Finding Apparent and Event Horizons

Jonathan Thornburg <jthorn@aei.mpg.de>
Peter Diener <diener@aei.mpg.de>

Max-Planck-Institut für Gravitationsphysik
Albert-Einstein-Institut
Golm, Germany
Why Find **Event** and **Apparent** Horizons?

simulate a *strongly relativistic system* ⇒

- may want to know about BHs to understand physics better (masses, spins, Gaussian curvature, isolated horizons, ...)
- may want to excise singularities from the grid
- may want to “steer” coordinates to “track” moving BHs

study initial data families

study cosmic censorship

“because they’re there” :)
Event and Apparent Horizons

**event horizon** (true definition of BH surface):
boundary between those events from which future-pointing null geodesics can reach $\mathcal{I}^+$, and those events from which they can’t
- defined non-locally in time: must know full spacetime to compute
- (continuous) null surface in spacetime
- generically smooth, but may have non-differentiable caustics

**apparent horizon** (⊂ event horizon, in practice ≈)
outermost closed spacelike **2-surface** whose future-directed outgoing null geodesics have zero expansion
- defined locally in time: can compute while evolution is in progress
- generically smooth in space (elliptic PDE)
- may jump discontinuously or appear/disappear during evolution
Finding Apparent Horizons
Jonathan Thornburg, gr-qc very soon, CACTUS thorn AHFINDERDIRECT

AHs are very good BH diagnostics
⇒ want to find AHs at each time step of a numerical evolution
typical (3D) evolutions today
  • last for $10^3 - 10^5$ time steps ⇒ need fast 3D AH finder
  • use Cartesian-topology grids

spherical symmetry or axisymmetry ⇒ ∃ fast AH finders
3D grid (no symmetry) ⇒ existing AH finders are generally
  very slow (∼ 1–10 minutes/horizon)

basic goal of this work:
a fast AH finder for 3D Cartesian grids
Algorithms for Finding Apparent Horizons

1D (spherical symmetry)
- zero-finding

2D (axisymmetry)
- shooting

3D (generic spacetime) or 2D (axisymmetry)
\[
\{ \begin{array}{l}
\quad \text{curvature/fast/level flow} \\
\quad \text{minimization} \\
\quad \text{elliptic PDE}
\end{array} \} \times \{ \begin{array}{l}
\quad \text{finite difference} \\
\quad \text{spectral}
\end{array} \}
\]
Parameterizing the Apparent Horizon Surface

assume we’re given a spacelike 3-slice $\Sigma$

choose local coordinate origin inside the horizon

assume horizon is a Strahlkörper
(“ray body”, or more commonly “star-shaped region”),
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.

parameterize horizon shape as $r = h(\text{angle})$
for some single-valued function $h : S^2 \rightarrow \mathbb{R}^+$
Solving the Apparent Horizon Equation

given \( r = h(\text{angle}) \) parameterization, AH equation
\[ \Theta \equiv \nabla_i n^i + K_{ij} n^i n^j - K = 0 \]
becomes a nonlinear elliptic PDE for \( h \) on \( S^2 \),
\[ \Theta(h, \partial_u h, \partial_{uv} h) = 0 \]
with coefficients which are algebraic functions of \( g_{ij}, \partial_k g_{ij}, K_{ij} \)

finite difference \( \Theta(h, \partial_u h, \partial_{uv} h) = 0 \) in angle on \( S^2 \)
\( (N_{\text{ang}} \text{ angular grid points on } S^2, \text{ typically } 300 \lesssim N_{\text{ang}} \lesssim 3000) \)
\( \Rightarrow \) set of \( N_{\text{ang}} \) nonlinear algebraic equations \( \{\Theta_k = 0\} \)

solve for \( \{h_K\} \) via **Newton’s method in \( N_{\text{ang}} \) dimensions**
\( \equiv \) Newton-Kantorovich linearization of original nonlinear PDE,
\( \) followed by finite differencing each of the resulting linear PDEs
Computing Derivatives Inside the Interpolation

geometry fields in $\Theta(h, \partial_u h, \partial_{uv} h; g_{ij}, \partial_k g_{ij}, K_{ij})$

must be evaluated at trial horizon surface

but we only know $g_{ij}, K_{ij}$ at (3D) Cartesian grid points

$\Rightarrow$ must interpolate to find $g_{ij}, \partial_k g_{ij}, K_{ij}$ at trial horizon surface

compute $\partial_k g_{ij}$ on the entire 3D grid, then interpolate $\Rightarrow$ very slow

instead, compute $\partial_k g_{ij}$ inside the interpolator

$\Rightarrow$ only compute $\partial_k g_{ij}$ at interpolation points, not on entire 3D grid

$\Rightarrow$ much faster

algorithm to interpolate a grid function $f$ and its derivative:

**input:** grid \{x_K\}, grid function $f$, interpolation point $x_{\text{interp}}$

- choose a molecule $M$ of grid points which is near $x_{\text{interp}}$
- compute interpolating function $I$ with $I(x_K) = f_K \ \forall K \in M$
- evaluate $I(x_{\text{interp}})$ and its derivative
Lagrange versus Hermite Interpolation

Lagrange polynomial interpolation

⇒ $g_{ij}, K_{ij}$ are only $C^0$, and $\partial_k g_{ij}$ has a jump discontinuity
   at each position where the interpolator switches
   the set of 3D grid points it use for the interpolation

instead, use **Hermite interpolation** to get better smoothness

⇒ $g_{ij}, K_{ij}$ are $C^1$ everywhere, $\partial_k g_{ij}$ is $C^0$ everywhere

Example:
1D cubic interpolation
grid spacing $\Delta x = 0.1$
$f(x) = \exp[\sin(x)]$
Computing the Jacobian Matrix

Newton’s method ⇒ need Jacobian matrix \( J_{ij} \equiv \partial \Theta_i / \partial h_j \)
\( (N_{\text{ang}} \times N_{\text{ang}} \text{ sparse matrix}) \)

numerical perturbation is very slow for large \( N_{\text{ang}} \)

instead, use **Symbolic Differentiation:**
compute \( J \) directly from \( \partial u, \partial uv \) molecule coefficients ⇒ very fast

algorithm for **Jacobian matrix** \( J \) of differential operator \( P = P(Q) \):

if \( P \) is **linear**, discretely approximated by molecule \( M \):

then \( P_i \equiv \sum_M M_M Q_{i+M} \Rightarrow J_{ij} \equiv \frac{\partial P_i}{\partial Q_j} = \begin{cases} M_{j-i} & \text{if } j-i \in M \\ 0 & \text{otherwise} \end{cases} \)

if \( P \) is **nonlinear**, Newton-Kantorovich–linearize it
⇒ same formula with \( M = \frac{\partial P}{\partial Q} I + \frac{\partial P}{\partial \partial_i Q} d_i + \frac{\partial P}{\partial \partial_{ij} Q} d_{ij} \)
Multiple Grid Patches

±z, ±x, ±y patches cover $S^2$ without coordinate singularities
Sample Results – 2BH Misner Evolution

2BH Misner initial data, $\mu = 2.0 \Rightarrow m_{\text{ADM}} = 1.27$
BSSN system, $1 + \log$ slicing, variant $\Gamma$-freezing shift
mean time to find all horizons in a slice (not using axisymmetry):
AHFINDER (flow) 55 seconds, AHFINDERDIRECT 5.2 seconds

Apparent Horizon Areas for Misner $\mu=2.0$ BH Collision

[[movie]]
Finding Event Horizons
Peter Diener, gr-qc/0305039, CACTUS thorn EHFinder

event horizon (true definition of BH surface):
boundary between those events where future-pointing null geodesics
can reach infinity, and those events where they can’t

- defined non-locally in time: must know full spacetime to compute
- (continuous) null surface in spacetime
- generically smooth, but may have non-differentiable caustics

non-locality ⇒ run an evolution (writing out data files of 4-metric) then post-process to find EH
Integrating Null Geodesics

obvious EH-finding algorithm:
integrate null geodesics in all directions from each event
⇒ very slow

better idea:
consider radial
null geodesics near EH:
they diverge exponentially
⇒ they converge exponentially
if integrated backwards in time
Integrating Null Surfaces

problems with integrating null geodesics:

- geodesics can drift sideways
- integrate geodesics ⇒ need numerical derivatives of $g_{ab}$

⇒ better to integrate null surfaces backwards in time:
  ... no sideways drift (a surface can only move $\perp$ to itself)
  ... no need for numerical derivatives of $g_{ab}$

algorithm summary:

- evolve spacetime to almost-stationary end state,
  writing out 4-metric on each $t =$ constant slice during evolution
- in final slice, choose trial 2-surfaces inside/outside EH
- evolve 2-surfaces backwards in time as null surfaces
  ⇒ converge exponentially to EH
How to Integrate Null Surfaces

define scalar "level set function" \( f(t, x^i) \)

choose \( f \) so surface is zero set \( \{ x^a | f(x^a) = 0 \} \)
(in practice \( f = \) signed distance inside/outside horizon)

null surface \( \Rightarrow g^{ab}(\partial_a f)(\partial_b f) = 0 \)

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

gradients in \( f \) tend to steepen in (backwards) evolution
\( \Rightarrow \) numerical problems
\( \Rightarrow \) periodically reinitialize to signed-distance function

note this algorithm handles topology changes ok
(given clever implementation and finite differencing tricks)

this algorithm is (or can be if careful) vary accurate
Sample Results – Kerr Spacetime

Kerr spacetime, \( a \equiv J/m^2 = 0.8, \Delta x = 0.2m, 0.1m, 0.05m, 0.025m \)
\( \Rightarrow O((\Delta x)^2) \) convergence, typically \( \lesssim 10^{-3}m \)
Sample Results – 3BH Brill-Lindquist Evolution

3BH Brill-Lindquist initial data, $m_1 = m_2 = m_3 = 0.5$ ($m_{\text{ADM}} = 1.5$) BSSN system, $K$-freezing slicing, $\Gamma$-freezing shift

$\Delta x = 0.2m$ and $0.1m$ resolution

![3 black hole collision (Brill–Lindquist)](movie)
Conclusions

apparent horizon:

- good approximation to BH surface
- can find during an evolution (e.g. for excision)
- `AHFINDERDIRECT` is very fast ($\sim$ few seconds/horizon)
  very accurate ($\sim 10^{-5}m$)

event horizon:

- true definition of BH surface
- find by postprocessing after evolution
- `EHFINDER` is fast relative to full evolution ($\sim$ hours/spacetime)
  accurate ($\lesssim 0.001m$)

`AHFINDERDIRECT` and `EHFINDER` are both `CACTUS` thorns, and will be freely available (anonymous CVS) starting later this summer