Finding apparent horizons in numerical relativity

Jonathan Thornburg*
Physics Department, University of British Columbia, Vancouver, British Columbia, Canada V6T 1Z1

(Received 7 August 1995)

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 $H \equiv \nabla_i n^i + K_{ij} n^i n^j - K = 0$ 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 variant to solve the finite difference equations. We describe a method for computing the Jacobian matrix of the finite differenced $H(h)$ function $\mathsf{H}(\mathsf{h})$ by symbolically differentiating the finite difference equations, giving the Jacobian elements directly in terms of the finite difference molecule coefficients used in computing $\mathsf{H}(\mathsf{h})$. 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 a number of different $\mathsf{H}(\mathsf{h})$ and Jacobian computation schemes. We find this symbolic differentiation method of computing the $\mathsf{H}(\mathsf{h})$ 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 $\mathsf{H}(\mathsf{h})=0$ equations, we find that Newton's method generally shows robust convergence. 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 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 of Newton's method roughly doubles the horizon finder's radius of convergence, but both the unmodified and modified methods' radia 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 $\mathsf{H}(\mathsf{h})=0$ 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 $\mathsf{H}(\mathsf{h})$ finite differencing scheme. Using fourth order finite differencing, we demonstrate that the error in the numerically computed horizon position shows the expected $O((\Delta\theta)^4)$ scaling with grid resolution $\Delta\theta$, and is typically $\sim 10^{-5}(10^{-6})$ for a grid resolution of $\Delta\theta = \frac{\pi/2}{50}(\frac{\pi/2}{100})$. 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. [S0556-2821(96)02616-1]

PACS number(s): 04.25.Dm, 02.60.Cb, 02.60.Lj, 02.70.Bf

I. INTRODUCTION

In $3+1$ numerical relativity, one often wishes to locate the black hole(s) in a (spacelike) slice. As discussed by Refs. [1,2], a black hole is rigorously defined in terms 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 [3], ''a number of nice properties,'' it is 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. [4,5], 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 outer trapped surface, is defined [1,2] locally in time, within a single slice, as a closed two-surface whose outgoing null geodesics have zero expansion. An apparent horizon is slicing-dependent: If we define a ''world tube'' by taking the union of the apparent horizon(s) in each slice of a slicing, this world tube will vary from one slicing to another. In a stationary spacetime event and apparent horizons coincide, although this generally is not the case in dynamic spacetimes. However, given certain technical assumptions, the existence of an apparent horizon in a slice implies the existence of an event horizon, and thus by definition a black hole, containing the apparent horizon. (Unfortunately, the converse does not always hold. Notably, Wald and Iyer [6] have constructed a family of angularly anisotropic slices in Schwarzschild spacetime which approach arbitrarily close to $r=0$ yet contain no apparent horizons.)

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. [4,7]), and

*Address for written correspondence: Box 8–7, Thetis Island, British Columbia, V0R 2Y0, Canada. Electronic address: thornbur@theory.physics.ubc.ca for use ''on the fly'' during numerical evolutions to help in choosing the coordinates and ''steering'' the numerical evolution [8–11]. 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 do not develop in the time evolution. Furthermore, the horizon must be re-located at each time step of the evolution, and so the horizon-finding algorithm should be as efficient as possible. Finally, when evolving multiple-black-hole spacetimes in this manner it is 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 Sec. 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) partial differential equation (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 Sec. 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 [12], 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. [13]. We use the standard $3+1$ formalism of Arnowitt, Deser, and Misner [14] (see Refs. [15,16] for recent reviews).

We assume that a specific spacetime and $3+1$ (spacelike) slice are given, 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 often refer to various sets in the slice as being one, two, or three 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 two dimensional.

We assume that the spatial coordinates $x^i \equiv (r,\theta,\phi)$ are such that in some neighborhood of the horizon, surfaces of constant r are topologically nested two-spheres with r increasing outward, and we refer to r as a ''radial'' coordinate and θ and ϕ as ''angular'' coordinates. For pedagogical convenience (only), we take θ and ϕ to be the usual polar spherical coordinates, so that if spacetime is axisymmetric (spherically symmetric), ϕ is (θ and ϕ are) the symmetry coordinate(s). However, we make no assumptions about the detailed form of the coordinates; i.e., we allow all components of the three-metric 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 [17] or multiple-coordinate-patch coordinate systems [18].

We use $ijkl$ for spatial (three)-indices, and $uvwxy$ for indices ranging over the angular coordinates only. g_{ij} denotes the three-metric in the slice, g its determinant, and ∇_i the associated three-covariant derivative operator. K_{ij} denotes the three-extrinsic curvature of the slice, and K its trace.

We use $\mathcal{A}$ to denote the two dimensional space of angular coordinates (θ,ϕ). We sometimes need to distinguish between field variables defined on $\mathcal{A}$ or on the (two dimensional) horizon, and field variables defined on a three 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 use italic letters H, h, etc., to denote *continuum* coordinates, functions, differential operators, and other quantities. We use sans serif 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_{I}. We use $\mathbf{J}[\mathsf{P}(\mathsf{Q})]$ to denote the Jacobian matrix of the grid function $\mathsf{P}=\mathsf{P}(\mathsf{Q})$, as defined by Eq. (17), and $\cdot$ to denote the product of two such Jacobians or that of a Jacobian and a grid function. We use $J[P(Q)]$ to denote the linearization of the differential operator $P=P(Q)$ about the point Q.

We use M as a generic finite difference molecule and M as a generic index for molecule coefficients. We write $\mathrm{M}\in\mathsf{M}$ to mean that M has a nonzero coefficient at position M. Temporarily taking $\langle \mathrm{M}\rangle$ to denote some particular coordinate component of M, we refer to $\max_{\mathrm{M}\in\mathsf{M}}|\langle \mathrm{M}\rangle|$ as the ''radius'' of M, and to the number of distinct $\langle \mathrm{M}\rangle$ values with $M\in\mathsf{M}$ as the ''diameter'' or ''number of points'' of M. (For example, the usual symmetric second-order three-point molecules for first and second derivatives both have radius 1 and diameter 3.) We often refer to a molecule as itself being a discrete operator, the actual application to a grid function being implicit.

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

III. APPARENT HORIZON EQUATION

As discussed by (for example) Ref. [19], an apparent horizon satisfies the equation

$$H \equiv {}^{(2)}H \equiv \nabla_i n^i + K_{ij} n^i n^j - K = 0, \tag{1}$$

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). [Notice that in order for the three-divergence $\nabla_i n^i$ to be meaningful, n^i must be (smoothly) continued off the horizon, and extended to a field ${}^{(3)}n^i$ in some three-dimensional neighborhood of the horizon. The off-horizon continuation is nonunique, but it is easy to see that this does not affect H on the horizon.]

To solve the apparent horizon equation (1), we begin by assuming that the horizon and coordinates are such that each radial coordinate line $\{(\theta,\phi)=\text{const}\}$ intersects the horizon in exactly one point. In other words, we assume that the horizon’s coordinate shape is a ‘‘Strahlkörper,’’ defined by Minkowski 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’’ [20]. Given this assumption, we can parametrize the horizon’s shape by $r=h(\theta,\phi)$ for some single-valued ‘‘horizon shape function’’ h defined on the two-dimensional domain $\mathcal{A}$ of angular coordinates (θ,ϕ).

Equivalently, we may write the horizon’s shape as ${}^{(3)}F=0$, where the scalar function ${}^{(3)}F$, defined on some three-dimensional neighborhood $\mathcal{N}$ of the horizon, satisfies ${}^{(3)}F=0$ if and only if $r=h(\theta,\phi)$, and we take ${}^{(3)}F$ to increase outward. In practice we take ${}^{(3)}F(r,\theta,\phi) = r-h(\theta,\phi)$.

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

$$s_i \equiv {}^{(3)}s_i = \nabla_i {}^{(3)}F, \tag{2}$$

i.e., by

$$s_r = 1, \tag{3a}$$

$$s_u = -\partial_u h, \tag{3b}$$

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

$$n^i \equiv {}^{(3)}n^i = \frac{s^i}{\|s^k\|} \tag{4}$$

$$= \frac{g^{ij}s_j}{\sqrt{g^{kl}s_k s_l}} \tag{5}$$

$$= \frac{g^{ir}-g^{iu}\partial_u h}{\sqrt{g^{rr}-2g^{ru}\partial_u h + g^{uv}(\partial_u h)(\partial_v h)}}. \tag{6}$$

Henceforth we drop the ${}^{(3)}$ prefixes on ${}^{(3)}s_i$ and ${}^{(3)}n^i$.

Substituting Eq. (6) into the apparent horizon equation (1), we see that the horizon function $H(h)$ depends on the (angular) second derivatives of h. In fact, the apparent horizon equation (1) is a second-order elliptic (boundary-value) PDE for h on the domain of angular coordinates $\mathcal{A}$. The apparent horizon equation (1) must, therefore, be augmented with suitable boundary conditions to define a (locally) unique solution. These are easily obtained by requiring the horizon’s three-dimensional shape to be smooth across the artificial boundaries $\theta=0$, $\theta=\pi$, $\phi=0$, and $\phi=2\pi$.

IV. ALGORITHMS FOR SOLVING THE APPARENT HORIZON EQUATION

We now survey various algorithms for solving the apparent horizon equation (1). Reference [21] reviews much of the previous work on this topic.

In spherical symmetry, the apparent horizon equation (1) degenerates into a one-dimensional nonlinear algebraic 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. [22,23,9,11]. (See also Ref. [24] for an interesting analytical study giving necessary and sufficient conditions for apparent horizons to form in nonvacuum spherically symmetric spacetimes.)

In an axisymmetric spacetime, the angular-coordinate space $\mathcal{A}$ is effectively one-dimensional, and so the apparent horizon equation (1) reduces to a nonlinear two-point boundary value ordinary differential equation (ODE) for the function $h(\theta)$, which may be solved either with a shooting method, or with one of the more general methods described below. Shooting methods have been used by a number of authors, for example, Refs. [25–31].

The remaining apparent-horizon-finding algorithms we discuss are all applicable to either axisymmetric spacetimes (two-dimensional codes) or fully general spacetimes (three-dimensional codes).

Tod [32] has proposed an interesting pair of ‘‘curvature flow’’ methods for finding apparent horizons. Bernstein [33] 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 parametrize the horizon shape. One algorithm rewrites the apparent horizon equation $H(a_k)=0$ as $\|H(a_k)\|=0$ and 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. [34,35] in axisymmetric spacetimes, and more recently by Ref. [36] in fully general spacetimes. Alternatively, Nakamura, Oohara, and Kojima [37–39] have suggested a functional iteration method 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 [40] have suggested and tested several modifications of 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 $\{(\theta_{\text{K}},\phi_{\text{K}})\}$ gives a set of simultaneous nonlinear algebraic equations for the unknown values $\{h(\theta_{\text{K}},\phi_{\text{K}})\}$, which are then solved by some variant of Newton’s method. This ‘‘Newton’s-method’’ algorithm (we continue to use this term even if a modification of Newton’s method is actually used) has been used in axisymmetric spacetimes by a number of authors, for example, Refs. [41–44,10], and is also applicable in fully general spacetimes when the coordinates have a (locally) polar spherical topology. Huq [45] 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 $\mathsf{H}(\mathsf{h})$, the computation of the discrete horizon function's Jacobian matrix $\mathbf{J}[\mathsf{H}(\mathsf{h})]$, and the solution of the simultaneous nonlinear algebraic equations $\mathsf{H}(\mathsf{h})=0$. 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 $\mathsf{H}(\mathsf{h})$. More precisely, first fix an angular-coordinate grid $\{(\theta_{\mathrm{K}},\phi_{\mathrm{K}})\}$. Then, given a "trial horizon surface" $r=h(\theta,\phi)$, which need not actually be an apparent horizon, we define $\mathsf{h}(\theta,\phi)$ to be the discretization of $h(\theta,\phi)$ to the angular-coordinate grid, and we consider the computation of $\mathsf{H}(\mathsf{h})$ on the discretized trial horizon surface, i.e., at the points $\{(r=\mathsf{h}(\theta_{\mathrm{K}},\phi_{\mathrm{K}}),\theta=\theta_{\mathrm{K}},\phi=\phi_{\mathrm{K}})\}$.

The apparent horizon equation (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 (three-dimensional) grid points of the underlying $3+1$ code of which the horizon finder is a part. We therefore extend ${}^{(2)}H$ to some (three-dimensional) neighborhood $\mathcal{N}$ of the trial horizon surface; i.e., we define an extended horizon function ${}^{(3)}H$ on $\mathcal{N}$:

$$ {}^{(3)}H=\nabla_i n^i+K_{ij}n^i n^j-K \tag{7}$$

$$ =\partial_i n^i+(\partial_i\ln\sqrt{g})n^i+K_{ij}n^i n^j-K. \tag{8}$$

To compute ${}^{(2)}\mathsf{H}(\mathsf{h})$ on the (discretized) trial horizon surface, we first compute ${}^{(3)}\mathsf{H}(\mathsf{h})$ on the underlying $3+1$ code's (three-dimensional) grid points in $\mathcal{N}$, and then radially interpolate these ${}^{(3)}\mathsf{H}$ values to the trial-horizon-surface position to obtain ${}^{(2)}\mathsf{H}(\mathsf{h})$,

$$ {}^{(2)}\mathsf{H}(\theta,\phi)=\mathrm{interp}({}^{(3)}\mathsf{H}(r,\theta,\phi),\ \ r=\mathsf{h}(\theta,\phi)) \tag{9a}$$

or, equivalently,

$$ {}^{(2)}\mathsf{H}_{\mathrm{I}}=\mathrm{interp}({}^{(3)}\mathsf{H}_{\langle r\mathrm{I}\rangle},r=\mathsf{h}_{\mathrm{I}}), \tag{9b}$$

where I is an angular grid-point index and the $\langle r\mathrm{I}\rangle$ subscript denotes that the interpolation is done independently at each angular coordinate along the radial coordinate line $\{\theta=\theta_{\mathrm{I}},\phi=\phi_{\mathrm{I}}\}$. In practice any reasonable interpolation method should work well here: References [43,44] report 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. Neglecting the interpolation error, we can also write Eq. (9) in the form

$$ {}^{(2)}\mathsf{H}(\theta,\phi)={}^{(3)}\mathsf{H}(r=\mathsf{h}(\theta,\phi),\theta,\phi). \tag{10}$$

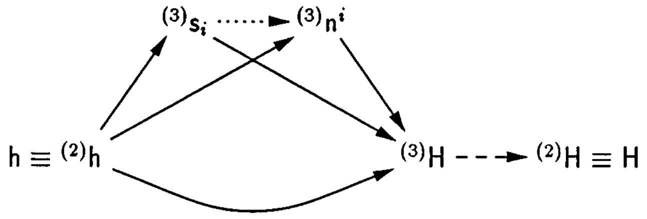
${}^{(3)}\mathsf{s}_i$
${}^{(3)}\mathsf{n}^i$
$\mathsf{h}\equiv{}^{(2)}\mathsf{h}$
${}^{(3)}\mathsf{H}$
${}^{(2)}\mathsf{H}\equiv\mathsf{H}$

FIG. 1. This figure illustrates the various two-stage and one-stage computation methods for the horizon-function $\mathsf{H}(\mathsf{h})$. 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 $r=\mathsf{h}(\theta,\phi)$. Each path from h to H represents a separate computation method. Notice that there are three distinct two-stage methods (using the upper arrows from ${}^{(2)}\mathsf{h}$ to ${}^{(3)}\mathsf{H}$ in the figure) and one one-stage method (using the lower arrow from ${}^{(2)}\mathsf{h}$ to ${}^{(3)}\mathsf{H}$).

We consider two basic types of methods for computing the extended horizon-function ${}^{(3)}\mathsf{H}(\mathsf{h})$.

A "two-stage" computation method uses two sequential numerical finite differencing stages, first explicitly computing s_i and/or n^i by numerically finite differencing h, and then computing ${}^{(3)}\mathsf{H}$ by numerically finite differencing s_i or n^i.

A "one-stage" computation method uses only a single numerical (second) finite differencing stage, computing ${}^{(3)}\mathsf{H}$ directly in terms of h's first and second angular derivatives.

Figure 1 illustrates this.

To derive the detailed equations for these methods, we substitute Eqs. (3) and Eqs. (5) into Eq. (8):

$$ {}^{(3)}H=\nabla_i n^i+K_{ij}n^i n^j-K \tag{11}$$

$$ =\partial_i n^i+(\partial_i\ln\sqrt{g})n^i+K_{ij}n^i n^j-K \tag{12}$$

$$ =\partial_i\frac{g^{ij}s_j}{(g^{kl}s_k s_l)^{1/2}}+(\partial_i\ln\sqrt{g})\frac{g^{ij}s_j}{(g^{kl}s_k s_l)^{1/2}}+\frac{K^{ij}s_i s_j}{g^{kl}s_k s_l}-K \tag{13}$$

$$ =\frac{A}{D^{3/2}}+\frac{B}{D^{1/2}}+\frac{C}{D}-K, \tag{14}$$

where the subexpressions A, B, C, and D are given by

$$ A=-(g^{ik}s_k)(g^{jl}s_l)\partial_i s_j-\tfrac{1}{2}(g^{ij}s_j)[(\partial_i g^{kl})s_k s_l], \tag{15a}$$

$$ B=(\partial_i g^{ij})s_j+g^{ij}\partial_i s_j+(\partial_i\ln\sqrt{g})(g^{ij}s_j), \tag{15b}$$

$$ C=K^{ij}s_i s_j, \tag{15c}$$

$$ D=g^{ij}s_i s_j, \tag{15d}$$

i.e.,

$$ A=(g^{ur}-g^{uw}\partial_w h)(g^{vr}-g^{vw}\partial_w h)\partial_{uv}h -\tfrac{1}{2}(g^{ir}-g^{iu}\partial_u h)[\partial_i g^{rr}-2(\partial_i g^{ru})\partial_u h +(\partial_i g^{uv})(\partial_u h)(\partial_v h)], \tag{16a}$$

$$ B=[\partial_i g^{ir}-(\partial_i g^{iu})\partial_u h]-g^{uv}\partial_{uv}h+(\partial_i\ln\sqrt{g})(g^{ir}-g^{iu}\partial_u h), \tag{16b}$$

$$C = K^{rr} - 2K^{ru}\partial_u h + K^{uv}(\partial_u h)(\partial_v h), \tag{16c}$$

$$D = g^{rr} - 2g^{ru}\partial_u h + g^{uv}(\partial_u h)(\partial_v h). \tag{16d}$$

Comparing the one-stage and two-stage methods, the two-stage methods' equations are somewhat simpler, and so these methods are somewhat easier to implement and somewhat cheaper (faster) 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 one-stage method, the two-stage methods double the effective radius of the net $\mathsf{H}(\mathsf{h})$ finite differencing molecules, and thus have 2 (4) times as many nonzero off-diagonal Jacobian elements for a two- (three-) dimensional code. In practice the cost of computing these extra Jacobian elements for the two-stage methods more than outweighs the slight cost saving in evaluating the horizon function. We discuss the relative costs of the different methods further in Sec. VI D.

VI. COMPUTING THE JACOBIAN

In this section we discuss the details of the computation of the Jacobian matrix $\mathbf{J}[\mathsf{H}(\mathsf{h})]$ of the horizon function $\mathsf{H}(\mathsf{h})$ on a given trial horizon surface.

A. Computing the Jacobian of a generic function $\mathsf{P}(\mathsf{Q})$

We consider first the case of a generic function $P(Q)$ in d dimensions, finite differenced using N-point molecules. We define the Jacobian matrix of the discrete $\mathsf{P}(\mathsf{Q})$ function by

$$\mathbf{J}[\mathsf{P}(\mathsf{Q})]_{\mathrm{IJ}} = \frac{\partial \mathsf{P}_{\mathrm{I}}}{\partial \mathsf{Q}_{\mathrm{J}}} \tag{17a}$$

or, equivalently, by the requirement that

$$\delta\mathsf{P}_{\mathrm{I}} \equiv [\mathsf{P}(\mathsf{Q}+\delta\mathsf{Q}) - \mathsf{P}(\mathsf{Q})]_{\mathrm{I}} = \mathbf{J}[\mathsf{P}(\mathsf{Q})]_{\mathrm{IJ}} \cdot \delta\mathsf{Q}_{\mathrm{J}} \tag{17b}$$

for any infinitesimal perturbation $\delta\mathsf{Q}$ of Q.

We assume that P is actually a *local* grid function of Q, and so the Jacobian matrix is sparse. (For example, this would preclude the nonlocal fourth-order "compact differencing" methods described by Refs. [46,47].) We assume that by exploiting the locality of the discrete $\mathsf{P}(\mathsf{Q})$ function, any single 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 $\mathbf{J}[\mathsf{P}(\mathsf{Q})]$. The first of these is the "numerical perturbation" method. This involves numerically perturbing Q and examining the resulting perturbation in $\mathsf{P}(\mathsf{Q})$,

$$\mathbf{J}[\mathsf{P}(\mathsf{Q})]_{\mathrm{IJ}} \approx \left[\frac{\mathsf{P}(\mathsf{Q}+\mu\mathsf{e}^{(\mathrm{J})}) - \mathsf{P}(\mathsf{Q})}{\mu}\right]_{\mathrm{I}}, \tag{18}$$

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

$$[\mathsf{e}^{(\mathrm{J})}]_{\mathrm{I}} = \begin{cases} 1 & \text{if } \mathrm{I}=\mathrm{J}, \\ 0 & \text{otherwise}, \end{cases} \tag{19}$$

and μ is a "small" perturbation amplitude. This computation of the Jacobian proceeds by columns: For each J, Q_{J} is perturbed, and the resulting perturbation in $\mathsf{P}(\mathsf{Q})$ gives the Jth column of the Jacobian matrix.

The perturbation amplitude μ should be chosen to balance the truncation error of the one-sided finite difference approximation (18) against the numerical loss of significance caused by subtracting the nearly equal quantities $\mathsf{P}(\mathsf{Q}+\mu\mathsf{e}^{(\mathrm{J})})$ and $\mathsf{P}(\mathsf{Q})$. References [48,49] discuss the choice of μ, and conclude that if $\mathsf{P}(\mathsf{Q})$ can be evaluated with an accuracy of ε, then $\mu \approx \sqrt{\varepsilon}$ "seems to work the best." In practice the choice of μ is not very critical for horizon finding. Values of 10^{-4}–10^{-6} seem to work well, and the inaccuracies in the Jacobian matrix resulting from these values of μ do not seem to be a significant problem.

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

Assuming that $\mathsf{P}(\mathsf{Q})$ is already known, computing $\mathbf{J}[\mathsf{P}(\mathsf{Q})]$ by numerical perturbation requires a total of N^d P_{I} evaluations at each grid point; i.e., it requires a perturbed-P_{I} evaluation for each nonzero Jacobian element.

2. Computing Jacobians by symbolic differentiation

An alternate method of computing the Jacobian matrix $\mathbf{J}[\mathsf{P}(\mathsf{Q})]$ is by "symbolic differentiation." This method makes explicit use of the finite differencing scheme used to compute the discrete $\mathsf{P}(\mathsf{Q})$ function.

Suppose first that the continuum $P(Q)$ function is a position-dependent local *linear* differential operator, discretely approximated by a position-dependent local finite difference molecule M:

$$\mathsf{P}_{\mathrm{I}} = \sum_{\mathrm{M} \in \mathsf{M}(\mathrm{I})} \mathsf{M}(\mathrm{I})_{\mathrm{M}} \mathsf{Q}_{\mathrm{I+M}}. \tag{20}$$

Differentiating this, we have

$$\mathbf{J}[\mathsf{P}(\mathsf{Q})]_{\mathrm{IJ}} \equiv \frac{\partial \mathsf{P}_{\mathrm{I}}}{\partial \mathsf{Q}_{\mathrm{J}}} = \begin{cases} \mathsf{M}(\mathrm{I})_{\mathrm{J-I}} & \text{if } \mathrm{J}-\mathrm{I} \in \mathsf{M}(\mathrm{I}), \\ 0 & \text{otherwise}, \end{cases} \tag{21}$$

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

More generally, suppose P is a position-dependent local nonlinear algebraic function of Q and some finite number of Q's derivatives, say,

$$P = P(Q, \partial_i Q, \partial_{ij} Q). \tag{22}$$

Logically, the Jacobian matrix $\mathbf{J}[\mathsf{P}(\mathsf{Q})]$ is defined [by Eq. (17)] in terms of the linearization of the discrete (finite differenced) $\mathsf{P}(\mathsf{Q})$ function. However, as illustrated in Fig. 2, if the discretization (the finite differencing scheme) commutes with the linearization, we can instead compute the Jacobian by first linearizing the continuum $P(Q)$ function, and then finite differencing this (continuum) linearized function. (This method of computing the Jacobian is essentially just the

$$\begin{tikzcd}
\begin{array}{c}\text{nonlinear}\\ \text{continuum}\\ P(Q)\end{array} \arrow[r, "\text{linearize}"] \arrow[d, "\begin{array}{l}\text{discretize}\\ \text{(finite difference)}\end{array}"] & \begin{array}{c}\text{linearized}\\ \text{continuum}\\ \delta P(\delta Q)\end{array} \arrow[d, "\begin{array}{l}\text{discretize}\\ \text{(finite difference)}\end{array}"] \\
\begin{array}{c}\text{nonlinear}\\ \text{discrete}\\ \text{(finite difference)}\\ \mathsf{P(Q)}\end{array} \arrow[r, "\text{linearize}"] & \begin{array}{c}\text{linearized}\\ \text{discrete}\\ \text{(finite difference)}\\ \delta\mathsf{P}(\delta\mathsf{Q})\end{array}
\end{tikzcd}$$

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 $\mathbf{J}[\mathsf{P(Q)}]$ is logically defined in terms of the lower-left path in the diagram; i.e., it is defined as the Jacobian of a nonlinear discrete (finite difference) approximation $\mathsf{P(Q)}$ 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 and then discretizing (finite differencing) this linearization $\delta P(\delta Q)$.

''Jacobian part'' of the Newton-Kantorovich algorithm for solving nonlinear elliptic PDEs.)

That is, we first linearize the continuum $P(Q)$ function:

$$\delta P = \frac{\partial P}{\partial Q}\delta Q + \frac{\partial P}{\partial(\partial_i Q)}\delta\partial_i Q + \frac{\partial P}{\partial(\partial_{ij} Q)}\delta\partial_{ij} Q \tag{23}$$

$$= \frac{\partial P}{\partial Q}\delta Q + \frac{\partial P}{\partial(\partial_i Q)}\partial_i \delta Q + \frac{\partial P}{\partial(\partial_{ij} Q)}\partial_{ij} \delta Q. \tag{24}$$

We then view the linearized function $\delta P(\delta Q)$ as a linear differential operator, and discretely approximate it by the position-dependent finite difference molecule

$$\mathsf{M} = \frac{\partial P}{\partial Q}\mathbf{I} + \frac{\partial P}{\partial(\partial_i Q)}\mathbf{d}_i + \frac{\partial P}{\partial(\partial_{ij} Q)}\mathbf{d}_{ij}, \tag{25}$$

where $\mathbf{I}$ is the identity molecule and $\mathbf{d}_i$ and $\mathbf{d}_{ij}$ are finite difference molecules discretely approximating ∂_i and ∂_{ij}, respectively. Finally, we apply Eq. (21) to the molecule M defined by Eq. (25) to obtain the desired Jacobian matrix $\mathbf{J}[\mathsf{P(Q)}]$.

In practice, there is no need to explicitly form the molecule M — the Jacobian matrix elements can easily be assembled directly from the known $\mathbf{I}$, $\mathbf{d}_i$, and $\mathbf{d}_{ij}$ molecule coefficients and the ''Jacobian coefficients'' $\partial P/\partial Q$, $\partial P/\partial(\partial_i Q)$, and $\partial P/\partial(\partial_{ij} Q)$. Once these 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. (25) and (21). The main cost of computing a Jacobian matrix by symbolic differentiation is thus the computation of the Jacobian coefficients themselves. Depending on the functional form of the $P(Q)$ function, there may be anywhere from 1 to 10 coefficients, although in practice these often have many common subexpressions.

In other words, where the numerical perturbation method requires a P_{I} evaluation per nonzero Jacobian *element*, the symbolic differentiation method requires the computation of ''a few'' Jacobian-coefficient subexpressions per Jacobian *row*. More precisely, suppose the computation of all the Jacobian coefficients at a single grid point is R times as costly as a P_{I} evaluation. Then the symbolic differentiation method is approximately N^d/R times more efficient than the numerical perturbation method.

B. Semantics of the horizon-function Jacobian

We now consider the detailed semantics of the horizon-function Jacobian. We define the Jacobian of $\mathsf{H(h)} \equiv {}^{(2)}\mathsf{H(h)}$, $\mathbf{J}[\mathsf{H(h)}] \equiv \mathbf{J}[{}^{(2)}\mathsf{H(h)}]$, by

$$\mathbf{J}[{}^{(2)}\mathsf{H(h)}]_{\mathrm{IJ}} = \frac{d^{(2)}\mathsf{H}_{\mathrm{I}}}{d\mathsf{h}_{\mathrm{J}}} \tag{26a}$$

or, equivalently, by the requirement that

$$\delta^{(2)}\mathsf{H}_{\mathrm{I}} \equiv [{}^{(2)}\mathsf{H}(\mathsf{h}+\delta\mathsf{h}) - {}^{(2)}\mathsf{H(h)}]_{\mathrm{I}} = \mathbf{J}[{}^{(2)}\mathsf{H(h)}]_{\mathrm{IJ}}\cdot\delta\mathsf{h}_{\mathrm{J}} \tag{26b}$$

for any infinitesimal perturbation $\delta\mathsf{h}$. Here I and J are both angular (two-dimensional) grid-point indices. Notice that this definition uses the *total* derivative $d^{(2)}\mathsf{H}/d\mathsf{h}$. This is because ${}^{(2)}\mathsf{H(h)}$ is defined to always be evaluated *at the position* $r=\mathsf{h}(\theta,\phi)$ *of the trial horizon surface*, and so the Jacobian $\mathbf{J}[{}^{(2)}\mathsf{H(h)}]$ must take into account not only the direct change in ${}^{(2)}\mathsf{H}$ at a fixed position due to a perturbation in h, but also the implicit change in ${}^{(2)}\mathsf{H}$ caused by the field-variable coefficients in ${}^{(2)}\mathsf{H}$ being evaluated at a perturbed position $r=\mathsf{h}(\theta,\phi)$.

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

$$\mathbf{J}[{}^{(3)}\mathsf{H(h)}]_{\mathrm{IJ}} = \frac{\partial^{(3)}\mathsf{H}_{\mathrm{I}}}{\partial\mathsf{h}_{\mathrm{J}}} \tag{27a}$$

or, equivalently, by the requirement that

$$\delta^{(3)}\mathsf{H}_{\mathrm{I}} \equiv [{}^{(3)}\mathsf{H}(\mathsf{h}+\delta\mathsf{h}) - {}^{(3)}\mathsf{H(h)}]_{\mathrm{I}} = \mathbf{J}[{}^{(3)}\mathsf{H(h)}]_{\mathrm{IJ}}\cdot\delta\mathsf{h}_{\mathrm{J}} \tag{27b}$$

for any infinitesimal perturbation $\delta\mathsf{h}$. Here I is a three-dimensional grid-point index for ${}^{(3)}\mathsf{H}$, while J is an (angular) two-dimensional grid-point index for h. In contrast with $\mathbf{J}[{}^{(2)}\mathsf{H(h)}]$, this definition uses the *partial* derivative $\partial^{(3)}\mathsf{H}/\partial\mathsf{h}$. This is because we take ${}^{(3)}\mathsf{H(h)}$ to be evaluated at a fixed position (a grid point in the neighborhood $\mathcal{N}$ of the trial horizon surface) *which does not change with perturbations in* h, and so $\mathbf{J}[{}^{(3)}\mathsf{H(h)}]$ need only take into account the direct change in ${}^{(3)}\mathsf{H}$ at a fixed position due to a perturbation in h.

$\mathbf{J}[{}^{(3)}\mathsf{H(h)}]$ thus has much simpler semantics than $\mathbf{J}[{}^{(2)}\mathsf{H(h)}]$. We have found $\mathbf{J}[{}^{(3)}\mathsf{H(h)}]$ very useful, both as an intermediate variable in the computation of $\mathbf{J}[{}^{(2)}\mathsf{H(h)}]$ (described in the next section), and also conceptually, as an aid to *thinking* about the Jacobians.

TABLE I. This table summarizes the various methods for computing the horizon function ${}^{(2)}\mathsf{H}(\mathsf{h})$ and its Jacobian $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$. The ''codes'' are shorthand labels for referring to the various methods. The relative CPU times are as measured for our implementation (described in Appendix B), and are per angular grid point, normalized relative to the one-stage ${}^{(2)}\mathsf{H}(\mathsf{h})$ computation. The notation ''$\mathsf{s}_i|\mathsf{n}^i$'' means whichever of s_i and/or n^i is appropriate, depending on the precise two-stage method used to compute the horizon function.

Code	Jacobian computation dimensions	Jacobian computation method	Horizon function method	Jacobian matrices used	Relative CPU time	Estimated implementation effort
H.1s			one-stage		$\equiv 1$	Moderate
H.2s			two-stage		0.7	Low
2d.np.1s	two-dimensional	Numerical perturbation of ${}^{(2)}\mathsf{H}(\mathsf{h})$	one-stage	$\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$	6	Low
2d.np.2s	two-dimensional	Numerical perturbation of ${}^{(2)}\mathsf{H}(\mathsf{h})$	two-stage	$\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$	8	Low
3d.np.1s	three-dimensional	Numerical perturbation of ${}^{(3)}\mathsf{H}(\mathsf{h})$	one-stage	$\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$, $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$	7	Low – Moderate
3d.sd.1s	three-dimensional	Symbolic differentiation of ${}^{(3)}\mathsf{H}(\mathsf{h})$	one-stage	$\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$, $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$	1.5	Moderate
3d.np.2s	three-dimensional	Numerical perturbation of ${}^{(3)}\mathsf{H}(\mathsf{h})$	two-stage	$\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$, $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$	8	Low – Moderate
3d.np2.2s	three-dimensional	Numerical perturbation of $\mathsf{s}_i|\mathsf{n}^i(\mathsf{h})$ and ${}^{(3)}\mathsf{H}(\mathsf{s}_i|\mathsf{n}^i)$	two-stage	$\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$, $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$, $\mathbf{J}[\mathsf{s}_i|\mathsf{n}^i(\mathsf{h})]$, $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{s}_i|\mathsf{n}^i)]$	14	Moderate – High
3d.sd2.2s	three-dimensional	Symbolic differentiation of $\mathsf{s}_i|\mathsf{n}^i(\mathsf{h})$ and ${}^{(3)}\mathsf{H}(\mathsf{s}_i|\mathsf{n}^i)$	two-stage	$\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$, $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$, $\mathbf{J}[\mathsf{s}_i|\mathsf{n}^i(\mathsf{h})]$, $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{s}_i|\mathsf{n}^i)]$	5	Moderate – High

C. Computing the horizon-function Jacobian

Table I (discussed further in Sec. VI D) summarizes all the Jacobian-computation methods in this paper, which we now describe in detail. We tag each method with a shorthand ''code,'' which gives the method's basic properties: whether it computes $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$ directly or computes $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$ as an intermediate step, whether it uses symbolic differentiation or numerical perturbation, and whether it uses a one-stage or a two-stage horizon function computation.

The simplest methods for computing $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$ are the ''two-dimensional'' ones, which work directly with ${}^{(2)}\mathsf{H}(\mathsf{h})$ in angular-coordinate space, without computing $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$ as an intermediate step. Since ${}^{(2)}\mathsf{H}(\mathsf{h})$ is not given by a simple molecule operation of the form Eq. (20), symbolic differentiation is not directly applicable here. However, numerical perturbation in angular-coordinate space is applicable, using either a one-stage or a two-stage method to compute ${}^{(2)}\mathsf{H}(\mathsf{h})$. We refer to the resulting Jacobian computation methods as the ''2d.np.1s'' and ''2d.np.2s'' methods, respectively.

Our remaining methods for computing $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$ are all ''three-dimensional'' ones, which first explicitly compute $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$ and then compute $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$ from this in the manner described below.

If ${}^{(3)}\mathsf{H}(\mathsf{h})$ is computed using the one-stage method, i.e., via Eqs. (14) and (16), then either numerical perturbation or symbolic differentiation may be used to compute $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$. We refer to these as the ''3d.np.1s'' and ''3d.sd.1s'' methods, respectively. The symbolic differentiation Jacobian coefficients for the 3d.sd.1s method are tabulated in Appendix A.

Alternatively, if ${}^{(3)}\mathsf{H}(\mathsf{h})$ is computed using a two-stage method, then $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$ may be computed either by the simple numerical perturbation of ${}^{(3)}\mathsf{H}(\mathsf{h})$ (the ''3d.np.2s'' method) or by separately computing the Jacobians of the individual stages and matrix-multiplying them together. For the latter case, either numerical perturbation or symbolic differentiation may be used to compute the individual-stage Jacobians, giving the ''3d.np2.2s'' and ''3d.sd2.2s'' methods, respectively. The symbolic differentiation Jacobian coefficients for the 3d.sd2.2s method are tabulated in Appendix A.

For any of the three dimensional methods, once $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$ is known, we compute $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$ as follows:

$$\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]_{\mathrm{IJ}} \equiv \frac{d\,{}^{(2)}\mathsf{H}_{\mathrm{I}}}{d\mathsf{h}_{\mathrm{J}}} \tag{28}$$

$$= \frac{d\,{}^{(2)}\mathsf{H}(\theta_{\mathrm{I}},\phi_{\mathrm{I}})}{d\mathsf{h}(\theta_{\mathrm{J}},\phi_{\mathrm{J}})} \tag{29}$$

$$= \frac{d\,{}^{(3)}\mathsf{H}(r=\mathsf{h}(\theta_{\mathrm{I}},\phi_{\mathrm{I}}),\theta_{\mathrm{I}},\phi_{\mathrm{I}})}{d\mathsf{h}(\theta_{\mathrm{J}},\phi_{\mathrm{J}})} \quad [\text{by Eq. (10)}] \tag{30}$$

$$= \left.\frac{\partial\,{}^{(3)}\mathsf{H}(r,\theta_{\mathrm{I}},\phi_{\mathrm{I}})}{\partial\mathsf{h}(\theta_{\mathrm{J}},\phi_{\mathrm{J}})}\right|_{r=\mathsf{h}(\theta_{\mathrm{I}},\phi_{\mathrm{I}})} + \left.\frac{\partial\,{}^{(3)}\mathsf{H}(r,\theta_{\mathrm{I}},\phi_{\mathrm{I}})}{r}\right|_{r=\mathsf{h}(\theta_{\mathrm{I}},\phi_{\mathrm{I}})} \tag{31}$$

$$= \mathrm{interp}(\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]_{\langle r\mathrm{I}\rangle\mathrm{J}}, r=\mathsf{h}_{\mathrm{I}}) + \mathrm{interp}'({}^{(3)}\mathsf{H}_{\langle r\mathrm{I}\rangle}, r=\mathsf{h}_{\mathrm{I}}), \tag{32}$$

where the $\langle r\mathrm{I}\rangle$ subscripts in Eq. (32) denote that the interpolations are done along the radial line $\{\theta=\theta_{\mathrm{I}},\phi=\phi_{\mathrm{I}}\}$, analogously to Eq. (9), and where we neglect the interpolation errors in Eq. (32).

Notice that the $\mathrm{interp}'(\cdots)$ term in Eq. (32) may be computed very cheaply using the same ${}^{(3)}\mathsf{H}$ data values used in computing ${}^{(2)}\mathsf{H}$; cf. Eq. (9). [The number of ${}^{(3)}\mathsf{H}$ 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 $\mathrm{interp}'(\cdots)$ term in Eq. (32) as in the $\mathrm{interp}(\cdots)$ term.] It is thus easy to compute $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$ once $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$ is known.

D. Comparing the methods

Table I summarizes all the horizon-function and Jacobian-computation methods described in Secs. V and VI C. The table also shows which Jacobian matrices the methods use, the methods' measured relative CPU times in our axisymmetric-spacetime (two-dimensional) code (discussed further in Appendix B), and our estimates of the methods' approximate implementation effort (programming complexity).

As can be seen from the table, for our implementation 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 numerical perturbation methods. In fact, the computation of the Jacobian $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$ by the 3d.sd.1s method is only 1.5–2 times more expensive than the simple evaluation of the horizon-function ${}^{(2)}\mathsf{H}(\mathsf{h})$.

The relative performance of the different methods will of course vary considerably from one implementation to another, and especially between axisymmetric-spacetime (two-dimensional) and fully-general-spacetime (three-dimensional) codes. However, counting the number of operations needed for each method shows that the 3d.sd.1s method should remain the fastest for any reasonable implementation. (We omit details of the counting in view of their length and lack of general interest.) Notably, the 3d.sd.1s method's relative advantage over the other methods should be approximately a factor of the molecule diameter *larger* for fully-general-spacetime (three-dimensional) codes than for axisymmetric-spacetime (two-dimensional) codes such as ours.

Considering now the implementation efforts required by the various methods, in general we find that these depend more on which Jacobian matrices are involved than on how the Jacobians are computed: The two-dimensional methods, involving only $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$, are the easiest to implement, while the three-dimensional methods involving (only) $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$ and $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$ are somewhat harder to implement. The three-dimensional methods involving the individual-stage Jacobians $\mathbf{J}[\mathsf{s}_i(\mathsf{h})]$, $\mathbf{J}[\mathsf{n}^i(\mathsf{h})]$, $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{s}_i)]$, and/or $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{n}^i)]$ are considerably more difficult to implement, due to these Jacobians' more complicated sparsity patterns.

All the Jacobian matrices are highly sparse, and for reasonable efficiency it is essential to exploit this sparsity in their storage and computation. We have done this in our code, and our CPU-time measurements and implementation-effort estimates all reflect this. We briefly describe our sparse-Jacobian storage scheme in Appendix C. This scheme is very efficient, but its programming is a significant fraction of the overall Jacobian implementation effort, especially for the individual-stage Jacobians.

Comparing numerical perturbation and symbolic differentiation methods, we had previously suggested [50] that symbolic-differentiation Jacobian computations would be very difficult to implement, necessarily requiring substantial support from a (computer) symbolic computation system. Several colleagues have expressed similar opinions to us. We had also previously suggested [50] that due to the structure of the $H(h)$ function, a Jacobian-coefficient formalism of the type described in Secs. VI A 2 and VI C would not be valid for the horizon function, and so symbolic differentiation methods would require explicitly differentiating the finite difference equations.

These suggestions have proven to be incorrect: Using the Jacobian-coefficient formalism described in Secs. VI A 2 and VI C, only the continuum equations need be differentiated, and this is easily done by hand. More generally, using this formalism we find the actual programming of the symbolic differentiation methods to be only moderately more difficult than that of the numerical perturbation methods. Some of the Jacobian coefficients tabulated in Appendix A are fairly complicated, but no more so than many other computations in $3+1$ numerical relativity.

In order to be confident of the correctness of any of the Jacobian-computation methods except the simple two-dimensional numerical perturbation ones, we feel that it is highly desirable to program an independent method (which may be programmed for simplicity at the expense of efficiency) and make an end-to-end comparison of the resulting Jacobian matrices. (We have successfully done this for each of the Jacobian matrices computed by each of the methods listed in Table I, and our implementation-effort estimates there include doing this.) If, and only if, the Jacobians agree to within the expected truncation error of the numerical-perturbation Jacobian approximation (18) can we then have a high degree of confidence that both calculations are correct. If they disagree, then we find the detailed pattern of which matrix elements differ to be a very useful debugging aid.

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

VII. CONVERGENCE TESTS

Before continuing our discussion of Newton's-method horizon finding, in this section we digress to consider the convergence of finite differencing computations to the continuum limit.

As has been forcefully emphasized by Choptuik [23,51,52], 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, and only with, such estimates available can we safely 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 that the (a) true (continuum) apparent horizon position h^* is known. For a convergence test in this case, we run the horizon finder twice, using a 1:2 ratio of grid resolutions. As discussed in detail by Ref. [51], if the code's numerical errors are dominated by truncation errors from nth-order finite differencing, the numerically computed horizon positions h must satisfy

$$\mathsf{h}[\Delta x]=h^*+(\Delta x)^n f+O((\Delta x)^{n+2}), \tag{33a}$$

$$\mathsf{h}[\Delta x/2]=h^{*}+(\Delta x/2)^{n}f+O((\Delta x)^{n+2}), \tag{33b}$$

at each grid point, where $\mathsf{h}[\Delta x]$ denotes the numerically computed horizon position using grid resolution Δx, and f is an $O(1)$ smooth function depending on various high-order derivatives of h^{*} and the field variables, but *not* on the grid resolution. [We are assuming centered finite differencing here in writing the higher-order terms as $O((\Delta x)^{n+2})$; otherwise, they would only be $O((\Delta x)^{n+1})$.] Neglecting the higher-order terms, i.e., in the limit of small Δx, we can eliminate f to obtain a direct relationship between the code's errors at the two resolutions,

$$\frac{\mathsf{h}[\Delta x/2]-h^{*}}{\mathsf{h}[\Delta x]-h^{*}}=\frac{1}{2^{n}}, \tag{34}$$

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

To test how well any particular set of (finite-resolution) numerical results satisfies this convergence criterion, we plot a scatterplot of the high-resolution errors $\mathsf{h}[\Delta x/2]-h^{*}$ against the low-resolution errors $\mathsf{h}[\Delta x]-h^{*}$ at the grid points common to the two grids. If, and given the arguments of Ref. [51], in practice *only* if, the error expansions Eq. (33) are valid with the higher-order error terms negligible, i.e., if and only if the errors are indeed dominated by the expected nth-order finite difference truncation errors, will all the points in the scatterplot fall on a line through the origin with slope $1/2^{n}$.

Now suppose the true (continuum) apparent horizon position h^{*} is unknown. For a convergence test in this case, we run the horizon finder 3 times, using a 1:2:4 ratio of grid resolutions. Analogously to the two-grid case, we now have

$$\mathsf{h}[\Delta x]=h^{*}+(\Delta x)^{n}f+O((\Delta x)^{n+2}), \tag{35a}$$

$$\mathsf{h}[\Delta x/2]=h^{*}+(\Delta x/2)^{n}f+O((\Delta x)^{n+2}), \tag{35b}$$

$$\mathsf{h}[\Delta x/4]=h^{*}+(\Delta x/4)^{n}f+O((\Delta x)^{n+2}), \tag{35c}$$

at each grid point, with f again independent of the grid resolution. Again neglecting the higher-order terms, we can eliminate both f and h^{*} to obtain the ''three-grid'' convergence criterion

$$\frac{\mathsf{h}[\Delta x/2]-\mathsf{h}[\Delta x/4]}{\mathsf{h}[\Delta x]-\mathsf{h}[\Delta x/2]}=\frac{1}{2^{n}}, \tag{36}$$

which must hold at each grid point common to the three grids. We test this criterion using a scatterplot technique analogous to that for the two-grid criterion (34).

We emphasize that for a three-grid convergence test of this type, the true continuum solution h^{*} need not be known. In fact, 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 (35) hold. We make use of this latter case in Secs. VIII C and IX B to apply three-grid convergence tests to intermediate Newton iterates (trial horizon surfaces) of our horizon finder.

For both the two-grid and the three-grid convergence test, we find that the *pointwise* nature of the scatterplot comparison makes it significantly more useful than a simple comparison of gridwise norms. In particular, the scatterplot 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.

Notice also that the parameter n, the order of the convergence, is (should be) 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 is not fitted to the data points, but is rather an *a priori* prediction with *no* adjustable parameters. Convergence tests of this type are thus a very strong test of the validity of the finite differencing scheme and the error expansions Eq. (33) or Eq. (35).

VIII. SOLVING THE NONLINEAR ALGEBRAIC EQUATIONS

Returning to our specific discussion of horizon finding, we now discuss the details of using Newton's method or a variant to solve the simultaneous nonlinear algebraic equations $\mathsf{H}(\mathsf{h})=0$.

A. Newton's method

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

$$\mathsf{H}(\mathsf{h}^{(k)}+\delta\mathsf{h})=\mathsf{H}(\mathsf{h}^{(k)})+\mathbf{J}[\mathsf{H}(\mathsf{h}^{(k)})]\cdot\delta\mathsf{h}+O(\|\delta\mathsf{h}\|^{2}), \tag{37}$$

where $\delta\mathsf{h}$ now denotes a finite perturbation in h, and where $\mathbf{J}[\mathsf{H}(\mathsf{h}^{(k)})]$ denotes the Jacobian matrix $\mathbf{J}[\mathsf{H}(\mathsf{h})]$ evaluated at the point $\mathsf{h}=\mathsf{h}^{(k)}$. We then neglect the higher-order (nonlinear) terms and solve for the perturbation $\delta\mathsf{h}^{(k)}$ such that $\mathsf{H}(\mathsf{h}^{(k)}+\delta\mathsf{h}^{(k)})=0$. This gives the simultaneous linear algebraic equations

$$\mathbf{J}[\mathsf{H}(\mathsf{h}^{(k)})]\cdot\delta\mathsf{h}^{(k)}=-\mathsf{H}(\mathsf{h}^{(k)}) \tag{38}$$

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

$$\mathsf{h}^{(k+1)}\leftarrow\mathsf{h}^{(k)}+\delta\mathsf{h}^{(k)} \tag{39}$$

and repeat the iteration until some convergence criterion is satisfied.

Notice that here we are using the word ''convergence'' in a very different sense from that of Sec. VII — here it refers to the ''iteration convergence'' of the Newton iterates $\mathsf{h}^{(k)}$ to the exact solution h^{*} of the discrete equations, whereas there it refers to the ''finite difference convergence'' of a finite difference computation result $\mathsf{h}[\Delta x]$ to its continuum limit h^{*} as the grid resolution is increased.

Once the current solution estimate $\mathsf{h}^{(k)}$ is reasonably close to h^{*}; i.e., in practice once the trial horizon surface is reasonably close to the (an) apparent horizon, Newton's method converges extremely rapidly. In particular, once the linear approximation in Eq. (37) is approximately valid, Newton's method roughly squares the relative error $\|\mathsf{h}-\mathsf{h}^{*}\|/\|\mathsf{h}^{*}\|$ at each iteration, and can thus bring the error down to a negligible value in only a few (more) iterations. (This rapid ''quadratic'' convergence depends critically on the mutual consistency of the horizon function and Jacobian matrix used in the computation, and is thus a useful diagnostic for monitoring the Jacobian's correctness.) (For a detailed discussion of Newton's method, including precise formulations and proofs of these statements, see, for example, Ref. [49].)

However, if the initial guess $\mathsf{h}^{(0)}$ for the horizon position, or more generally any Newton iterate (trial horizon surface) $\mathsf{h}^{(k)}$, differs sufficiently from h^* so that the linear approximation in Eq. (37) is not approximately valid, then Newton's method may converge poorly, or fail to converge at all.

B. Modifications of Newton's method

Unfortunately, as discussed in Sec. IX B, for certain types of initial guesses Newton's method fails to converge unless the initial guess is very close to the exact solution of the finite difference equations. There is an extensive numerical analysis literature on more robust ''modified Newton'' algorithms for solving nonlinear algebraic equations, for example, Refs. [53–59]. We have found Ref. [55] 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 is important for the nonlinear-algebraic-equations solver to support treating the Jacobian as either a band matrix (for axisymmetric-spacetime codes) or a fully general sparse matrix (for fully-general-spacetime codes). It is also desirable for the nonlinear-algebraic-equations 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 three-dimensional grid. Unfortunately, these requirements rule out many nonlinear-algebraic-equations 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. [55,57]. However, a much better long-term solution would be to use an extant nonlinear-algebraic-equations code embodying high-quality implementations of more sophisticated algorithms, such as the GIANT code described by Refs. [58,59]. 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. [55,57], is identical to the basic Newton's-method algorithm, except that the Newton's-method update, Eq. (39), is modified to $\mathsf{h}^{(k+1)} \leftarrow \mathsf{h}^{(k)} + \lambda\, \delta\mathsf{h}^{(k)}$, where $\lambda \in (0,1]$ is chosen at each ''outer'' iteration by an inner ''line search'' iteration to ensure that $\|\mathsf{H}\|_2$ decreases monotonically. References [55,57] show that such a choice of λ 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. (37) is good.

The line-search algorithm described by Refs. [55,57] 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 λ if necessary to ensure that $\mathsf{h}^{(k)} + \lambda\, \delta\mathsf{h}^{(k)}$ lies within the radial extent of our code's main (three-dimensional) numerical grid at each angular grid coordinate. Our implementation of the algorithm also enforces an upper bound (typically 10%) on the relative change $\|\lambda\, \delta\mathsf{h}^{(k)}/\mathsf{h}^{(k)}\|_\infty$ in any component of $\mathsf{h}^{(k)}$ in a single outer iteration. However, it is not clear whether or not this latter restriction is a good idea: Although it makes the algorithm more robust when the $\mathsf{H}(\mathsf{h})$ function is highly nonlinear, it may slow the algorithm's convergence when the $\mathsf{H}(\mathsf{h})$ function is only weakly nonlinear and the error in the initial guess is large. We give an example of this latter behavior in Sec. X.

C. Newton-Kantorovich method

We have described the Newton's-method algorithm, and the more robust modified versions of it, in terms of solving the discrete $\mathsf{H}(\mathsf{h}) = 0$ equations. However, these algorithms can also be interpreted directly in terms of solving the continuum $H(h) = 0$ equations. This ''Newton-Kantorovich'' method and its relationship to the discrete Newton's method are discussed in detail by Ref. [60].

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

$$H(h^{(k)} + \delta h) = H(h^{(k)}) + J[H(h^{(k)})](\delta h) + O(\|\delta h\|^2), \tag{40}$$

where δh is now a finite perturbation in h, and where the linear differential operator $J[H(h^{(k)})]$ is now the 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 $H(h^{(k)} + \delta h^{(k)}) = 0$. This gives the linear differential equation

$$J[H(h^{(k)})](\delta h^{(k)}) = -H(h^{(k)}) \tag{41}$$

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

$$h^{(k+1)} \leftarrow h^{(k)} + \delta h^{(k)} \tag{42}$$

and repeat the iteration until some convergence criterion is satisfied.

Now suppose we discretely approximate this continuum Newton-Kantorovich algorithm by finite differencing the iteration equation (41). If the finite differencing and the linearization commute in the manner discussed in Sec. VI A 2, then *this finite difference approximation to the Newton-Kantorovich algorithm is in fact identical to the discrete Newton's-method algorithm applied to the (discrete)* $\mathsf{H}(\mathsf{h}) = 0$ *equations obtained by finite differencing the continuum* $H(h) = 0$ *equation*. (In a simpler context, our Jacobian-coefficient formalism described in Sec. VI A 2 essentially just exploits the ''Jacobian part'' of this identity.)

Therefore, when using the discrete Newton's method to solve the $\mathsf{H}(\mathsf{h}) = 0$ equations, we can equivalently view each Newton iterate (trial horizon surface) $\mathsf{h}^{(k)}[\Delta x]$ as being a finite difference approximation to the corresponding continuum Newton-Kantorovich iterate (trial horizon surface)

$h^{(k)}$. As the grid resolution is increased, each Newton iterate $\mathsf{h}^{(k)}[\Delta x]$ should therefore show proper finite difference convergence *regardless of the iteration convergence or iteration divergence of the Newton or Newton-Kantorovich iteration itself.*

Moreover, once we verify the individual Newton iterates' finite differencing convergence (with a three-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, as opposed to (say) a finite differencing artifact. We make use of this in Sec. IX B.

IX. GLOBAL CONVERGENCE OF THE HORIZON FINDER

We now consider the global convergence behavior of the Newton's-method horizon-finding algorithm. That is, how close must the initial guess $\mathsf{h}^{(0)}$ be to the (an) exact solution h^* of the finite difference equations in order for the iterates (trial horizon surfaces) $\mathsf{h}^{(k)}$ to converge to h^*? In other words, 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 is 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 are not maximal: K is nonzero and spatially variable throughout the slices.)

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

$$H=\frac{2(r-2)}{r^{3/2}\sqrt{r+2}} \tag{43}$$

for 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 $r=r^{\max}=\frac{1}{2}(3+\sqrt{33})\approx 4.372$, and in particular that for $r>r^{\max}$, $H>0$ and $dH/dr<0$. Because of this, almost any algorithm — including Newton's method and its variants — which tries to solve $H(r)=0$ using only local information about $H(r)$, and which maintains the spherical symmetry, 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 one-parameter family of topologically two-spherical nested trial horizon surfaces starting at the outermost apparent horizon and extending outward towards the two-sphere at spatial infinity. $H=0$ for the horizon and for the two-sphere at spatial

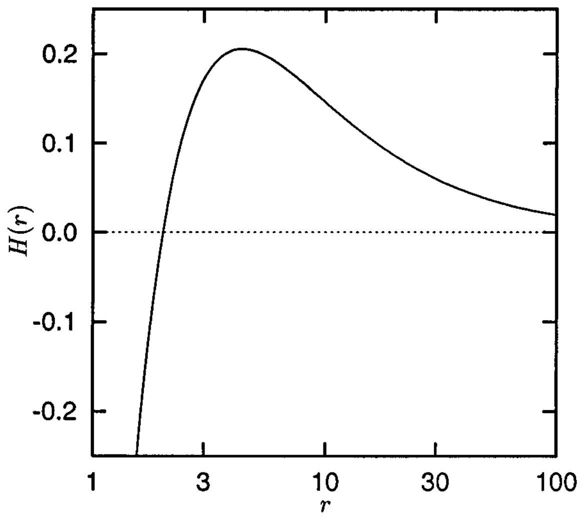

FIG. 3. This figure shows $H(r)$ for spherical trial horizon surfaces with coordinate radius r in an Eddington-Finkelstein slice of a unit-mass Schwarzschild spacetime. Notice that for $r>r^{\max}\approx 4.372$, $H>0$ and $dH/dr<0$, and so Newton's method diverges in this region.

infinity, and 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 is not completely rigorous, since the algorithm could move inward in an angularly anisotropic manner, but this seems unlikely.

Fortunately, in practice this is not a problem: The black hole area theorem places an upper bound on the size of an apparent horizon, and this lets us avoid overly large initial guesses, or restart the Newton iteration if any intermediate iterate (trial horizon surface) is too large.

B. Global convergence in the presence of high-spatial-frequency errors

Assuming that 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 initial guess's error $\mathsf{h}^{(0)}-\mathsf{h}^*$: 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* $\mathsf{h}^{(0)}-\mathsf{h}^*$ *is very small.*

This behavior is *not* an artifact of insufficient resolution in the finite difference grid. Rather, it appears to be caused by a strong nonlinearity in the continuum $H(h)$ function for high-spatial-frequency components in h. In this context there is 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.

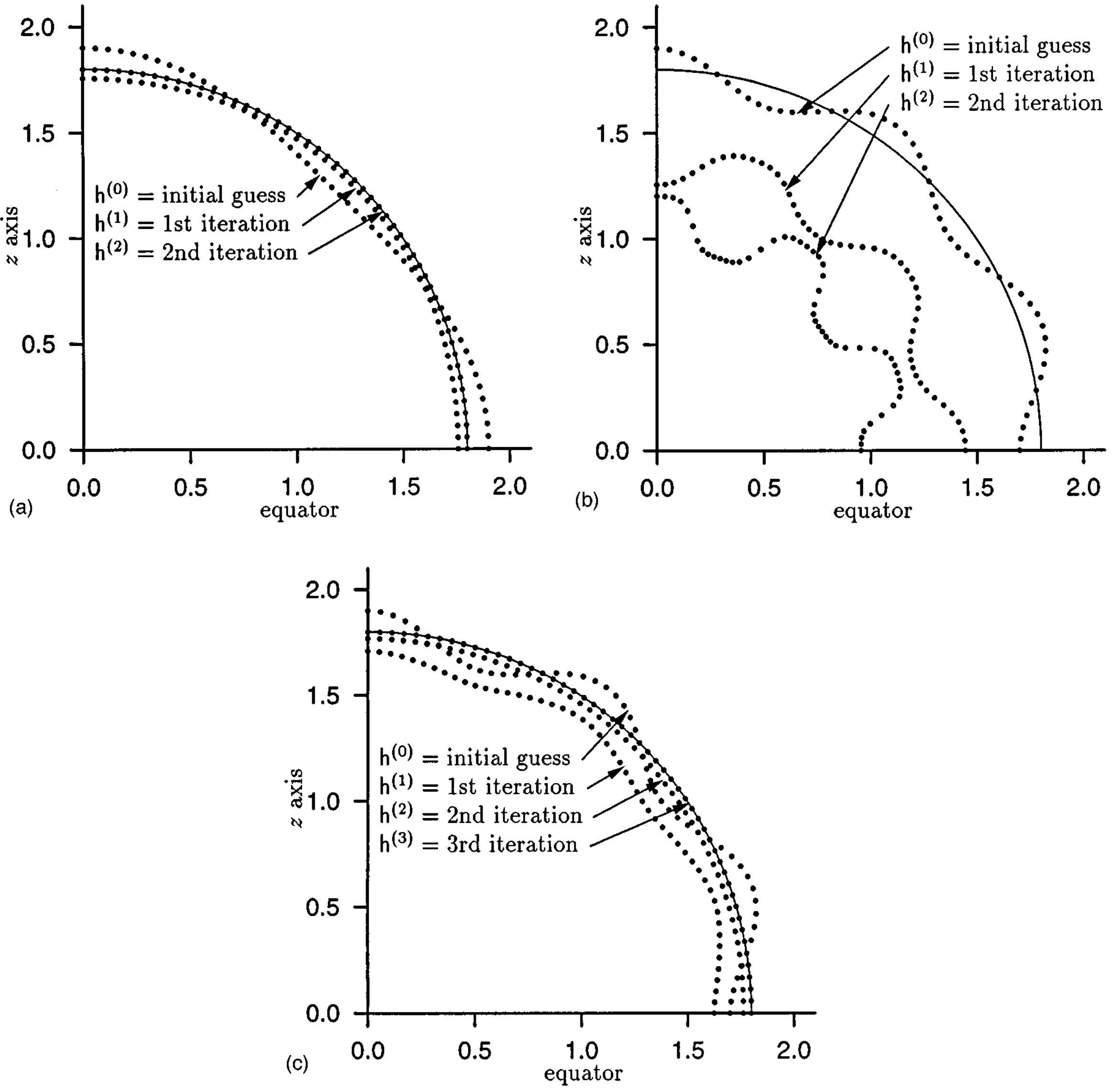

FIG. 4. This figure illustrates how the convergence behavior of the basic and modified-Newton iterations depends on the spatial-frequency spectrum of the initial guess's error $\mathsf{h}^{(0)}-\mathsf{h}^*$. In each part of the figure, the true continuum horizon h^* is plotted as a solid line, while the horizon finder's first few iterates (trial horizon surfaces) $\mathsf{h}^{(k)}$ 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 iteration is converging, the final iterates shown are indistinguishable from the true continuum horizon at the scale of the figure. In part (b), where the iteration is diverging, the computed values for the next iterate $\mathsf{h}^{(3)}$ (not shown) are almost all far outside the scale of the figure; many of them are in fact negative.

1. Example

As an example of this behavior, consider Kerr spacetime with dimensionless angular momentum $a \equiv J/M^2 = 0.6$. 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 slices.) Taking the black hole to be of dimensionless unit mass, the (only) apparent horizon in such a 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 all the trial horizon surfaces occurring in the example.

Figure 4(a) 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.

Figure 4(b) 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 hori-

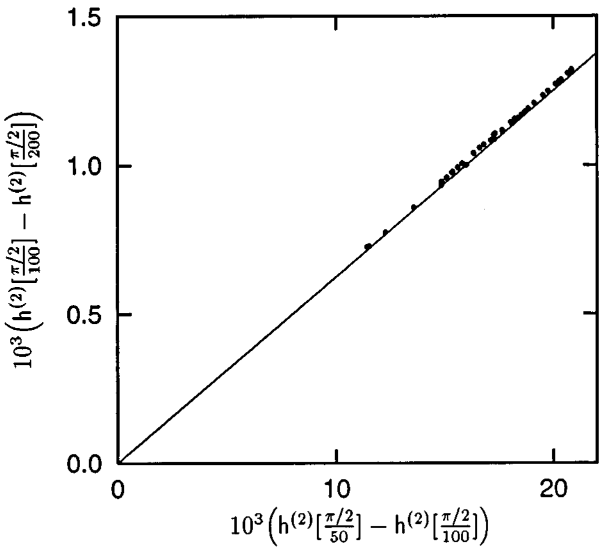

FIG. 5. This figure shows the results of a three-grid covergence test for the second-iteration Newton iterate (trial horizon surface) $\mathsf{h}^{(2)}$ plotted in Fig. 4(b). The line has slope $\frac{1}{16}$, appropriate for fourth-order convergence. (Recall that this line is not fitted to the data, but is rather an *a priori* prediction with *no* adjustable parameters.) (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 $\mathsf{h}^{(1)}$, and the very strong angular variation in both iterates $\mathsf{h}^{(1)}$ and $\mathsf{h}^{(2)}$.)

zon surfaces) $\mathsf{h}^{(k)}$ move farther and farther away from the horizon, and rapidly become more and more nonspherical.

Figure 4(c) shows the behavior of the modified Newton's method for this same high-spatial-frequency-error initial guess. Although the first iteration still moves the trial horizon surface somewhat inward from the horizon, the nonsphericity damps rapidly, and the successive iterates (trial horizon surfaces) quickly converge to the horizon.

Notice that all the intermediate iterates (trial horizon surfaces) in this example are well resolved by the finite difference grid. To verify that insufficient grid resolution is not a factor in the behavior of the horizon finder here, we have rerun all three parts of this example with several higher grid resolutions, obtaining results essentially identical to those plotted here.

More quantitatively, following our discussion of the Newton-Kantorovich method in Sec. VIII C , we have made three-grid convergence tests of each intermediate iterate (trial horizon surface) in this example. For example, Fig. 5 shows a three-grid convergence test for the Newton iterate (trial horizon surface) $\mathsf{h}^{(2)}$ plotted in Fig. 4(b), using grids with resolutions

$$\Delta\theta = \frac{\pi/2}{50} : \frac{\pi/2}{100} : \frac{\pi/2}{200}.$$

Notice that despite the iteration divergence of the Newton iteration, this iterate shows excellent fourth-order finite difference convergence. The other Newton and modified-Newton iterates (trial horizon surfaces) in our example all similarly show excellent fourth-order finite difference convergence.

We conclude that the iteration divergence of Newton's method seen in Fig. 4(b) is in fact an inherent property of the continuum Newton-Kantorovich algorithm for this initial guess and slice. Looking at the internal structure of this algorithm, we see that its only approximation is the linearization of the continuum $H(h)$ function in Eq. (40), and so the algorithm's iteration divergence must (can only) be due to nonlinearity in the continuum $H(h)$ function.

2. Horizon-perturbation survey

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 both algorithms' behavior over a range of different initial-guess-error spatial frequency spectra.

For this survey we first fix a particular horizon-finding algorithm. Suppose we are given a slice containing an apparent horizon at the continuum position h^*, and consider running the horizon finder with the generic perturbed initial guess

$$h = h^* + \sum_{\substack{m=0 \\ m \text{ even}}}^{M} c_m \cos m\theta \tag{44}$$

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

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 $\mathsf{H}(\mathsf{h})=0$ equations only by the small $\mathsf{H}(\mathsf{h})$ finite differencing error.

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

$$R_M = \begin{cases} V_0 & \text{if } M=0, \\ \dfrac{V_M}{V_{M-2}} & \text{if } M \geq 2, \end{cases} \tag{45}$$

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

3. Results and discussion

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 position of the inner grid boundary, which is at $r=1$ for these tests), the relative trends in the data should be fairly generic. These

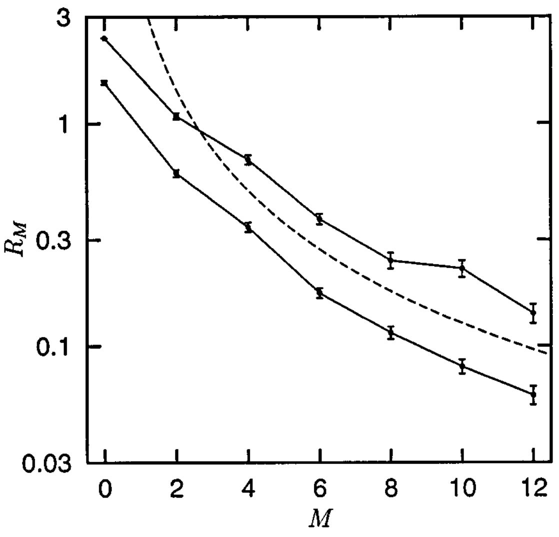

FIG. 6. This figure shows the volume ratios R_M for the horizon-perturbation survey. These measure the average radius of convergence of the horizon finder as a function of the initial-guess-error's maximum spatial frequency M. The points and solid lines show the results for the modified-Newton (upper) 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.

tests use a grid with $\Delta\theta = \frac{\pi/2}{50}$, which is adequate to resolve all the perturbed trial horizon surfaces.

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' radia of convergence still fall rapidly with increasing spatial frequency, approximately as $1/M^{3/2}$, although the rate is slightly slower for the modified-Newton than for the Newton algorithm. The radius of convergence of Newton's method falls below 0.1 ($\sim$5% of the horizon radius) by $M \gtrsim 10$, and the data suggest that the radius of convergence of the modified-Newton method would be similarly small by $M \gtrsim 18$.

Since the grid resolution is adequate, we again conclude that the small radius of convergence of Newton's method must be due to a strong high-spatial-frequency nonlinearity in the continuum $H(h)$ function. Our horizon-perturbation survey covers only a single axisymmetric initial slice and generic axisymmetric perturbations of the initial guess, but it seems unlikely that the nonlinearity would diminish for more general cases. Huq [61] has made limited tests with nonaxisymmetric spacetimes and high-spatial-frequency perturbations, and has found (poor) convergence of Newton's method similar to our results.

Although we write the continuum horizon function as $H=H(h)$, it is more accurate to write this as $H=H(g_{ij},K_{ij},h)$, since H also depends on the slice's field variables and their spatial derivatives. Examining the functional form of the $H(g_{ij},K_{ij},h)$ function in Eqs. (14) and (16), we see that H depends on the g^{ij} components in a manner broadly similar to its dependence on h. We thus conjecture that the $H(g_{ij},K_{ij},h)$ function may exhibit strong high-spatial-frequency nonlinearity in the field variables, in particular in the g^{ij} components, similar to its nonlinear dependence on h.

If this is the case, then high-spatial-frequency variations in the field variables, such as would be caused by high-frequency gravitational radiation, might well impair the convergence of Newton's method in a manner similar to high-spatial-frequency perturbations in h. Further investigation of this possibility, either by analytical study of the nonlinear structure of the $H(g_{ij},K_{ij},h)$ function, or by numerical investigations, would be very interesting. Fortunately, however, those (few) dynamic black hole spacetimes which have been explicitly computed thus far (for example, Ref. [62]) seem to contain mainly low-frequency gravitational radiation.

In general, how serious a problem is the poor high-spatial-frequency convergence of Newton's method? Given a sufficiently good initial guess, Newton's method still converges very rapidly (quadratically), and so the key question is, how good is the initial guess in practice? Two cases seem to be of particular importance: If the horizon finder is being used to update a horizon's position at each time step of a time evolution, then the previous time step's horizon position probably provides a sufficiently good initial guess for Newton's method to converge well. In contrast, if the horizon finder is being used on initial data, or in a time evolution where there is no nearby horizon in the previous time step, then significant initial-guess errors can be expected, and Newton's method may converge poorly.

X. ACCURACY OF THE HORIZON FINDER

We now consider the accuracy of the Newton's-method horizon-finding algorithm. That is, assuming the Newton or modified-Newton iteration converges, how close is the horizon finder's final numerically computed horizon position to the (a) true continuum horizon position h^*?

The horizon finder computes Newton or modified-Newton iterates (trial horizon surfaces) $\mathsf{h}^{(k)}$ 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. Sec. VIII A), there is little extra cost in using a very strict convergence criterion, i.e., in solving the discrete $\mathsf{H}(\mathsf{h})=0$ equations to very high accuracy. In our horizon finder we typically require $\|\mathsf{H}(\mathsf{h}^{(p)})\|_\infty < 10^{-10}$.

We denote the exact solution of the discrete $\mathsf{H}(\mathsf{h})=0$ equations by h^*. Given that $\|\mathsf{H}(\mathsf{h}^{(p)})\|$ is reasonably small, then from standard matrix-perturbation theory (see, for example, Refs. [63,64]), $\|\mathsf{h}^{(p)}-\mathsf{h}^*\| \lesssim \kappa \|\mathsf{H}(\mathsf{h}^{(p)})\|$, where κ is the condition number of the (presumably nonsingular) Jacobian matrix $\mathbf{J}[\mathsf{H}(\mathsf{h})]$ at the horizon position.

If we take the convergence tolerance to be strict enough for $\|\mathsf{h}^{(p)}-\mathsf{h}^*\|$ to be negligible, then the overall accuracy of the horizon finder, i.e., the external error $\|\mathsf{h}^{(p)}-h^*\|$ in the computed horizon position, is thus limited only by the closeness with which the discrete $\mathsf{H}(\mathsf{h})=0$ equations approximate the continuum $H(h)=0$ equations, i.e., by the accuracy of the $\mathsf{H}(\mathsf{h})$ 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 spatial coordinate transformation

$$r \to r + \frac{b^2}{b^2 + r^2}(a_2 \cos 2\theta + a_4 \cos 4\theta) \qquad (46a)$$

to the slice, where the parameters are given by

$$b = 5, \quad a_2 = 0.75, \quad a_4 = 0.05. \qquad (46b)$$

As shown in Fig. 7(a), in the transformed coordinates this gives a strongly nonspherical ''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. (The convergence took nine iterations, but would have taken only six iterations in the absence of our 10% restriction on the relative change in any component of h in a single outer iteration; cf. Sec. VIII B.) Figure 7(a) shows the initial guess and the final numerically computed horizon position.

Figure 7(b) shows the results of a two-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 fourth-order convergence. Moreover, the numerically computed horizon positions are very accurate, with $\|\mathsf{h}^{(p)} - h^*\| \sim 10^{-5}(10^{-6})$ for a grid resolution of $\Delta\theta = \frac{\pi/2}{50}$ $(\frac{\pi/2}{100})$. Errors of this magnitude are typical of what we find for Newton's-method horizon finding using fourth-order finite differencing, as long as the grid adequately resolves the horizon shape.

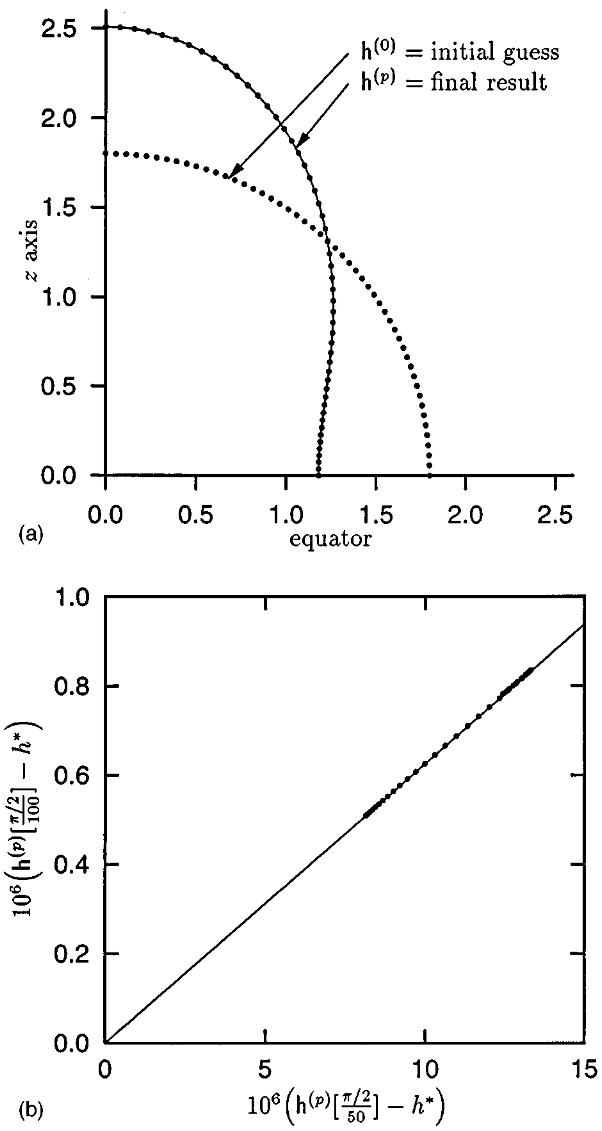

FIG. 7. This figure illustrates the accuracy of our horizon finder for a test case where the horizon's coordinate shape is strongly nonspherical. The figure is plotted using the transformed radial coordinate defined by Eq. (46). Part (a) of the figure shows the ''peanut-shaped'' true continuum horizon position h^*, plotted as a solid line, and the initial guess $\mathsf{h}^{(0)}$ and the final numerically computed horizon position $\mathsf{h}^{(p)}$, plotted with dots at the grid points. At this scale the numerically computed horizon position $\mathsf{h}^{(p)}$ is indistinguishable from the true continuum position h^*. Part (b) of the figure shows the results of a two-grid convergence test for the numerically computed horizon position $\mathsf{h}^{(p)}$. The line has slope $\frac{1}{16}$, appropriate for fourth-order convergence. (Recall again that this line is not fitted to the data, but is rather an *a priori* prediction with *no* adjustable parameters.)

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 is 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. [8–11]. 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. [10], 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 known for finding or recognizing outermost apparent horizons in nonspherical spacetimes. (For spherical spacetimes, a one-dimensional search on $H(r)$ suffices.) If started with a very large two-sphere as the initial guess, the curvature flow method might well converge to the outermost horizon in the slice, but as mentioned in Sec. 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 Sec. IV, including the Newton's-method one, we know of no better method for locating or recognizing outermost horizons than trying the local horizon finder with a number of different initial guesses near the suspected position of an outermost horizon. If this method succeeds, it locates a horizon, but there is still no assurance that this horizon is the outermost one in the slice. Moreover, if all the local-horizon-finding trials fail, this may mean that there is no horizon in the vicinity of the initial guesses, or it may only mean that a horizon is present nearby but the method failed to converge to it. It is also not clear how many local-horizon-finding trials should be made, or just how their 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

We find Newton's method to be an excellent horizon-finding algorithm: It handles fully generic slices, it is fairly easy to implement, it is very efficient, it is generally robust in its convergence, and it is very accurate. These properties are all well known, and Newton's method is widely used for horizon finding. In this paper we focus on two key aspects of this algorithm: the computation of the Jacobian matrix and the algorithm's global convergence behavior.

Traditionally, the Newton's-method Jacobian matrix is computed by a numerical perturbation technique. In this paper we present a much more efficient ''symbolic differentiation'' technique. Conceptually, this entails differentiating the actual finite difference equations used to compute the discrete horizon function $\mathsf{H}(\mathsf{h})$. However, provided the finite differencing scheme commutes with linearization, the computation can instead be done by first differentiating the continuum horizon function $H(h)$ and then finite differencing. (This is essentially just the ''Jacobian part'' of the Newton-Kantorovich method for solving nonlinear PDEs.)

In our axisymmetric-spacetime (two-dimensional) numerical code, this method is about a factor of 5 faster than any other Jacobian computation method. In fact, the Jacobian computation using this method is only 1.5–2 times more expensive than the simple evaluation of $\mathsf{H}(\mathsf{h})$. We expect the symbolic differentiation method's relative advantage over other Jacobian computation methods to be roughly similar for other axisymmetric-spacetime (two-dimensional) codes, and an additional factor of $\sim 3-5$ larger for fully-general-spacetime (three-dimensional) codes.

We had previously suggested [10] 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 turns out not to be the case: We computed all the symbolic differentiation Jacobian coefficients for our horizon finder by hand in only a few few pages of algebra. Some of the coefficients are fairly complicated, but no more so than many other computations in $3+1$ numerical relativity.

We find the actual programming of the symbolic differentiation Jacobian computation to be only moderately more difficult than that of a numerical perturbation computation. In order to be confident of the correctness of a symbolic differentiation Jacobian computation, we feel that it is highly desirable to program an independent numerical perturbation method and make an end-to-end comparison of the resulting Jacobian matrices. The comparison Jacobian computation may be programmed for simplicity at the expense of efficiency, and so it need not add much to the overall symbolic-differentiation implementation effort.

Turning now to the convergence behavior of Newton's method, we find that as 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 (good) 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 small (poor) radius of convergence; i.e., it may fail to converge even when the initial guess is quite close to the true horizon position. In this context there is 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.

Using a Monte Carlo survey of initial-guess-error Fourier-coefficient space, we find that the radius of convergence for Newton's method falls rapidly with increasing spatial frequency, approximately as $1/m^{3/2}$. A simple ''line-search'' modification of Newton's method roughly doubles the horizon finder's radius of convergence, and slightly slows the rate of decline with spatial frequency. Using a robust nonlinear-algebraic-equations code to solve the discrete $\mathsf{H}(\mathsf{h})=0$ equations would probably give some further improvement, but we doubt that it would change the overall trend.

Using quantitative 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. If so, then the presence of high-frequency gravitational radiation might well also impair the convergence of Newton's method, and possibly other horizon-finding methods as well. Further investigation of this possibility would be very interesting.

Fortunately, if the horizon finder is being used to update a horizon's position at each time step of a time evolution, then the previous time step's horizon position probably provides a sufficiently good initial guess for Newton's method to converge well.

Provided it converges, the Newton's-method algorithm for horizon finding is potentially very accurate, in practice limited only by the accuracy of the $\mathsf{H}(\mathsf{h})$ finite differencing scheme. Using fourth-order finite differencing, we demonstrate that the error in the numerically computed horizon position, i.e., the deviation of h from the true continuum horizon position, shows the expected $O((\Delta\theta)^4)$ scaling with grid resolution $\Delta\theta$, and is typically $\sim 10^{-5}$ (10^{-6}) for a grid resolution of $\Delta\theta = \frac{\pi/2}{50}$ $(\frac{\pi/2}{100})$.

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 various 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. We thank J. Wolfgang for major assistance in setting up computer facilities, and G. Rodgers for financial support.

APPENDIX A: SYMBOLIC-DIFFERENTIATION JACOBIAN COEFFICIENTS

In this appendix we tabulate all the nonzero symbolic differentiation Jacobian coefficients for ${}^{(3)}H(h)$ and its subfunctions. These are used in the 3d.sd.1s and 3d.sd2.2s Jacobian-computation methods. All the coefficients are obtained by straightforward, if somewhat tedious, linearizations in the manner of Eq. (24), starting from the defining equations noted.

For ${}^{(3)}H(h)$, starting from Eqs. (14) and (16), the coefficients are

$$\frac{\partial^{(3)}H}{\partial(\partial_x h)} = \frac{1}{D^{3/2}}\frac{\partial A}{\partial(\partial_x h)} + \frac{1}{D^{1/2}}\frac{\partial B}{\partial(\partial_x h)} + \frac{1}{D}\frac{\partial C}{\partial(\partial_x h)} - \left(\frac{3}{2}\frac{A}{D^{5/2}} + \frac{1}{2}\frac{B}{D^{3/2}} + \frac{C}{D^2}\right)\frac{\partial D}{\partial(\partial_x h)}, \tag{A1a}$$

$$\frac{\partial^{(3)}H}{\partial(\partial_{xy} h)} = \frac{1}{D^{3/2}}\frac{\partial A}{\partial(\partial_{xy} h)} + \frac{1}{D^{1/2}}\frac{\partial B}{\partial(\partial_{xy} h)}, \tag{A1b}$$

where

$$\frac{\partial A}{\partial(\partial_x h)} = -[g^{ux}(g^{vr} - g^{vw}\partial_w h) + g^{vx}(g^{ur} - g^{uw}\partial_w h)]\partial_{uv}h + \tfrac{1}{2}g^{ix}[\partial_i g^{rr} - 2(\partial_i g^{ru})\partial_u h + (\partial_i g^{uv})(\partial_u h)(\partial_v h)] + (g^{ir} - g^{iu}\partial_u h)[\partial_i g^{xr} - (\partial_i g^{xv})\partial_v h], \tag{A1c}$$

$$\frac{\partial B}{\partial(\partial_x h)} = -\partial_i g^{ix} - (\partial_i \ln\sqrt{g})g^{ix}, \tag{A1d}$$

$$\frac{\partial C}{\partial(\partial_x h)} = -2(K^{xr} - K^{xu}\partial_u h), \tag{A1e}$$

$$\frac{\partial D}{\partial(\partial_x h)} = -2(g^{xr} - g^{xu}\partial_u h), \tag{A1f}$$

$$\frac{\partial A}{\partial(\partial_{xy} h)} = (g^{xr} - g^{xu}\partial_u h)(g^{yr} - g^{yu}\partial_u h), \tag{A1g}$$

$$\frac{\partial B}{\partial(\partial_{xy} h)} = -g^{xy}. \tag{A1h}$$

For $s_i(h)$, starting from Eq. (3), the coefficients are

$$\frac{\partial s_u}{\partial(\partial_x h)} = \begin{cases} -1 & \text{if } u = x, \\ 0 & \text{otherwise.}\end{cases} \tag{A2}$$

For $n^i(h)$, starting from Eq. (6), the coefficients are

$$\frac{\partial n^i}{\partial(\partial_x h)} = -\frac{g^{ix}}{D^{1/2}} + \frac{(g^{ir} - g^{iu}\partial_u h)(g^{xr} - g^{xv}\partial_v h)}{D^{3/2}}. \tag{A3}$$

For $n^i(s_j)$, starting from Eq. (5), the coefficients are

$$\frac{\partial n^i}{\partial s_u} = \frac{g^{iu}}{(g^{kl}s_k s_l)^{1/2}} - \frac{(g^{ik}s_k)(g^{ul}s_l)}{(g^{kl}s_k s_l)^{3/2}}. \tag{A4}$$

For ${}^{(3)}H(s_i)$, starting from Eqs. (14) and (15), the coefficients are

$$\frac{\partial^{(3)}H}{\partial s_x} = \frac{1}{D^{3/2}}\frac{\partial A}{\partial s_x} + \frac{1}{D^{1/2}}\frac{\partial B}{\partial s_x} + \frac{1}{D}\frac{\partial C}{\partial s_x} - \left(\frac{3}{2}\frac{A}{D^{5/2}} + \frac{1}{2}\frac{B}{D^{3/2}} + \frac{C}{D^2}\right)\frac{\partial D}{\partial s_x}, \tag{A5a}$$

$$\frac{\partial^{(3)}H}{\partial(\partial_x s_y)} = \frac{1}{D^{3/2}}\frac{\partial A}{\partial(\partial_x s_y)} + \frac{1}{D^{1/2}}\frac{\partial B}{\partial(\partial_x s_y)}, \tag{A5b}$$

where

$$\frac{\partial A}{\partial s_x} = -[g^{ix}(g^{jk}s_k) + g^{jx}(g^{ik}s_k)]\partial_i s_j - \tfrac{1}{2}g^{ix}[(\partial_i g^{kl})s_k s_l] - (g^{ij}s_j)[(\partial_i g^{xk})s_k], \tag{A5c}$$

$$\frac{\partial B}{\partial s_x} = -\partial_i g^{ix} + (\partial_i \ln\sqrt{g})g^{ix}, \tag{A5d}$$

$$\frac{\partial C}{\partial s_x} = 2K^{xi}s_i, \tag{A5e}$$

$$\frac{\partial D}{\partial s_x} = 2g^{xi}s_i, \tag{A5f}$$

$$\frac{\partial A}{\partial(\partial_x s_y)} = -(g^{xk}s_k)(g^{yl}s_l), \tag{A5g}$$

$$\frac{\partial B}{\partial(\partial_x s_y)} = g^{xy}. \tag{A5h}$$

For ${}^{(3)}H(n^i)$, starting from Eq. (8), the coefficients are

$$\frac{\partial{}^{(3)}H}{\partial n^x} = \partial_x \ln\sqrt{g} + 2K_{xi}n^i, \tag{A6a}$$

$$\frac{\partial{}^{(3)}H}{\partial(\partial_x n^y)} = \begin{cases} 1 & \text{if } x=y, \\ 0 & \text{otherwise.} \end{cases} \tag{A6b}$$

APPENDIX B: 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, as summarized in Table I. It is part of a larger $3+1$ code under development, designed to time evolve 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. [8–11]. The code uses fourth-order centered finite differencing (five-point molecules) for finite differencing, on a two-dimensional polar-spherical-coordinate grid. (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 that for the horizon function and Jacobian computations, from a high-level tensor-differential-operator specification 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 is 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 about 1K lines of Awk. The code for the horizon finder itself is about 6K lines of C and 2K lines of Maple, but a large part of this is due to its supporting many different combinations of finite differencing schemes and horizon-function and Jacobian computation methods. We estimate that an implementation supporting only a single differencing scheme and horizon-function and Jacobian-computation method, supplemented by a not-optimized-for-efficiency independent Jacobian computation for debugging purposes (cf. Sec. VI D), would be a factor of ~ 4 smaller.

The code takes the metric, extrinsic curvature, and other $3+1$ field tensors to be algebraically 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. [10], for the subset of the slice containing the code's (two-dimensional) grid, this hybrid coordinate system combines the convenient topology of polar spherical coordinates with the singularity-free nature of Cartesian coordinates.

For present purposes, the key consequence of this z-axis-handling method is that in this work we have made no effort to avoid expressions which would be singular on the z axis if polar spherical coordinates were used as a tensor basis. We have not investigated this case in detail, but we suspect such singularities would be widespread.

APPENDIX C: OUR SPARSE-JACOBIAN STORAGE SCHEME

As mentioned in Sec. VI D, all the Jacobian matrices involved in horizon finding are highly sparse, and for reasonable efficiency this sparsity *must* be exploited in storing and computing the Jacobians. In this appendix we briefly describe our sparse-Jacobian storage scheme. This scheme stores the Jacobian by rows, and is applicable to all of the Jacobian matrices which arise in our horizon-finding algorithm.

We consider first the storage of $\mathbf{J}[{}^{(2)}\mathsf{H}(\mathsf{h})]$. Which elements in a specified row I of this Jacobian are nonzero? From the basic definition Eq. (26), we see that the nonzero elements J are precisely those where ${}^{(2)}\mathsf{H}_{\mathrm{I}}$ depends on h_{J}, i.e., those for which h_{J} enters into the computation of ${}^{(2)}\mathsf{H}_{\mathrm{I}}$. That is, for a one- (two-) stage ${}^{(3)}\mathsf{H}(\mathsf{h})$ computation, the nonzero-Jacobian J values are precisely those within one (two) molecule radia of I. This makes it easy to store the Jacobian: For each grid point I, we simply store a molecule-sized (twice-molecule-sized) array of Jacobian elements.

In practice, for an axisymmetric-spacetime (two-dimensional) code, where I and J are both one-dimensional (θ) grid indices and the Jacobian is a band matrix, we would store the Jacobian as a two-dimensional array with indices I and $\mathrm{J}-\mathrm{I}$. For a fully-general-spacetime (three-dimensional) code, where I and J are both two-dimensional (θ and ϕ) grid indices, we would store the Jacobian as a four-dimensional array with indices I_θ, I_ϕ, $\mathrm{J}_\theta-\mathrm{I}_\theta$, and $\mathrm{J}_\phi-\mathrm{I}_\phi$, where we temporarily use subscripts for coordinate components, and where for pedagogical simplicity we ignore the artificial grid boundaries at $\theta=\{0,\pi\}$ and $\phi=\{0, 2\pi\}$.

A similar storage scheme may be used for more complicated Jacobians. For example, consider the storage of $\mathbf{J}[{}^{(3)}\mathsf{H}(\mathsf{h})]$. Here I is a three-dimensional grid point index for ${}^{(3)}\mathsf{H}$, while J is a two-dimensional grid point index for h. For a one- (two-) stage ${}^{(3)}\mathsf{H}(\mathsf{h})$ computation, the nonzero Jacobian elements in a specified Jacobian row I are now precisely those J within one (two) angular molecule radia of the angular components of I. Thus for an axisymmetric-spacetime (two-dimensional) code we would store this Jacobian as a three-dimensional array with indices I_r, I_θ, and $\mathrm{J}_\theta-\mathrm{I}_\theta$, while for a fully-general-spacetime (three-dimensional) code we would store the Jacobian as a five-dimensional array with indices I_r,I_θ, I_ϕ, $\mathrm{J}_\theta-\mathrm{I}_\theta$, and $\mathrm{J}_\phi-\mathrm{I}_\phi$.

Notice that with this storage scheme the Jacobian's structure, i.e., the location of its nonzero elements, is stored implicitly. This makes this scheme considerably more efficient in both space and time than generic ''sparse matrix'' storage schemes (for example, those of Refs. [65,66]), which invariably require the storage of large integer or pointer arrays to record a sparse matrix's structure.

APPENDIX D: DETAILS OF OUR HORIZON-PERTURBATION SURVEY

In this appendix we describe our Monte Carlo horizon-perturbation survey (cf. Sec. 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, hyperparallelepipeds) 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 in each dimension, to provide a safety margin against missing disconnected ''islands'' or fractal zones near the boundary of the convergence region. [These are quite plausible; 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, and 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 Fig. 6 required 15 000 trials each.) It is this effect which ultimately limits the maximum value of M attainable in practice by a survey of this type.

[1] S. W. Hawking, in *Black Holes*, Proceedings of the 23rd Les Houches Summer School of Theoretical Physics, Les Houches, 1972, edited by C. DeWitt and B. S. DeWitt (Gordon and Breach, New York, 1973), pp. 1–56.

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

[3] S. W. Hawking and G. F. R. Ellis, *The Large Scale Structure of Space-Time* [2], p. 319.

[4] P. Anninos, D. Bernstein, S. Brandt, J. Libson, J. Massó, E. Seidel, L. L. Smarr, W.-M. Suen, and P. Walker, Phys. Rev. Lett. **74**, 630 (1995).

[5] J. Libson, J. Massó, E. Seidel, W.-M. Suen, and P. Walker, Phys. Rev. D **53**, 4335 (1996).

[6] R. M. Wald and V. Iyer, Phys. Rev. D **44**, R3719 (1991).

[7] P. Anninos, D. Bernstein, S. Brandt, D. Hobill, E. Seidel, and L. L. Smarr, Phys. Rev. D **50**, 3801 (1984).

[8] J. Thornburg, talk at the CITA Workshop on Numerical Relativity, Toronto, Canada, 1991 (unpublished).

[9] E. Seidel and W.-M. Suen, Phys. Rev. Lett. **69**, 1845 (1992).

[10] J. Thornburg, Ph.D. thesis, University of British Columbia, 1993.

[11] P. Anninos, G. Daues, J. Massó, E. Seidel, and W.-M. Suen, Phys. Rev. D **51**, 5562 (1995).

[12] C. W. Misner, K. S. Thorne, and J. A. Wheeler, *Gravitation* (Freeman, San Francisco, 1973).

[13] R. M. Wald, *General Relativity* (University of Chicago Press, Chicago, 1984).

[14] R. Arnowitt, S. Deser, and C. W. Misner, in *Gravitation: An Introduction to Current Research*, edited by L. Witten (Wiley, New York, 1962), pp. 227–265.

[15] J. W. York, Jr., in *Sources of Gravitational Radiation*, edited by L. L. Smarr (Cambridge University Press, Cambridge, England, 1979), pp. 83–126.

[16] J. W. York, Jr., in *Gravitational Radiation*, edited by N. Deruelle and T. Piran (North-Holland, Amsterdam, 1983), pp. 175–201.

[17] A. Čadež, Ph.D. thesis, University of North Carolina at Chapel Hill, 1971.

[18] J. Thornburg, Class. Quantum Grav. **4**, 1119 (1987).

[19] J. W. York, Jr., in *Frontiers in Numerical Relativity*, edited by C. R. Evans, L. S. Finn, and D. W. Hobill (Cambridge University Press, London, 1989), pp. 89–109.

[20] M. R. Schroeder, *Number Theory in Science and Communication* (Springer-Verlag, Berlin, 1986), p. 108.

[21] T. Nakamura, K. Oohara, and Y. Kojima, Prog. Theor. Phys. Suppl. **90**, 1 (1987).

[22] L. I. Petrich, S. L. Shapiro, and S. A. Teukolsky, Phys. Rev. D **31**, 2459 (1985).

[23] M. W. Choptuik, Ph.D. thesis, University of British Columbia, 1986.

[24] P. Bizon, E. Malec, and N. Ò Murchadha, Phys. Rev. Lett. **61**, 1147 (1988).

[25] A. Čadež, Ann. Phys. (N.Y.) **83**, 449 (1974).

[26] P. G. Dykema, Ph.D. thesis, University of Texas at Austin, 1980.

[27] A. M. Abrahams and C. R. Evans, Phys. Rev. D **46**, R4117 (1992).

[28] N. T. Bishop, Gen. Relativ. Gravit. **14**, 717 (1982).

[29] N. T. Bishop, Gen. Relativ. Gravit. **16**, 589 (1984).

[30] S. L. Shapiro and S. A. Teukolsky, Phys. Rev. D **45**, 2739 (1992).

[31] A. M. Abrahams, K. R. Heiderich, S. L. Shapiro, and S. A. Teukolsky, Phys. Rev. D **46**, 2452 (1992).

[32] K. P. Tod, Class. Quantum Grav. **8**, L115 (1991).

[33] D. Bernstein, National Center for Supercomputing Applications report, 1993 (unpublished).

[34] D. R. Brill and R. W. Lindquist, Phys. Rev. **131**, 471 (1963).

[35] K. R. Eppley, Phys. Rev. D **16**, 1609 (1977).

[36] J. Libson, J. Massó, E. Seidel, and W-.M. Suen, in *General Relativity*, Proceedings of the Seventh Marcel Grossmann Meeting, Stanford, California, 1995, edited by R. Ruffini and M. Keiser (World Scientific, Singapore, 1995).

[37] T. Nakamura, Y. Kojima, and K. Oohara, Phys. Lett. **106A**, 235 (1984).

[38] K. Oohara, T. Nakamura, and Y. Kojima, Phys. Lett. **107A**, 452 (1985).

[39] K. Oohara, in *Gravitational Collapse and Relativity*, edited by H. Sato and T. Nakamura (World Scientific, Singapore, 1986), pp. 313–319.

[40] A. J. Kemball and N. T. Bishop, Class. Quantum Grav. **8**, 1361 (1991).

[41] D. M. Eardley, report (unpublished), as discussed (p. 135, p. 149) in K. R. Eppley, Ph.D. thesis, Princeton University, 1975.

[42] G. B. Cook, Ph.D thesis, University of North Carolina at Chapel Hill, 1990.

[43] G. B. Cook and J. W. York, Jr., Phys. Rev. D **41**, 1077 (1990).

[44] G. B. Cook and A. M. Abrahams, Phys. Rev. D **46**, 702 (1992).

[45] M. Huq, talk at the Pennsylvania State Numerical Relativity Workshop, University Park, Pennsylvania, 1993 (unpublished).

[46] M. Ciment and S. H. Leventhal, Math. Comput. **29**, 985 (1975).

[47] R. S. Hirsh, J. Comput. Phys. **19**, 90 (1975).

[48] A. R. Curtis and J. K. Reid, J. Inst. Math. Appl. **13**, 121 (1974).

[49] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis* (Springer-Verlag, New York, 1980), Sec. 5.3.

[50] J. Thornburg, [10], Sec. 4.4.

[51] M. W. Choptuik, Phys. Rev. D **44**, 3124 (1991).

[52] M. W. Choptuik, D. S. Goldwirth, and T. Piran, Class. Quantum Grav. **9**, 721 (1992).

[53] R. E. Bank and D. J. Rose, SIAM J. Numer. Anal. **17**, 806 (1980).

[54] R. E. Bank and D. J. Rose, Numer. Math. **37**, 279 (1981).

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

[56] B. S. Garbow, K. E. Hillstrom, and J. J. Moré, ''MINPACK — A software package for Nonlinear Equations and Nonlinear Least Squares Problems,'' available from the NETLIB on-line software repository.

[57] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran*, 2nd ed. (Cambridge University Press, New York, 1992).

[58] U. Nowak and L. Weimann, '' GIANT — A Software Package for the Numerical Solution of Very Large Systems of Highly Nonlinear Equations,'' Konrad-Zuse-Zentrum für Informationstechnik Berlin Report No. TR-90-11, 1990 (unpublished).

[59] U. Nowak and L. Weimann, ''A Family of Newton Codes for Systems of Highly Nonlinear Equations,'' Konrad-Zuse-Zentrum für Informationstechnik Berlin Report No. TR-91-10, 1991 (unpublished).

[60] J. P. Boyd, *Chebyshev & Fourier Spectral Methods*, Lecture Notes in Engineering Vol. 49 (Springer-Verlag, Berlin, 1989), Appendix C.

[61] M. Huq (private communication).

[62] P. Anninos, D. Hobill, E. Seidel, W.-M. Suen, and L. L. Smarr, Phys. Rev. D **52,** 2044 (1995).

[63] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, LINPACK Users' Guide (SIAM, Philadelphia, 1979), pp. I-7 – I-12.

[64] G. H. Golub and C. Van Loan, *Matrix Computations*, 2nd ed. (Johns Hopkins University Press, Baltimore, 1989), pp. 79–80.

[65] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1981).

[66] ILUCG: A computer code implementing Kernshaw's Incomplete LU-Decomposition — Conjugate Gradient iteration, written by P. Madderom and supplied to the present author in 1985 by T. Nicol.