Finding Apparent Horizons in Numerical Relativity

Jonathan Thornburg

Physics Department, University of British Columbia
Vancouver, British Columbia, V6T 1Z1, Canada
(August 8, 1995)

Abstract

We review various algorithms for finding apparent horizons in 3+1 numerical relativity. We then focus on one particular algorithm, in which we pose the apparent horizon equation \( H = \nabla_i n^i + K_{ij} n^i n^j - K = 0 \) as a nonlinear elliptic (boundary-value) PDE on angular-coordinate space for the horizon shape function \( r = h(\theta, \phi) \), finite difference this PDE, and use Newton's method or a modification thereof to solve the finite difference equations.

We describe a method for computing the Jacobian matrix of the finite differenced \( H(h) \) function \( H(h) \) by symbolically differentiating the finite difference equations, giving the Jacobian elements directly in terms of the finite difference molecule coefficients used in computing \( H(h) \). Assuming the finite differencing scheme commutes with linearization, we show how the Jacobian elements may be computed by first linearizing the continuum \( H(h) \) equations, then finite differencing the linearized (continuum) equations. (This is essentially just the "Jacobian part" of the Newton-Kantorovich method for solving nonlinear PDEs.) We tabulate the resulting Jacobian coefficients for several different \( H(h) \) computation schemes.

We find this symbolic differentiation method of computing the \( H(h) \) Jacobian to be much more efficient than the usual numerical-perturbation method, and also much easier to implement than is commonly thought.

When solving the (discrete) \( H(h) = 0 \) equations, we find that Newton's method generally converges very rapidly. However, we find that it has a small (poor) radius of convergence if the initial guess for the horizon position contains significant high-spatial-frequency error components, i.e. angular Fourier components varying as (say) \( \cos m\theta \) with \( m \gtrsim 8 \). (Such components occur naturally if spacetime contains significant amounts of high-frequency gravitational radiation.) We show that this poor convergence behavior is not an artifact of insufficient resolution in the finite difference grid; rather, it

---

*Address for correspondence: Box 129, Thetis Island, British Columbia, V0R 2Y0, Canada, E-mail thornbur@physics.ubc.ca
appears to be caused by a strong nonlinearity in the continuum $H(h)$ function for high-spatial-frequency error components in $h$. We find that a simple “line search” modification to Newton’s method roughly doubles the horizon finder’s radius of convergence, but both the unmodified and modified methods’ radii of convergence still fall rapidly with increasing spatial frequency, approximately as $1/m^{3/2}$. Further research is needed to explore more robust numerical algorithms for solving the $H(h) = 0$ equations.

Provided it converges, the Newton’s-method algorithm for horizon finding is potentially very accurate, in practice limited only by the accuracy of the $H(h)$ finite differencing scheme. Using 4th order finite differencing, we demonstrate that the resulting accuracies for the horizon position show the expected $O((\Delta \theta)^4)$ scaling with angular grid resolution $\Delta \theta$; we find typical accuracies in the $10^{-5}$ range for $\Delta \theta = \pi/16$.

Finally, we briefly discuss the global problem of finding or recognizing the outermost apparent horizon in a slice. We argue that this is an important problem, and that no reliable algorithms currently exist for it except in spherical symmetry.

04.25.Dm, 02.70.Bf, 02.60.Cb, 02.60.Lj
In 3+1 numerical relativity, one often wishes to locate the black hole(s) in a (spacelike) slice. As discussed by Refs. [31,32], a black hole is rigorously defined as the interior of its event horizon, the boundary of future null infinity's causal past. Although the event horizon has, in the words of Hawking and Ellis (Ref. [33]), "a number of nice properties", it's defined in an inherently acausal manner: it can only be determined if the entire future development of the slice is known. (As discussed by Refs. [3,38], in practice the event horizon may be located to good accuracy given only the usual numerically generated approximate development to a nearly stationary state, but the fundamental acausality remains.)

In contrast, an apparent horizon (also known as a marginally trapped surface) is defined locally in time, within a single slice, as a closed 2-surface whose outgoing null geodesics have zero expansion. Given certain technical assumptions, the presence of an apparent horizon in a slice implies the presence of an event horizon (and thus by definition a black hole) containing the apparent horizon. (Unfortunately, the converse doesn't always hold: Wald and Iyer (Ref. [57]) have constructed a family of (angularly anisotropic) slices in Schwarzschild spacetime which approach arbitrarily close to $r = 0$ yet contain no apparent horizons.) In a stationary spacetime event and apparent horizons coincide, although this generally isn't the case in dynamic spacetimes.

There is thus considerable interest in numerical algorithms to find apparent horizons in numerically computed slices, both as diagnostic tools for locating black holes and studying their behavior (see, for example, Refs. [3,4]), and for use "on the fly" during numerical evolutions to help in choosing the coordinates and "steering" the numerical evolution (Refs. [53,49,54,5]). This latter context makes particularly strong demands on a horizon-finding algorithm: Because the computed horizon position is used in the coordinate conditions, the horizon must be located quite accurately to ensure that spurious finite difference instabilities don't develop in the time evolution. Furthermore, the horizon must be relocated at each time step of the evolution, so the horizon-finding algorithm should be as efficient as possible. Finally, when evolving multiple-black-hole spacetimes in this manner it's desirable to have a means of detecting the appearance of a new outermost apparent horizon surrounding two black holes which are about to merge. We discuss this last problem further in section XI.

In this paper we give a detailed discussion of the "Newton's method" apparent-horizon-finding algorithm. This algorithm poses the apparent horizon equation as a nonlinear elliptic (boundary-value) PDE on angular-coordinate space for the horizon shape function $r = h(\theta, \phi)$, finite differences this PDE, and uses some variant of Newton's method to solve the resulting set of simultaneous nonlinear (algebraic) equations for the values of $h$ at the angular-coordinate grid points. This algorithm is suitable for both axisymmetric and fully-general spacetimes, and we discuss both cases. As explained in section II, we assume a locally polar spherical topology for the coordinates and finite differencing, though we make no assumptions about the basis used in taking tensor components.
II. NOTATION

Our notation generally follows that of Misner, Thorne, and Wheeler (Ref. [41]), with \(G = c = 1\) units and a \((- , +, +, +\) spacetime metric signature. We assume the usual Einstein summation convention for repeated indices (regardless of their tensor character), and we use the Penrose abstract-index notation, as described by (for example) Ref. [56]. We use the standard \(3 + 1\) formalism of Arnowitt, Deser, and Misner (Ref. [7]) (see Refs. [58,59] for recent reviews).

Except in section XI, for the remainder of this paper we assume that a specific \(3 + 1\) (spacelike) slice has been chosen, and all our discussions take place within this slice. We use the term “horizon” to refer to the (an) apparent horizon in this slice. We refer to various sets in the slice as being 1-, 2-, or 3-dimensional, meaning the number of spatial dimensions – the time coordinate is never included in the dimensionality count. For example, we refer to the horizon itself as 2-dimensional.

We use \(i, j, k,\) and \(l\) for spatial (3-) indices. \(g_{ij}\) denotes the 3-metric in the slice, \(g\) its determinant, and \(\nabla_i\) the associated 3-covariant derivative operator. \(K_{ij}\) denotes the 3-extrinsic curvature of the slice, and \(K\) its trace.

We assume that the spatial coordinates \(x^i = (r, x^2, x^3)\) are such that in some neighborhood of the horizon, surfaces of constant \(r\) are topologically nested 2-spheres with \(r\) increasing outward, and we refer to \(r\) as a “radial” coordinate and \(x^2\) and \(x^3\) as “angular” coordinates. We use \(u, v, w, a, b,\) and \(c\) for indices ranging over the angular coordinates only. For pedagogical convenience (only), we take \(x^2\) and \(x^3\) to be the usual polar spherical coordinates \(\theta\) and \(\phi\), and if spacetime is axisymmetric (spherically symmetric) we take \(\phi (\theta\) and \(\phi)\) to be the symmetry coordinate(s). However, we make no assumptions about the detailed form of the coordinates, i.e. we allow all components of the 3-metric \(g_{ij}\) to be nonzero.

We emphasize that although our assumptions about the local topology of \(r\) are fundamental, our assumptions about the angular coordinates are for pedagogical convenience only, and could easily be eliminated. In particular, all our discussions carry over unchanged to multiple black hole spacetimes, using (for example) either Čadež conformal-mapping equipotential coordinates (Ref. [16]) or multiple-coordinate-patch coordinate systems (Ref. [52]).

We use \(A\) to denote the 2-dimensional space of angular coordinates \((\theta, \phi)\). We sometimes need to distinguish between field variables defined on \(A\) or on the (2-dimensional) horizon, and field variables defined on a 3-dimensional neighborhood \(\mathcal{N}\) of the horizon. This distinction is often clear from context, but where ambiguity might arise we use prefixes \((^2)\) and \((^3)\) respectively, as in \((^2)H\) and \((^3)H\).

We generally use italic Roman letters \(H, h,\) etc., to denote continuum coordinates, functions, differential operators, and other quantities. We use sans serif Roman letters \(H, h,\) etc., to denote grid functions, and small capital Roman indices \(i, j,\) and \(k\) to index grid points. We use subscript grid-point indices to denote the evaluation of a continuum or grid function at a particular grid point, as in \(H_i\) or \(H_j\). We use \(J[P(Q)]\) to denote the (continuum) linearization of the differential operator \(P(Q)\). We use \(J[P(Q)]\) to denote the Jacobian matrix of the grid function \(P = P(Q)\), and \(\cdot\) to denote the product of two such Jacobians or that of a Jacobian and a grid function.

We use \(M\) as a generic finite difference molecule and \(m\) as a generic index for molecule
coefficients. We write $M \in M$ to mean that an index-$M$ coefficient exists in $M$, and we refer to the largest such value of $||M||_\infty$, i.e. the largest-magnitude such component of $M$ in any coordinate direction, as the "radius" of $M$. (For example, a symmetric 3-point molecule has radius 1, and a symmetric 5-point molecule has radius 2.) In a slight abuse of mathematical correctness, we use the term "convolution" to describe the application of a finite difference molecule to a grid function, as defined by Eq. (20). We often refer to a molecule as itself being a discrete operator, the actual application to (in our terminology, convolution with) a grid function being implicit.

Given a grid function $f$ and a set of points $\{x_k\}$ in its domain, we use $\text{interp}(f(x), x = a)$ to mean an interpolation of the values $f(x_k)$ to the point $x = a$ and $\text{interp}'(f(x), x = a)$ to mean the derivative of the same interpolant at this point. More precisely, taking $I$ to be a smooth interpolating function such that $I(x_k) = f(x_k)$ for each $k$, $\text{interp}(f(x), x = a)$ denotes $I(a)$ and $\text{interp}'(f(x), x = a)$ denotes $(\partial I/\partial x)|_{x=a}$. In practice we take $I$ to be a Lagrange interpolating polynomial, so both $\text{interp}(f(x), x = a)$ and $\text{interp}'(f(x), x = a)$ are linear combinations of the $f(x_k)$ values.

III. THE APPARENT HORIZON EQUATION

As discussed by (for example) Ref. [60], an apparent horizon satisfies the equation

$$H \equiv (2)H = \nabla n^i + K_{ij}n^j - K = 0,$$

where $n^i$ is the outward-pointing unit normal to the horizon, all the field variables are evaluated on the horizon surface, and where for future use we define the "horizon function" $H \equiv (2)H$ as the left hand side of Eq. (1).

(The form of this definition of $H$ raises an interesting question: given that $n^i$ is only defined on the horizon, a 2-surface, what does the 3-divergence $\nabla n^i$ mean? The answer to this question is that $n^i$ must be (smoothly) continued off the horizon, and extended to a field $(^3)n^i$ in some 3-dimensional neighborhood of the horizon. It's easy to see that $H$ on the horizon is independent of the off-horizon $(^3)n^i$ values, so the non-uniqueness of the off-horizon continuation isn't a problem.)

To solve the apparent horizon equation Eq. (1), we begin by assuming that the horizon and coordinates are such that each radial coordinate line $\{(\theta, \phi) = \text{constant}\}$ intersects the horizon in exactly one point, i.e. that the horizon's coordinate shape is a "Strahlkörper", as described by Ref. [48]:

Another intriguing geometrical concept [invented] by Minkowski is that of a Strahlkörper (literally: ray body) defined as a region in $n$-dimensional Euclidean space containing the origin and whose surface, as seen from the origin, exhibits only one point in any direction. In other words, if the inner region was [sic] made of transparent glass and only the [horizon] surface was made opaque, then the origin would be visible from each surface point of the Strahlkörper (i.e. there are no intervening surface points).

Given this assumption, then we may parameterize the horizon's shape by $r = h(\theta, \phi)$ for some (single-valued) "horizon shape function" $h$ defined on the 2-dimensional domain $A$ of angular coordinates $(\theta, \phi)$.
Equivalently, we may write the horizon’s shape as \((^3F = 0)\), where the scalar function \((^3F)\), defined on some 3-dimensional neighborhood \(\mathcal{N}\) of the horizon, satisfies \((^3F = 0)\) if and only if \(r = h(\theta, \phi)\), and \((^3F)\) increases outward. In practice we take \((^3F(r, \theta, \phi) = r - h(\theta, \phi))\).

We define the non-unit outward-pointing normal (field) to the horizon by

\[
s_i \equiv (^3 s_i) = \nabla_i (^3 F),
\]

i.e.

\[
s_r = 1, \quad s_u = -\partial_u h.
\]

and the (contravariant) outward-pointing unit normal (field) to the horizon by

\[
^i n = (^3 n^i) = \frac{s^i}{\|s^i\|}
\]

\[
= \frac{g^{ij} s_j}{\sqrt{g^{kl} s_k s_l}}
\]

\[
= \frac{g^{ir} - g^{iu} \partial_u h}{\sqrt{g^{rr} - 2g^{ru} \partial_u h + g^{uv}(\partial_u h)(\partial_v h)}}.
\]

Henceforth we drop the \((^3)\) prefixes on \((^3 F)\), \((^3 s_i)\), and \((^3 n^i)\).

Substituting Eq. (6) into the apparent horizon equation Eq. (1), we see that the horizon function \(H(h)\) depends on the (angular) 2nd derivatives of \(h\). In fact, the apparent horizon equation Eq. (1) is a 2nd order elliptic (boundary-value) PDE for \(h\) on the domain of angular coordinates \(\mathcal{A}\).

The apparent horizon equation Eq. (1) must therefore be augmented with suitable boundary conditions to define a (locally) unique solution. In a fully-general spacetime (one with no continuous symmetries), only the obvious continuity and smoothness conditions across the artificial boundaries at \(\theta = \{0, \pi\}\) and \(\phi = \{0, 2\pi\}\) need be imposed on \(h\), while in an axisymmetric spacetime the (stricter) Neumann condition \(\partial_s h = 0\) must be imposed at the \(z\) axis boundaries \(\theta = \{0, \pi\}\) to ensure the horizon’s 3-dimensional smoothness there.

IV. ALGORITHMS FOR SOLVING THE APPARENT HORIZON EQUATION

We now survey various algorithms for solving the apparent horizon equation Eq. (1). Ref. [43] reviews much of the previous work on this topic.

In spherical symmetry, the apparent horizon equation Eq. (1) degenerates into a 1-dimensional transcendental equation for the horizon radius \(h\). This is easily solved by zero-finding on the horizon function \(H(h)\). This technique has been used by a number of authors, for example Refs. [46,18,49,5]. (See also Ref. [13] for an interesting analytical study giving necessary and sufficient conditions for apparent horizons to form in non-vacuum spherically symmetric spacetimes.)

In an axisymmetric spacetime, the angular-coordinate space \(\mathcal{A}\) is effectively 1-dimensional, so the apparent horizon equation Eq. (1) reduces to a nonlinear 2-point boundary
value (ODE) problem for the function \( h(\theta) \), which may be solved either with a shooting method, or with one of the more general algorithms described below. Shooting algorithms have been used by a number of authors, for example Refs. [17,27,1,11,12,2].

The remaining apparent-horizon-finding algorithm we discuss are all applicable to either axisymmetric or fully general spacetimes.

Tod (Ref. [55]) has proposed an interesting pair of “curvature flow” methods for finding apparent horizons. Bernstein (Ref. [10]) has tested these methods in several axisymmetric spacetimes, and reports favorable results. Unfortunately, the theoretical justification for these methods’ convergence is only valid in time-symmetric \((K_{ij} = 0)\) slices.

The next two algorithms we discuss are both based on a pseudospectral expansion of the horizon shape function \( h(\theta, \phi) \) in some complete set of basis functions (typically spherical harmonics or symmetric trace-free tensors), using some finite number of the expansion coefficients \( \{a_k\} \) to parameterize of the horizon shape. One algorithm rewrites the apparent horizon equation \( H(a_k) = 0 \) as \( \|H(a_k)\| = 0 \), then uses a general-purpose function-minimization routine to search \( \{a_k\}\)-space for a minimum of \( \|H\| \). This algorithm has been used by Refs. [15,29] in axisymmetric spacetimes, and more recently by Ref. [37] in fully general spacetimes. Alternatively, Nakamura, Oohara, and Kojima (Refs. [42,45,44]) have suggested a functional iteration scheme for directly solving the apparent horizon equation \( H(a_k) = 0 \) for the expansion coefficients \( \{a_k\} \), and have used it in a number of fully general spacetimes. Kemball and Bishop (Ref. [39]) have suggested and tested several modifications to this latter algorithm to improve its convergence properties.

The final algorithm we discuss, and the main subject of this paper, poses the apparent horizon equation \( H(h) = 0 \) as a nonlinear elliptic (boundary-value) PDE for \( h \) on the angular-coordinate space \( \mathcal{A} \). Finite differencing this PDE on an angular-coordinate grid \{\((\theta_K, \phi_K)\)\} gives a set of simultaneous nonlinear algebraic equations for the unknown values \( \{h(\theta_K, \phi_K)\} \), which may be solved by Newton’s method or a modification thereof. This “Newton’s-method” algorithm (we continue to use this term even when a modification of Newton’s method is actually used) has been used in axisymmetric spacetimes by a number of authors, for example Refs. [28,22,23,54], and is also applicable in fully general spacetimes when the coordinates have a (locally) polar spherical topology. Huq (Ref. [35]) has extended this algorithm to fully general spacetimes with Cartesian-topology coordinates and finite differencing, and much of our discussion remains applicable to his extension.

The Newton’s-method algorithm has three main parts: the computation of the discrete horizon function \( H(h) \), the computation of the discrete horizon function’s Jacobian matrix \( J[H(h)] \), and the solution of the simultaneous nonlinear (algebraic) \( H(h) = 0 \) equations. We now discuss these in more detail.

V. COMPUTING THE HORIZON FUNCTION

In this section we discuss the details of the computation of the discrete horizon function \( H(h) \). More precisely, first fix an angular-coordinate grid \{\((\theta_K, \phi_K)\)\}. Then, given a “trial horizon surface” \( r = h(\theta, \phi) \), which need not actually be an apparent horizon, we define \( h = h(\theta, \phi) \) to be the discretization of \( h = h(\theta, \phi) \) to this angular-coordinate grid, and we discuss how \( H \equiv H(h) \) may be computed on the discretized trial horizon surface, i.e. at the points \{\((r = h(\theta_K, \phi_K), \theta = \theta_K, \phi = \phi_K)\)\}.

7
The apparent horizon equation Eq. (1) defines \( H \equiv (2)H \) in terms of the field variables and their spatial derivatives on the trial horizon surface. However, these are typically known only at the grid points. We therefore extend \((2)H\) to some (3-dimensional) neighborhood \( \mathcal{N} \) of the trial horizon surface, i.e. we define an extended horizon function \((3)H\) on \( \mathcal{N} \),

\[
(3)H = \nabla n^i + K_{ij}n^in^j - K
\]

\[
= \partial_i n^i + (\partial_i \ln \sqrt{g})n^i + K_{ij}n^in^j - K \quad (7)
\]

To compute \((2)H(h)\) on the (discretized) trial horizon surface, we first compute \((3)H(h)\) on a (3-dimensional) grid in \( \mathcal{N} \), then radially interpolate these \((3)H\) values to the trial-horizon-surface position to obtain \((2)H(h)\),

\[
(2)H(\theta, \phi) = \text{interp}\left((3)H(r, \theta, \phi), r = h(\theta, \phi)\right), \quad (9a)
\]

or equivalently

\[
(2)H_\theta = \text{interp}\left((3)H(r_\theta), r = h_\theta\right), \quad (9b)
\]

where \(i\) is an angular grid-point index and the \((r_\theta)\) subscript denotes that the interpolation is done (independently at each angular coordinate) along the radial coordinate line \( \{ \theta = \theta_\theta, \phi = \phi_\theta \} \).

So long as the grid in \( \mathcal{N} \) where \((3)H\) is computed surrounds the trial-horizon-surface position roughly symmetrically, the requirements on this grid’s spacing for the interpolation to be accurate are basically the same as those for the radial finite differencing to be accurate, so in practice any reasonable interpolation method should be acceptable. Ref. [23] reported satisfactory results using a spline interpolant; in this work we use a Lagrange (polynomial) interpolant centered on the trial-horizon-surface position, also with satisfactory results.

We consider two basic methods for computing the extended horizon function \((3)H(h)\) on the horizon-neighborhood domain \( \mathcal{N} \). One possibility is to first compute \(s_i(h)\) by finite differencing \(h\) in Eq. (3), then compute \(n^i\) via Eq. (5), then compute \((3)H(n^i)\) by finite differencing \(n^i\) in Eq. (8). We refer to this as a “2-stage” computation scheme for computing \((3)H(h)\) and \(H(h)\), because it uses two separate (sequential) finite differencing stages. Figure 1 illustrates this.

(As shown in the figure, there are actually three slightly different variants of the basic 2-stage computation scheme: the one just described, using both \(s_i\) and \(n^i\) explicitly as intermediate variables; one using only \(s_i\) explicitly as an intermediate variable, where we finite difference Eqs. (3) and Eqs. (13) and (14) (below); and one using only \(n^i\) explicitly as an intermediate variable, where we finite difference Eqs. (6) and (8). In practice these variants all behave very similarly, and we usually don’t distinguish between them.)

Alternatively, substituting Eqs. (5) and (3) into Eq. (8), we can write \((3)H\) directly in terms of \(h\) and its angular (1st and 2nd) derivatives,

\[
(3)H = \nabla n^i + K_{ij}n^in^j - K
\]

\[
= \partial_i n^i + (\partial_i \ln \sqrt{g})n^i + K_{ij}n^in^j - K \quad (10)
\]

\[
= \partial_i \left( g^{ij} s_j \right) + (\partial_i \ln \sqrt{g})\left( g^{ij} s_j \right) + K_{ij} s_i s_j - K \quad (11)
\]

\[
= \partial_i \left( g^{kl} s_k s_l \right)^{1/2} + (\partial_i \ln \sqrt{g})\left( g^{kl} s_k s_l \right)^{1/2} + K_{ij} s_i s_j - K \quad (12)
\]

\[
= A N^{3/2} + B N^{1/2} + C N - K, \quad (13)
\]
where the subexpressions $A$, $B$, $C$, and $N$ are given by

\begin{align}
A &= -\left( g^{ik} s_k \right) \left( g^{i} s_j \right) \partial_i s_j - \frac{1}{2} \left( g^{ij} s_j \right) \left[ \left( \partial g^{kl} s_k s_l \right) \right] \\
B &= \left( \partial g^{ij} s_j \right) + g^{ij} \partial_i s_j + \left( \partial_i \ln \sqrt{g} \right) \left( g^{ij} s_j \right) \\
C &= K^{ij} s_j \\
N &= g^{ij} s_j,
\end{align}

i.e.

\begin{align}
A &= \left( g^{uv} - g^{uw} \partial_u h \right) \left( g^{wr} - g^{uw} \partial_w h \right) \partial_u h \\
&\quad - \frac{1}{2} \left( g^{ir} - g^{iu} \partial_u h \right) \left[ \partial_i g^{rr} - 2 \left( \partial_i g^{ru} \right) \partial_u h + \left( \partial g^{uv} \right) \left( \partial_u h \right) \left( \partial_v h \right) \right] \\
B &= \left[ \partial g^{ir} - \left( \partial_i g^{ru} \right) \right] \partial_u h \\
&\quad - g^{uv} \partial_u h + \left( \partial_i \ln \sqrt{g} \right) \left( g^{ir} - g^{ir} \partial_u h \right) \\
C &= K^{rr} - 2K^{ru} \partial_u h + K^{uv} \left( \partial_u h \right) \left( \partial_v h \right) \\
N &= g^{rr} - 2g^{ru} \partial_u h + g^{uv} \left( \partial_u h \right) \left( \partial_v h \right).
\end{align}

Using Eqs. (13) and (15), we can compute $(3)^{\text{c}} H(h)$ by directly finite differencing $h$. We refer to this as a “1-stage” evaluation scheme for $(3)^{\text{c}} H(h)$ and $H(h)$, because it uses only a single finite differencing stage. Figure 1 also illustrates this scheme.

Comparing these schemes, the equations for the 2-stage scheme (Eqs. (3), (5), and (8)) for the variant using both $s_i$ and $n_i$, Eqs. (3), (13), and (14) for the variant using only $s_i$, and Eqs. (6) and (8) for the variant using only $n_i$) are somewhat simpler than those for the 1-stage scheme (Eqs. (13) and (15)). This makes the 2-stage scheme somewhat easier to implement, and the smaller number of operations performed makes it somewhat cheaper to compute.

However, for a proper comparison the cost of computing the horizon function must be considered in conjunction with the cost of computing the horizon function’s Jacobian. Compared to the 1-stage scheme, the 2-stage scheme doubles the effective radius of the net $H(h)$ finite differencing molecules, so the 2-stage Jacobian matrix has 2 (4) times as many nonzero off-diagonal elements as the 1-stage one for an axisymmetric (fully general) spacetime. In practice the cost of computing these extra Jacobian elements for the 2-stage scheme more than outweighs the slight cost saving in evaluating the horizon function. We discuss the relative costs of the different schemes further in section VI C.

VI. COMPUTING THE JACOBIAN

In this section we discuss the details of the computation of the Jacobian matrix $J[H(h)]$ of the horizon function $H(h)$. Analogously to our discussion in the previous section, we assume that a trial horizon surface has been given, which need not actually be an apparent horizon.

A. Computing the Jacobian of a Generic Function $P(Q)$

We consider first the case of a generic function $P(Q)$ in $d$ dimensions, finite differenced using $M$-point molecules. We define the Jacobian of the discrete $P(Q)$ function by
\[
\mathbf{J}[\mathbf{P}(\mathbf{Q})]_{IJ} = \frac{\partial \mathbf{P}_I}{\partial \mathbf{Q}_J},
\]  

(16)

or equivalently by the requirement that

\[
\delta \mathbf{P}_I \equiv \left[ \mathbf{P}(\mathbf{Q}) + \delta \mathbf{Q} - \mathbf{P}(\mathbf{Q}) \right]_I = \mathbf{J}[\mathbf{P}(\mathbf{Q})]_{IJ} \cdot \delta \mathbf{Q}_J 
\]

(17)

for any infinitesimal perturbation \( \delta \mathbf{Q} \) of \( \mathbf{Q} \).

We assume that \( \mathbf{P} \) is actually a local grid function of \( \mathbf{Q} \), so the Jacobian matrix is sparse. (For example, this would preclude the nonlocal 4th order “compact differencing” schemes described by Refs. [21,34].) We assume that by exploiting the locality of the discrete \( \mathbf{P}(\mathbf{Q}) \) function, any single \( \mathbf{P}_I \) can be computed in \( O(1) \) time, independent of the grid size.

1. Computing Jacobians by Numerical Perturbation

We consider two general methods for computing the Jacobian matrix \( \mathbf{J}[\mathbf{P}(\mathbf{Q})] \). The first of these is the “numerical perturbation” method. This involves numerically perturbing \( \mathbf{Q} \) and examining the resulting perturbation in \( \mathbf{P}(\mathbf{Q}) \),

\[
\mathbf{J}[\mathbf{P}(\mathbf{Q})]_{IJ} \approx \left[ \frac{\mathbf{P}(\mathbf{Q} + \mu \mathbf{e}^{(J)}) - \mathbf{P}(\mathbf{Q})}{\mu} \right]_I,
\]

(18)

where \( \mathbf{e}^{(J)} \) is a Kronecker-delta vector defined by

\[
[\mathbf{e}^{(J)}]_I = \begin{cases} 1 & \text{if } I = J \\ 0 & \text{otherwise} \end{cases},
\]

and \( \mu \) is a “small” perturbation amplitude. This computation of the Jacobian proceeds by columns: for each \( J \), \( \mathbf{Q}_J \) is perturbed, and the resulting perturbation in \( \mathbf{P}(\mathbf{Q}) \) gives the \( J \)-th column of the Jacobian matrix.

The perturbation amplitude \( \mu \) should be chosen to balance the truncation error of the one-sided finite difference approximation Eq. (18) against the numerical loss of significance caused by subtracting the nearly equal quantities \( \mathbf{P}(\mathbf{Q} + \mu \mathbf{e}^{(J)}) \) and \( \mathbf{P}(\mathbf{Q}) \). Refs. [51,24] discuss the choice of \( \mu \), and conclude that if \( \mathbf{P}(\mathbf{Q}) \) can be evaluated with an accuracy of \( \varepsilon \), then \( \mu \approx \sqrt{\varepsilon} \) “seems to work the best.” In practice the choice of \( \mu \) isn’t very critical for horizon finding. Values of \( 10^{-4} \) to \( 10^{-6} \) seem to work well, and the inaccuracies in the Jacobian matrix resulting from these values of \( \mu \) don’t seem to be a significant problem.

This method of computing Jacobians requires no knowledge of the \( \mathbf{P}(\mathbf{Q}) \) function’s internal structure. In particular, the \( \mathbf{P}(\mathbf{Q}) \) function may involve arbitrary nonlinear (local) computations, including multiple sequential stages of finite differencing and/or interpolation. This method is thus directly applicable to the \( (2)^{\text{nd}} \mathbf{H}(\mathbf{h}) \) computation.

Computing \( \mathbf{J}[\mathbf{P}(\mathbf{Q})] \) by numerical perturbation requires a total of \( M^d + 1 \) \( \mathbf{P}_I \) evaluations at each grid point: 1 for the initial unperturbed \( \mathbf{P} \), and \( M^d \) perturbed-\( \mathbf{P} \) reevaluations.
An alternate method of computing the Jacobian matrix $J[P(Q)]$ is by "symbolic differentiation". This method makes explicit use of the finite differencing scheme used to compute the discrete $P(Q)$ function.

We consider first the case where the continuum $P(Q)$ function is a position-dependent (local) linear differential operator, discretely approximated by a position-dependent (local) finite difference molecule $M$,

$$P_I = \sum_{M \in M(I)} M(i) M_Q I_M^i.$$

(20)

Differentiating this, we have

$$J[P(Q)]_{ij} \equiv \frac{\partial P_I}{\partial Q_j} = \begin{cases} M(i)_{j-I} & \text{if } j-I \in M(i) \\ 0 & \text{otherwise} \end{cases},$$

(21)

so that the molecule coefficients at each grid point give the corresponding row of the Jacobian matrix.

We now consider the more general case where the continuum $P(Q)$ function is a position-dependent (local) nonlinear algebraic function of $P$ and some finite number of $P$'s derivatives, say

$$P = P(Q, \partial_i Q, \partial_{ij} Q).$$

(22)

Logically, the Jacobian matrix $J[P(Q)]$ is defined (by Eq. (17)) in terms of the linearization of the discrete (finite differenced) $P(Q)$ function. However, if the discretization (the finite differencing scheme) commutes with the linearization, then as illustrated in figure 2, we can instead compute the Jacobian by first linearizing the continuum $P(Q)$ function, then finite differencing this (continuum) linearized function. (This method of computing the Jacobian is essentially just the "Jacobian part" of the Newton-Kantorovich algorithm for solving nonlinear elliptic PDEs. We discuss this in detail in section VIII B.)

In detail, we first linearize the (continuum) differential operator $P(Q)$, obtaining

$$\delta P = \frac{\partial P}{\partial Q} \delta Q + \frac{\partial P}{\partial(\partial_i Q)} \delta \partial_i Q + \frac{\partial P}{\partial(\partial_{ij} Q)} \delta \partial_{ij} Q$$

(23a)

$$= \frac{\partial P}{\partial Q} \delta Q + \frac{\partial P}{\partial(\partial_i Q)} \delta \partial_i Q + \frac{\partial P}{\partial(\partial_{ij} Q)} \delta \partial_{ij} Q$$

(23b)

for any infinitesimal perturbation $\delta Q$. We then discretely approximate the (continuum) linear differential operator $\delta P(\delta Q)$ by the position-dependent finite difference molecule

$$M = \frac{\partial P}{\partial Q} I + \frac{\partial P}{\partial(\partial_i Q)} d_i + \frac{\partial P}{\partial(\partial_{ij} Q)} d_{ij},$$

(24)

where $I$ is the identity molecule and $d_i$ and $d_{ij}$ are finite difference molecules discretely approximating $\partial_i$ and $\partial_{ij}$ respectively.
Finally, we apply Eq. (21) to the molecule $M$ defined by Eq. (24), to obtain the Jacobian matrix $J[P(Q)]$. In practice, there’s no need to explicitly form the molecule $M$ – the Jacobian matrix elements can easily be assembled directly from the “Jacobian coefficients” $\partial P/\partial Q$, $\partial P/\partial (\partial_i Q)$, and $\partial P/\partial (\partial_{ij} Q)$, and the known $I$, $d_i$, and $d_{ij}$ molecule coefficients.

Once the Jacobian coefficients are known, the assembly of the actual Jacobian matrix elements is very cheap, requiring only a few arithmetic operations per matrix element to evaluate Eqs. (24) and (21). The main cost of computing a Jacobian matrix by this scheme is therefore the computation of the Jacobian coefficients. Depending on how many $Q$ derivatives $P$ depends on, there may be anywhere from 1 to 10 coefficients, although in practice these often have many common subexpressions. We discuss the cost of symbolic differentiation Jacobian computations for horizon finding in detail in section VI C.

B. Computing the Jacobian of the Horizon Function $H(h)$

We now return to the computation of the Jacobian (matrix) $J[H(h)] \equiv J^{(2)}[H(h)]$. We define this Jacobian by

$$J^{(2)}[H(h)]_{IJ} = \frac{\partial^{(2)} H_I}{\partial h_J},$$

(25)

(where $I$ and $J$ are angular grid-point indices), or equivalently by the requirement that

$$\delta^{(2)} H_I \equiv \left[ (^{(2)} H(h + \delta h) - (^{(2)} H(h)) \right]_I = J^{(2)}[H(h)]_{IJ} \cdot \delta h_J$$

(26)

for any infinitesimal perturbation $\delta h$.

It’s also useful to consider the Jacobian $J^{(3)}[H(h)]$ of the extended horizon function $^{(3)} H(h)$, which we define analogously by

$$J^{(3)}[H(h)]_{IJ} = \frac{\partial^{(3)} H_I}{\partial h_J},$$

(27)

or equivalently by the requirement that

$$\delta^{(3)} H_I \equiv \left[ (^{(3)} H(h + \delta h) - (^{(3)} H(h)) \right]_I = J^{(3)}[H(h)]_{IJ} \cdot \delta h_J$$

(28)

for any infinitesimal perturbation $\delta h$. Here $I$ is a 3-dimensional grid-point index for $^{(3)} H$, while $J$ is an (angular) 2-dimensional grid-point index for $h$.

In addition to the obvious difference in dimensionality, there’s a subtle but very important difference in the semantics of $J^{(2)}[H(h)]$ and $J^{(3)}[H(h)]$: We define $^{(2)} H(h)$ as being evaluated at the position $r = h(\theta, \phi)$ of the trial horizon surface, so $J^{(2)}[H(h)]$ must take into account not only the direct change in $^{(2)} H$ at a fixed position due to a perturbation in $h$, but also...
the implicit change in \( (2)^{\text{H}} \) caused by the field-variable coefficients in \( (2)^{\text{H}} \) being evaluated at the perturbed position \( r = h(\theta, \phi) \). In contrast, we define \( (3)^{\text{H}}(h) \) to be evaluated at a fixed position (a grid point in the neighborhood \( \mathcal{N} \) of the trial horizon surface), which need not lie on the trial horizon surface, and whose position doesn’t change with perturbations in \( h \). Therefore \( J \left[ (3)^{\text{H}}(h) \right] \) need only take into account the direct change in \( (3)^{\text{H}} \) at a fixed position due to a perturbation in \( h \).

\( J \left[ (3)^{\text{H}}(h) \right] \) thus has much simpler semantics than \( J \left[ (2)^{\text{H}}(h) \right] \). Independently of its use in our computational schemes (below), we find \( J \left[ (3)^{\text{H}}(h) \right] \) to be very useful in thinking about the meanings of the various Jacobians. (In fact, before we realised the semantic distinction between \( J \left[ (3)^{\text{H}}(h) \right] \) and \( J \left[ (2)^{\text{H}}(h) \right] \) we had great difficulty in determining any clear semantics for the “amalgamated” Jacobian \( J \left[ (h) \right] \). In hindsight this was to be expected, since two quite distinct semantics were being blurred together.)

Corresponding to this difference in semantics, we can compute \( J \left[ (2)^{\text{H}}(h) \right] \) either by working directly with \( (2)^{\text{H}}(h) \) in angular-coordinate space (we refer to this as a “2-dimensional” computation), or indirectly, computing \( J \left[ (3)^{\text{H}}(h) \right] \) first as an intermediate step (we refer to this as a “3-dimensional” computation). We also categorize methods for computing \( J \left[ (2)^{\text{H}}(h) \right] \) by whether they use numerical perturbation or symbolic differentiation, and by whether they use a 2-stage or a 1-stage scheme to compute the horizon function. As we describe each Jacobian computation method, we tag it with a shorthand “code” based on these attributes, which we use later to refer to the method. Table I (discussed in section VI C) summarizes all the Jacobian computation methods.

The simplest way to compute \( J \left[ (2)^{\text{H}}(h) \right] \) is by direct numerical perturbation in angular-coordinate space, using either a 1-stage or a 2-stage scheme to compute \( (2)^{\text{H}}(h) \). We refer to the resulting Jacobian computation methods as the “2d.np.1s” and “2d.np.2s” methods respectively.

Symbolic differentiation isn’t directly applicable to 2-dimensional Jacobian computations, both because of the position-dependent coefficient problem just described, and because \( (2)^{\text{H}}(h) \) isn’t given by a simple molecule convolution of the form Eq. (20). We therefore consider 3-dimensional Jacobian computations.

If \( (3)^{\text{H}}(h) \) is computed using a 1-stage scheme, either numerical perturbation or symbolic differentiation may be used to compute \( J \left[ (3)^{\text{H}}(h) \right] \). We refer to these as the “3d.np.1s” and “3d.sd.1s” schemes respectively. For the latter, a straightforward but somewhat tedious calculation gives the (only) nonzero Jacobian coefficients of \( (3)^{\text{H}}(h) \) as

\[
\frac{\partial (3)^{\text{H}}}{\partial (\partial x h)} = \frac{1}{N^{3/2}} \frac{\partial A}{\partial (\partial x h)} + \frac{1}{N^{1/2}} \frac{\partial B}{\partial (\partial x h)} + \frac{1}{N} \frac{\partial C}{\partial (\partial x h)} - \left( \frac{3}{2} \frac{A}{N^{5/2}} + \frac{1}{2} \frac{B}{N^{3/2}} + \frac{C}{N^2} \right) \frac{\partial N}{\partial (\partial x h)} \tag{29a}
\]

\[
\frac{\partial (3)^{\text{H}}}{\partial (\partial y h)} = \frac{1}{N^{3/2}} \frac{\partial A}{\partial (\partial y h)} + \frac{1}{N^{1/2}} \frac{\partial B}{\partial (\partial y h)} \tag{29b}
\]

where the subexpressions \( A, B, C, \) and \( N \) are given by Eq. (15), and their (only) nonzero Jacobian coefficients are given by
\[
\frac{\partial A}{\partial (\partial_x h)} = - [g^{ux} (g^{vr} - g^{uv} \partial_w h) + g^{uv} (g^{wr} - g^{wu} \partial_v h)] \partial_{uw} h
\]
\[
+ \frac{1}{2} g^{uv} \left[ \partial_i g^{vr} - 2 (\partial_i g^{uv}) \partial_u h + (\partial_i g^{uv}) (\partial_v h) (\partial_w h) \right] 
\]
\[
+ (g^{ir} - g^{iu} \partial_u h) \left[ \partial_i g^{vr} - (\partial_i g^{uv}) \partial_v h \right]
\]  
(30a)

\[
\frac{\partial B}{\partial (\partial_x h)} = - \partial_i g^{iw} - (\partial_i \ln \sqrt{g}) g^{iw}
\]  
(30b)

\[
\frac{\partial C}{\partial (\partial_x h)} = - 2 (K^{xw} - K^{zw} \partial_u h)
\]  
(30c)

\[
\frac{\partial N}{\partial (\partial_x h)} = - 2 (g^{wr} - g^{wu} \partial_u h)
\]  
(30d)

\[
\frac{\partial A}{\partial (\partial_x u) h} = (g^{wr} - g^{wu} \partial_u h)(g^{vr} - g^{vu} \partial_v h)
\]  
(30e)

\[
\frac{\partial B}{\partial (\partial_x u) h} = - g^{xy}.
\]  
(30f)

Alternatively, if \( (3)^{\text{H}}(h) \) is computed using a 2-stage scheme, then \( \mathbf{J}[^{(3)}\text{H}(h)] \) may be computed either by the simple numerical perturbation of \( (3)^{\text{H}}(h) \) (the “3d.np.2s” scheme), or by separately computing the Jacobians of the individual stages and matrix-multiplying them together,

\[
\mathbf{J}[^{(3)}\text{H}(h)]_{JK} = \mathbf{J}[s_i(h)]_{IJ} \cdot \mathbf{J}[^{(3)}\text{H}(s_i)]_{JK} 
\]
\[
= \mathbf{J}[s_J(h)]_{IJ} \cdot \mathbf{J}[n^i(s_j)]_{JJ} \cdot \mathbf{J}[^{(3)}\text{H}(n^i)]_{JK} 
\]
\[
= \mathbf{J}[n^i(h)]_{IJ} \cdot \mathbf{J}[^{(3)}\text{H}(n^i)]_{JK},
\]  
(31a)

where the three equations correspond to the three variant 2-stage computation schemes for the horizon function (illustrated in figure 1), and where \( i, j, \) and \( k \) are 3-dimensional grid-point indices. (Notice that \( \mathbf{J}[n^i(s_j)] \) is diagonal in the grid-point indices.) Either numerical perturbation or symbolic differentiation may be used to compute the individual-stage Jacobians, giving the “3d.np.2s” and “3d.sd.2s” schemes respectively. For the latter, an easy series of calculations give the (only) nonzero Jacobian coefficients as follows:

For \( s_i(h) \),

\[
\frac{\partial s_i}{\partial (\partial_x h)} = \begin{cases} 
1 & \text{if } i = x \\
0 & \text{otherwise}
\end{cases}
\]  
(32)

For \( n^i(h) \),

\[
\frac{\partial n^i}{\partial (\partial_x h)} = - \frac{g^{ix}}{N^{1/2}} + \frac{(g^{ir} - g^{iu} \partial_u h)(g^{xr} - g^{xr} \partial_v h)}{N^{3/2}}.
\]  
(33)

For \( n^i(s_j) \),

\[
\frac{\partial n^i}{\partial s_j} = \frac{g^{ij}}{(g^{kl}s_k s_i)^{1/2}} - \frac{(g^{ik}s_k)(g^{jl}s_l)}{(g^{kl}s_k s_l)^{3/2}}.
\]  
(34)
For $^{(3)}H(s_i)$,

$$\frac{\partial^{(3)}H}{\partial s_x} = \frac{1}{N^{3/2}} \frac{\partial A}{\partial s_x} + \frac{1}{N^{1/2}} \frac{\partial B}{\partial s_x} + \frac{1}{N} \frac{\partial C}{\partial s_x}$$
$$- \left( \frac{3}{2} \frac{A}{N^{3/2}} + \frac{1}{2} \frac{B}{N^{1/2}} + \frac{C}{N^2} \right) \frac{\partial N}{\partial s_x}$$ \hfill (35a)

$$\frac{\partial^{(3)}H}{\partial(\partial x s_y)} = \frac{1}{N^{3/2}} \frac{\partial A}{\partial(\partial x s_y)} + \frac{1}{N^{1/2}} \frac{\partial B}{\partial(\partial x s_y)},$$ \hfill (35b)

where the subexpressions $A$, $B$, $C$, and $N$ are again given by Eqs. (14) or (15) and their (only) nonzero Jacobian coefficients are given by

$$\frac{\partial A}{\partial s_x} = - \left[ g^{ik} (g^j s_k) + g^{ik} (g^j s_k) \right] \delta_i s_j$$
$$- \frac{1}{2} g^{ik} \left[ (\partial_i g^{kl}) s_k s_l - (\partial_i g^{lk}) s_k s_l \right]$$ \hfill (36a)

$$\frac{\partial B}{\partial s_x} = - \delta_i g^{ik} + (\partial_i \ln \sqrt{g}) g^{ik} \tag{36b}$$

$$\frac{\partial C}{\partial s_x} = 2 K^{i x} s_i \tag{36c}$$

$$\frac{\partial N}{\partial s_x} = 2 g^{i x} s_i; \tag{36d}$$

$$\frac{\partial A}{\partial(\partial x s_y)} = -(g^{ik} s_k)(g^{i l} s_l) \tag{36e}$$

$$\frac{\partial B}{\partial(\partial x s_y)} = g^{i y} \tag{36f}.$$

For $^{(3)}H(n^i)$,

$$\frac{\partial^{(3)}H}{\partial n^x} = \partial x \ln \sqrt{g} + 2 K^{x i} n^i \tag{37a}$$

$$\frac{\partial^{(3)}H}{\partial(\partial x n^y)} = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}. \tag{37b}$$

Finally, we consider the computation of $J[H(h)] = J^{(2)}H(h)$ once $J^{(3)}H(h)$ is known. Since $H(\theta, \phi) = ^{(3)}H(r = h(\theta, \phi), \theta, \phi)$, we have

$$J[H(h)]_{ij} = \frac{\partial H_{ij}}{\partial h_{ij}}, \tag{38}$$

$$= \frac{\partial H(\theta_i, \phi_i)}{\partial h(\theta_i, \phi_i)} \tag{39}$$

$$= \frac{\partial^{(3)}H(r = h(\theta_i, \phi_i), \theta_i, \phi_i)}{\partial h(\theta_i, \phi_i)} \tag{40}$$

$$= \left. \frac{\partial^{(3)}H(r, \theta_i, \phi_i)}{\partial h(\theta_i, \phi_i)} \right|_{r = h(\theta_i, \phi_i)} + \left. \frac{\partial^{(3)}H(r, \theta_i, \phi_i)}{\partial r} \right|_{r = h(\theta_i, \phi_i)} \tag{41}$$

$$= \text{interp} \left( J^{(3)}H(h) \right)_{[r \mapsto 1], r = h(t)} + \text{interp}' \left( ^{(3)}H(r), r = h(t) \right), \tag{42}$$
where the \( \{ r_t \} \) subscripts in Eq. (42) denote that the interpolations are done along the radial line \( \{ \theta = \theta_t, \phi = \phi_t \} \), analogously to Eq. (9).

Notice that the \( \text{interp}'(\ldots) \) term in Eq. (42) may be computed very cheaply from the same \( (3)^{\text{H}} \) data values which were used to compute \( H \equiv (2)^{\text{H}} \), cf. Eq. (9). (The number of \( (3)^{\text{H}} \) data points used in the radial interpolation at each angular grid position will probably have to be increased by one to retain the same order of accuracy in the \( \text{interp}'(\ldots) \) term in Eq. (42) as in the \( \text{interp}(\ldots) \) term.) It's thus easy to compute \( J[H[h]] = J[(2)^{\text{H}}(h)] \) once \( H(h) \equiv (2)^{\text{H}}(h) \) and \( J[(3)^{\text{H}}(h)] \) are known.

C. Comparing the Horizon-Function and Jacobian Computation Methods

As described in appendix A, we have implemented all the horizon-function and Jacobian computation methods described in sections V and VI B, in the context of a larger 3 + 1 code. (For the 2-stage \( (3)^{\text{H}}(h) \) computation, our code uses only \( n^i \) as an explicit intermediate variable.) Our code assumes that spacetime is axisymmetric, and uses 4th order finite differencing (5-point molecules) on a (2-dimensional) nonuniform polar-coordinate grid.

Using this code, we have made a number of tests of the internal consistency of the different horizon-function and Jacobian computation methods. In particular, we have explicitly verified that:

- the 1-stage and 2-stage \( (3)^{\text{H}}(h) \) schemes yield identical results up to finite differencing truncation errors,
- the symbolic-differentiation Jacobian coefficients given by Eqs. (30) are in fact identical to the explicit derivatives (computed by our "PDE compiler" symbolic computation system, cf. appendix A) of the subexpressions \( A, B, C, \) and \( N \) as given by (15),
- the symbolic-differentiation Jacobian coefficients given by Eqs. (33) and (37) are similarly identical to the derivatives of \( n^i(h) \) and \( (3)^{\text{H}}(n^i) \) as given by Eqs. (6) and (8) respectively, and
- the symbolic-differentiation and numerical-perturbation Jacobian matrices for each of \( (3)^{\text{H}}(h) \), \( n^i(h) \), and \( (3)^{\text{H}}(n^i) \) are identical up to the expected truncation error of the numerical-perturbation Jacobian approximation Eq. (18).

Table I summarizes all these different methods for computing the horizon function and its Jacobian. As well as their basic properties, the table also shows the methods’ measured relative CPU times in our implementation and our estimates of the relative effort needed to implement them. The CPU times are per angular grid point, and are normalized relative to the 1-stage \( (2)^{\text{H}}(h) \) computation.

As can be seen from the table, in our code we find that the 3d.sd.1s method is by far the most efficient of the Jacobian computation methods, being about a factor of 5 faster than any of the others. Notice, in particular, that the computation of the Jacobian by this method is only 1.5–2 times more expensive than the simple evaluation of the horizon function.

The relative performance of the different methods will of course vary considerably from one implementation to another. However, we expect most implementations designed for
axisymmetric spacetimes (i.e. using 2-dimensional grids) to have relative timings broadly similar to ours, with the numerical perturbation methods being a factor of \( \sim 1.5 \) more efficient relative to the symbolic differentiation methods for codes using 2nd order finite differencing (3-point molecules), due to the smaller number of numerical-perturbation function reevaluations needed with the smaller molecules. For the same reason, we expect codes assuming fully general spacetimes (i.e. using 3-dimensional grids) to have the numerical perturbation methods be a factor of \( \sim 3-5 \) less efficient relative to the symbolic differentiation methods. We thus expect the 3d.sd.1s method to remain much more efficient than any of the others in all "reasonable" implementations.

As well as the measured CPU times, table I also shows our estimates of the approximate implementation effort needed for each of the methods. For computing the horizon function, we find the 2-stage scheme to be somewhat easier to implement than the 1-stage scheme, simply because the equations are somewhat simpler. However, the difference in implementation effort between the horizon-function schemes is considerably smaller than those between the Jacobian computation methods.

Broadly speaking, we find that the implementation effort for computing the Jacobian depends more on which Jacobian matrices are involved than on how the Jacobians are computed: The (2-dimensional) methods involving only \( J^{(2)}H(h) \) are the easiest to implement (these appear to have been used in many previous implementations of the Newton’s-method horizon-finding algorithm), the (3-dimensional) methods involving \( J^{(3)}H(h) \) are still fairly easy to implement but somewhat less so than those involving only \( J^{(2)}H(h) \), and the (3-dimensional) methods involving the individual-stage Jacobians \( J[s_i(h)], J[n^i(h)], J^{(3)}H(s_i) \), and/or \( J^{(3)}H(n^i) \) are moderately difficult to implement due to their more complicated sparsity patterns.

Comparing the numerical-perturbation and symbolic-differentiation Jacobian computation methods, our main conclusion is that the symbolic differentiation methods are much easier to implement than we had expected or than seems to be commonly believed. We had previously suggested (Ref. [54]) that symbolic-differentiation Jacobian computations would necessarily require substantial support from a (computer) symbolic computation system, and several colleagues have expressed similar opinions to us. We find that this is not the case. Using the Jacobian-coefficient scheme described in sections VI A 2 and VI B, only the continuum equations need be differentiated, and this is easily done by hand. (In fact, we derived all the Jacobian coefficients given in section VI B by hand, using only a few pages of algebra.)

As discussed in section VIII A, the rapid "quadratic" convergence of a Newton’s-method horizon finder provides a strong verification of the overall mutual consistency of \( H(h) \) and \( J[H(h)] \). However, for the detailed debugging of symbolic-differentiation Jacobian computations, we find by far the most useful technique to be an explicit comparison of the Jacobian matrix elements with those of a corresponding numerical-perturbation Jacobian computation. By localizing any discrepancies (beyond the expected truncation error of the numerical-perturbation Jacobian approximation Eq. (18)) to specific Jacobian matrix elements, we find that such a comparison makes it relatively easy to locate and resolve programming errors in Jacobian computations.

Summarizing our comparisons, we find that the best Jacobian computation method is
clearly the 3d.sd.1s one. It's much more efficient than any of the other methods, and still quite easy to implement.

VII. CONVERGENCE TESTS

As has been forcefully emphasized by Choptuik (Refs. [18–20]), a careful comparison of a finite differencing code's numerical results at different grid resolutions can yield very stringent tests of the code's numerical performance and correctness. In particular, such “convergence tests” can yield reliable numerical estimates of a code’s external errors, i.e. of the deviation of the code’s results from those that would be obtained by exactly solving the continuum equations. With such estimates available, we can then securely draw inferences about solutions of the continuum equations from the code’s (finite-resolution) numerical results.

To apply this technique in the horizon-finding context, suppose first we have a slice where the (a) true (continuum) apparent horizon position \( h^* \) is known. For a convergence test in such a slice we run the horizon finder twice on this slice, using grids with resolutions in a 1:2 ratio. (It's essential here that the resolutions differ, but the particular choice of a 1:2 ratio isn’t essential, although its convenience makes it the usual choice.) When constructing various-resolution grids, we always maintain the same grid aspect ratio (relative grid resolution in the different coordinate dimensions) as we vary the resolution, so any given grid’s resolution can be parameterized by a single scale \( \Delta x \).

As discussed in detail by Ref. [19], if the code’s numerical errors are dominated by truncation errors from \( n \)th order finite differencing, the numerically computed horizon position \( h \) will satisfy

\[
\begin{align*}
    h(\Delta x) &= h^* + (\Delta x)^n f + O((\Delta x)^{n+2}) \\
    h(\Delta x/2) &= h^* + (\Delta x/2)^n f + O((\Delta x)^{n+2})
\end{align*}
\]

(43a)

(43b)

at each grid point, where (changing notation somewhat from previous sections of this paper) \( h(\Delta x) \) and \( h(\Delta x/2) \) denote the numerically computed horizon positions using grid resolutions of \( \Delta x \) and \( \Delta x/2 \) respectively, and where \( f \) is an (in theory computable but in practice unknown) \( O(1) \) smooth function of position and various high order derivatives of \( h^* \) and the field variables, but not of the grid resolution. (We’re assuming centered differencing here in writing the higher order terms as \( O((\Delta x)^{n+2}) \), otherwise they would only be \( O((\Delta x)^{n+1}) \).)

Because \( f \) in Eq. (43) is independent of the grid resolution, and hence is the same in Eqs. (43a) and (43b), if we neglect the higher order error terms we can eliminate \( f \) to obtain a direct relationship between the code’s errors at the two resolutions,

\[
\frac{h(\Delta x/2) - h^*}{h(\Delta x) - h^*} = \frac{1}{2^n},
\]

(44)

which must hold at each grid point common to the two grids.

Because we neglected the higher order error terms in Eq. (43) in deriving the convergence criterion Eq. (44), we only expect this result to hold in the limit of small \( \Delta x \). To test how well any particular set of numerical results satisfies this convergence criterion, we plot
a scatterplot of the high-resolution errors \( h(\Delta x/2) - h^* \) against the low-resolution errors \( h(\Delta x) - h^* \) at the grid points common to the two grids. If (and given the arguments of Ref. [19], in practice only if) the error expansions Eq. (43) are valid with the higher order error terms negligible, i.e. if (and in practice only if) the errors are indeed dominated by the expected \( n \)th order finite difference truncation errors, then all the points in the scatterplot will fall on a line through the origin with slope \( 1/2^n \).

(We find that the pointwise nature of this type of scatterplot comparison makes it significantly more useful than a simple comparison of gridwise norms. In particular, a pointwise error comparison clearly shows convergence problems which may occur only in a small subset of the grid points (for example near a boundary), which would be “washed out” in a comparison of gridwise norms.)

Now suppose the true (continuum) apparent horizon position \( h^* \) is unknown. For a convergence test in this case we run the horizon finder three times on the slice, using grids with resolutions in a 1:2:4 ratio. Analogously to the 2-grid case, we now have

\[
\begin{align*}
    h(\Delta x) &= h^* + (\Delta x)^n f + O((\Delta x)^{n+2}) \\
    h(\Delta x/2) &= h^* + (\Delta x/2)^n f + O((\Delta x)^{n+2}) \\
    h(\Delta x/4) &= h^* + (\Delta x/4)^n f + O((\Delta x)^{n+2}),
\end{align*}
\]

at each grid point, with \( f \) again being independent of the grid resolution. Again neglecting the higher order terms, we now eliminate both \( f \) and \( h^* \) to obtain the “3-grid” convergence criterion

\[
    \frac{h(\Delta x/2) - h(\Delta x/4)}{h(\Delta x) - h(\Delta x/2)} = \frac{1}{2^n}
\]

which must hold at each grid point common to the three grids. We test this criterion using an analogous scatterplot technique to that for the 2-grid criterion Eq. (44).

We emphasize that for a 3-grid convergence test of this type, the true continuum solution \( h^* \) need not be known. As well, nothing in the derivation actually requires \( h^* \) to be the true continuum horizon position – it need only be the true continuum solution to some continuum equation such that the truncation error formulas Eq. (45) hold. We make use of this latter case in sections VIII B and IX B to apply 3-grid convergence tests to intermediate Newton-Kantorovich iterates (trial horizon surfaces) of our horizon finder.

For both the 2-grid and the 3-grid convergence tests it’s important to realise that the parameter \( n \), the order of the convergence, is known in advance from the form of the finite differencing scheme. Thus the slope-1/\( 2^n \) line with which the scatterplot points are compared isn’t fitted to the data points, but is rather an a priori prediction with no adjustable parameters. Requiring that all the data points fall on this line, is thus a very strong test of the validity of the finite differencing scheme and the error expansions Eq. (43) or Eq. (45).

Given the error expansions Eq. (45), then as well as eliminating \( f \), we can solve for \( h^* \) and for the actual code errors, for example

\[
\begin{align*}
    h(\Delta x) - h^* &= \frac{2^n}{2^n - 1} \left[ h(\Delta x) - h(\Delta x/2) \right] + O((\Delta x)^{n+2}) \\
    h^* &= h(\Delta x) - \frac{2^n}{2^n - 1} \left[ h(\Delta x) - h(\Delta x/2) \right] + O((\Delta x)^{n+2}).
\end{align*}
\]
If the higher order terms are negligible (which we can determine from a 3-grid convergence test), then we use the Richardson extrapolation formula Eq. (48) as an effectively $(n+2)$-th order differencing scheme. Such schemes are discussed in detail by Refs. [18–20]. However, in this work we follow the simpler procedure of only neglecting the $O(1)$ coefficient in Eq. (47) and taking $\|h(\Delta x) - h(\Delta x/2)\|$ as an approximate error estimate for $h(\Delta x)$ when a 3-grid convergence test shows the higher order terms to be negligible.

### VIII. SOLVING THE NONLINEAR ALGEBRAIC EQUATIONS

In this section we discuss the details of using Newton’s method or a modification thereof to solve the simultaneous nonlinear algebraic equations $H(h) = 0$.

#### A. Newton’s Method

The basic Newton’s-method algorithm is well known: At each iteration, we first linearize the discrete $H(h)$ function about the current approximate solution $h^{(k)}$,

$$H(h^{(k)} + \delta h) = H(h^{(k)}) + J[H(h^{(k)})] \cdot \delta h + O(\|\delta h\|^2),$$

(49)

where $\delta h$ now denotes a finite perturbation in $h$, and where $J[H(h^{(k)})]$ denotes the Jacobian matrix $J[H(h)]$ evaluated at the point $h = h^{(k)}$. We then neglect the higher order (nonlinear) terms and solve for the perturbation $\delta h^{(k)}$ such that $h^{(k)} + \delta h^{(k)}$ satisfies the (discrete) linearization of Eq. (49). This gives the simultaneous linear algebraic equations

$$J[H(h^{(k)})] \cdot \delta h^{(k)} = -H(h^{(k)})$$

(50)

to be solved for $\delta h^{(k)}$. Finally, we update the approximate solution via

$$h^{(k+1)} \leftarrow h^{(k)} + \delta h^{(k)},$$

(51)

and repeat the iteration until some convergence criterion is satisfied.

Notice that here we’re using the word “convergence” in a very different sense from that of section VII – here it refers to the “iteration-convergence” of the Newton iterates $h^{(k)}$ to the exact solution $h^*$ of the discrete equations, whereas there it refers to the “finite-difference-convergence” of the finite difference computation result $h(\Delta x)$ to the continuum limit $h^*$ as the grid resolution is increased.

Once the current solution estimate $h^{(k)}$ is reasonably close to the (an) exact solution of the discrete equations, $h^*$, i.e. in practice once the trial horizon surface is reasonably close to the (an) apparent horizon, Newton’s method converges extremely rapidly. (Here we’re assuming that the exact solution of the discrete equations is very close to the true continuum apparent horizon, which should be true for all reasonable finite differencing schemes.)

In particular, once the linear approximation in Eq. (49) is approximately valid, Newton’s method roughly squares the relative error $\|h - h^*\|/\|h^*\|$ at each iteration, and can thus bring the error down to a negligible value in only a few (more) iterations. (For a detailed discussion...
of Newton’s method, including precise formulations and proofs of these statements, see, for example, Ref. [50].) (Note also that because this rapid “quadratic” convergence depends critically on the validity (in the linear approximation) of Eq. (49), this convergence provides a strong verification of the mutual consistency of the $H(h)$ function and its Jacobian matrix $J[H(h)]$, which can be quite useful in program verification.)

However, if the initial guess $h^{(0)}$ for the horizon position, or more generally any iterate (trial horizon surface) $h^{(k)}$, is far enough from the (an) actual apparent horizon position $h^*$ so that the linear approximation in Eq. (49) isn’t approximately valid, then Newton’s method may converge poorly, or fail to converge at all. We discuss this case further in section IX.

B. The Newton-Kantorovich Method

We have described the Newton’s-method algorithm in terms of solving the discrete $H(h) = 0$ equations. However, the algorithm can also be interpreted in terms of solving the continuum $H(h) = 0$ equations. This continuum “Newton-Kantorovich” method, and its relationship to the discrete Newton’s method, are discussed in detail by Ref. [14].

For the Newton-Kantorovich algorithm, at each iteration, we first linearize the (continuum) differential operator $H(h)$ about the current (continuum) approximate solution $h^{(k)}$,

$$H(h^{(k)} + \delta h) = H(h^{(k)}) + J[H(h^{(k)})](\delta h) + O(\|\delta h\|^2),$$

where $\delta h$ is now a finite perturbation in $h$, and where $J[H(h^{(k)})]$ is now the (continuum) linearization of the differential operator $H(h)$ about the point $h = h^{(k)}$. We then neglect the higher order (nonlinear) terms and solve for the perturbation $\delta h^{(k)}$ such that $h^{(k)} + \delta h^{(k)}$ satisfies the (continuum) linearization of Eq. (52). This gives the linear differential equation

$$J[H(h^{(k)})](\delta h^{(k)}) = -H(h^{(k)})$$

(53)

to be solved for $\delta h^{(k)}$. Finally, we update the approximate solution via

$$h^{(k+1)} \leftarrow h^{(k)} + \delta h^{(k)},$$

(54)

and repeat the iteration until some convergence criterion is satisfied.

Notice that this is an entirely continuum algorithm. As such, its iteration-convergence or iteration-divergence is clearly determined (only) by the initial guess and the (continuum) nonlinearity of the $H(h)$ differential operator.

Now suppose we discretely approximate this continuum Newton-Kantorovich algorithm by finite differencing using a grid resolution $\Delta x$. If the discretization (finite differencing) and linearization commute in the manner discussed in section VI A 2, then the discretization of the Newton-Kantorovich algorithm is in fact identical to the (discrete) Newton’s-method algorithm applied to the $H(h) = 0$ equations obtained by finite differencing the (continuum) $H(h) = 0$ equation with a grid resolution $\Delta x$. (In a simpler context, our technique of computing symbolic-differentiation Jacobians by first computing the continuum Jacobian coefficients, then assembling the Jacobian from these and the partial-derivative molecule coefficients (cf. section VI A 2), essentially just exploits the “Jacobian part” of this identity.)
Therefore, when using the standard (discrete) Newton’s method to solve the \( H(h) = 0 \) equations, we can equivalently view each Newton iterate (trial horizon surface) \( h^{(k)}(\Delta x) \) as being a finite difference approximation to the corresponding (continuum) Newton-Kantorovich iterate (trial horizon surface) \( h^{(k)} \). If the finite differencing is properly convergent in the sense of section VII and Ref. [19], then as the grid resolution is increased, the discrete approximations \( h^{(k)}(\Delta x) \) must thus satisfy the 3-grid finite-difference-convergence error expansions Eq. (45). (We use the 3-grid criterion here rather than the 2-grid one since the (continuum) Newton-Kantorovich iterate \( h^{(k)} \) generally isn’t known.)

Notice that this finite-difference-convergence must hold regardless of the iteration-convergence or iteration-divergence of the discrete Newton (Newton-Kantorovich) iteration itself. Moreover, once we verify the finite-differencing-convergence with a 3-grid convergence test, we can safely extrapolate the iteration-convergence or iteration-divergence of this discrete iteration to that of the continuum Newton-Kantorovich algorithm applied to the (continuum) \( H(h) = 0 \) equations. In other words, by this procedure we can ascribe the iteration-convergence or iteration-divergence of Newton’s method to inherent properties of the continuum \( H(h) = 0 \) equations, rather than (say) to some artifact of our finite differencing techniques. We make use of this in section IX B.

C. Modifications of Newton’s Method

Unfortunately, as discussed in section IXB, the continuum \( H(h) \) function, and thus its discretization \( H(h) \), appears to be highly nonlinear for certain classes of errors in the initial guess. Newton’s method is well known to exhibit poor global convergence behavior in such situations, i.e. it often fails to converge unless the initial guess \( h^{(0)} \) is very close to the exact solution of the finite difference equations, \( h^* \). There’s an extensive numerical analysis literature on more robust “modified Newton” algorithms for solving nonlinear algebraic equations, for example Refs. [8,9,25,40,47,61,62]. We have found Ref. [25] to be a particularly useful introduction to this topic.

For horizon finding, the Jacobian matrix’s size is the number of (angular) grid points on the horizon surface. This is generally large enough that it’s important for the nonlinear-algebraic-equations solver to support treating the Jacobian as either a band matrix (for axisymmetric spacetimes) or a fully general sparse matrix (for fully general spacetimes). It’s also desirable for the nonlinear-algebraic-equation solver to permit explicit bounds on the solution vector, so as to ensure the trial horizon surfaces never fall outside the radial extent of the code’s main (3-dimensional) numerical grid. Unfortunately, these requirements rule out many nonlinear-algebraic-equation software packages.

For the sake of expediency, in the present work we chose to write our own implementation of a relatively simple modified-Newton algorithm, the “line search” algorithm described by Refs. [25,47]. However, a much better long-term solution would be to use an extant nonlinear-algebraic-equations code embodying (higher-quality implementations of) more sophisticated algorithms, such as GIANT package described by Refs. [61,62]. We would expect (Newton’s-method) horizon-finding codes using such software to be considerably more robust and efficient than our present code.

The modified-Newton algorithm used in this work, the line-search algorithm of Refs. [25,47], is identical to the basic Newton’s-method algorithm, except that the Newton’s-
method update Eq. (51) is modified to \( h^{(k+1)} \leftarrow h^{(k)} + \lambda \delta h^{(k)} \), where \( \lambda \in (0, 1] \) is chosen at each "outer" iteration by an inner "line search" iteration to ensure that \( \| H \|_2 \) decreases.

Refs. [25,47] show that such a choice of \( \lambda \) is always possible, and describe an efficient algorithm for it. Sufficiently close to the solution \( h^* \), this algorithm always chooses \( \lambda = 1 \), and so takes the same steps as Newton’s method. The overall modified-Newton algorithm thus retains the extremely rapid convergence of Newton’s method once the linear approximation in Eq. (49) is good.

The line-search algorithm described by Refs. [25,47] always begins by trying the basic Newton step \( \lambda = 1 \). For horizon finding, we have slightly modified the algorithm to decrease the starting value of \( \lambda \) if necessary to ensure that \( h^{(k)} + \lambda \delta h^{(k)} \) lies within the radial extent of our code’s main (3-dimensional) numerical grid at each angular grid coordinate. Our implementation of the algorithm also enforces an upper bound (typically 10%) on the relative change \( \| \lambda \delta h^{(k)}/h^{(k)} \|_\infty \) in any component of \( h^{(k)} \) in a single outer iteration. (This latter restriction is of somewhat dubious merit: although it makes the algorithm more robust when the \( H(h) \) function is highly nonlinear, it greatly slows the algorithm’s convergence when the \( H(h) \) function is only weakly nonlinear and the error in the initial guess is large. We give an example of this latter behavior in section X.)

IX. GLOBAL CONVERGENCE OF THE HORIZON FINDER

We now consider the global convergence behavior of our horizon-finding algorithm. That is, how close must the initial guess \( h^{(0)} \) be to the exact solution \( h^* \) of the finite difference equations in order for the algorithm to converge to the (correct) horizon? Equivalently, how large is the algorithm’s radius of convergence?

A. Global Convergence for Schwarzschild Spacetime

To gain a general picture of the qualitative behavior of \( H(h) \) and its implications for Newton’s-method horizon finding, it’s useful to consider Schwarzschild spacetime. We use the Eddington-Finkelstein slicing, where the time coordinate is defined by requiring \( t + r \) to be an ingoing null coordinate. (These slices aren’t maximal: \( K \) is nonzero and spatially variable throughout the slice.)

Taking the black hole to be of dimensionless unit mass, the (only) apparent horizon in the slice(s) is the coordinate sphere \( r = 2 \). More generally, a straightforward calculation gives

\[
H = \frac{2(r - 2)}{r^{3/2}\sqrt{r + 2}}
\]

for a spherical trial horizon surfaces with coordinate radius \( r \). Figure 3 shows \( H(r) \) for these surfaces. As expected, \( H = 0 \) for the horizon \( r = 2 \). However, notice that \( H \) reaches a maximum value at \( r = r^{\text{max}} = \frac{1}{2}(3 + \sqrt{33}) \approx 4.372 \), and in particular that for \( r > r^{\text{max}} \), \( H > 0 \) and \( dH/dr < 0 \). Because of this, any algorithm trying to solve \( H(r) = 0 \) using only local information about \( H(r) \) (and which maintains the spherical symmetry), including Newton’s method and any reasonable modification of it, will diverge towards infinity when...
started from within this region, or if any intermediate iterate (trial horizon surface) ever enters it.

In fact, we expect broadly similar behavior for $H$ in any black hole spacetime: Given an asymptotically flat slice containing an apparent horizon or horizons, consider any 1-parameter family of (topologically 2-spherical) nested trial horizon surfaces starting at the outermost apparent horizon and extending outward towards the 2-sphere at spatial infinity. $H = 0$ for the horizon and for the 2-sphere at spatial infinity, so $\|H\|$ must attain a maximum for some finite trial horizon surface somewhere between these two surfaces. We thus expect the same general behavior as in the Schwarzschild-slice case, i.e., divergence to infinity if the initial guess or any intermediate iterate (trial horizon surface) lies outside the maximum $\|H\|$ surface. (This argument isn't completely rigorous, since the algorithm could move inward in an angularly anisotropic manner, but this seems unlikely.)

Fortunately, in practice this problem doesn't seem to be serious: in all slices we have examined, this divergence to infinity only occurs for (coordinate-spherical) trial horizon surfaces too large (by virtue of the black hole area theorem) to be candidates for being apparent horizons.

**B. Global Convergence in the Presence of High-Spatial-Frequency Errors**

Assuming the initial guess is close enough to the horizon for the divergence-to-infinity phenomenon not to occur, we find the global convergence behavior of Newton's method to depend critically on the (angular) spatial frequency spectrum of the error in the initial guess, $h^{(0)} - h^*$: If the error has only low-spatial-frequency components (in a sense to be clarified below), then Newton's method has a large radius of convergence, i.e., it will converge even for a rather inaccurate initial guess. However, if the error has significant high-spatial-frequency components, then we find that Newton's method has a very small radius of convergence, i.e., it often fails to converge even when the error $h^{(0)} - h^*$ is very small.

We demonstrate (below) that this behavior is not an artifact of insufficient resolution in the finite difference grid. Rather, it appears to be caused by the (a) strong nonlinearity of the continuum $H(h)$ function for high-spatial-frequency components in $h$. In this context there's no sharp demarcation between "low" and "high" spatial frequencies, but in practice we use the terms to refer to angular Fourier components varying as (say) $\cos m \theta$ with $m \lesssim 4$ and $m \gtrsim 8$ respectively.

As an example of this behavior, consider Kerr spacetime with dimensionless angular momentum $J/M^2 = 0.6$. We use the Kerr slicing, where the time coordinate is defined by requiring $t + r$ to be an ingoing null coordinate. (These slices generalize the Eddington-Finkelstein slices of Schwarzschild spacetime, and are similarly nonmaximal, with $K$ nonzero and spatially variable throughout the slice.) Taking the black hole to be of dimensionless unit mass, the (only) apparent horizon in each slice is the coordinate sphere $r = h^*(\theta, \phi) = 1 + \sqrt{1 - a^2} = 1.8$.

For this example we consider two different initial guesses for the horizon position: one containing only low-spatial-frequency errors, $r = h^{(0)}(\theta, \phi) = 1.8 + 0.1 \cos 4 \theta$, and one containing significant high-spatial-frequency errors, $r = h^{(0)}(\theta, \phi) = 1.8 + 0.1 \cos 10 \theta$. Notice that both initial guesses are quite close to the actual horizon shape, differing from it by
slightly less than 5%. We use a finite difference grid with \( \Delta \theta = \frac{\pi/2}{50} \), which is ample to resolve both the low- and high-spatial-frequency-error initial guesses.

Part (a) of figure 4 shows the behavior of Newton’s method for the low-spatial-frequency-error initial guess. As can be seen, here Newton’s method converges without difficulty.

Part (b) of figure 4 shows the behavior of Newton’s method for the high-spatial-frequency-error initial guess. Here Newton’s method fails to converge: the successive iterates (trial horizon surfaces) \( h^{(k)} \) move farther and farther away from the true horizon position, and they also become more and more nonspherical. (The computed values for the next iterate \( h^{(3)} \) are almost all far outside the scale of the figure; many of them are in fact negative!)

Part (c) of figure 4 shows the behavior of the modified Newton’s method for this same high-spatial-frequency-error initial guess. Although the 1st (outer) iteration still moves the trial horizon surface somewhat inward from the horizon, the successive iterates (trial horizon surfaces) become steadily more spherical, and quickly converge to the horizon.

Notice that all of the intermediate iterates (trial horizon surfaces) shown in the figure are well-resolved by the finite difference grid. To verify that insufficient grid resolution isn’t a factor in the behavior of the horizon finder in these examples, we have rerun all three parts with several higher grid resolutions, obtaining essentially identical results to those plotted in the figure.

More quantitatively, following our discussion in section VIII B, we have made 3-grid convergence tests of each Newton or modified-Newton iterate (trial horizon surface) \( h^{(k)} \) plotted in figure 4. For example, figure 5 shows 3-grid convergence test results for the 2nd-iteration Newton iterate (trial horizon surface) \( h^{(2)} \) plotted in part (b) of figure 4, using grids with resolutions \( \Delta \theta = \frac{\pi/2}{50}, \frac{\pi/2}{100}, \frac{\pi/2}{200} \). As can be seen, this iterate shows excellent 4th order convergence. (Recall that the slope \( \frac{1}{16} \) line shown isn’t a fit to the data points, but is rather an a priori prediction with no adjustable parameters.) (Note that the absolute magnitude of the errors shown here is much larger than is typical for our horizon finder, due to a combination of the compounding of smaller errors in the earlier Newton iterate \( h^{(1)} \), and the strong angular variation in both iterates \( h^{(1)}(\theta, \phi) \) and \( h^{(2)}(\theta, \phi) \).)

The convergence results for the other Newton and modified-Newton iterates (trial horizon surfaces) plotted in figure 4 are similar: they all show excellent 4th order convergence. We conclude that the divergence of Newton’s method seen in this example, is in fact an inherent property of the continuum Newton-Kantorovich algorithm for this initial guess and slice. As discussed in section VIII B, this algorithm posits solving the linearized (continuum) \( H(h) = 0 \) equation Eq. (52) exactly at each iteration, so clearly its divergence is (must be) due to a strong nonlinearity in the continuum \( H(h) \) function when \( h \) contains high spatial frequencies.

To investigate how general the poor convergence of Newton’s method seen in this example is, and to what extent it also occurs for the modified Newton’s method, we have made a Monte Carlo numerical survey of the two algorithms’ behavior over a range of different initial-guess-error spatial frequency spectra.

In particular, fix a particular horizon-finding algorithm, and suppose we are given a slice containing an apparent horizon at the (continuum) position \( h^* \). Consider running the horizon finder with the generic perturbed initial guess.
\[ h = h^* + \sum_{m=0}^{M} c_m \cos m\theta \]  

for some set of initial-guess-error Fourier coefficients \( \{c_m\} \). (Here we include only even-\( m \) cosine terms in \( \theta \) so as to preserve axisymmetry and equatorial reflection symmetry, both of which our code assumes.)

For each value of \( M \) we define the horizon finder’s “convergence region” in \( \{c_m\} \)-space to be the set of coefficients \( \{c_m\} \) for which the horizon finder converges (we presume to the correct solution). For example, the convergence region will in practice always include the origin in \( \{c_m\} \)-space, since there \( h = h^* \), so the initial guess differs from the exact solution of the discrete \( H(h) = 0 \) equations only by the (very small) \( H(h) \) finite differencing error.

We define \( V_M \) to be the (hyper)volume of the convergence region. As described in detail in appendix B, we estimate \( V_M \) by Monte Carlo sampling in \( \{c_m\} \)-space. Given \( V_M \), we then define the “volume ratio”

\[
R_M = \begin{cases} 
V_0 & \text{if } M = 0 \\
V_M & \text{if } M \geq 2 \\
V_M^{-2} & \text{if } M \geq 2 
\end{cases}
\]

so that \( R_M \) measures the average radius of convergence of the horizon finder in the \( c_M \) dimension.

We have carried out such a horizon-perturbation survey for the same Kerr slices of the unit-mass spin-0.6 Kerr spacetime used in the previous example, for both the Newton and the modified-Newton algorithms, for \( M = 0, 2, 4, \ldots, 12 \). Figure 6 shows the resulting volume ratios. Although the precise values are somewhat dependent on the details of our implementation and on the test setup (in particular on the inner grid boundary, which was at \( r = 1 \) for these tests), the relative trends in the data should be fairly generic. We used a grid with \( \Delta \theta = \pi/2 \) for this computation, but the results would be very similar with higher resolution, since even an \( m = 12 \) perturbation is still well-resolved by this grid.

As can be seen from the figure, the modified-Newton algorithm is clearly superior to the Newton algorithm, increasing the radius of convergence by a factor of 2–3 at high spatial frequencies. However, both algorithms’ radii of convergence still fall rapidly with increasing spatial frequency, approximately as \( 1/M^{3/2} \), although the rate is somewhat slower for the modified-Newton than for the Newton algorithm. It’s clear that the Newton algorithm is essentially unusable for \( M \gtrsim 10 \) unless very good initial guesses are available, and the data suggests that the modified-Newton algorithm would have a similarly small radius of convergence by \( M \gtrsim 18 \).

As discussed in section VIII C, our present modified-Newton algorithms is a relatively simple and unsophisticated one, and much better (more robust and efficient) algorithms and implementations are available. Further research is needed to test and apply these to solving the (discrete) \( H(h) = 0 \) equations.

Our horizon-perturbation survey covers only axisymmetric spacetimes and perturbations of the initial guess. Huq (Ref. [36]) has investigated nonaxisymmetric spacetimes and perturbations, and finds similarly poor convergence of Newton’s method to our findings.

Although we write the horizon function as \( H = H(h) \), it’s more accurate to write \( H = H(g_{ij}, K_{ij}, h) \), since \( H \) also depends on the field variables in the slice and their spatial
derivatives. Examining the functional form of $H$ in Eqs. (13) and (15), we see that $H$ depends on the $g^{ij}$ components in a manner broadly similar to its dependence on $h$. We thus conjecture that $H$ may exhibit strong high-spatial-frequency nonlinearity in the field variables, in particular in the $g^{ij}$ components, similar to its dependence on $h$. If this is the case, then high-spatial-frequency perturbations in the field variables may well impair the convergence of Newton’s method in a manner similar to such perturbations in $h$. Further investigation of this possibility, either by analytical study of the nonlinear structure of the $H = H(g_{ij}, K_{ij}, h)$ function, or by numerical investigations, would be very interesting.

If spacetime contains significant amounts of high-frequency gravitational radiation, then high-spatial-frequency perturbations in all the field variables, as well as in the initial-guess error, are almost inevitable. Thus Newton’s method (and possibly other horizon-finding methods) may perform poorly in such spacetimes. On the other hand, if the horizon finder is being used at each time step of a time evolution, with the previous time step’s horizon position being used as the (horizon finder’s) initial guess, then even for such spacetimes the initial guess should be sufficiently good for Newton’s method to converge well (the initial guess error would be of the order of the time step).

More generally, it’s important to realise that the poor convergence of Newton’s method occurs only for quite high spatial frequencies. So long as only low spatial frequencies are present – which seems to be the usual situation in those (few) dynamic black hole spacetimes which have been computed thus far (for example Ref. [6]) – Newton’s method has excellent convergence behavior.

X. ACCURACY OF THE HORIZON FINDER

We now consider the accuracy (“local convergence”) of a Newton’s-method horizon finder. That is, assuming the Newton or modified-Newton iteration converges, how close is the horizon finder’s numerically computed horizon position to the (a) true (continuum) apparent horizon position $h^*$?

The horizon finder computes Newton or modified-Newton iterates (trial horizon surfaces) $h^{(k)}$ for $k = 0, 1, 2, \ldots$, until some convergence criterion is satisfied, say at $k = p$. Because of the extremely rapid convergence of the Newton and modified-Newton iterations once the error is sufficiently small (cf. section VIII A), there’s little extra cost in using a very strict convergence criterion, i.e. in solving the (discrete) $H(h) = 0$ equations to very high accuracy. In our horizon finder we typically require $\|H(h^{(p)})\|_\infty \leq 10^{-10}$.

We denote the exact solution of the (discrete) $H(h) = 0$ equations by $h^*$. Given that $\|H(h^{(p)})\|$ is reasonably small, then from standard matrix-perturbation theory (see, for example, Refs. [26,30]), $\|h^{(p)} - h^*\| \lesssim \kappa \|H(h^{(p)})\|$, where $\kappa$ is the condition number of the (presumably nonsingular) Jacobian matrix $J[H(h)]$ at the horizon position.

If we take the convergence tolerance to be strict enough for $\|h^{(p)} - h^*\|$ to be negligible, the overall accuracy of a Newton’s-method horizon finder, i.e. the external error $\|h^{(p)} - h\|$ in the computed horizon position, is thus limited only by the closeness with which the (discrete) $H(h) = 0$ equations approximate the (continuum) $H(h) = 0$ equations, i.e. by the accuracy of the $H(h)$ finite differencing. This potential for very high accuracy is one of the main advantages of the Newton’s-method horizon-finding algorithm.
For an example of the accuracy attainable in practice, we again consider the Kerr slices of the unit-mass spin-0.6 Kerr spacetime. However, to make the horizon deviate from a coordinate sphere and hence be a more significant test case for our horizon finder, we apply the transformation

\[ r \rightarrow r + \frac{b^2}{b^2 + r^2} \left( a_2 \cos 2\theta + a_4 \cos 4\theta \right) \]  

(58a)
to the spatial coordinates, where

\[ b = 5 \]  

(58b)
\[ a_2 = 0.75 \]  

(58c)
\[ a_4 = 0.05. \]  

(58d)

As shown in part (a) of figure 7, in the transformed coordinates this gives a "peanut-shaped" horizon, similar in shape to those around a pair of coalescing black holes.

We have run our horizon finder on this slice, using the (warped-coordinate) coordinate sphere \( r = 1.8 \) as an initial guess and a grid resolution of \( \Delta \theta = \frac{\pi/2}{50} \). We used the modified-Newton algorithm, which converged to the horizon without difficulty. (However, the convergence would have been much faster in the absence of our 10% restriction on the relative change in any component of \( h \) in a single outer iteration, cf. section VIII C.) Part (a) of figure 7 shows the initial guess and the final numerically computed horizon position (which at the scale of the figure is indistinguishable from the true horizon position).

Part (b) of figure 7 shows the results of a 2-grid convergence test of the final numerically computed horizon position for this example, using grids with resolutions \( \Delta \theta = \frac{\pi/2}{50}, \frac{\pi/2}{100} \). As can be seen, the numerically computed solution shows excellent 4th order convergence. (Recall again that the slope-1/10 line shown isn't a fit to the data points, but is rather an a priori prediction with no adjustable parameters.) The figure also shows that the errors in the numerically computed horizon position are very small, \( \| h^{(p)} - h^* \| \sim 10^{-5} \) for a grid resolution of \( \Delta \theta = \frac{\pi/2}{50} \) and \( \| h^{(p)} - h^* \| \sim 10^{-6} \) for a grid resolution of \( \Delta \theta = \frac{\pi/2}{100} \). Errors of this magnitude are typical of what we find for horizon finding with our code (using 4th order finite differencing), so long as the grid resolution is sufficient to adequately resolve the horizon shape.

XI. FINDING OUTERMOST APPARENT HORIZONS

The main focus of this paper is on locally finding apparent horizons, i.e. on finding an apparent horizon in a neighborhood of the initial guess. However, there's a related global problem of some interest which has heretofore attracted little attention, that of finding or recognizing the outermost apparent horizon in a slice. (By "recognizing" the outermost apparent horizon we mean the problem of determining whether or not a given apparent horizon is in fact the outermost one in a slice.)

These global problems are of particular interest when apparent horizons are used to set the inner boundary of a black-hole-excluding grid in the numerical evolution of a multiple-black-hole spacetime, as discussed by Refs. [53,49,54,5]. In this context, we can use the appearance of a new outermost apparent horizon surrounding the previously-outermost apparent horizons around two black holes as a diagnostic that the black holes have collided.
and coalesced into a single (distorted) black hole. As suggested by Ref. [54], we can then generate a new numerical grid and attach it to the new outermost apparent horizon, and continue the evolution on the exterior of the new (distorted) black hole.

So far as we know, no reliable algorithms are current known for finding or recognizing outermost apparent horizons in nonspherical spacetimes. (For spherical spacetimes, a 1-dimensional search on \( H(r) \) suffices.) If started with a very large 2-sphere as the initial guess, the curvature flow method might well converge to the outermost horizon in the slice, but as mentioned in section IV, the theoretical justification for this method’s convergence is only valid in time-symmetric \((K_{ij} = 0)\) slices.

For the remaining local-horizon-finding algorithms surveyed in section IV, including the Newton’s-method one, no better method of locating or recognizing outermost horizons seems available than trying the local-horizon-finder with several different initial guesses near the suspected position of an outermost apparent horizon. Unfortunately, if all the local-horizon-finding trials fail, this may mean that there’s no horizon in the vicinity of the initial guesses, or it may only mean that such a horizon exists but the method failed to converge to it. Moreover, it’s not clear how many local-horizon-finding trials should be made, nor just how the initial guesses should be chosen.

This is clearly not a satisfactory algorithm. Further research to develop reliable algorithms for finding or recognizing outermost apparent horizons in generic (nonspherical, nonmaximal) slices would be very useful.

XII. CONCLUSIONS

The major result of this paper is the symbolic differentiation method of computing the Jacobian matrix. This method is much more efficient than the usual numerical perturbation method. In our axisymmetric-spacetime numerical code (using a 2-dimensional numerical grid), this method is about a factor of 5 faster than any other Jacobian computation method we have tried. In fact, in our code the computation of the Jacobian by this method is only 1.5–2 times more expensive than the simple evaluation of the horizon function. We expect the symbolic differentiation method’s relative advantage over the other Jacobian computation methods to be roughly similar for any 2-dimensional code, and an additional factor of \(\sim 3–5\) larger for a 3-dimensional code.

We have previously suggested (Ref. [54]) that symbolic-differentiation Jacobian computations would be quite difficult, necessarily requiring substantial support from a (computer) symbolic computation system. Several colleagues have expressed similar opinions to us. However, this has turned out not to be the case: Using the method described in section VI A 2 of first linearizing the continuum equations, then finite differencing the (continuum) linearized equations, all the coefficients for a symbolic-differentiation Jacobian are easily computed by hand. In fact, although our code incorporates a “PDE compiler” written in a symbolic computation system (cf. appendix A), we actually derived all the the symbolic-differentiation Jacobian coefficients in this paper by hand, using only a few pages of algebra.

We assess the performance of the overall Newton’s-method algorithm for horizon finding in two main ways: convergence (does the algorithm converge?) and accuracy (if it converges, how accurate are the answers?).
We find that so long as the error in the initial guess (its deviation from the true horizon position) contains only low-spatial-frequency components, a Newton's-method horizon finder has a large radius of convergence, i.e. it converges even for rather inaccurate initial guesses. However, if the error in the initial guess contains significant high-spatial-frequency components, then we find that Newton's method has a very small radius of convergence, i.e. it may fail to converge even when the initial guess is very close to the true horizon position. In this context there's no sharp demarcation between "low" and "high" spatial frequencies, but in practice we use the terms to refer to angular Fourier components varying as (say) \( \cos m\theta \) with \( m \lesssim 4 \) and \( m \gtrsim 8 \) respectively.

High-spatial-frequency signals of this type occur naturally in all the field variables (and the horizon position) if spacetime contains significant amounts of high-frequency gravitational radiation, so this is potentially a serious limitation on Newton's-method horizon finding. However, if a Newton's-method horizon finder is used at each time step of a time evolution, then even for such a spacetime the previous time step's horizon position should be an accurate enough initial guess for Newton's method to converge well.

Using convergence tests, we demonstrate that the poor high-spatial-frequency convergence behavior of Newton's method is not an artifact of insufficient resolution in the finite difference grid. Rather, it appears to be inherent in the (a) strong nonlinearity of the continuum \( H(h) \) function for high-spatial-frequency components in \( h \). We conjecture that \( H \) may be similarly nonlinear in its high-spatial-frequency dependence on the inverse-metric components, suggesting that such perturbations in the metric might well also impair the convergence of Newton's method, and possibly other horizon-finding methods as well.

To try to improve the convergence behavior of the horizon finder, we have tried a simple "line-search" modification of Newton's method. This enlarges the radius of convergence by a factor of 2–3 over the unmodified Newton's method. However, a much better long-term solution would be for the horizon finder to use an exact nonlinear-algebraic-equations code embodying (higher-quality implementations of) more sophisticated algorithms, such as GIANT package described by Refs. [61,62]. We would expect (Newton's-method) horizon-finding codes using such software to be considerably more robust and efficient than our present code.

To investigate how general the poor convergence of Newton's method is, and to what extent it also occurs for the modified Newton's method, we have made a Monte Carlo numerical survey of the two algorithms' behavior over a range of different initial-guess-error spatial frequency spectra. We find that both algorithms' radii of convergence fall rapidly with increasing spatial frequency, roughly as \( 1/m^{3/2} \).

Further research is needed to explore more robust algorithms for solving the (discrete) \( H(h) = 0 \) equations. However, it's important to realise that the poor convergence of Newton's method and its variants occurs only for quite high spatial frequencies. So long as only low spatial frequencies are present – which seems to be the usual situation in dynamic black hole spacetimes – Newton's method has excellent convergence behavior.

Provided it converges, Newton's method for horizon finding offers the potential for very high accuracy, in practice limited only by the accuracy of the \( H(h) \) finite differencing scheme. Using 4th order finite differencing, our code typically achieves accuracies (deviation of the numerically computed horizon position from the true continuum horizon position) of \( \sim 10^{-5} \) for a grid resolution of \( \Delta \theta = \frac{\pi/2}{90} \), and \( \sim 10^{-6} \) for \( \Delta \theta = \frac{\pi/2}{100} \).

30
Comparing Newton’s method to the other known horizon-finding methods, we see the main advantages of Newton’s method as:

- It’s relatively easy to implement.
- Given a reasonable initial guess, it generally has excellent convergence properties.
- It’s very efficient.
- It has the potential to be extremely accurate.
- All of these properties continue to hold in fully generic slices, including nonmaximal ones.

The main drawback we have found to Newton’s method is its poor convergence in the presence of high-spatial-frequency errors in the initial guess, (or, we hypothesize, in the field variables, especially the 3-metric components), such as might arise if spacetime contains high-frequency gravitational radiation. However, in view of our arguments about this poor convergence being due to inherent nonlinearities in the (continuum) $H(h)$ function, we suspect that other horizon-finding algorithms may encounter similar convergence problems in such spacetimes.

Finally, we have argued that considerable further research is needed to develop algorithms for finding or recognizing the outermost apparent horizon in a slice. This is an important problem for the numerical evolution of multiple-black-hole spacetimes with the black holes excluded from the numerical evolution, but so far as we know no reliable algorithms are known for it except in spherical symmetry.

ACKNOWLEDGMENTS

We thank M. Huq for numerous useful conversations on horizon finding, and for helpful comments on earlier drafts of this paper. We thank D. Bernstein for communicating unpublished research notes on the curvature-flow method to us. We thank W. G. Unruh and the University of British Columbia Physics Department for their hospitality and the use of their research facilities, and J. Ezell for major assistance in setting up computer facilities. We thank G. Rodgers for financial support.

APPENDIX A: DETAILS OF OUR HORIZON-FINDING CODE

In this appendix we outline those details of our horizon-finding code relevant to the remainder of this paper.

Our horizon finder implements all the horizon-function and Jacobian-computation methods discussed in this paper (these are summarized in table I). It’s part of a larger 3 + 1 code designed to simulate the time evolution of an asymptotically flat axisymmetric vacuum spacetime containing a single black hole present in the initial data. The black hole is excluded from the numerical grid in the manner described by Refs. [53,49,54,5]. The code uses 4th order centered finite differencing (5-point molecules) for finite differencing,
on a 2-dimensional polar-coordinate radially-nomuniform tensor-product grid. (As well as
axisymmetry, the code also assumes equatorial reflection symmetry, but this is merely for
convenience and could easily be changed.)

The code uses a "PDE compiler" to automatically generate all the finite differencing and
other grid-computation code, including the horizon function and Jacobian computations,
from high-level tensor-differential-operator specifications of the 3 + 1 equations.

The entire code is freely available on request from the author, and may be modified
and/or redistributed under the terms of the GNU Public License. The code should be
easily portable to any modern computing platform. It's mainly written in ANSI C (about
30K lines) and the Maple symbolic-computation language (about 9K lines for the PDE
compiler itself, and about 6K lines for the 3 + 1 equations), together with some "new" Awk
(about 1K lines). The code for the horizon finder itself is about 6K lines of C and 2K lines
of Maple, although we estimate that an implementation supporting only a single horizon-
function and Jacobian-computation method would be a factor of 2–4 smaller.

The code takes the metric, extrinsic curvature, and other 3 + 1 field tensors to be al-
gebraically fully general, i.e. it permits all their coordinate components to be nonzero. To
avoid z axis coordinate singularities, the code uses a hybrid of polar spherical and Cartesian
coordinates as a tensor basis. As discussed in detail by Ref. [54], for the subset of the slice
containing the code's (2-dimensional) grid, this hybrid coordinate system combines the con-
vienient topology of polar spherical coordinates with the singularity-free nature of Cartesian
coordinates.

For present purposes, the key consequence of this z-axis-handling scheme is that in this
paper we have made no effort to avoid expressions which are singular on the z axis when
standard polar spherical coordinates are used as a tensor basis. We haven't investigated
how common such singularities are, but suspect they might be widespread. (Note that Huq
(Ref. [35]) has extended the Newton's-method algorithm to the case of Cartesian-topology
coordinates and finite differencing; much of our discussion in this paper remains applicable
to his extension.)

With the black hole excluded, the code's grid covers (only) the region \( r_{\text{min}} \leq r \leq r_{\text{max}} \),
where typically the inner boundary is at \( r_{\text{min}} = 1.0 \) and the outer boundary is far enough
from the black hole not to be a factor for horizon-finding. The horizon finder requires that
at all times the trial horizon surface position \( h \) must be entirely within the grid's radial
extent, and in fact that all of the radial finite difference molecules used for \( h \) and \( n^i \) must
be within the grid's radial extent. (The code's 2-stage scheme for \( H(h) \) uses only \( n^i \) as an
intermediate variable.)

**APPENDIX B: DETAILS OF THE HORIZON-PARAMETER-SPACE SURVEY**

In this appendix we describe our Monte Carlo horizon-parameter-space survey (cf. sec-
tion IX B) in more detail. Given the maximum initial-guess-error spatial frequency \( M \),
the goal of the survey procedure is to estimate \( V_M \), the (hyper)volume in \( \{c_m\} \)-space of the
horizon finder's convergence region.

To do this, we first start from the origin in \( \{c_m\} \)-space, and search outwards along each
c\( m \) axis until we find coefficients for which the horizon finder fails to converge. This gives
the intersection of the \( c_m \) coordinate axes with the boundary of the convergence region.
We then construct a sequence of nested hypercubes (strictly speaking, hyper-parallelepips) $C_1$, $C_2$, $C_3$, ... in $\{c_m\}$-space, starting with $C_1$ just containing the $c_m$-coordinate-axis boundaries of the convergence region, and expanding outwards. We use the obvious Monte Carlo sampling algorithm to estimate the volume of the convergence region contained within the first hypercube $C_1$, and then within the differences $C_{k+1} - C_k$ of the succeeding hypercubes. We continue this process until one of the differences contains no convergence-region volume. We include one additional hypercube in the sequence after this, typically 25–50% larger than the previous one, to provide a margin of safety against missing disconnected "islands" or fractal zones near the boundary of the convergence region. (Recall that the (fractal) Julia set is just the convergence region of a simple Newton's-method iteration.) Finally, we compute an estimate for $V_M$ by simply adding the convergence-region-volume estimates for $C_1$ and each $C_{k+1} - C_k$.

Unfortunately, as $M$ and hence the dimensionality of $\{c_m\}$-space increases, we find that the fraction of the hypercubes and hypercube differences occupied by the convergence region decreases rapidly, so a very large number of horizon-finding trials is needed to obtain a reasonable statistical accuracy for $V_M$. (For example, the $M = 12$ points in figure 6 required 15,000 trials each.) It's this effect which ultimately limits the maximum value of $M$ attainable in practice by a survey of this type.
REFERENCES


FIG. 1. This figure illustrates the various 2-stage and 1-stage computation schemes for the horizon function $H(h)$. The solid arrows denote finite differencing operations, the dotted arrow denotes an algebraic computation, and the dashed arrow denotes a radial interpolation to the horizon position $r = h(\theta, \phi)$. Each path from $h$ to $H$ represents a separate computation scheme. Notice that there are three distinct 2-stage schemes (using the upper arrows from $^2h$ to $^3H$ in the figure) and one 1-stage scheme (using the lower arrow from $^2h$ to $^3H$).

FIG. 2. This commutative diagram illustrates the two different ways a Jacobian matrix can be computed. Given a nonlinear (continuum) function $P(Q)$, the Jacobian matrix $J_{P(Q)}$ is logically defined in terms of the lower-left path in the diagram, i.e., it's defined as the Jacobian of a (nonlinear) discrete (finite difference) approximation $P(Q)$ to $P(Q)$. However, if the operations of discretization (finite differencing) and linearization commute, we can instead compute the Jacobian by the upper-right path in the diagram, i.e., by first linearizing the continuum $P(Q)$ function, then discretizing (finite differencing) this linearization $\delta P(\delta Q)$.

FIG. 3. This figure shows $H(r)$ for spherical trial horizon surfaces with coordinate radius $r$ in an Eddington-Finkelstein slice of a unit-mass Schwarzschild spacetime. Notice that for $r > r_{\text{max}} \approx 4.372$, $H > 0$ and $dH/dr < 0$, so Newton's method diverges in this region.

FIG. 4. This figure illustrates how the convergence behavior of the basic and modified Newton iterations depends on the spatial-frequency spectrum of the error $h - h^*$ in the initial guess. In each part of the figure, the true (continuum) horizon $h^*$ is plotted as a solid line, while the horizon finder's first few iterates (trial horizon surfaces) $h^{(k)}$ are plotted with dots at the grid points. Part (a) of the figure shows the behavior of Newton's method for an initial-guess-error containing only low spatial frequencies, part (b) shows the behavior of Newton's method for an initial-guess-error containing significant high spatial frequencies, and part (c) shows the behavior of the modified Newton iteration for the same initial guess as part (b). In parts (a) and (c), where the iterates $h^{(k)}$ are converging, the final iterates shown are indistinguishable from the true (continuum) horizon at the scale of the figure.

FIG. 5. This figure shows the results of a 3-grid convergence test for the 2nd-iteration Newton iterate (trial horizon surface) $h^{(2)}$ plotted in part (b) of figure 4. The line has slope $1/16^3$, appropriate for 4th order convergence.

FIG. 6. This figure shows the volume ratios $R_M$ for the horizon-perturbation survey. These measure the average radius of convergence of the horizon finder as a function of the initial guess error's maximum spatial frequency $M$. The points and solid lines show the results for the modified-Newton (upper) and Newton (lower) algorithms, with $\pm 1\sigma$ statistical error bars from the Monte Carlo estimation procedure. The dashed line shows an $R_M \sim 1/M^{3/2}$ falloff.
FIG. 7. This figure illustrates the accuracy of our horizon finder for a test case where the horizon’s coordinate shape is significantly non-spherical. The figure is plotted using the transformed radial coordinate defined by Eq. (58). Part (a) of the figure shows the “peanut-shaped” true (continuum) horizon position $h^*$, plotted as a solid line, and the initial guess $h^{(0)}$ and the final result of the horizon finder (the numerically computed horizon position) $h^{(p)}$, plotted with dots at the grid points. At the scale of the figure, the numerically computed horizon position $h^{(p)}$ is indistinguishable from the true (continuum) position $h^*$. Part (b) shows the results of a 2-grid convergence test for the numerically computed horizon position $h^{(p)}$. The line has slope $\frac{1}{16}$, appropriate for 4th order convergence.
TABLE I. This table summarizes the various methods for computing the horizon function $(^{(2)}H(h))$ and its Jacobian $J^{[^{(2)}H(h)]}$. The “codes” are shorthand labels which we use to refer to the various methods. The relative CPU times are as measured for our implementation (described in appendix A), and are per angular grid point, normalized relative to the 1-stage $(^{(2)}H(h))$ computation.

<table>
<thead>
<tr>
<th>Code</th>
<th>Dimensions</th>
<th>Jacobian Computation Method</th>
<th>Horizon Function Scheme</th>
<th>Relative CPU time</th>
<th>Estimated Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_{1s}$</td>
<td></td>
<td>1-stage</td>
<td></td>
<td>$\equiv 1$</td>
<td>Moderate</td>
</tr>
<tr>
<td>$H_{2s}$</td>
<td></td>
<td>2-stage</td>
<td></td>
<td>0.7</td>
<td>Low</td>
</tr>
<tr>
<td>2d.np.1s</td>
<td>2-dimensional</td>
<td>Numerical perturbation of $(^{(2)}H(h))$</td>
<td>1-stage</td>
<td>6</td>
<td>Low</td>
</tr>
<tr>
<td>2d.np.2s</td>
<td>2-dimensional</td>
<td>Numerical perturbation of $(^{(2)}H(h))$</td>
<td>2-stage</td>
<td>8</td>
<td>Low</td>
</tr>
<tr>
<td>3d.np.1s</td>
<td>3-dimensional</td>
<td>Numerical perturbation of $(^{(3)}H(h))$</td>
<td>1-stage</td>
<td>7</td>
<td>Low – Moderate</td>
</tr>
<tr>
<td>3d.np.2s</td>
<td>3-dimensional</td>
<td>Numerical perturbation of $(^{(3)}H(h))$</td>
<td>2-stage</td>
<td>8</td>
<td>Low – Moderate</td>
</tr>
<tr>
<td>3d.sd.1s</td>
<td>3-dimensional</td>
<td>Symbolic differentiation of $(^{(3)}H(h))$</td>
<td>1-stage</td>
<td>1.5</td>
<td>Moderate</td>
</tr>
<tr>
<td>3d.np2.2s</td>
<td>3-dimensional</td>
<td>Numerical perturbation of $s_i(h)$ or $n_i(h)$</td>
<td>2-stage</td>
<td>14</td>
<td>Moderate – High</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and of $(^{(3)}H(s_i))$ or $(^{(3)}H(n_i))$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3d.sd2.2s</td>
<td>3-dimensional</td>
<td>Symbolic differentiation of $s_i(h)$ or $n_i(h)$</td>
<td>2-stage</td>
<td>5</td>
<td>Moderate – High</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and of $(^{(3)}H(s_i))$ or $(^{(3)}H(n_i))$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>