE apparent-horizon finder described in section 3.2.5.7) typically takes on the order of 5 to 10 binary-search iterations.[29,30] The cost of pretracking is thus on the order of 5 to 10 times that of finding a single apparent horizon. This is substantial, but not prohibitive, particularly if the pretracking algorithm isn't run at every time step.

[27] There is one complication here: Any local apparent-horizon finding algorithm may fail if the initial guess isn't good enough, *even if the desired surface is actually present*. The solution is to use the constant-expansion surface for a slightly larger expansion $E$ as an initial guess, gradually "walking down" the value of $E$ to find the minimum value $E_*$. Thornburg [140, appendix C] describes such a "continuation-algorithm binary search" algorithm in detail.

[28] So far as I know this is the only case that has so far been considered for horizon pretracking. Extension to other types of apparent-horizon finders might be a fruitful area for further research.

[29] This refers to the period before a common apparent horizon is found. Once a common apparent horizon is found, then pretracking can be disabled as the apparent-horizon finder can easily "track" the apparent horizon's motion from one time step to the next.

[30] With a binary search the number of iterations depends only weakly (logarithmically) on the pretracking algorithm's accuracy tolerance. It might be possible to replace the binary search by a more sophisticated 1-dimensional search algorithm (I discuss such algorithms in appendix A), potentially cutting the number of iterations substantially. This might be a fruitful area for further research.

#### 3.2.6.5 Sample Results

As an example of the results obtained from horizon pretracking, figure 3.5 shows the expansion $\Theta$ for various pretracking surfaces (i.e. various choices for the shape function $H$ in a head-on binary black hole collision. Notice how all three of the shape functions (3.16) result in pretracking surfaces whose expansions converge smoothly to zero just when the apparent horizon appears (at about $t = 1.1$).

As a further example, figure 3.6 shows the pretracking surfaces at various times in a spiraling binary black hole collision, projected into the black hole's orbital plane.

#### 3.2.6.6 Summary of Horizon Pretracking

Pretracking is a very valuable addition to the horizon-finding repertoire: it essentially gives a local algorithm (in this case an elliptic-PDE algorithm) most of the robustness of a global algorithm in terms of finding a common apparent horizon as soon as it appears. It's implemented as a higher-level algorithm which uses a slightly-modified elliptic-PDE apparent-horizon finding algorithm as a "subroutine".

The one significant disadvantage of pretracking is its cost: each pretracking search typically takes 5 to 10 times as long as finding an apparent horizon. Further research to reduce the cost of pretracking would be useful.

Schnetter *et al.*'s pretracking implementation [121] is implemented as a set of modifications to Thornburg's AHFinderDirect [140] apparent-horizon finder. Like the original AHFinderDirect, the modified version is a "thorn" in the Cactus toolkit and is freely available by anonymous CVS.

### 3.2.7 Flow Algorithms

Flow algorithms define a "flow" on 2-surfaces, i.e., they define an evolution of 2-surfaces in some pseudo-time $\lambda$, such that the apparent horizon is the $\lambda \to \infty$ limit of a (any) suitable starting surface. Flow algorithms are different from other apparent-horizon finding algorithms (except for zero-finding in spherical symmetry), in that their convergence doesn't depend on having a good initial guess. In other words, flow algorithms are global algorithms (section 3.1.7).

To find the (an) apparent horizon, i.e., an *outermost* MOTS, the starting surface should be outside the largest possible MOTS in the slice. In practice, it generally suffices to start with a 2-sphere of areal radius substantially greater than $2m_{\text{ADM}}$.

The global convergence property requires that a flow algorithm always flow from a large starting surface into the apparent horizon. This means that the algorithm gains no particular benefit from already knowing the approximate position of the apparent horizon. In particular, flow algorithms are no faster when "tracking" the apparent horizon (repeatedly finding it at frequent intervals) in a numerical time evolution. (In contrast, in this situation a local apparent-horizon finding algorithm

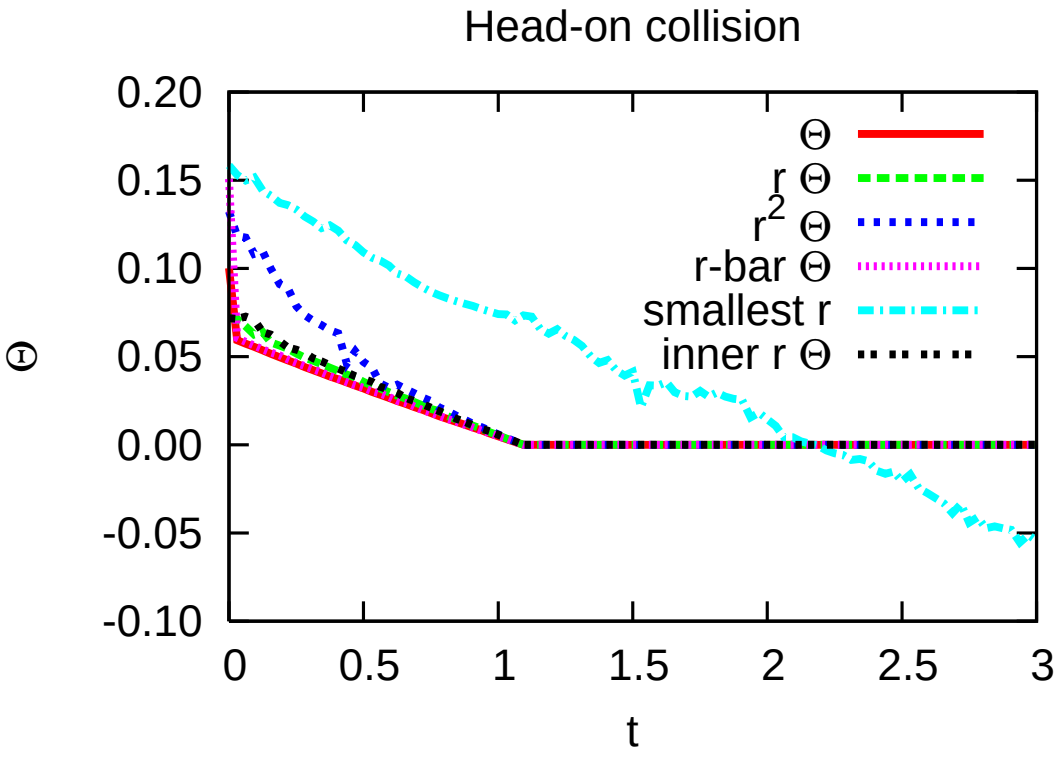


Figure 3.5: This figure shows the expansion $\Theta$ for various pretracking surfaces, i.e., for various choices for the shape function $H$, in a head-on binary black hole collision. Notice how the three shape functions (3.16) (here labelled $\Theta$, $r\Theta$, and $r^2\Theta$) result in pretracking surfaces whose expansions converge smoothly to zero just when the apparent horizon appears (at about $t = 1.1$). Notice also that these three expansions have all converged to each other somewhat before the common apparent horizon appears. Figure reprinted with permission from Schnetter, Herrmann, and Pollney, *Physical Review D* **71**, 044033 (2005). Copyright 2005 by the American Physical Society.

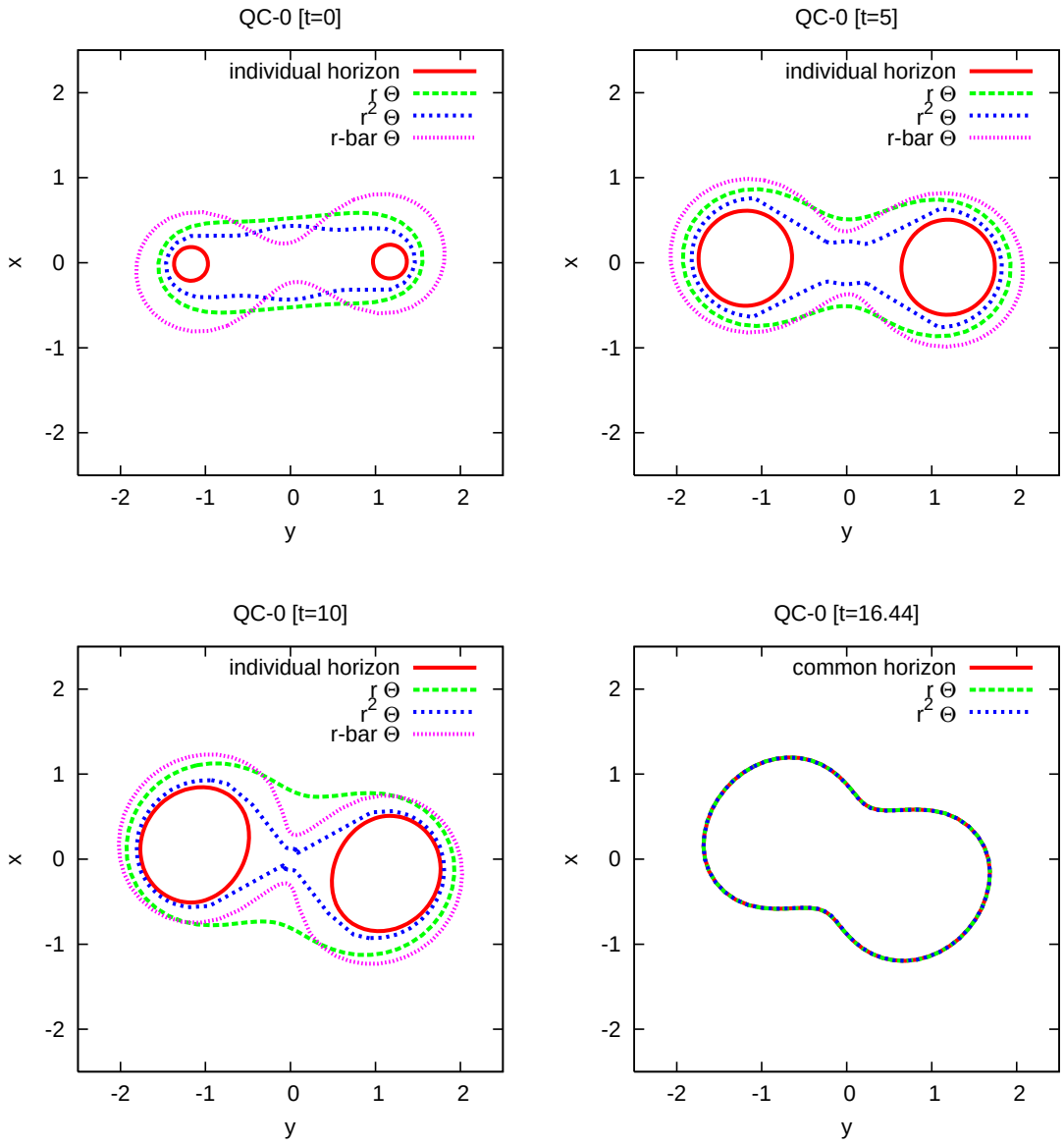


Figure 3.6: This figure shows the pretracking surfaces at various times in a spiraling binary black hole collision, projected into the black holes' orbital plane. Notice how, even well before the common apparent horizon first appears ($t = 16.44 m_{\mathrm{ADM}}$, bottom right plot), the $r\Theta$ pretracking surface is already a reasonable approximation to the eventual common apparent-horizon's shape. Figure reprinted with permission from Schnetter, Herrmann, and Pollney, *Physical Review D* **71**, 044033 (2005). Copyright 2005 by the American Physical Society.

can use the most recent previously-found apparent horizon as an initial guess, greatly speeding the algorithm's convergence.)

Flow algorithms were first proposed for apparent-horizon finding by Tod [142]. He initially considered the case of a time-symmetric slice (one where $K_{ij} = 0$). In this case a marginally outer trapped surface (and thus an apparent horizon) is a surface of minimal area, and may be found by a "mean curvature flow"

$$\partial_\lambda x^i = -\kappa s^i \tag{3.20}$$

where $x^i$ are the spatial coordinates of a horizon-surface point, $s^i$ is as before the outward-pointing unit 3-vector normal to the surface, and $\kappa \equiv \nabla_k s^k$ is the mean curvature of the surface as embedded in the slice. This is a gradient flow for the surface area, and Grayson [71] has proven that if the slice contains a minimum-area surface, this will in fact be the stable $\lambda \to \infty$ limit of this flow. Unfortunately, this proof is valid only for the time-symmetric case.

For non–time-symmetric slices, Tod [142] proposed generalizing the mean curvature flow to the "expansion flow"

$$\partial_\lambda x^i = -\Theta s^i \tag{3.21}$$

There is no theoretical proof that this flow will converge to the (an) apparent horizon, but since the flow velocity is zero there, and the flow is identical to the mean curvature flow (3.20) in the principle part, convergence is at least theoretically plausible. Numerical experiments by Bernstein [24], Shoemaker *et al.* [133, 134], and others show that that the expansion flow (3.21) does in fact converge robustly to the apparent horizon.

In the following subsections I discuss a number of important implementation details for, and refinements of, this basic algorithm.

#### 3.2.7.1 Implicit Pseudo-Time Stepping

Assuming the Strahlkörper surface parameterization (1.4), the expansion flow (3.21) is a *parabolic* equation for the horizon shape function $h$.[31] This means that any fully explicit scheme to integrate it (in the pseudo-time $\lambda$) must severely restrict its pseudo-time step $\Delta\lambda$ for stability, and this restriction grows (quadratically) worse at higher spatial resolutions.[32] This makes the horizon-finding process very slow.

To avoid this restriction, practical implementations of flow algorithms use implicit pseudo-time integration schemes; these can have large pseudo-time steps and still be stable. Because we only care about the $\lambda \to \infty$ limit, a highly accurate pseudo-time integration isn't important; only the accuracy of approximating

[31] Linearizing the $\Theta(h)$ function (3.2) gives a negative Laplacian in $h$ as the principal part.

[32] For a spatial resolution $\Delta x$, an explicit scheme is generally limited to a pseudo-time step $\Delta\lambda \lesssim (\Delta x)^2$.

the spatial derivatives matters. Bernstein [24] used a modified Du Fort-Frankel scheme [58],[33] but found some problems with the surface shape gradually developing high-spatial-frequency noise. Pasch [108] reports that an "exponential" integrator (Hochbruck *et al.* [77]) works well, provided the flow's Jacobian matrix is computed accurately.[34,35] The most common choice is probably that of Shoemaker *et al.* [133, 134], who use the iterated Crank-Nicholson ("ICN") scheme.[36] They report that this works very well; in particular, they don't report any noise problems.

By refining his finite-element grid (section 1.2.3) in a hierarchical manner, Metzger [98] is able to use standard conjugate-gradient elliptic solvers in a multigrid-like fashion,[37] using each refinement level's solution as an initial guess for the next higher refinement level's iterative solution. This greatly speeds the flow integration: Metzger reports that the performance of the overall surface-finding algorithm is "of the same order of magnitude" as that of Thornburg's AHFinderDirect [140] elliptic-PDE apparent-horizon finder (described in section 3.2.5.7).

In a more general context than numerical relativity, Osher and Sethian [106] have discused a general class of numerical algorithms for integrating "fronts propagating with curvature-dependent speed". These flow a level-set function (section 1.2.1) which implicitly locates the actual "front".

#### 3.2.7.2 Varying the Flow Speed

Another important performance optimization of the standard expansion flow (3.21) is to replace $\Theta$ in the right-hand side by a suitable nonlinear function of $\Theta$, chosen so the surface shrinks faster when it's far from the apparent horizon. For example, Shoemaker *et al.* [133, 134] use the flow

$$\partial_\lambda x^i = -\left[(\Theta - c)\arctan^2\left(\frac{\Theta - c}{\Theta_0}\right)\right] s^i \tag{3.22}$$

for this purpose, where $\Theta_0$ is the value of $\Theta$ on the initial-guess surface, and $c$ (which is gradually decreased towards 0 as the iteration proceeds) is a "goal" value for $\Theta$.

[33]Richtmyer and Morton [115, section 7.5] give a very clear presentation and analysis of the Du Fort-Frankel scheme.

[34]More precisely, Pasch [108] found that that an exponential integrator worked well when the flow's Jacobian matrix was computed exactly (using the symbolic-differentiation technique described in section 3.2.5.4). However, when the Jacobian matrix was approximated using the numerical-perturbation technique described in section 3.2.5.4, Pasch found that the pseudo-time integration became unstable at high numerical resolutions.

[35]Pasch [108] also notes that the exponential integrator uses a very large amount of memory.

[36]Teukolsky [136] and Leiler and Rezzolla [91] have analyzed ICN's stability under various conditions.

[37]See the references cited in footnote 21 on page 44 for general introductions to multigrid algorithms for elliptic PDEs.

#### 3.2.7.3 Surface Representation and the Handling of Bifurcations

Since a flow algorithm starts with (topologically) a single large 2-sphere, if there are multiple apparent horizons present the surface must change topology (bifurcate) at some point in the flow. Depending on how the surface is represented, this may be easy or difficult.

Pasch [108] and Shoemaker *et al.* [133, 134] use a level-set function approach (section 1.2.1). This automatically handles any topology or topology change. However, it has the drawback of requiring the flow to be integrated throughout the entire volume of the slice (or at least in some neighborhood of each surface). This is likely to be much more expensive than only integrating the flow on the surface itself. Shoemaker *et al.* also generate an explicit Strahlkörper surface representation (section 1.2.2), monitoring the surface shape to detect an imminent bifurcation and reparameterizing the shape into 2 separate surfaces if a bifurcation happens.

Metzger [98] uses a finite-element surface representation (section 1.2.3), which can represent any topology. However, if the flow bifurcates, then to explicitly represent each apparent horizon the code must detect that the surface self-intersects, which may be expensive.

#### 3.2.7.4 Gundlach's "Fast Flow"

Gundlach [72] introduced the important concept of a "fast flow". He observed that the subtraction and inversion of the flat-space Laplacian in Nakamura *et al.*'s spectral integral-iteration algorithm (section 3.2.4) is an example of "a standard way of solving nonlinear elliptic problems numerically, namely subtracting a simple linear elliptic operator from the nonlinear one, inverting it by pseudo-spectral algorithms and iterating". Gundlach then interpreted Nakamura *et al.*'s algorithm as a type of flow algorithm where each pseudo-time step of the flow corresponds to a single functional-iteration step of the Nakamura *et al.* algorithm.

In this framework, Gundlach defines a 2-parameter family of flows interpolating between Nakamura *et al.*'s algorithm and Tod's [142] expansion flow (3.21),

$$\partial_\lambda h = -A(1 - B\nabla^2)^{-1}\rho\Theta \tag{3.23}$$

where $A \geq 0$ and $B \geq 0$ are parameters, $\rho > 0$ is a weight functional which depends on $h$ through at most 1st derivatives, $\nabla^2$ is the flat-space Laplacian operator, and $(1 - B\nabla^2)^{-1}$ denotes inverting the operator $(1 - B\nabla^2)$.[38]

Gundlach then argues that intermediate "fast flow" members of this family should be a useful compromises between the speed of Nakamura *et al.*'s algorithm, and the robustness of Tod's expansion flow. Based on numerical experiments, Gundlach suggests a particular choice for the weight functional $\rho$ and the parameters $A$

[38]The inversion is only formal, because Nakamura *et al.*'s algorithm treats the $a_{00}$ spectral coefficient specially. Gundlach [72] discusses this in more detail.

and $B$. The resulting algorithm updates high-spatial-frequency components of $h$ essentially the same as Nakamura *et al.*'s algorithm, but should reduce low-spatial-frequency error components faster.

Alcubierre's AHFinder [4] horizon finder includes an implementation of Gundlach's fast flow algorithm.[39] AHFinder is implemented as a module ("thorn") in the Cactus computational toolkit, and is freely available by anonymous CVS (it's part of the CactusEinstein set of thorns included with the standard Cactus distribution). AHFinder has been used by a large number of research groups.

#### 3.2.7.5 Summary of Flow Algorithms/Codes

Flow algorithms are the only truely global apparent-horizon finding algorithms, and as such can be much more robust than local algorithms. In particular, flow algorithms can guarantee convergence to the *outermost* MOTS in a slice. Unfortunately, these convergence guarantees hold only for time-symmetric slices.

In the forms which have strong convergence guarantees, flow algorithms tend to be very slow. (Metzger's algorithm [98] is a notable exception: it's very fast.) There are modifications which can make flow algorithms much faster, but then their convergence is no longer guaranteed. In particular, practical experience has shown that in some binary black hole coalescence simulations (Alcubierre *et al.* [5], Diener *et al.* [56]), "fast flow" algorithms (section 3.2.7.4) can miss common apparent horizons which are found by other (local) algorithms.

Alcubierre's apparent-horizon finder AHFinder [4] includes a "fast flow" algorithm based on the work of Gundlach [72].[39] It's implemented as a module ("thorn") in the Cactus computational toolkit, and is freely available by anonymous CVS (it's part of the CactusEinstein set of thorns included with the standard Cactus distribution). It has been used by a number of research groups.

## 3.3 Summary of Algorithms/Codes for Finding Apparent Horizons

### 3.3.1 Summary of Apparent-Horizon Finding Algorithms

There are a large number of apparent-horizon finding algorithms, with differing trade-offs between speed, robustness of convergence, accuracy, and ease of programming.

In spherical symmetry, zero-finding (section 3.2.1) is fast, robust, and easy to program. In axisymmetry, shooting algorithms (section 3.2.2) work well and are

[39] AHFinder also includes a minimization algorithm (section 3.2.3).

fairly easy to program. Alternatively, any of the algorithms for generic slices (summarized below) can be used with implementations tailored to the axisymmetry.

Minimization algorithms (section 3.2.3) are fairly easy to program, but are susceptible to spurious local minima, have relatively poor accuracy, and tend to be very slow unless axisymmetry is assumed.

Nakamura *et al.*'s spectral integral-iteration algorithm (section 3.2.4) and elliptic-PDE algorithms (section 3.2.5) are both fast and accurate, but are moderately difficult to program. Their main disadvantage is the need for a fairly good initial guess for the horizon position/shape.

In many cases Schnetter's "pretracking" algorithm (section 3.2.6) can greatly improve an elliptic-PDE algorithm's robustness, by determining – *before* it appears – approximately where (in space) and when (in time) a new outermost apparent horizon will appear. Pretracking is implemented as a modification of an existing elliptic-PDE algorithm, and is moderately slow: it typically has a cost 5 to 10 times that of finding a single horizon with the elliptic-PDE algorithm.

Finally, flow algorithms (section 3.2.7) are generally quite slow (Metzger's algorithm [98] is a notable exception), but can be very robust in their convergence. They are moderately easy to program. Flow algorithms are global algorithms, in that their convergence does not depend on having a good initial guess.

### 3.3.2 Summary of Publicly-Available Apparent-Horizon Finding Codes

I know of 3 publicly-available apparent-horizon finding codes, all implemented as modules ("thorns") in the Cactus computational toolkit:

**AHFinder**
: Alcubierre's AHFinder [4] includes both a "fast flow" algorithm based on the work of Gundlach [72], and a minimization algorithm based on the work of Anninos *et al.* [7]. AHFinder is part of the CactusEinstein set of thorns included with the standard Cactus distribution, and has been used by many research groups.

**AHFinderDirect**
: Thornburg's AHFinderDirect [140] uses an elliptic-PDE algorithm, and has been used by many research groups, as well as ported to at least two non-Cactus numerical relativity codes. Schnetter's pretracking algorithm [121] is implemented as a set of modifications to AHFinderDirect.

**TGRapparentHorizon2D**
: Schnetter's TGRapparentHorizon2D [118, 119] uses an elliptic-PDE algorithm. It's no longer maintained, but remains freely available.

# Acknowledgements

I thank the many researchers who answered my E-mail queries on various aspects of their work. I thank Scott Caveny and Peter Diener for useful conversations on event-horizon finders, and Badri Krishnan for useful conversations on the properties of apparent, isolated, and dynamical horizons. I thank Peter Diener, Luciano Rezzolla, and Virginia J. Vitzthum for helpful comments on various drafts of this paper. I thank Peter Diener and Edward Seidel for providing unpublished figures.

I think the many authors named in this review for granting permission to reprint figures from their published work. I thank the American Astronomical Society, the American Physical society, and IOP Publishing for granting permission to reprint figures published in their journals. The American Physical Society requires the following disclaimer regarding such reprinted material:

> Readers may view, browse, and/or download material for temporary copying purposes only, provided these uses are for noncommercial personal purposes. Except as provided by law, this material may not be further reproduced, distributed, transmitted, modified, adapted, performed, displayed, published, or sold in whole or part, without prior written permission from the publisher.

I thank the Alexander von Humboldt foundation, the AEI visitors program, and the AEI postdoctoral fellowship program for financial support.

# Appendix A

# Solving A Single Nonlinear Algebraic Equation

In this appendix I briefly outline numerical algorithms and codes for solving a single 1-dimensional nonlinear algebraic equation $f(x) = 0$, where the function $f : \Re \to \Re$ is given.

The process generally begins by evaluating $f$ on a suitable grid of points and looking for sign changes. Assuming $f$ to be continuous, each sign change must then bracket at least one root $x_*$:

Given a pair of ordinates $x_-$ and $x_+$ which bracket a root, there are a variety of different algorithms available to accurately and efficiently find the (a) root:

If $|x_+ - x_-|$ is small, say on the order of a finite-difference grid spacing, then closed-form approximations are probably accurate enough:

- The simplest approximation is a simple linear interpolation of $f$ between $x_-$ and $x_+$.
- A slightly more sophisticated algorithm, "inverse quadratic interpolation", is to use 3 ordinates, two of which bracket a root, and estimate the root as the root of the (unique) parabola which passes through the 3 given $\big(x, f(x)\big)$ points.[1]

For larger $|x_+ - x_-|$, iterative algorithms are necessary to obtain an accurate root:

- Bisection (binary search on the sign of $f$), is a well-known iterative scheme which is very robust, but rather slow if high accuracy is desired.
- Newton's method can be used, but it requires that the the derivative $f'$ be available. Alternatively, the secant algorithm (similar to Newton's method but

[1]The parabola generically has two roots, but normally only one of them lies between $x_-$ and $x_+$.

estimating $f'$ from the most recent pair of function evaluations) gives similarly fast convergence without requiring $f'$ to be available. Unfortunately, if $|f'|$ is small enough at any iteration point, both these algorithms can fail to converge, or more generally they can generate "wild" trial ordinates.

- Probably the most sophisticated algorithm is that of van Wijngaarden, Dekker, and Brent. This is a carefully engineered hybrid of the bisection, secant, and inverse quadratic interpolation algorithms, and generally combines the rapid convergence of the secant algorithm with the robustness of bisection. The van Wijngaarden-Dekker-Brent algorithm is described by Forsythe, Malcolm, and Moler [63, chapter 7], Kahaner, Moler, and Nash [83, chapter 7], and Press *et al.* [111, section 9.3]. An excellent implementation of this, the Fortran subroutine ZEROIN, is freely available from `http://www.netlib.org/fmm/`.

# Appendix B

# The Numerical Integration of Ordinary Differential Equations

The time-integration problem[1,2] for ordinary differential equations (ODEs) is traditionally written as follows: We are given an integer $n > 0$ (the number of ODEs to integrate), a "right-hand-side" function $\mathsf{f} : \Re^n \times \Re \to \Re^n$, and the value $\mathsf{y}(0)$ of a function $\mathsf{y} : \Re \to \Re^n$ satisfying the ODE

$$\frac{d\mathsf{y}}{dt} = f(\mathsf{y}, t) \tag{B.1}$$

We wish to know (or approximate) $\mathsf{y}(t)$ for some finite interval $t \in [0, t_{\max}]$.

This is a well-studied problem in numerical analysis. See Forsythe, Malcolm, and Moler [63, chapter 6] or Kahaner, Moler, and Nash [83, chapter 8] for a general overview of ODE integration algorithms and codes, or Shampine and Gordon [125], Hindmarsh [76], or Brankin, Gladwell, and Shampine [34] for detailed technical accounts.

For our purposes, it suffices to note that highly accurate, efficient, and robust ODE-integration codes are widely available. In fact, there is a strong tradition in numerical analysis of free availability of such codes. Notably, the RKF45 code described by Forsythe, Malcolm, and Moler [63, chapter 6] is freely available at `http://www.netlib.org/ode/rkf45.f`, the ODE code described by Shampine and Gordon [125] is freely available at `http://www.netlib.org/ode/ode.f`, the ODEPACK/LSODE suite of codes described by Hindmarsh [76] are freely

[1]The numerical-analysis literature usually refers to this as the "initial value problem". Unfortunately, in a relativity context this terminology often causes confusion with the "initial data problem" of solving the ADM constraint equations. I use the term "time-integration problem for ODEs" to (try to) avoid this confusion.

[2]In this appendix sans-serif lower-case letters $\mathsf{abc}\ldots\mathsf{z}$ denote variables and functions in $\Re^n$ (for some fixed dimension $n$), and sans-serif upper-case letters $\mathsf{ABC}\ldots\mathsf{Z}$ denote $n \times n$ real-valued matrices.

available at `http://www.netlib.org/odepack/`, and the RKSUITE suite of codes described by Brankin, Gladwell, and Shampine [34] are freely available at `http://www.netlib.org/ode/rksuite/`.

As well as being of high numerical quality, these codes are also very easy to use, employing sophisticated adaptive algorithms to automatically adjust step size and/or the precise integration scheme used.[3] These codes can generally be relied upon to produce accurate results both more efficiently and more easily than a hand-crafted integrator. I have used the LSODE solver in several research projects with excellent results.

[3]LSODE can also automatically detect and handle stiff systems of ODEs.

# Bibliography


[1] Abrahams, Andrew M., Cook, Gregory B., Shapiro, Stuart L., and Teukolsky, Saul A., "Solving Einstein's Equations for Rotating Spacetimes: Evolution of Relativistic Star Clusters", *Phys. Rev. D*, **49**(10), 5153–5164, (15 May, 1994).

[2] Abrahams, Andrew M., and Evans, Charles R., "Trapping a Geon: Black Hole Formation by an Imploding Gravitational Wave", *Phys. Rev. D*, **46**, R4117–R4121, (1992).

[3] Abrahams, Andrew M., Heiderich, Karen R., Shapiro, Stuart L., and Teukolsky, Saul A., "Vacuum Initial Data, Singularities, and Cosmic Censorship", *Phys. Rev. D*, **46**(6), 2452–2463, (15 September, 1992).

[4] Alcubierre, Miguel, Brandt, S., Brügmann, B., Gundlach, C., Massó, Joan, Seidel, E., and Walker, P., "Test-beds and applications for apparent horizon finders in numerical relativity", *Class. Quantum Grav.*, **17**, 2159–2190, (2000).

[5] Alcubierre, Miguel, Brügmann, Bernd, Diener, Peter, Guzmán, Francisco Siddhartha, Hawke, Ian, Hawley, Scott, Herrmann, Frank, Koppitz, Michael, Pollney, Denis, Seidel, Edward, and Thornburg, Jonathan, "Dynamical evolution of quasi-circular binary black hole data", *Phys. Rev. D*, **72**, 044004, (5 August, 2005).

[6] Anninos, Peter, Bernstein, D., Brandt, S., Libson, J., Massó, Joan, Seidel, E., Smarr, L., Suen, W.-M., and Walker, P., "Dynamics of Apparent and Event Horizons", *Phys. Rev. Lett.*, **74**(5), 630–633, (30 January, 1995).

[7] Anninos, Peter, Camarda, Karen, Libson, Joseph, Massó, Joan, Seidel, E., and Suen, W.-M., "Finding Apparent Horizons in Dynamic 3D Numerical Spacetimes", *Phys. Rev. D*, **58**, 024003, (1998).

[8] Anninos, Peter, Daues, G., Massó, Joan, Seidel, E., and Suen, W.-M., "Horizon Boundary Conditions for Black Hole Spacetimes", *Phys. Rev. D*, **51**(10), 5562–5578, (1995).

[9] Ansorg, M., Kleinwächter, A., and Meinel, R., "Highly accurate calculation of rotating neutron stars: Detailed description of the numerical methods", *Astron. Astrophys.*, **405**, 711, (2003).

[10] Ansorg, Marcus, Brügmann, Bernd, and Tichy, Wolfgang, "A single-domain spectral method for black hole puncture data", *Phys. Rev. D*, **70**, 064011, (2004).

[11] Ansorg, Marcus, and Petroff, David, "Black holes surrounded by uniformly rotating rings", *Physical Review D*, **72**, 024019, (2005).

[12] Arnowitt, R., Deser, S., and Misner, C. W., "The Dynamics of General Relativity", in Witten, L., ed., *Gravitation: An Introduction to Current Research*, 227–265, (John Wiley, New York, 1962).

[13] Ashtekar, Abhay, Beetle, Christopher, and Fairhurst, Stephen, "Isolated Horizons: A Generalization of Black Hole Mechanics", *Class. Quantum Grav.*, **16**, L1–L7, (1999).

[14] Ashtekar, Abhay, and Galloway, Greg, "Some uniqueness results for dynamical horizons", *Advances in Theoretical and Mathematical Physics*, **to appear**, (2005).

[15] Ashtekar, Abhay, and Krishnan, Badri, "Dynamical Horizons: Energy, Angular Momentum, Fluxes, and Balance Laws", *Phys. Rev. Lett.*, **89**, 261101, (2002).

[16] Ashtekar, Abhay, and Krishnan, Badri, "Dynamical horizons and their properties", *Phys. Rev. D*, **68**, 104030, (2003).

[17] Ashtekar, Abhay, and Krishnan, Badri, "Isolated and dynamical horizons and their applications", *Living Rev. Rel.*, **7**, 10, (2004).

[18] Baiotti, Luca, Hawke, Ian, Montero, Pedro J., Löffler, Frank, Rezzolla, Luciano, Stergioulas, Nikolaos, Font, José A., and Seidel, Ed, "Three-dimensional relativistic simulations of rotating neutron star collapse to a Kerr black hole", *Phys. Rev. D*, **71**, 024035, (2005).

[19] Balay, Satish, Buschelman, Kris, Gropp, William D., Kaushik, Dinesh, Knepley, Matt, McInnes, Lois Curfman, Smith, Barry F., and Zhang, Hong, "PETSc home page", `http://www.mcs.anl.gov/petsc`, (2001).

[20] Balay, Satish, Buschelman, Kris, Gropp, William D., Kaushik, Dinesh, Knepley, Matt, McInnes, Lois Curfman, Smith, Barry F., and Zhang, Hong, *PETSc Users Manual*, (Argonne National Laboratory, Argonne, USA, 2002), Report no. ANL-95/11 - Revision 2.1.5.

[21] Balay, Satish, Gropp, William D., McInnes, Lois Curfman, and Smith, Barry F., "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries", in Arge, E., Bruaset, A. M., and Langtangen, H. P., eds., *Modern Software Tools in Scientific Computing*, 163–202. Birkhauser Press, (1997).

[22] Baumgarte, Thomas W., Cook, Gregory B., Scheel, Mark A., Shapiro, Stuart L., and Teukolsky, Saul A., "Implementing an apparent-horizon finder in three dimensions", *Phys. Rev. D*, **54**(8), 4849–4857, (1996).

[23] Baumgarte, Thomas W., and Shapiro, Stuart L., "Numerical Relativity and Compact Binaries", *Physics Reports*, **376**(2), 41–131, (March, 2003).

[24] Bernstein, David, *Notes on the Mean Curvature Flow Method for Finding Apparent Horizons*, (National Center for Supercomputing Applications, Urbana-Champaign, USA, 1993), Report no. unnumbered.

[25] Bishop, Nigel T., "The Closed Trapped Region and the Apparent Horizon of Two Schwarzschild Black Holes", *Gen. Rel. Grav.*, **14**(9), 717–723, (1982).

[26] Bishop, Nigel T., "The horizons of two Schwarzschild black holes", *General Relativity and Gravitation*, **16**, 589–593, (June, 1984).

[27] Bishop, Nigel T., "The Event Horizons of Two Schwarzschild black holes", *General Relativity and Gravitation*, **20**(6), 573–581, (1988).

[28] Bizoń, Piotr, Malec, Edward, and Ó Murchadha, Niall, "Trapped Surfaces in Spherical Stars", *Phys. Rev. Lett.*, **61**(10), 1147–1150, (5 September, 1988).

[29] Bonazzola, S., and Marck, J.-A., "Pseudo-Spectral Methods Applied to Gravitational Collapse", in Evans, C., Finn, L., and Hobill, D., eds., *Frontiers in Numerical Relativity*, 239–253, (Cambridge University Press, Cambridge, England, 1989).

[30] Bonazzola, Silvano, Frieben, Joachim, Gourgoulhon, Eric, and Marck, Jean-Alain, "Spectral methods in general relativity – toward the simulation of 3D-gravitational collapse of neutron stars", in *Proceedings of the Third International Conference on Spectral and High Order Methods, Houston Journal of Mathematics (1996), University of Houston*, (1996).

[31] Bonazzola, Silvano, Gourgoulhon, Eric, and Marck, Jean-Alain, "Spectral methods in general relativistic astrophysics", *Journal of Computational and Applied Mathematics*, **109**, 433–483, (1999).

[32] Booth, Ivan, "Black hole boundaries", gr-qc/0508107, (2005).

[33] Boyd, John P., *Chebyshev and Fourier Spectral Methods (Second Edition, Revised)*, (Dover Publications, New York, 2001).

[34] Brankin, Richard W., Gladwell, Ian, and Shampine, Lawrence F., *RKSUITE: A Suite of Runge-Kutta Codes for the Initial Value Problem for ODEs*, (Department of Mathematics, Southern Methodist University, Dallas, USA, 1992), Report no. Softreport 92-S1. `http://www.netlib.org/ode/rksuite/index.html`.

[35] Brent, Richard P., *Algorithms for Minimization Without Derivatives*, (Dover, New York, 2002), reprinted edition.

[36] Brewin, Leo C., "Is the Regge Calculus a Consistent Approximation to General Relativity?", *General Relativity and Gravitation*, **32**(5), 897–918, (May, 2000).

[37] Brewin, Leo C., and Gentle, Adrian P., "On the Convergence of Regge Calculus to General Relativity", *Classical and Quantum Gravity*, **18**(3), 517–525, (7 February, 2001).

[38] Briggs, William L., Henson, Van Emden, and McCormick, Steve F., *A Multigrid Tutorial*, (SIAM Press, Philadelphia, 2000), 2nd edition.

[39] Brill, D., and Lindquist, R., "Interaction Energy in Geometrostatics", *Phys. Rev.*, **131**(1), 471–476, (1963).

[40] Caveny, Scott A., *Tracking Black Holes in Numerical Relativity: Foundations and Applications*, PhD thesis, (University of Texas at Austin, Austin, USA, 2002).

[41] Caveny, Scott A., Anderson, Matthew, and Matzner, Richard A., "Tracking Black Holes in Numerical Relativity", *Phys. Rev. D*, **68**, 104009, (2003).

[42] Caveny, Scott A., and Matzner, Richard A., "Adaptive Event Horizon Tracking and Critical Phenomena in Binary Black Hole Coalescence", *Phys. Rev. D*, **68**, 104003, (2003).

[43] Choptuik, M. W., *A Study of Numerical Techniques for Radiative Problems in General Relativity*, PhD thesis, (University of British Columbia, Vancouver, Canada, 1986).

[44] Choptuik, M. W., "Experiences with an Adaptive Mesh Refinement Algorithm in Numerical Relativity", in Evans, C., Finn, L., and Hobill, D., eds., *Frontiers in Numerical Relativity*, 206–221, (Cambridge University Press, Cambridge, England, 1989).

[45] Chruściel, Piotr T., and Galloway, Gregory J., "Horizons Non-Differentiable on a Dense Set", *Communications in Mathematical Physics*, **193**(2), 449–470, (April, 1998).

[46] Cook, Gregory B., *Initial Data for the Two-Body Problem of General Relativity*, PhD thesis, (University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, 1990).

[47] Cook, Gregory B., and Abrahams, Andrew M., "Horizon Structure of Initial-Data Sets for Axisymmetric Two-Black-Hole Collisions", *Phys. Rev. D*, **46**(2), 702–713, (15 July, 1992).

[48] Cook, Gregory B., and York, Jr., James W., "Apparent Horizons for Boosted or Spinning Black Holes", *Phys. Rev. D*, **41**(4), 1077–1085, (15 February, 1990).

[49] Curtis, A. R., and Reid, J. K., "The Choice of Step Lengths When Using Differences to Approximate Jacobian Matrices", *J. Inst. Math. Appl.*, **13**(1), 121–126, (1974).

[50] Davis, T. A., *Algorithm 8xx: UMFPACK V3.2, an unsymmetric-pattern multifrontal method with a column pre-ordering strategy*, (Univ. of Florida, CISE Dept., Gainesville, FL, 2002), Report no. TR-02-002. (www.cise.ufl.edu/tech-reports. Submitted to *ACM Trans. Math. Softw.*).

[51] Davis, T. A., *A column pre-ordering strategy for the unsymmetric-pattern multifrontal method*, (Univ. of Florida, CISE Dept., Gainesville, FL, 2002), Report no. TR-02-001. (www.cise.ufl.edu/tech-reports. Submitted to *ACM Trans. Math. Softw.*).

[52] Davis, Timothy A., and Duff, Iain S., "An unsymmetric-pattern multifrontal method for sparse LU factorization", *SIAM J. Matrix Anal. Applic.*, **18**(1), 140–158, (1997).

[53] Davis, Timothy A., and Duff, Iain S., "A combined unifrontal/multifrontal method for unsymmetric sparse matrices", *ACM Trans. Math. Softw.*, **25**(1), 1–19, (1999).

[54] Diener, P., "A New General Purpose Event Horizon Finder for 3D Numerical Spacetimes", *Class. Quantum Grav.*, **20**(22), 4901–4917, (2003).

[55] Diener, Peter, "personal communication", Diener is enhancing EHFinder to work with Schnetter's Carpet mesh-refinement driver ([120, 117])., (2005).

[56] Diener, Peter, Herrmann, Frank, Pollney, Denis, Schnetter, Erik, Seidel, Edward, Takahashi, Ryoji, Thornburg, Jonathan, and Ventrella, Jason, "Accurate Evolution of Orbiting Binary Black Holes", gr-qc/0512108, (2006).

[57] Dreyer, Olaf, Krishnan, Badri, Shoemaker, Deirdre, and Schnetter, Erik, "Introduction to Isolated Horizons in Numerical Relativity", *Phys. Rev. D*, **67**, 024018, (2003).

[58] Du Fort, E. C., and Frankel, S. P., "Stability Conditions in the Numerical Treatment of Parabolic Differential Equations", *Mathematical Tables and Other Aids to Computation*, **7**, 135, (1953).

[59] Duff, Iain S., Erisman, A. M., and Reid, John K., *Direct Methods for Sparse Matrices*, (Oxford University Press, Oxford, UK, 1986).

[60] Dykema, P. G., *The Numerical Simulation of Axially Symmetric Gravitational Collapse*, PhD thesis, (University of Texas at Austin, Austin, USA, 1980).

[61] Eardley, Douglas M., "Gravitational Collapse of Marginally Bound Spheroids: Initial Conditions", *Phys. Rev. D*, **12**(10), 3072–3076, (15 November, 1975).

[62] Eppley, Kenneth R., "Evolution of time-symmetric gravitational waves: Initial data and apparent horizons", *Phys. Rev. D*, **16**(6), 1609–1614, (1977).

[63] Forsythe, George E., Malcolm, Michael A., and Moler, Cleve B., *Computer Methods for Mathematical Computations*, (Prentice-Hall, Englewood Cliffs, 1977). software available at `http://www.netlib.org/fmm/`.

[64] Gentle, Adrian P., "Regge Calculus: A Unique Tool for Numerical Relativity", *General Relativity and Gravitation*, **34**, 1701–1718, (2002).

[65] Gentle, Adrian P., and Miller, Warner A., "A fully (3+1)-Dimensional Regge Calculus Model of the Kasner Cosmology", *Classical and Quantum Gravity*, **15**(2), 389–405, (February, 1998).

[66] Goodale, T., Allen, G., Lanfermann, G., Massó, J., Radke, T., Seidel, E., and Shalf, J., "The Cactus Framework and Toolkit: Design and Applications", in *Vector and Parallel Processing - VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science*, (Springer, Berlin, 2003).

[67] Gottlieb, David, and Orszag, Steven A., *Numerical Analysis of Spectral Methods*, volume 26 of *CMBS-NSF Regional Conference Series in Applied Mathematics*, (Society for Industrial and Applied Mathematics, Philadelphia, 1977, 1981, 1983, 1986, 1989).

[68] Gourgoulhon, Eric, and Jaramillo, Josè Luis, "A $3 + 1$ perspective on null hypersurfaces and isolated horizons", gr-qc/0503113, (2005).

[69] Grandclément, Philippe, Bonazzola, Silvano, Gourgoulhon, Eric, and Marck, Jean-Alain, "A multi-domain spectral method for scalar and vectorial Poisson equations with non-compact sources", *Journal of Computational Physics*, **170**, 231–260, (2001).

[70] Grandclément, Philippe, Gourgoulhon, Eric, and Bonazzola, Silvano, "Binary black holes in circular orbits. II. Numerical methods and first results", *Phys. Rev. D*, **65**, 044021, (2002).

[71] Grayson, Matthew A., "The Heat Equation Shrinks Embedded Plane Curves to Round Points", *Journal of Differential Geometry*, **26**(2), 285–314, (1987).

[72] Gundlach, C., and Walker, P., "Causal differencing of flux-conservative equations applied to black hole spacetimes", *Class. Quantum Grav.*, **16**, 991–1010, (1999).

[73] Hawking, S. W., "The Event Horizon", in DeWitt, C., and DeWitt, B. S., eds., *Black Holes*, 1–55, (Gordon and Breach, New York, 1973).

[74] Hawking, S. W., and Ellis, G. F. R., *The Large Scale Structure of Spacetime*, (Cambridge University Press, Cambridge, England, 1973).

[75] Hayward, Sean A., "General laws of black hole dynamics", *Phys. Rev. D*, **49**(12), 6467–6474, (15 June, 1994).

[76] Hindmarsh, Alan C., "ODEPACK, A Systematized Collection of ODE Solvers", *IMACS Transactions on Scientific Computing*, **1**, 55–64, (1983). article also available at http://www.llnl.gov/CASC/nsde/pubs/u88007.pdf.

[77] Hochbruck, Marlis, Lubich, Christian, and Selhofer, Hubert, "Exponential Integrators for Large Systems of Differential Equations", *SIAM Journal of Scientific Computing*, **19**(5), 1552–1574, (September, 1998).

[78] Hughes, Scott, Keeton, Charles R., II, Walker, Paul, Walsh, Kevin, Shapiro, Stuart L., and Teukolsky, Saul A., "Finding Black Holes in Numerical Spacetimes", *Phys. Rev. D*, **49**(8), 4004–4015, (15 April, 1994).

[79] Huq, Mijan F., *Apparent Horizon Location in Numerical Spacetimes*, PhD thesis, (The University of Texas at Austin, Austin, USA, 1996).

[80] Huq, Mijan F., Choptuik, M. W., and Matzner, Richard A., "Locating Boosted Kerr and Schwarzschild Apparent Horizons", *Phys. Rev. D*, **66**, 084024, (2002).

[81] Husa, Sascha, and Winicour, Jeffrey, "The asymetric merger of black holes", *Phys. Rev. D*, **60**(8), 084019, (1999).

[82] Jr., J. E. Dennis, and Schnabel, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, (Prentice-Hall, Englewood Cliffs, 1978).

[83] Kahaner, David, Moler, Cleve B., and Nash, Stephen, *Numerical Methods and Software*, (Prentice-Hall, Englewood Cliffs, 1989).

[84] Kemball, A. J., and Bishop, Nigel T., "The numerical determination of apparent horizons", *Class. Quantum Grav.*, **8**(7), 1361–1367, (July, 1991).

[85] Kershaw, David S., "The Incomplete Cholesky – Conjugate Gradient Method for Interative Solution of Linear Equations", *J. Comput. Phys.*, **26**(1), 43–65, (January, 1978).

[86] Kidder, L. E., and Finn, L. S., "Spectral Methods for Numerical Relativity. The Initial Data Problem", *Phys. Rev. D*, **62**, 084026, (2000).

[87] Kidder, L. E., Scheel, Mark A., Teukolsky, Saul A., Carlson, E. D., and Cook, Gregory B., "Black hole evolution by spectral methods", *Phys. Rev. D*, **62**, 084032, (2000).

[88] Kidder, Lawrence, Scheel, Mark, Teukolsky, Saul, and Cook, Greg, "Spectral Evolution of Einstein's Equations", in *Miniprogram on Colliding Black Holes: Mathematical Issues in Numerical Relativity*, (Institute for Theoretical Physics, UCSB, Santa Barbara, CA, 2000).

[89] Lehner, Luis, Bishop, Nigel T., Gómez, Roberto, Szilagyi, Bela, and Winicour, Jeffrey, "Exact solutions for the intrinsic geometry of black hole coalescence", *Phys. Rev. D*, **60**(4), 044005, (15 August, 1999).

[90] Lehner, Luis, Gómez, Roberto, Husa, Sascha, Szilagyi, Bela, Bishop, Nigel T., and Winicour, Jeffrey, "Bagels Form When Black Holes Collide", `http://www.psc.edu/research/graphics/gallery/winicour.html`.

[91] Leiler, Gregor, and Rezzolla, Luciano, "On the iterated Crank-Nicolson method for hyperbolic and parabolic equations in numerical relativity", to appear in Physical Review D, (2006).

[92] Libson, J., Massó, Joan, Seidel, E., and Suen, W.-M., "A 3D Apparent Horizon Finder", in Jantzen, R. T., Keiser, G. M., and Ruffini, R., eds., *The Seventh Marcel Grossmann Meeting: On Recent Developments in Theoretical and Experimental General Relativity, Gravitation, and Relativistic Field Theories*, 631, (World Scientific, Singapore, 1996).

[93] Libson, Joseph, Massó, Joan, Seidel, Edward, Suen, Wai-Mo, and Walker, Paul, "Event horizons in numerical relativity: Methods and tests", *Phys. Rev. D*, **53**(8), 4335–4350, (1996).

[94] Lorensen, William E., and Cline, Harvey E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *Computer Graphics*, **21**(4), 163–169, (1987). Proceedings of the SIGGRAPH'87 Conference.

[95] MacNeice, Peter, Olson, Kevin M., Mobarry, Clark, de Fainchtein, Rosalinda, and Packer, Charles, "PARAMESH: A parallel adaptive mesh refinement community toolkit", *Computer Physics Communications*, **126**(3), 330–354, (11 April, 2000).

[96] Madderom, P., "Incomplete LU-Decomposition – Conjugate Gradient", Fortran 66 subroutine implementing the method of [85], (1984).

[97] Matzner, Richard A., Seidel, E., Shapiro, Stuart L., Smarr, L., Suen, W.-M., Teukolsky, S., and Winicour, Jeffrey, "Geometry of a Black Hole Collision", *Science*, **270**, 941–947, (November 10, 1995).

[98] Metzger, Jan, "Numerical computation of constant mean curvature surfaces using finite elements", *Class. Quantum Grav.*, **21**(19), 4625–4646, (2004).

[99] Miller, Mark A., "Regge Calculus as a Fourth Order Method in Numerical Relativity", *Classical and Quantum Gravity*, **12**(12), 3037–3051, (December, 1995).

[100] Misner, Charles W., and Sharp, D. H., "Relativistic Equations for Adiabatic, Spherically Symmetric Gravitational Collapse", *Phys. Rev. B*, **136**(2), 571–576, (1964).

[101] Misner, Charles W., Thorne, Kip S., and Wheeler, John A., *Gravitation*, (W. H. Freeman, San Francisco, 1973).

[102] Nakamura, Takashi, Kojima, Yasufumi, and Oohara, Ken-ichi, "A Method of Determining Apparent Horizons in Three-Dimensional Numerical Relativity", *Physics Letters A*, **106**(5-6), 235–238, (10 December, 1984).

[103] Nakao, Ken-ichi, "personal communication to Masaru Shibata, cited in [131]".

[104] Oohara, Ken-ichi, "Apparent Horizon of Initial Data for Black Hole Collisions", in Sato, Humitaka, and Nakamura, Takashi, eds., *Gravitational Collapse and Relativity*, 313–319, (World Scientific, Singapore, 1986).

[105] Oohara, Ken-ichi, Nakamura, Takashi, and Kojima, Yasufumi, "Apparent Horizons of Time-Symmetric Initial Value for Three Black Holes", *Physics Letters A*, **107**(9), 452–455, (4 March, 1985).

[106] Osher, Stanley, and Sethian, James A., "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations", *Journal of Computational Physics*, **79**(1), 12–49, (November, 1988).

[107] Parashar, M., and Browne, J. C., "IMA Volume on Structured Adaptive Mesh Refinement (SAMR) Grid Methods", chapter System Engineering for High Performance Computing Software: The HDDA/DAGH Infrastructure for Implementation of Parallel Structured Adaptive Mesh Refinement, 1–18, (Springer-Verlag, 2000).

[108] Pasch, Eberhard, *The level set method for the mean curvature flow on* $(\Re^3, g)$, (University of Tübingen, Tübingen, Germany, 1997), Report no. 63. report of the Sonderforschungsbereich 382 project, `http://www.uni-tuebingen.de/uni/opx/reports/pasch_63.ps.gz`.

[109] Petrich, Loren I., Shapiro, Stuart L., and Teukolsky, Saul A., "Oppenheimer-Snyder Collapse with Maximal Time Slicing and Isotropic Coordinates", *Phys. Rev. D*, **31**(10), 2459–2469, (15 May, 1985).

[110] Pfeiffer, Harald P., Kidder, Lawrence E., Scheel, Mark A., and Teukolsky, Saul A., "A multidomain spectral method for solving elliptic equations", *Comput. Phys. Commun.*, **152**, 253–273, (2003).

[111] Press, W. H., Flannery, B. P., Teukolsky, Saul A., and Vetterling, W. T., *Numerical Recipes*, (Cambridge University Press, New York, 1992), 2nd edition.

[112] Pretorius, Frans, and Choptuik, Matthew W., "Adaptive Mesh Refinement for Coupled Elliptic-Hyperbolic Systems", gr-qc/0508110, (2005).

[113] Pretorius, Frans, and Lehner, Luis, "Adaptive mesh refinement for characteristic codes", *Journal of Computational Physics*, **198**, 10–34, (2004).

[114] Regge, T., "General Relativity without Coordinates", *Nuovo Cim.*, **19**, 558–571, (1961).

[115] Richtmyer, Robert D., and Morton, K.W., *Difference Methods for Initial Value Problems*, (Kreiger, Malabar, Florida, USA, 1994), 2nd edition.

[116] Saad, Yousef, *Iterative Methods for Sparse Linear Systems*, (SIAM Press, Philadelphia, USA, 2003), 2nd edition.

[117] Schnetter, Erik, CARPET: A Mesh Refinement driver for CACTUS, http://www.carpetcode.org.

[118] Schnetter, Erik, "A fast apparent horizon algorithm", gr-qc/0206003, (2002).

[119] Schnetter, Erik, "Finding Apparent Horizons and other Two-Surfaces of Constant Expansion", *Class. Quantum Grav.*, **20**(22), 4719–4737, (2003).

[120] Schnetter, Erik, Hawley, Scott H., and Hawke, Ian, "Evolutions in 3D numerical relativity using fixed mesh refinement", *Class. Quantum Grav.*, **21**(6), 1465–1488, (21 March, 2004).

[121] Schnetter, Erik, Herrmann, Frank, and Pollney, Denis, "Horizon Pretracking", *Phys. Rev. D*, **71**, 044033, (2005).

[122] Schnetter, Erik, and Krishnan, Badri, "Non-symmetric trapped surfaces in the Schwarzschild and Vaidya spacetimes", gr-qc/0511017.

[123] Schroeder, Manfred R., *Number Theory in Science and Communication*, (Springer-Verlag, Berlin, 1986), 2nd enlarged edition.

[124] Seidel, E., and Suen, W.-M., "Towards a Singularity-Proof Scheme in Numerical Relativity", *Phys. Rev. Lett.*, **69**(13), 1845–1848, (1992).

[125] Shampine, Lawrence F., and Gordon, Marilyn Kay, *Computer solution of Ordinary Differential Equations*, (W. H. Freeman and Company, San Francisco, 1975).

[126] Shapiro, Stuart L., and Teukolsky, Saul A., "Gravitational Collapse of Supermassive Stars to Black Holes: Numerical Solution of the Einstein Equations", *Astrophysical Journal Letters*, **234**, L177–L181, (December 15, 1979).

[127] Shapiro, Stuart L., and Teukolsky, Saul A., "Gravitational Collapse to Neutron Stars and Black Holes: Computer Generation of Spherical Spacetimes", *Astrophysical Journal*, **235**, 199–215, (1980).

[128] Shapiro, Stuart L., and Teukolsky, Saul A., "Relativistic stellar dynamics on the computer. I. Motivation and Numerical Method", *Astrophys. J.*, **298**, 34–57, (November 1, 1985).

[129] Shapiro, Stuart L., and Teukolsky, Saul A., "Relativistic stellar dynamics on the computer. II. Physical applications", *Astrophys. J.*, **298**, 58–79, (November 1, 1985).

[130] Shapiro, Stuart L., and Teukolsky, Saul A., "Collision of relativistic clusters and the formation of black holes", *Phys. Rev. D*, **45**(8), 2739–2750, (1992).

[131] Shibata, Masaru, "Apparent horizon finder for a special family of spacetimes in 3D numerical relativity", *Phys. Rev. D*, **55**(4), 2002–2013, (15 February, 1997).

[132] Shibata, Masaru, and Uryū, Kōji, "Apparent Horizon Finder for General Three-Dimensional Spaces", *Phys. Rev. D*, **62**, 087501, (2000).

[133] Shoemaker, Deirdre M., *Apparent Horizons in Binary Black Hole Spacetimes*, PhD thesis, (The University of Texas at Austin, Austin, USA, 1999).

[134] Shoemaker, Deirdre M., Huq, Mijan F., and Matzner, Richard A., "Generic tracking of multiple apparent horizons with level flow", *Phys. Rev. D*, **62**, 124005, (2000).

[135] Stoer, J., and Bulirsch, R., *Introduction to Numerical Analysis*, (Springer-Verlag, Berlin and New York, 1980).

[136] Teukolsky, S., "On the Stability of the Iterated Crank-Nicholson Method in Numerical Relativity", *Phys. Rev. D*, **61**, 087501, (2000).

[137] Thornburg, Jonathan, "Finding apparent horizons in numerical relativity", *Phys. Rev. D*, **54**(8), 4899–4918, (October 15, 1996).

[138] Thornburg, Jonathan, "A 3+1 Computational Scheme for Dynamic Spherically Symmetric Black Hole Spacetimes – I: Initial Data", *Phys. Rev. D*, **59**(10), 104007, (1999).

[139] Thornburg, Jonathan, "A 3+1 Computational Scheme for Dynamic Spherically Symmetric Black Hole Spacetimes – II: Time Evolution", gr-qc/9906022, (1999).

[140] Thornburg, Jonathan, "A Fast Apparent-Horizon Finder for 3-Dimensional Cartesian Grids in Numerical Relativity", *Class. Quantum Grav.*, **21**(2), 743–766, (21 January, 2004).

[141] Thorne, Kip S., "suggestion (cited in [6, 93, 147]) to solve for the *union* of event horizon and all its null generators (including those which haven't yet joined the surface), since this *is* a smooth surface", (1994).

[142] Tod, K. P., "Looking for marginally trapped surfaces", *Class. Quantum Grav.*, **8**, L115–L118, (1991).

[143] Trottenberg, Ulrich, Oosterlee, Cornelis, and Schüller, Anton, *Multigrid*, (Academic Press, San Diego, 2001).

[144] Čadež, A., "Apparent Horizons in the Two-Black-Hole Problem", *Ann. Phys.*, **83**, 449–457, (1974).

[145] Wald, Robert M., *General Relativity*, (The University of Chicago Press, Chicago, 1984).

[146] Wald, Robert M., and Iyer, Vivek, "Trapped Surfaces in the Schwarzschild Geometry and Cosmic Censorship", *Phys. Rev. D*, **44**, R3719–R3722, (1991).

[147] Walker, Paul, *Horizons, Hyperbolic Systems, and Inner Boundary Conditions in Numerical Relativity*, PhD thesis, (University of Illinois at Urbana-Champaign, Urbana, Illinois, 1998).

[148] York, J., "Initial Data for Collisions of Black Holes and Other Gravitational Miscellany", in Evans, C., Finn, L., and Hobill, D., eds., *Frontiers in Numerical Relativity*, 89–109, (Cambridge University Press, Cambridge, England, 1989).

[149] York, James W., "Kinematics and Dynamics of General Relativity", in Smarr, Larry L., ed., *Sources of Gravitational Radiation*, 83–126, (Cambridge University Press, Cambridge, UK, 1979).