A Fast Apparent-Horizon Finder for 3D Cartesian Grids in Numerical Relativity

(gr-qc/0306056)

Jonathan Thornburg <jthorn@aei.mpg.de>

http://www.aei.mpg.de/~jthorn

Max-Planck-Institut für Gravitationsphysik

Albert-Einstein-Institut

Golm, Germany

Why Find Apparent Horizons?

AHs are excellent black hole diagnostics

may want to know about BHs to understand physics better
(masses, spins, Gaussian curvature, isolated horizons, . . .)

may want to do excision

may may want to "steer" coordinates to "track" moving BHs

study initial data families

study cosmic censorship

"because they're there" :)

Apparent Horizons

Definition

outermost closed spacelike **2-surface** in a slice, whose
future-pointing **outgoing null geodesics** have **zero expansion**

Properties

- $\exists$ AH $\Rightarrow$ $\exists$ BH with AH $\subseteq$ event horizon (in practice often AH $\approx$ EH)
- AH is defined locally in time: can compute while evolution is in progress and thus use for excision, "steering" coordinates, etc.
- AH is generically smooth in space (elliptic PDE)
- AH may fail to exist if the slicing is "bad"
- AH may jump discontinuously or appear/disappear during evolution

Why a new Apparent-Horizon Finder?

AHs are very good BH diagnostics and are defined locally in time
$\Rightarrow$ want to find AHs at **each time step** of a numerical evolution

typical (3D) evolutions today

- last for $O(10^3)$ time steps (hopefully $O(10^5)$ in the future)
  $\Rightarrow$ AH finder should be **fast**
- use Cartesian-topology grids

spherical symmetry or axisymmetry $\Rightarrow$ $\exists$ fast AH finders
3D grid (no symmetry) $\Rightarrow$ existing AH finders are generally
very slow ($\sim$ 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)

$$\left\{ \begin{array}{l} \bullet \text{ functional iteration} \\ \bullet \text{ curvature/fast/level flow} \\ \bullet \text{ minimization} \\ \bullet \ \underline{\textbf{elliptic PDE}} \end{array} \right\} \times \left\{ \begin{array}{l} \bullet \ \underline{\textbf{finite difference}} \\ \bullet \text{ spectral} \end{array} \right\}$$

Parameterizing the Apparent Horizon Surface

assume we're given a spacelike 3-slice Σ

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 $\boldsymbol{r = h(\text{angle})}$
for some **single-valued** function $h : S^2 \to \Re^+$

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** $\boldsymbol{S^2}$,

$$\Theta(h, \partial_u h, \partial_{uv} h) = 0$$

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 (4th order)
($N_{\rm ang}$ angular grid points on S^2, typically $300 \lesssim N_{\rm ang} \lesssim 3000$)
$\Rightarrow$ set of $N_{\rm ang}$ nonlinear algebraic equations $\{\Theta_{\rm K} = 0\}$

solve for $\{h_{\rm K}\}$ via **Newton's method in** $\boldsymbol{N_{\rm ang}}$ **dimensions**
$\left(\equiv \text{Newton-Kantorovich linearization of original nonlinear PDE, followed by finite differencing each of the resulting linear PDEs}\right)$

$\Rightarrow$ (potentially) **very fast and accurate**

[Erik Schnetter uses PETSc in his AH finder $\Rightarrow$ also fast]

Computing the Jacobian Matrix

Newton's method $\Rightarrow$ need Jacobian matrix ($\mathbf{J}_{\mathrm{IJ}} \equiv \partial\Theta_{\mathrm{I}}/\partial h_{\mathrm{J}}$)

($N_{\mathrm{ang}} \times N_{\mathrm{ang}}$ sparse matrix)

numerical perturbation is slow for large N_{ang}

instead, use **Symbolic Differentiation**:
compute $\mathbf{J}$ directly from ∂_u, ∂_{uv} molecule coefficients $\Rightarrow$ very fast

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

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

$$\text{then } P_{\mathrm{I}} \equiv \sum_{\mathrm{M}} \mathsf{M}_{\mathrm{M}} Q_{\mathrm{I}+\mathrm{M}} \Rightarrow \mathbf{J}_{\mathrm{IJ}} \equiv \frac{\partial P_{\mathrm{I}}}{\partial Q_{\mathrm{J}}} = \begin{cases} \mathsf{M}_{\mathrm{J}-\mathrm{I}} & \text{if } \mathrm{J}-\mathrm{I} \in \mathsf{M} \\ 0 & \text{otherwise} \end{cases}$$

if P is **nonlinear**, Newton-Kantorovich–linearize it

$$\Rightarrow \text{same formula with } \mathsf{M} = \frac{\partial P}{\partial Q}\mathbf{I} + \frac{\partial P}{\partial(\partial_i Q)}\mathbf{d}_i + \frac{\partial P}{\partial(\partial_{ij} Q)}\mathbf{d}_{ij}$$

Lagrange versus Hermite Interpolation

Lagrange polynomial interpolation

$\Rightarrow$ viewed as functions of the interpolation point, the interpolated

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

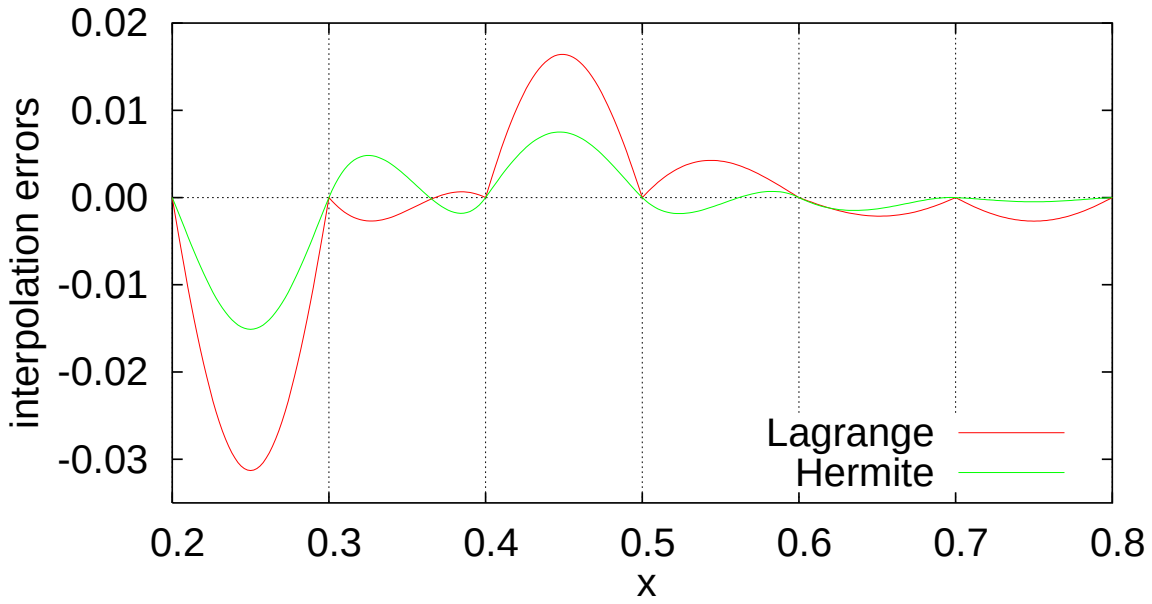
$\Rightarrow$ interpolated g_{ij}, K_{ij} are C^1 everywhere, $\partial_k g_{ij}$ is C^0 everywhere

Example:

1D cubic interpolation

$f(x) = \exp[\sin(x)]$

grid spacing $\Delta x = 0.1$



Computing Derivatives Inside the Interpolation

geometry fields in $\Theta(h, \partial_u h, \partial_{uv} h;\ \boldsymbol{g_{ij}}, \boldsymbol{\partial_k g_{ij}}, \boldsymbol{K_{ij}})$
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 $\boldsymbol{\partial_k g_{ij}}$ 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 derivatives:
input: grid $\{x_{\mathrm{K}}\}$, grid function f, interpolation point x_{interp}

- choose a molecule M of grid points which is near x_{interp}
- compute interpolating function I with $I(x_{\mathrm{K}}) = f_{\mathrm{K}}\ \forall \mathrm{K} \in \mathsf{M}$
- evaluate $I(x_{\mathrm{interp}})$ and its derivatives

Choice of Angular Coordinates/Grid on S^2

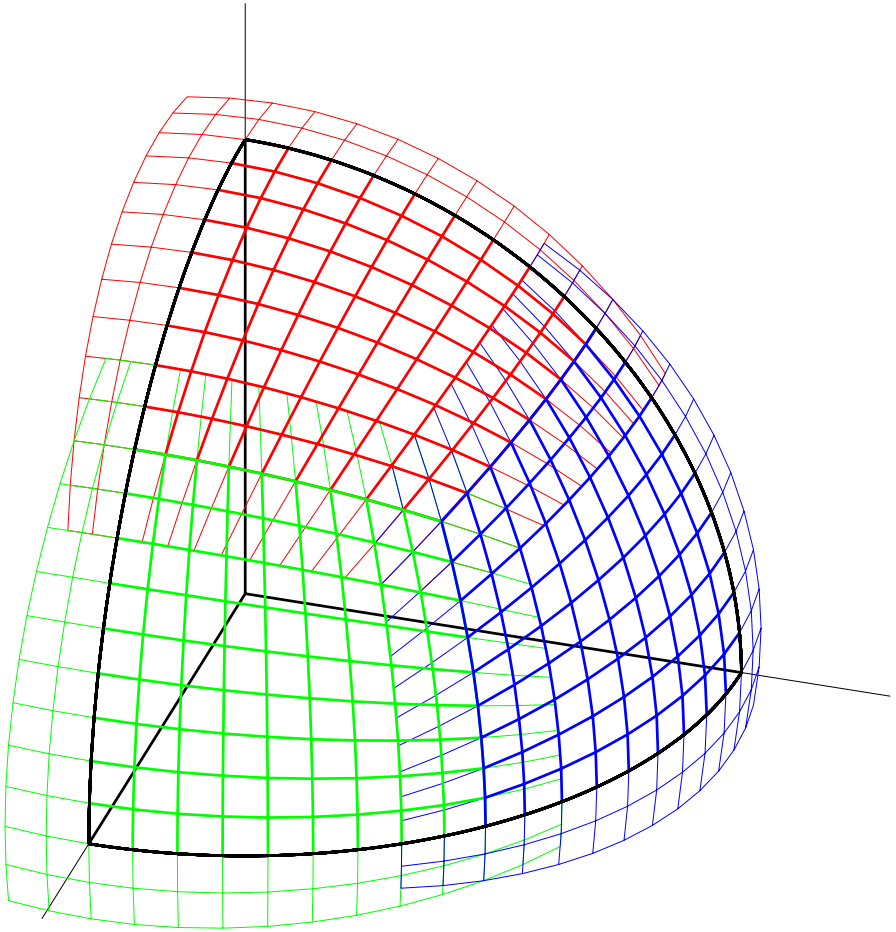
How to choose the angular coordinates/grid x^u on S^2?

usual polar spherical (θ, ϕ)

$\Rightarrow$ z axis coordinate singularity

$\Rightarrow$ possible numerical problems

[but this works ok in Erik Schnetter's AH finder]



multiple grid patches

figure shows 3 patches covering $(+,+,+)$ octant of S^2 with **no** coordinate singularities

more generally, 6 patches (neighborhoods of $\pm z$, $\pm x$, $\pm y$) can cover all of S^2, again with **no** coordinate singularities

Sample Results – Boosted Kerr Slices

take Kerr-Schild slice of Kerr spacetime (BH spin in z direction)
then Lorentz-boost in x direction ($\perp$ to spin)
$\Rightarrow$ no symmetries, yet AH still known analytically (triaxial ellipsoid)

			AHFinderDirect					Fast Flow			
a	v_x	Δxyz	$\Delta\rho\sigma$	interp	time	$\|\|\delta h\|\|_{\rm rms}$	$(\delta A)/A$	$\ell_{\rm max}$	interp	time	$(\delta A)/A$
0.8	0.8	0.15	5°	H3	4.0	3.4×10^{-4}	4.7×10^{-4}	10	L3	26	6.5×10^{-3}
0.8	0.8	0.10	5°	H3	4.1	5.4×10^{-5}	7.8×10^{-5}	10	L3	33	2.2×10^{-3}
0.8	0.8	0.05	5°	H3	4.2	2.5×10^{-5}	4.1×10^{-5}	10	L3	96	4.6×10^{-4}
0.8	**0.8**	**0.20**	**5°**	**H2**	**1.8**	$\mathbf{3.6\times10^{-3}}$	$\mathbf{4.7\times10^{-3}}$	10	L2	24	1.1×10^{-2}
0.8	**0.8**	**0.10**	**5°**	**H2**	**1.9**	$\mathbf{6.5\times10^{-4}}$	$\mathbf{1.0\times10^{-3}}$	10	L2	32	2.1×10^{-3}
0.8	**0.8**	**0.03**	**5°**	**H2**	**2.0**	$\mathbf{3.4\times10^{-5}}$	$\mathbf{4.7\times10^{-5}}$	10	L2	350	1.0×10^{-3}
0.8	0.8	0.05	3.0°	H3	13	4.4×10^{-6}	7.3×10^{-6}	15	L2	170	6.9×10^{-4}
0.8	0.8	0.05	1.8°	H3	43	1.3×10^{-6}	1.8×10^{-6}	20	L2	280	1.3×10^{-3}
0.8	0.8	0.05	1.0°	H3	220	9.5×10^{-7}	1.3×10^{-6}	28	L2	2600	failed
0.999	0.8	0.03	3°	H3	16	9.5×10^{-6}	1.4×10^{-5}	15	L2	1600	failed
0.999	0.95	0.02	3°	H3	13	4.0×10^{-4}	6.5×10^{-4}	not tested on this slice			
0.999	0.98	0.02	3°	H3	12	1.3×10^{-4}	1.1×10^{-4}	not tested on this slice			

$\Rightarrow$ AHFinderDirect is generally very fast and very accurate

Sample Results – Brill-Lindquist & Misner "Critical" AHs

Consider Brill-Lindquist or Misner initial data, and numerically find the largest parameter for which a common AH is present:

	Alcubierre *et al.* (2000)	AHFinderDirect	
Test Problem (parameter)	critical parameter	critical parameter	critical AH area
Misner (μ)	1.364	$1.365\,071\,172 \pm 3$	$409.549\,358 \pm 3$
Brill-Lindquist 2-throat (R)	0.766	$0.766\,197\,45 \pm 5$	$196.407\,951 \pm 3$
Brill-Lindquist 3-throat (R)	1.18 ± 4	$1.195\,499\,53 \pm 5$	$444.756\,224 \pm 3$

other values reported for 2-throat Brill-Lindquist data:

- $R = 0.767 \pm 1$ [Čadež (1974)]
- $R = 0.768$ [Shoemaker, Huq, and Matzner (2000)]
  [critical AH area disagrees by $\sim 6\%$:( ]

$\Rightarrow$ AHFinderDirect values are consistent and **much more accurate**

Sample Results – 2BH Misner Evolution

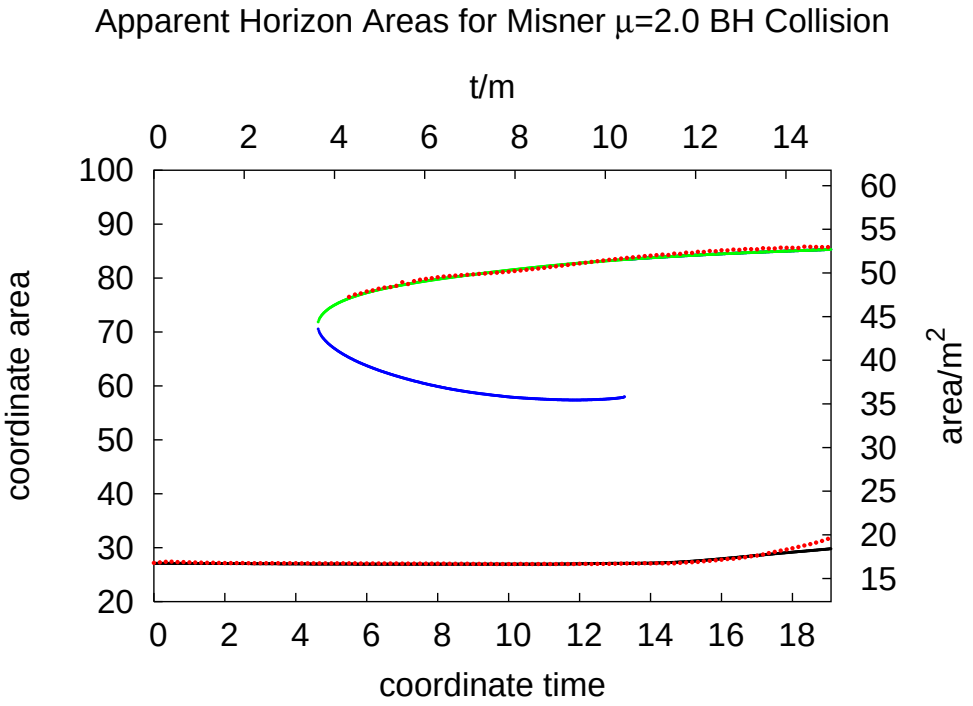
2BH Misner initial data, $\mu = 2.0 \Rightarrow m_{\rm ADM} = 1.27$

BSSN system, $1 + \log$ slicing, modified Γ-freezing shift

good resolution ($\Delta x = 0.039 m_{\rm ADM}$ near BHs, $0.12 m_{\rm ADM}$ near outer boundary)

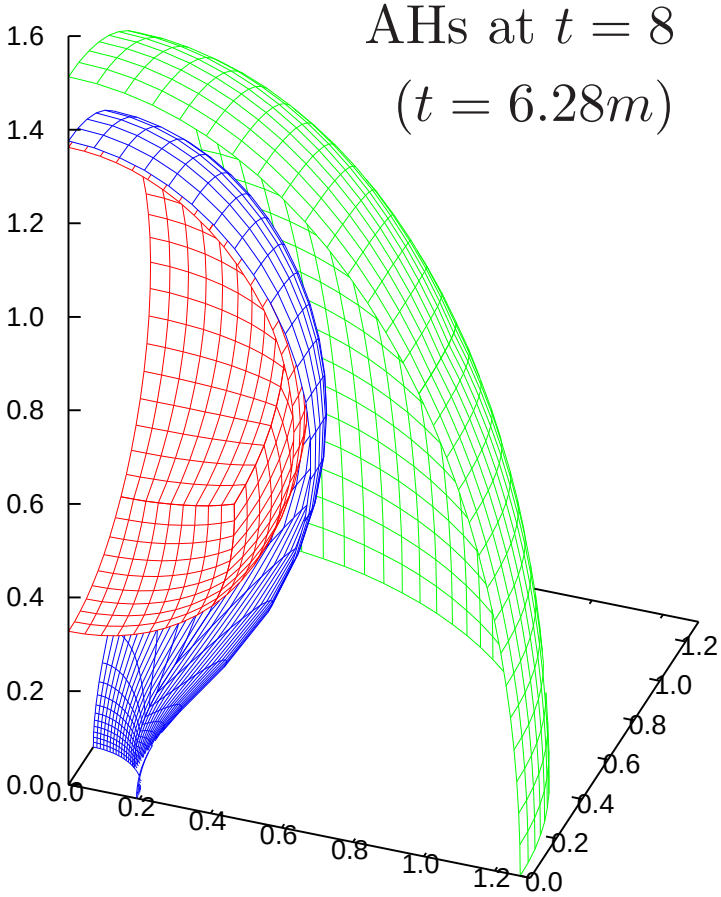
very small grid (outer boundary at $5.2 m_{\rm ADM}$)

⟦movie⟧



AHs at $t = 8$

$(t = 6.28m)$



Sample Results – Orbiting 2BH Collision

initial data

Baumgarte-Cook "ISCO": 2 Brandt-Brügmann "puncture" BHs, each with AH mass $0.52 m_{\rm ADM}$, total $J/m^2_{\rm ADM} = 0.78$, proper separation $5.0 m_{\rm ADM}$

evolution

AEI SNO (a.k.a. BSSN) (gr-qc/0003071), "Lego" excision @ $0.8 r_{\rm AH} - 5\,\Delta x$
$\Delta x = 0.08 m_{\rm ADM}$ (inner) to $0.32 m_{\rm ADM}$ (outer); outer boundary $45 m_{\rm ADM}$

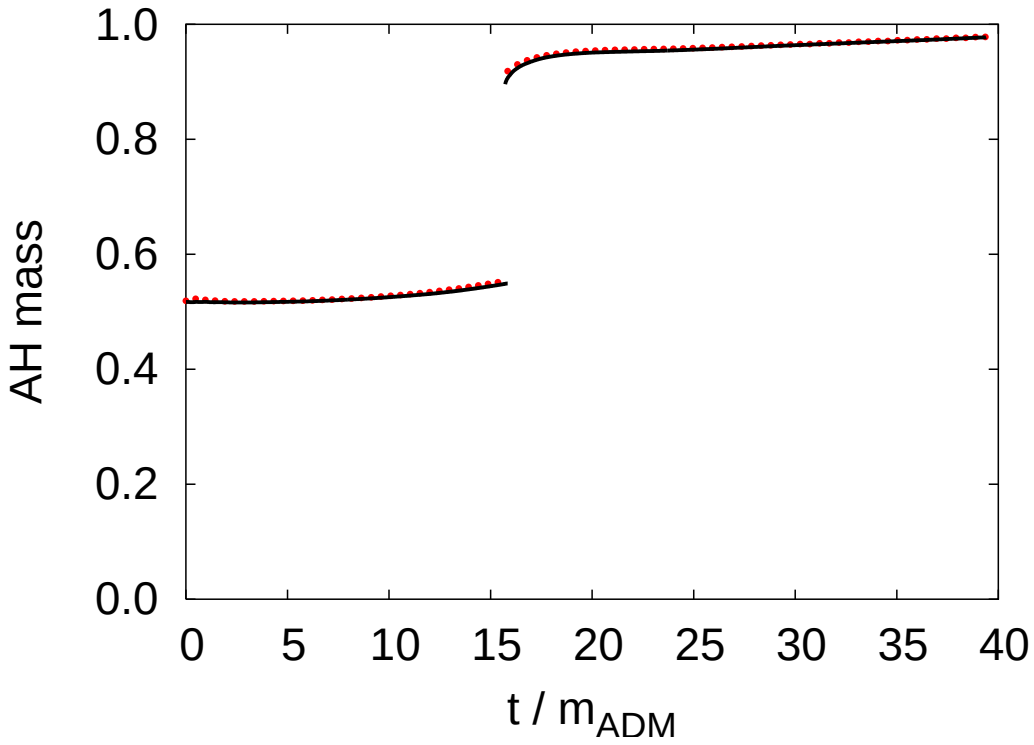
Curvature-flow AH finder

61 seconds/time step
(run every 24 time steps)

AHFinderDirect

1.7 seconds/time step
(run every time step)

⟦movie⟧



Conclusions

AHFinderDirect is

- very fast ($\sim$ few seconds to find all AHs)
  $\Rightarrow$ it's practical to use it to find AHs at each time step of an evolution
- very accurate (typically 10^{-3} to $10^{-5}m$)
  (can be made much better by high resolution & tight error tolerances)

AHFinderDirect is described in more detail in gr-qc/0306056.
It's implemented as a "thorn" in the Cactus computational toolkit (`http://www.cactuscode.org`), and is freely available for use by other groups.

AHFinderDirect is now entering production use at AEI, both as an analysis tool, and as part of our evolution code, controlling excision and corotating shift parameters (e.g. Baumgarte-Cook evolutions).