Numerical Simulations of Black Holes

26 Aug 2009

Frank Herrmann (fherrman@umd.edu)

Department of Physics & Center for Scientific Computation and Mathematical Modeling, University of Maryland

Group Members: J. Silberholz, E. Pazos, C. Galley, A. Zenginoglu, M. Tiglio

supported by NSF, NASA and NVIDIA

Overview

Introduction

Moving Punctures

BH Results

Multi-Block & Matter Results

Outlook & Conclusions

Numerical Relativity

In my universe $T_{\mu\nu} = 0$ (mostly)

Numerical Relativity

solve Einstein's equations with a computer (for vacuum: $G_{\mu\nu} = 0$)

important only for strong-field non-linear regime

- merger of compact objects

Much more than just black hole evolutions

- Formulations
- Hyperboloidal Slicing
- Coordinate conditions (gauges)
- Outer boundary
- Well-posedness analysis for different systems including gauge
- Use of different discretization systems
- Astrophysical initial data
- Matter (BH-Neutron Star, NS-NS, Supernovae, . . .)
- High-energy collisions
- . . .

Solving Einstein's Equations

plug g_{ab} into $G_{ab} = 0$

get 10 coupled quasi-linear 2nd order PDEs

problems

- no definite mathematical character,
  i.e. not hyperbolic, parabolic, elliptic
- admit no well-posed initial value problem

fix character of coordinates x^a

- typically 1 time-like, 3 space-like ("3+1"-split)
- get elliptic and hyperbolic PDEs
  - ▷ elliptic: constraints on space-like slices (no time-derivatives), initial-data
  - ▷ hyperbolic: to evolve these slices

Form of Equations

There are two sets of equations which people use

Start from $G_{\mu\nu} = 0$

Harmonic [Pretorius]	BSSN [many groups]
$\Box x^\mu = H^\mu$	$ds^2 = -\alpha^2 dt^2 + \gamma_{ij}(dx^i + \beta^i dt)(dx^j + \beta^j dt)$
$g^{\gamma\delta} g_{\alpha\beta,\gamma\delta} + \ldots = 0$	Evolution and Constraint equations

Generalized Harmonic

- used by Pretorius, Cornell/Caltech, UMD

BSSN

- used by AEI, FAU, GT, Jena, NASA, PSU, RIT, UMD

evolution method 1: Generalized Harmonic

Einstein's equations (vacuum)

$$0 = R_{ab} = -\frac{1}{2}\Box g_{ab} + \nabla_{(a}\Gamma_{b)} + \dots$$

with $\Gamma_a = -g_{ab}\Box x^b$

$$\Box x^a = 0$$

- 4 independent equations

principal part for each metric element becomes a scalar wave equation for that particular element!

- $\Box g_{ab} = \dots$

used for a long time, but not in numerical relativity (singularities)

gauge source functions: $\Box x^a = H^a$ and constraint damping needed

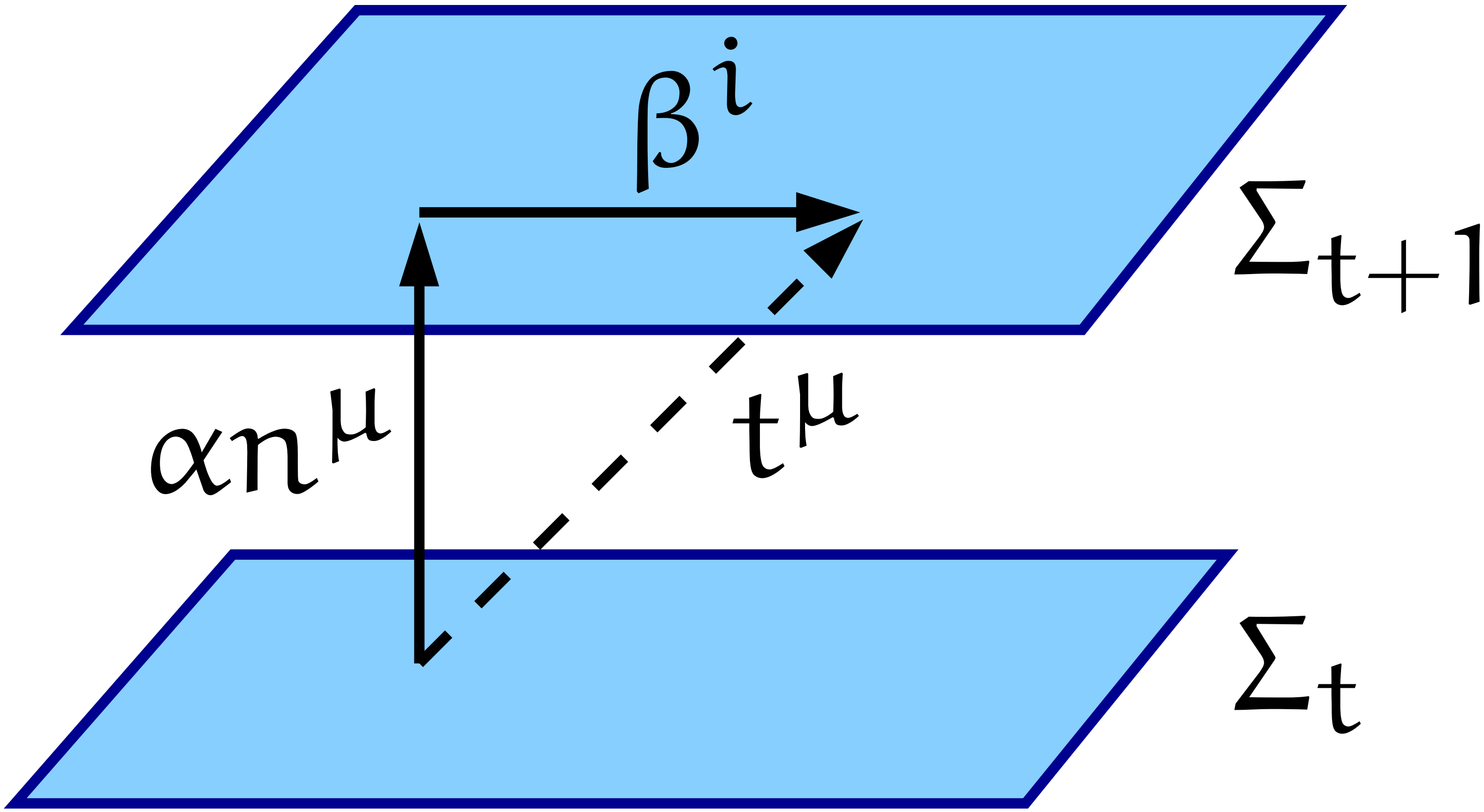
- Pretorius 2005

time-space split

Lapse α, Shift vector β^i, 3-metric γ_{ij}, extrinsic curvature K_{ij}

$$ds^2 = -\alpha^2 dt^2 + \gamma_{ij}(dx^i + \beta^i dt)(dx^j + \beta^j dt)$$

extrinsic curvature $K_{ij} =$ "$\dot{\gamma}_{ij}$"



$t^\mu = \alpha n^\mu + \beta^\mu$

evolution method 2: BSSN

$$
\begin{aligned}
\partial_0 \alpha &= -2\alpha K \\
\partial_0 \beta^a &= B^a \\
\partial_0 B^a &= 3/4 \partial_0 \tilde{\Gamma}^a - \eta B^a \\
\partial_0 \tilde{\gamma}_{ij} &= -2\alpha \tilde{A}_{ij} \\
\partial_t \chi &= \frac{2}{3}\chi\left(\alpha K - \partial_a \beta^a\right) + \beta^i \partial_i \chi, \\
\partial_0 \tilde{A}_{ij} &= \chi\left(-D_i D_j \alpha + \alpha R_{ij}\right)^{TF} + \alpha\left(K\tilde{A}_{ij} - 2\tilde{A}_{ik}\tilde{A}^k_j\right) \\
\partial_0 K &= -D^i D_i \alpha + \alpha\left(\tilde{A}_{ij}\tilde{A}^{ij} + \frac{1}{3}K^2\right) \\
\partial_t \tilde{\Gamma}^i &= \tilde{\gamma}^{jk}\partial_j\partial_k\beta^i + \frac{1}{3}\tilde{\gamma}^{ij}\partial_j\partial_k\beta^k + \beta^j\partial_j\tilde{\Gamma}^i - \tilde{\Gamma}^j\partial_j\beta^i + \frac{2}{3}\tilde{\Gamma}^i\partial_j\beta^j - 2\tilde{A}^{ij}\partial_j\alpha + 2\alpha\left(\tilde{\Gamma}^i{}_{jk}\tilde{A}^{jk} + 6\tilde{A}^{ij}\partial_j\phi - \frac{2}{3}\tilde{\gamma}^{ij}\partial_j K\right)
\end{aligned}
$$

$\partial_0 = \partial_t - \mathcal{L}_\beta$, TF indicates that only the trace-free part of the tensor, $R_{ij} = \tilde{R}_{ij} + R^\phi{}_{ij}$ is given by

$$
\begin{aligned}
R^\phi{}_{ij} &= -2\tilde{D}_i\tilde{D}_j\phi - 2\tilde{\gamma}_{ij}\tilde{D}^k\tilde{D}_k\phi + 4\tilde{D}_i\phi\tilde{D}_j\phi - 4\tilde{\gamma}_{ij}\tilde{D}^k\phi\tilde{D}_k\phi, \\
\tilde{R}_{ij} &= -\frac{1}{2}\tilde{\gamma}^{lm}\partial_l\partial_m\tilde{\gamma}_{ij} + \tilde{\gamma}_{k(i}\partial_{j)}\tilde{\Gamma}^k + \tilde{\Gamma}^k\tilde{\Gamma}_{(ij)k} + \tilde{\gamma}^{lm}\left(2\tilde{\Gamma}^k{}_{l(i}\tilde{\Gamma}_{j)km} + \tilde{\Gamma}^k{}_{im}\tilde{\Gamma}_{klj}\right),
\end{aligned}
$$

$\tilde{\Gamma}^i$ is replaced by $-\partial_j\tilde{\gamma}^{ij}$ wherever it is not differentiated. $\partial_i\phi = -1/(4\chi)\partial_i\chi$ and $\partial_{ij}\phi = \frac{1}{4}(-\partial_{ij}\chi/\chi + \partial_i\chi\partial_j\chi/\chi^2)$ Lie derivatives of non-tensorial quantities ($\tilde{\gamma}_{ij}$, and $\tilde{A}_{ij}$) are given by

$$
\begin{aligned}
\mathcal{L}_\beta\tilde{\gamma}_{ij} &= \beta^k\partial_k\tilde{\gamma}_{ij} + \tilde{\gamma}_{ik}\partial_j\beta^k + \tilde{\gamma}_{jk}\partial_i\beta^k - \frac{2}{3}\tilde{\gamma}_{ij}\partial_k\beta^k, \\
\mathcal{L}_\beta\tilde{A}_{ij} &= \beta^k\partial_k\tilde{A}_{ij} + \tilde{A}_{ik}\partial_j\beta^k + \tilde{A}_{jk}\partial_i\beta^k - \frac{2}{3}\tilde{A}_{ij}\partial_k\beta^k.
\end{aligned}
$$

BSSN equations: code

```
...

/* inverse conformal metric h^ij */
double deth = (h11*h22-(h12*h12))*h33-h11*(h23*h23)+2.0*h12*h13*h23-(h13*h13)*h22;
double oodeth = 1.0/deth;
double hup11 = (h22*h33-(h23*h23))*oodeth;
double hup12 = (-h12*h33+h13*h23)*oodeth;
double hup22 = (h11*h33-(h13*h13))*oodeth;
double hup13 = (h12*h23-h13*h22)*oodeth;
double hup23 = (-h11*h23+h12*h13)*oodeth;
double hup33 = (h11*h22-(h12*h12))*oodeth;


/* christoffel */
double chr111 = (-d_h311+2.0*d_h113)*half*hup13+(-d_h211+2.0*d_h112)*half*hup12+d_h111
    *half*hup11;
double chr211 = (-d_h311+2.0*d_h113)*half*hup23+(-d_h211+2.0*d_h112)*half*hup22+d_h111
    *half*hup12;
double chr311 = (-d_h311+2.0*d_h113)*half*hup33+(-d_h211+2.0*d_h112)*half*hup23+d_h111
    *half*hup13;
double chr112 = (-d_h312+d_h213+d_h123)*half*hup13+d_h122*half*hup12+d_h211*half*hup11;
double chr212 = (-d_h312+d_h213+d_h123)*half*hup23+d_h122*half*hup22+d_h211*half*hup12;
double chr312 = (-d_h312+d_h213+d_h123)*half*hup33+d_h122*half*hup23+d_h211*half*hup13;
double chr122 = (-d_h322+2.0*d_h223)*half*hup13+d_h222*half*hup12+(2.0*d_h212-d_h122)
    *half*hup11;
double chr222 = (-d_h322+2.0*d_h223)*half*hup23+d_h222*half*hup22+(2.0*d_h212-d_h122)
    *half*hup12;
double chr322 = (-d_h322+2.0*d_h223)*half*hup33+d_h222*half*hup23+(2.0*d_h212-d_h122)
    *half*hup13;
double chr113 = d_h133*half*hup13+(d_h312-d_h213+d_h123)*half*hup12+d_h311*half*hup11;
double chr213 = d_h133*half*hup23+(d_h312-d_h213+d_h123)*half*hup22+d_h311*half*hup12;
double chr313 = d_h133*half*hup33+(d_h312-d_h213+d_h123)*half*hup23+d_h311*half*hup13;
```

```
double chr123 = d_h233*half*hup13+d_h322*half*hup12+(d_h312+d_h213-d_h123)*half*hup11;
double chr223 = d_h233*half*hup23+d_h322*half*hup22+(d_h312+d_h213-d_h123)*half*hup12;
double chr323 = d_h233*half*hup33+d_h322*half*hup23+(d_h312+d_h213-d_h123)*half*hup13;
double chr133 = d_h333*half*hup13+(2.0*d_h323-d_h233)*half*hup12+(2.0*d_h313-d_h133)*half
    *hup11;
double chr233 = d_h333*half*hup23+(2.0*d_h323-d_h233)*half*hup22+(2.0*d_h313-d_h133)*half
    *hup12;
double chr333 = d_h333*half*hup33+(2.0*d_h323-d_h233)*half*hup23+(2.0*d_h313-d_h133)*half
    *hup13;

/* A^ij */
double Aup11 = hup13*hup13*A33+2.0*hup12*hup13*A23+hup12*hup12*A22+2.0*hup11*hup13*A13+2.0*hup11
    *hup12*A12+hup11*hup11*A11;
double Aup12 = hup13*hup23*A33+(hup12*hup23+hup13*hup22)*A23+hup12*hup22*A22+(hup11*hup23+hup12
    *hup13)*A13+(hup11*hup22+hup12*hup12)*A12+hup11*hup12*A11;
double Aup22 = hup23*hup23*A33+2.0*hup22*hup23*A23+hup22*hup22*A22+2.0*hup12*hup23*A13+2.0*hup12
    *hup22*A12+hup12*hup12*A11;
double Aup13 = hup13*hup33*A33+(hup12*hup33+hup13*hup23)*A23+hup12*hup23*A22+(hup11*hup33+hup13
    *hup13)*A13+(hup11*hup23+hup12*hup13)*A12+hup11*hup13*A11;
double Aup23 = hup23*hup33*A33+(hup22*hup33+hup23*hup23)*A23+hup22*hup23*A22+(hup12*hup33+hup13
    *hup23)*A13+(hup12*hup23+hup13*hup22)*A12+hup12*hup13*A11;
double Aup33 = hup33*hup33*A33+2.0*hup23*hup33*A23+hup23*hup23*A22+2.0*hup13*hup33*A13+2.0*hup13
    *hup23*A12+hup13*hup13*A11;

/* Gamma^i recomputed */
double GamRC1 = (d_h333*hup13+d_h323*hup12+d_h313*hup11)*hup33+((d_h323+d_h233)*hup13+(d_h322
    +d_h223)*hup12+(d_h312+d_h213)*hup11)*hup23+(d_h223*hup13+d_h222*hup12+d_h212
    *hup11)*hup22+(d_h313+d_h133)*(hup13*hup13)+((d_h312+d_h213+2.0*d_h123)*hup12
    +(d_h311+2.0*d_h113)*hup11)*hup13+(d_h212+d_h122)*(hup12*hup12)+(d_h211+2.0*d_h112)
    *hup11*hup12+d_h111*(hup11*hup11);
double GamRC2 = (d_h333*hup23+d_h323*hup22+d_h313*hup12)*hup33+(d_h323+d_h233)*(hup23*hup23)+((d_h322
    +2.0*d_h223)*hup22+(d_h313+d_h133)*hup13+(d_h312+2.0*d_h213+d_h123)*hup12+d_h113
    *hup11)*hup23+d_h222*(hup22*hup22)+((d_h312+d_h123)*hup13+(2.0*d_h212+d_h122)
    *hup12+d_h112*hup11)*hup22+(d_h311+d_h113)*hup12*hup13+(d_h211+d_h112)*(hup12
    *hup12)+d_h111*hup11*hup12;
double GamRC3 = d_h333*(hup33*hup33)+((2.0*d_h323+d_h233)*hup23+d_h223*hup22+(2.0*d_h313+d_h133)
    *hup13+(d_h213+d_h123)*hup12+d_h113*hup11)*hup33+(d_h322+d_h223)*(hup23*hup23)
```

```
    +(d_h222*hup22+(2.0*d_h312+d_h213+d_h123)*hup13+(d_h212+d_h122)*hup12+d_h112*hup11)
    *hup23+d_h212*hup13*hup22+(d_h311+d_h113)*(hup13*hup13)+((d_h211+d_h112)*hup12
    +d_h111*hup11)*hup13;


/* ricci */
double R11 = h13*d_Gam13+h12*d_Gam12+h11*d_Gam11+(-d_d_h3311*half+chr313*chr313*h33+2.0
    *chr213*chr313*h23+chr213*chr213*h22+(2.0*chr313*chr333+2.0*chr213*chr323+4.0
    *chr113*chr313)*h13+(2.0*chr233*chr313+2.0*chr213*chr223+4.0*chr113*chr213)*h12
    +(2.0*chr133*chr313+2.0*chr123*chr213+3.0*(chr113*chr113))*h11)*hup33+(-2.0*d_d_h2311
    *half+2.0*chr312*chr313*h33+(2.0*chr212*chr313+2.0*chr213*chr312)*h23+2.0*chr212
    *chr213*h22+(2.0*chr312*chr333+(2.0*chr313+2.0*chr212)*chr323+2.0*chr213*chr322
    +4.0*chr112*chr313+4.0*chr113*chr312)*h13+(2.0*chr223*chr313+2.0*chr233*chr312
    +2.0*chr212*chr223+2.0*chr213*chr222+4.0*chr112*chr213+4.0*chr113*chr212)*h12
    +(2.0*chr123*chr313+2.0*chr133*chr312+2.0*chr122*chr213+2.0*chr123*chr212+6.0
    *chr112*chr113)*h11)*hup23+(-d_d_h2211*half+chr312*chr312*h33+2.0*chr212*chr312
    *h23+chr212*chr212*h22+(2.0*chr312*chr323+2.0*chr212*chr322+4.0*chr112*chr312)
    *h13+(2.0*chr223*chr312+2.0*chr212*chr222+4.0*chr112*chr212)*h12+(2.0*chr123*chr312
    +2.0*chr122*chr212+3.0*(chr112*chr112))*h11)*hup22+(-2.0*d_d_h1311*half+2.0*chr311
    *chr313*h33+(2.0*chr211*chr313+2.0*chr213*chr311)*h23+2.0*chr211*chr213*h22+(2.0
    *chr311*chr333+2.0*chr211*chr323+2.0*(chr313*chr313)+4.0*chr111*chr313+2.0*chr213
    *chr312+4.0*chr113*chr311)*h13+(2.0*chr213*chr313+2.0*chr233*chr311+2.0*chr211
    *chr223+(2.0*chr212+4.0*chr111)*chr213+4.0*chr113*chr211)*h12+(2.0*chr113*chr313
    +2.0*chr133*chr311+2.0*chr112*chr213+2.0*chr123*chr211+6.0*chr111*chr113)*h11)
    *hup13+(-2.0*d_d_h1211*half+2.0*chr311*chr312*h33+(2.0*chr211*chr312+2.0*chr212
    *chr311)*h23+2.0*chr211*chr212*h22+(2.0*chr311*chr323+2.0*chr211*chr322+2.0*chr312
    *chr313+(2.0*chr212+4.0*chr111)*chr312+4.0*chr112*chr311)*h13+(2.0*chr213*chr312
    +2.0*chr223*chr311+2.0*chr211*chr222+2.0*(chr212*chr212)+4.0*chr111*chr212+4.0
    *chr112*chr211)*h12+(2.0*chr113*chr312+2.0*chr123*chr311+2.0*chr112*chr212+2.0
    *chr122*chr211+6.0*chr111*chr112)*h11)*hup12+(-d_d_h1111*half+chr311*chr311
    *h33+2.0*chr211*chr311*h23+chr211*chr211*h22+(2.0*chr311*chr313+2.0*chr211*chr312
    +4.0*chr111*chr311)*h13+(2.0*chr213*chr311+2.0*chr211*chr212+4.0*chr111*chr211)
    *h12+(2.0*chr113*chr311+2.0*chr112*chr211+3.0*(chr111*chr111))*h11)*hup11+(chr313
    *GamRC3+chr312*GamRC2+chr311*GamRC1)*h13+(chr213*GamRC3+chr212*GamRC2+chr211*GamRC1)
    *h12+(chr113*GamRC3+chr112*GamRC2+chr111*GamRC1)*h11;
double R12 = (h13*d_Gam23+h12*d_Gam22+h11*d_Gam21+h23*d_Gam13+h22*d_Gam12+h12*d_Gam11+(
    -2.0*d_d_h3312*half+2.0*chr313*chr323*h33+(2.0*chr313*chr333+4.0*chr213*chr323
```

```
+(2.0*chr223+2.0*chr113)*chr313)*h23+(2.0*chr233*chr313+4.0*chr213*chr223+2.0
*chr113*chr213)*h22+(2.0*chr323*chr333+(2.0*chr223+2.0*chr113)*chr323+4.0*chr123
*chr313)*h13+(2.0*chr233*chr323+2.0*chr133*chr313+2.0*(chr223*chr223)+2.0*chr113
*chr223+6.0*chr123*chr213+2.0*(chr113*chr113))*h12+(2.0*chr133*chr323+2.0*chr123
*chr223+4.0*chr113*chr123)*h11)*hup33+(-4.0*d_d_h2312*half+(2.0*chr312*chr323
+2.0*chr313*chr322)*h33+(2.0*chr312*chr333+(2.0*chr313+4.0*chr212)*chr323+4.0
*chr213*chr322+(2.0*chr222+2.0*chr112)*chr313+(2.0*chr223+2.0*chr113)*chr312)
*h23+(2.0*chr223*chr313+2.0*chr233*chr312+4.0*chr212*chr223+4.0*chr213*chr222
+2.0*chr112*chr213+2.0*chr113*chr212)*h22+(2.0*chr322*chr333+2.0*(chr323*chr323)
+(2.0*chr222+2.0*chr112)*chr323+(2.0*chr223+2.0*chr113)*chr322+4.0*chr122*chr313
+4.0*chr123*chr312)*h13+(2.0*chr223*chr323+2.0*chr233*chr322+2.0*chr123*chr313
+2.0*chr133*chr312+(4.0*chr222+2.0*chr112)*chr223+2.0*chr113*chr222+6.0*chr122
*chr213+6.0*chr123*chr212+4.0*chr112*chr113)*h12+(2.0*chr123*chr323+2.0*chr133
*chr322+2.0*chr122*chr223+2.0*chr123*chr222+4.0*chr112*chr123+4.0*chr113*chr122)
*h11)*hup23+(-2.0*d_d_h2212*half+2.0*chr312*chr322*h33+(2.0*chr312*chr323+4.0
*chr212*chr322+(2.0*chr222+2.0*chr112)*chr312)*h23+(2.0*chr223*chr312+4.0*chr212
*chr222+2.0*chr112*chr212)*h22+(2.0*chr322*chr323+(2.0*chr222+2.0*chr112)*chr322
+4.0*chr122*chr312)*h13+(2.0*chr223*chr322+2.0*chr123*chr312+2.0*(chr222*chr222)
+2.0*chr112*chr222+6.0*chr122*chr212+2.0*(chr112*chr112))*h12+(2.0*chr123*chr322
+2.0*chr122*chr222+4.0*chr112*chr122)*h11)*hup22+(-4.0*d_d_h1312*half+(2.0*chr311
*chr323+2.0*chr312*chr313)*h33+(2.0*chr311*chr333+4.0*chr211*chr323+2.0*(chr313
*chr313)+(2.0*chr212+2.0*chr111)*chr313+4.0*chr213*chr312+(2.0*chr223+2.0*chr113)
*chr311)*h23+(2.0*chr213*chr313+2.0*chr233*chr311+4.0*chr211*chr223+(4.0*chr212
+2.0*chr111)*chr213+2.0*chr113*chr211)*h22+(2.0*chr312*chr333+(2.0*chr313+2.0
*chr212+2.0*chr111)*chr323+4.0*chr112*chr313+(2.0*chr223+2.0*chr113)*chr312+4.0
*chr123*chr311)*h13+(2.0*chr213*chr323+2.0*chr113*chr313+2.0*chr233*chr312+2.0
*chr133*chr311+(4.0*chr212+2.0*chr111)*chr223+6.0*chr112*chr213+2.0*chr113*chr212
+6.0*chr123*chr211+4.0*chr111*chr113)*h12+(2.0*chr113*chr323+2.0*chr133*chr312
+2.0*chr112*chr223+2.0*chr123*chr212+4.0*chr111*chr123+4.0*chr112*chr113)*h11)
*hup13+(-4.0*d_d_h1212*half+(2.0*chr311*chr322+2.0*(chr312*chr312))*h33+(2.0*chr311
*chr323+4.0*chr211*chr322+2.0*chr312*chr313+(6.0*chr212+2.0*chr111)*chr312+(2.0
*chr222+2.0*chr112)*chr311)*h23+(2.0*chr213*chr312+2.0*chr223*chr311+4.0*chr211
*chr222+4.0*(chr212*chr212)+2.0*chr111*chr212+2.0*chr112*chr211)*h22+(2.0*chr312
*chr323+(2.0*chr313+2.0*chr212+2.0*chr111)*chr322+(2.0*chr222+6.0*chr112)*chr312
+4.0*chr122*chr311)*h13+(2.0*chr213*chr322+(2.0*chr223+2.0*chr113)*chr312+2.0
*chr123*chr311+(4.0*chr212+2.0*chr111)*chr222+8.0*chr112*chr212+6.0*chr122*chr211
+4.0*chr111*chr112)*h12+(2.0*chr113*chr322+2.0*chr123*chr312+2.0*chr112*chr222
```

```
+2.0*chr122*chr212+4.0*chr111*chr122+4.0*(chr112*chr112))*h11)*hup12+(-2.0*d_d_h1112
*half+2.0*chr311*chr312*h33+(2.0*chr311*chr313+4.0*chr211*chr312+(2.0*chr212+2.0
*chr111)*chr311)*h23+(2.0*chr213*chr311+4.0*chr211*chr212+2.0*chr111*chr211)*h22
+(2.0*chr312*chr313+(2.0*chr212+2.0*chr111)*chr312+4.0*chr112*chr311)*h13+(2.0
*chr213*chr312+2.0*chr113*chr311+2.0*(chr212*chr212)+2.0*chr111*chr212+6.0*chr112
*chr211+2.0*(chr111*chr111))*h12+(2.0*chr113*chr312+2.0*chr112*chr212+4.0*chr111
*chr112)*h11)*hup11+(chr313*GamRC3+chr312*GamRC2+chr311*GamRC1)*h23+(chr213*GamRC3
+chr212*GamRC2+chr211*GamRC1)*h22+(chr323*GamRC3+chr322*GamRC2+chr312*GamRC1)
*h13+((chr223+chr113)*GamRC3+(chr222+chr112)*GamRC2+(chr212+chr111)*GamRC1)*h12
+(chr123*GamRC3+chr122*GamRC2+chr112*GamRC1)*h11)/2.0;
```

...

high arithmetic intensity

Operator	Number of times used
$*$	12,961
$+$	5,398
$-$	3,438
$/$	69

Black Hole Singularities

Computers don't like the singularities inside of black holes

- Hawking & Penrose showed that all BHs contain singularities

techniques for handling singularities

- excision
- puncture
- stuffing
- singularity avoiding gauge

initial data for black holes

- collapse scalar field & excise (Pretorius)
- thin-sandwich
  - ▷ excise (Caltech/Cornell/UMD)
  - ▷ stuff (AEI/PSU/Jena/FAU/UMD)
- puncture

Stuffing

if singularity inside: why not put a regular solution instead?

- Bona 1999, Misner 2001

problem: constraint violations are not bound by causality!

- can escape from BH & influence outside

BUT they are bound by the rules of PDEs

- figure out characteristic speeds
- show that characteristic speeds of constraint violating modes are at most c
- then constraint violating modes can't get out!

depends on evolution system used (Brown et al 2008)

potential issue of discretization

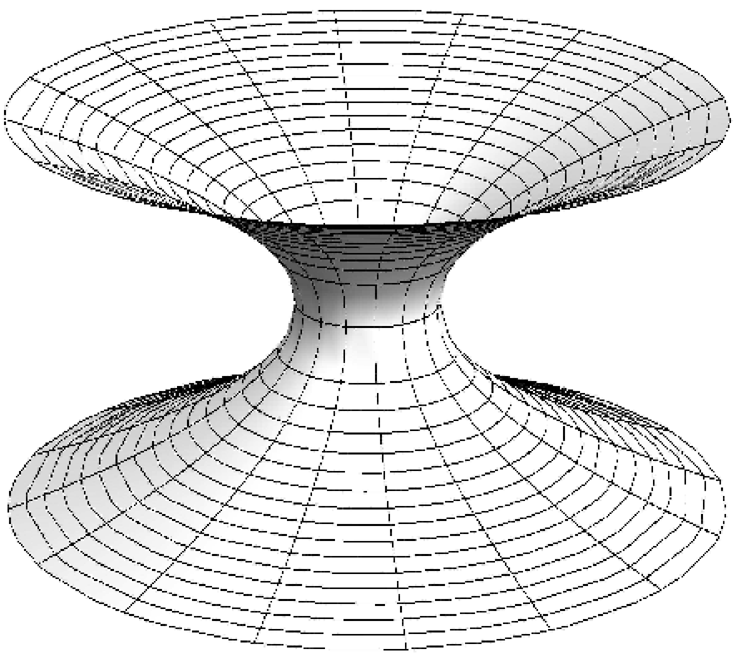
seems to work really well in practice

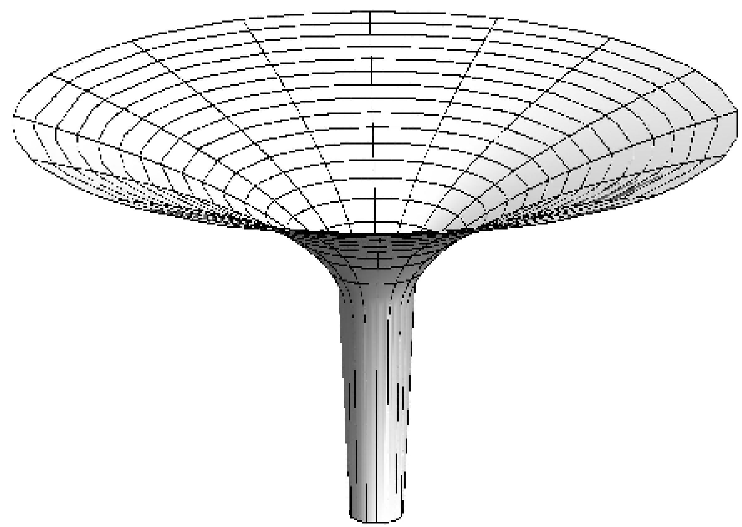
Single Puncture

Schwarzschild spacetime in isotropic coordinates

$$ds^2 = -\frac{1 - M/2r}{1 + M/2r}dt^2 + (1 + M/2r)^4(dr^2 + dS^2) \qquad r_S = r(1 + M/2r)^2$$

Wormhole becomes Trumpet. (Hannam et al 2008)





other end becomes cylindrical. $R = 1.31\,M$ (inside $R_S = 2\,M$)

Stages of BH merger

Newtonian: BHs far separated, GW emission would not lead to merger in Hubble time

- n-body interactions most likely to produce stellar mass BBHs
- probably not inspiral of stars (Belczynski et al 2007)
- gas interactions for supermassive BBHs

Inspiral: GW emission becomes dominant process, PN approximation works well

Plunge/Merger: Orbital evolution on longer adiabatic. Full numerical simulation required.

- this phase is very short, 1-2 GW cycles
- 1-10% of energy radiated

Ringdown: merger remnant settles down to single Kerr BH

- characteristic quasi-normal modes

Comparison to post-Newtonian (PN)

weak field, slow motion

$x = (\dot{\phi}^{2/3})$ orbital frequency, ϕ orbital phase

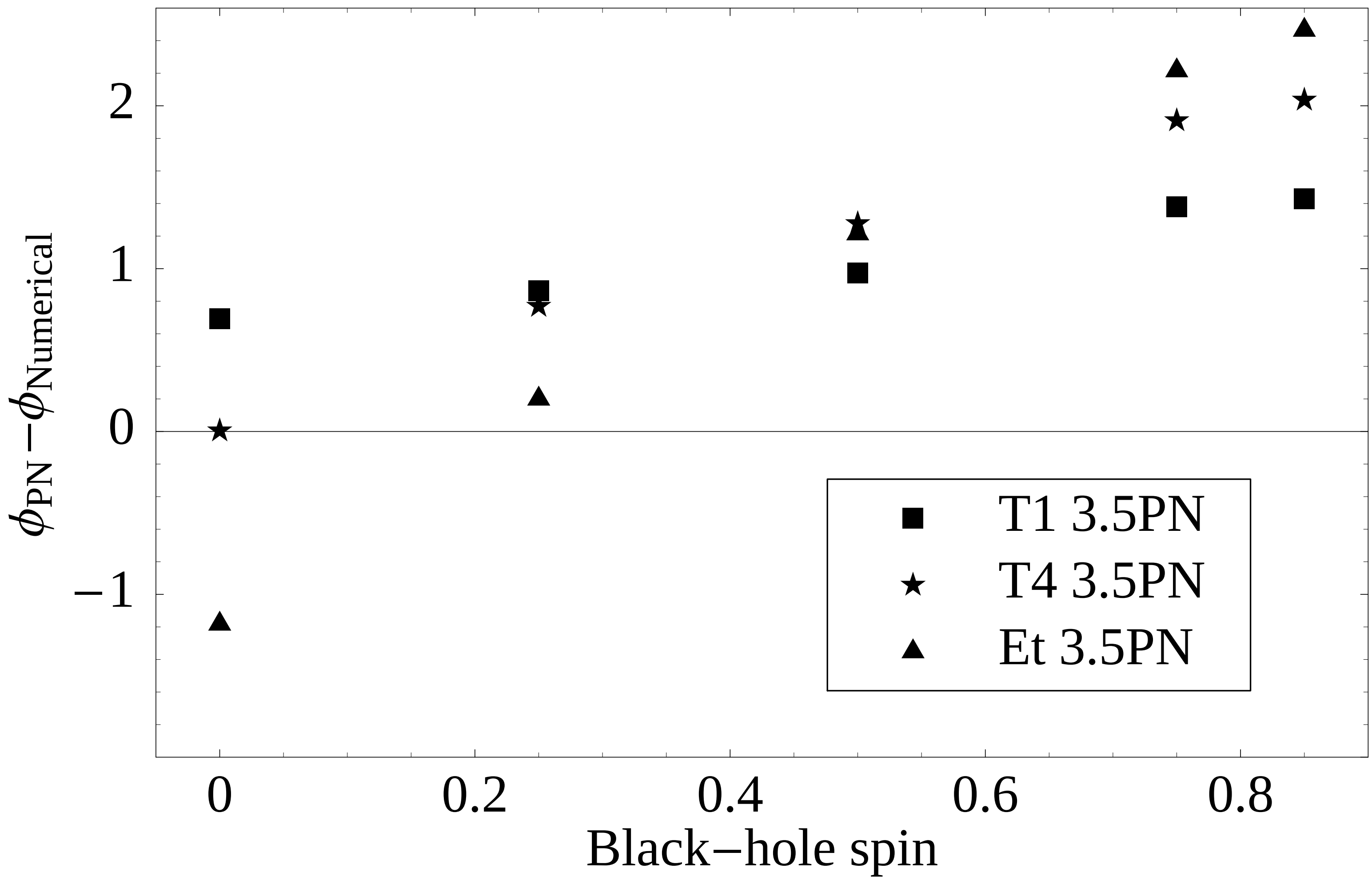
EOM for x, ϕ contain energy E, GW energy flux F

PN Taylor approximants (Damour et al 2001)

- TaylorT1: numerical integration of $dx/dt = -\frac{F}{dE/dx}$ & $d\phi/dt$
- TaylorT2: analytical integration of dx/dt & $d\phi/dt$
- TaylorT3: like T2 except introduce different variable
- TaylorT4: like T1 except $dx/dt = \texttt{Expand}\left(-\frac{F}{dE/dx}\right)$ (Buonanno, Cook, Pretorius 2007)

phase error of 0.05 radians shortly before merger ($\omega = 0.1$)

Comparison to PN: spin



comparison for different spins (Hannam et al 2008)

direct GW flux comparison shows no clear superiority of T4 (Boyle et al 2008)

Comparison to PN: eccentricity

Hinder et al 2008

PN: 3 eccentricities: e_t, e_r, e_ϕ

- represents deviations from circular motion in t, r and ϕ
- all 3 are related by PN equations (identical to Newtonian order); used e_t

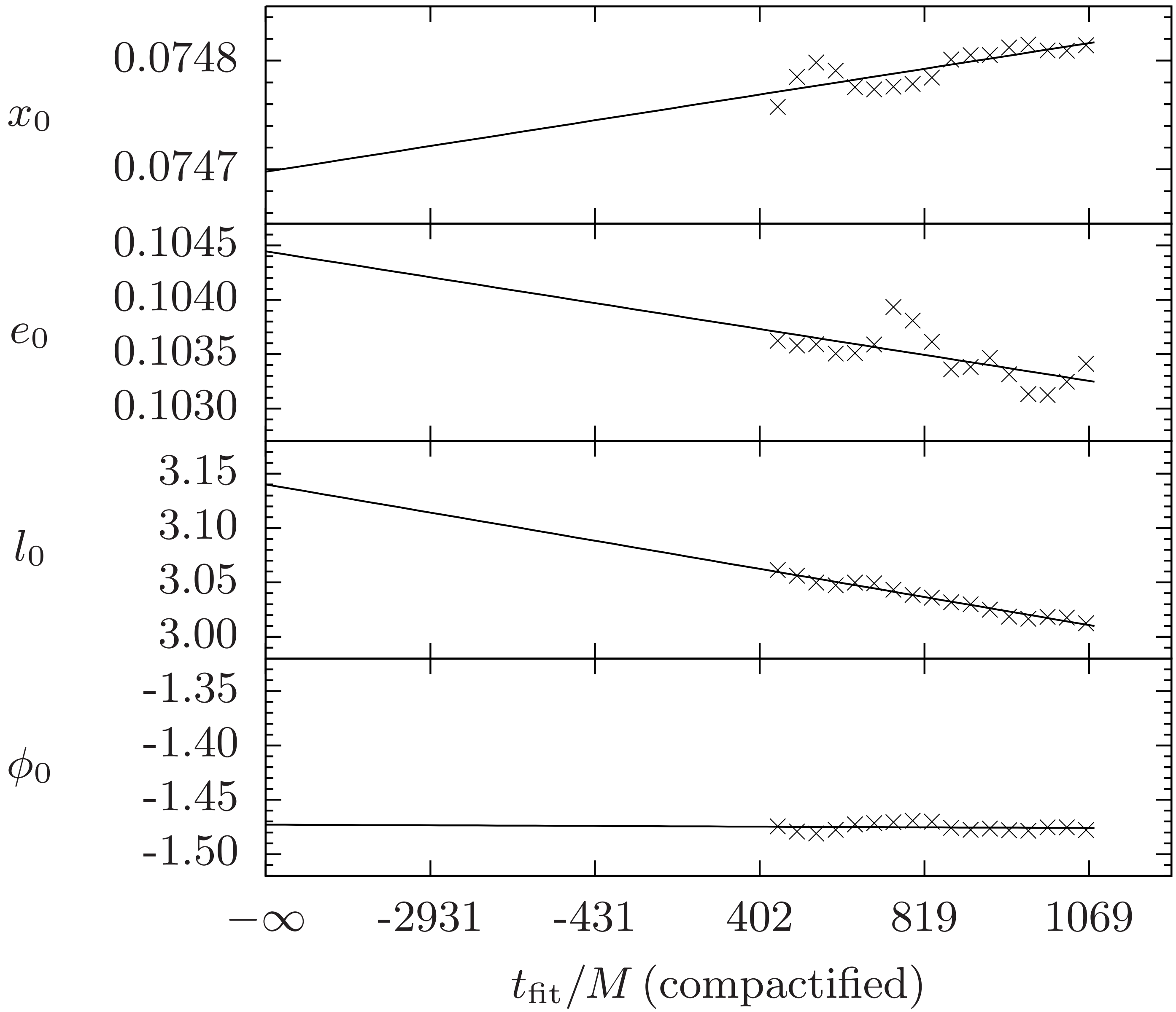
orbital period P: time to go from pericenter to pericenter

- due to precession this is not equivalent to $\phi \rightarrow \phi + 2\pi$

use $x = \omega^{2/3}$ instead of $n = 2\pi/P$ (as in circular case)

$r = a(1 - e\cos u)$, eccentric anomaly u, $l = u - e\sin u$ (Kepler's equation) mean anomaly l

Extrapolated Data



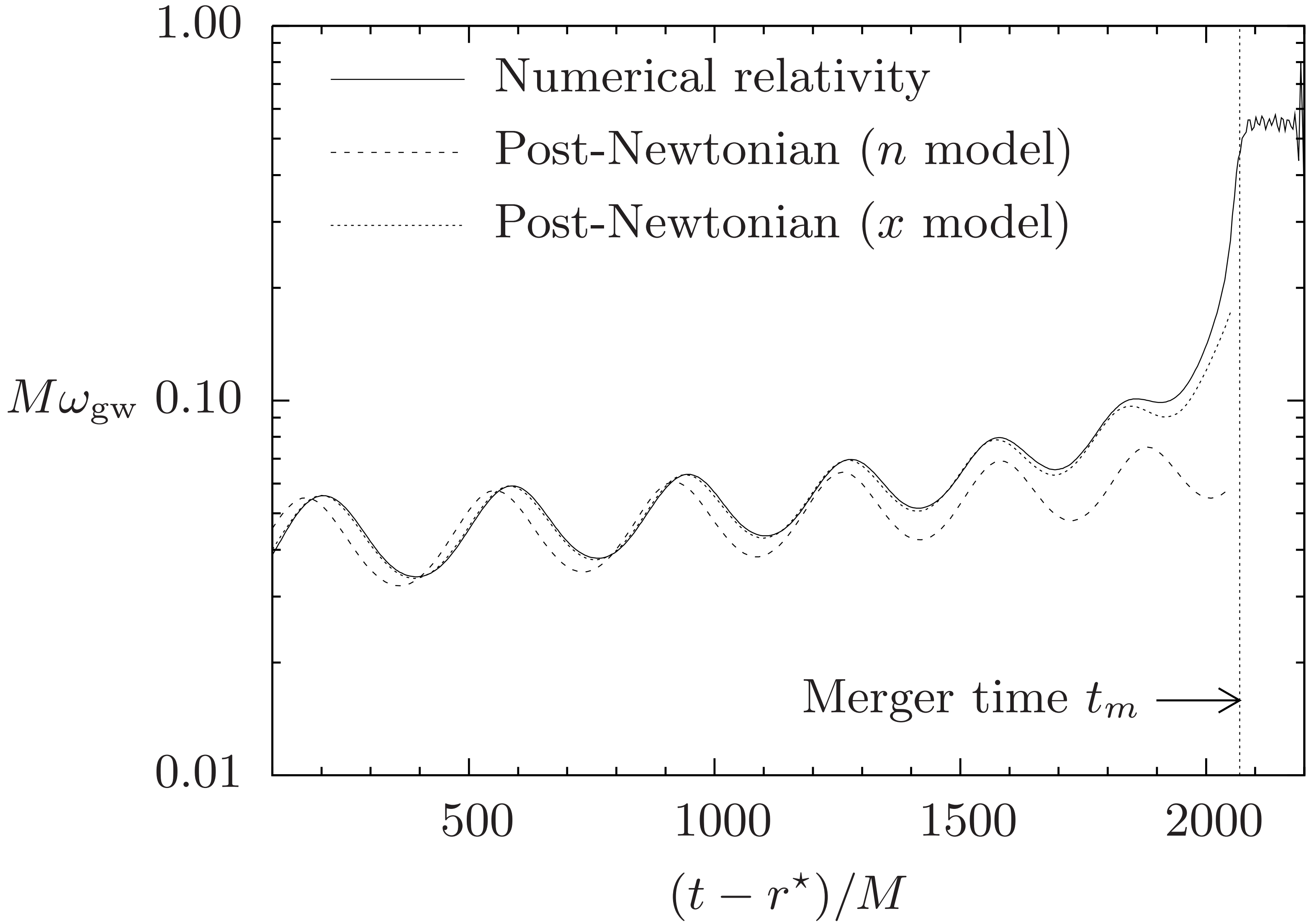
center of fitting window is modified, size is kept fixed ($\pm 250\,M$)

Extrapolated Data

Parameter	Extrapolated value	Initial data value
x_0	$0.07470(3)$	0.0740853
e_0	$0.1041(4)$	0.1
l_0	$3.14(1)$	$\pi = 3.1416$
ϕ_0	$-1.47(1)$	0

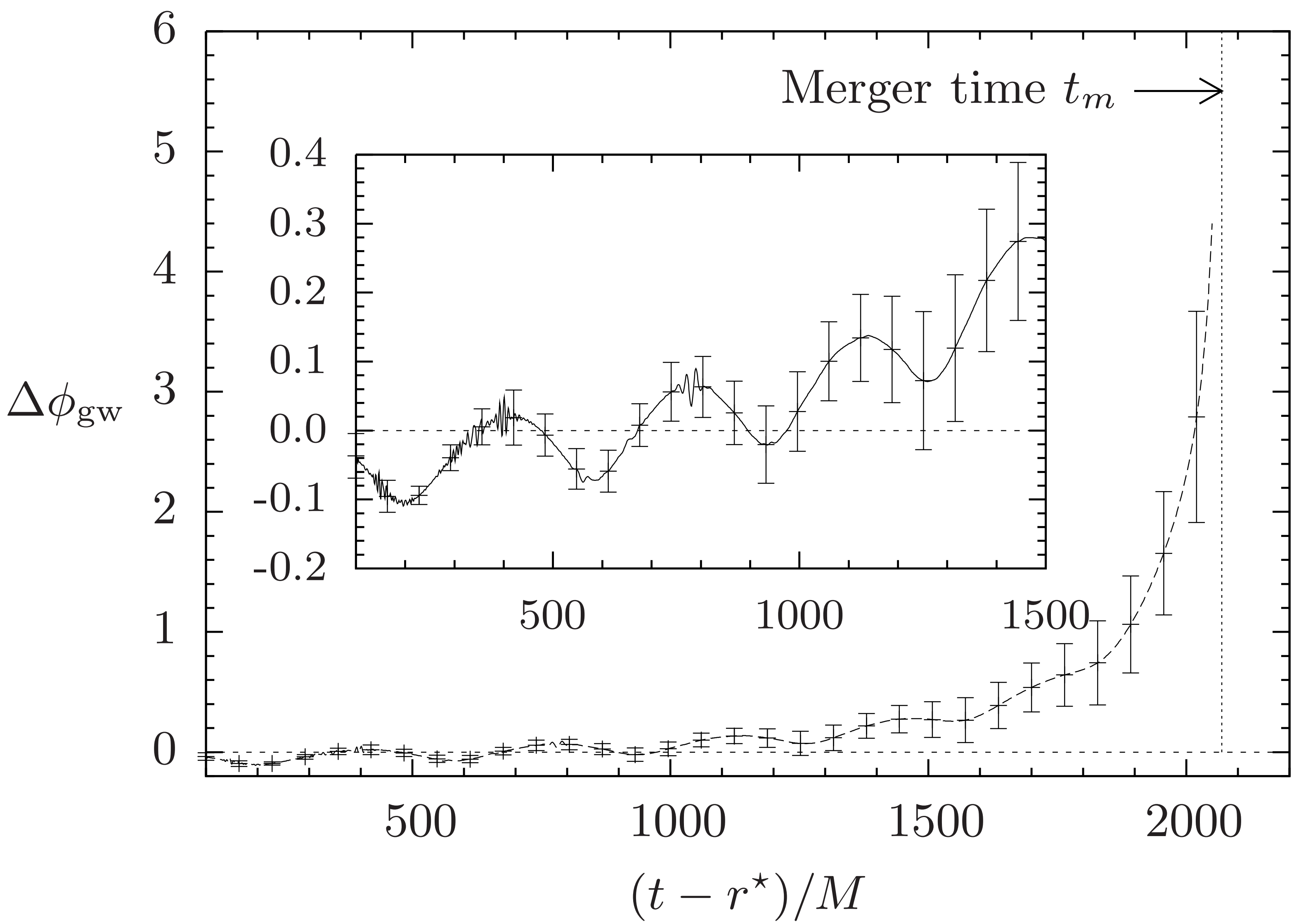
agreement is not required

Agreement in ω



$n = 2\pi/P$, eccentricity oscillations

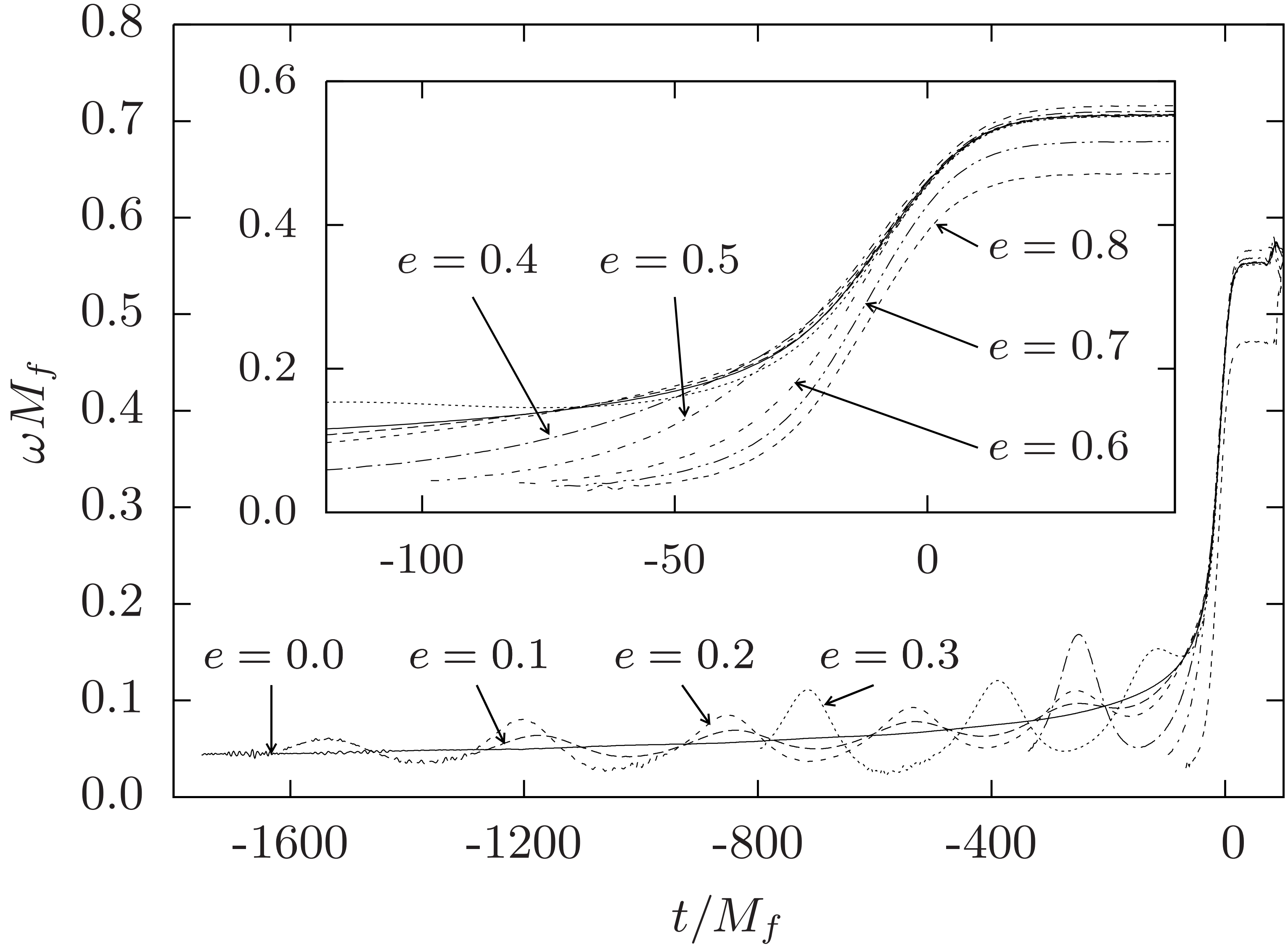
Agreement in ϕ_{GW}



at $\omega = 0.1$ there is $\Delta\phi_{GW} = 0.8$ radians. for TaylorT4 in circular: 0.3 radians for 2PN, (0.05 radians at 3.5PN)

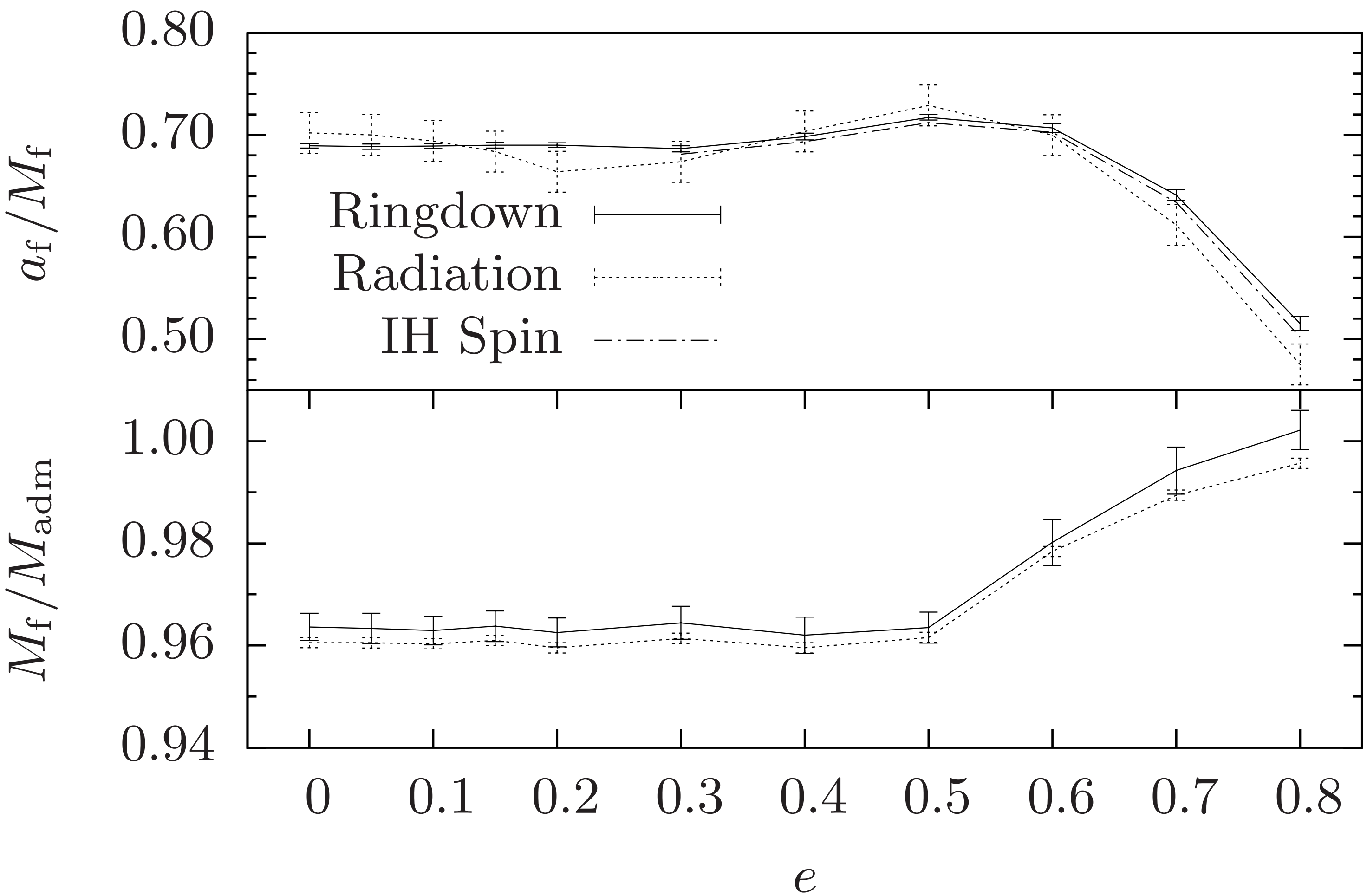
Eccentricity lost in merger

eccentricity radiated away



Final BH has no memory of inspiral

eccentricity radiated away



Gravitational Recoil

asymmetric radiation of GW can carry net linear momentum

recoil can come from unequal masses or from spins

- unequal-mass recoil up to 170 km/s (Gonzales et al 2007) $\propto \eta^2$ ($\eta = m_1 m_2/(m_1 + m_2)^2$)
- spin recoil $\propto S_1 - S_2$

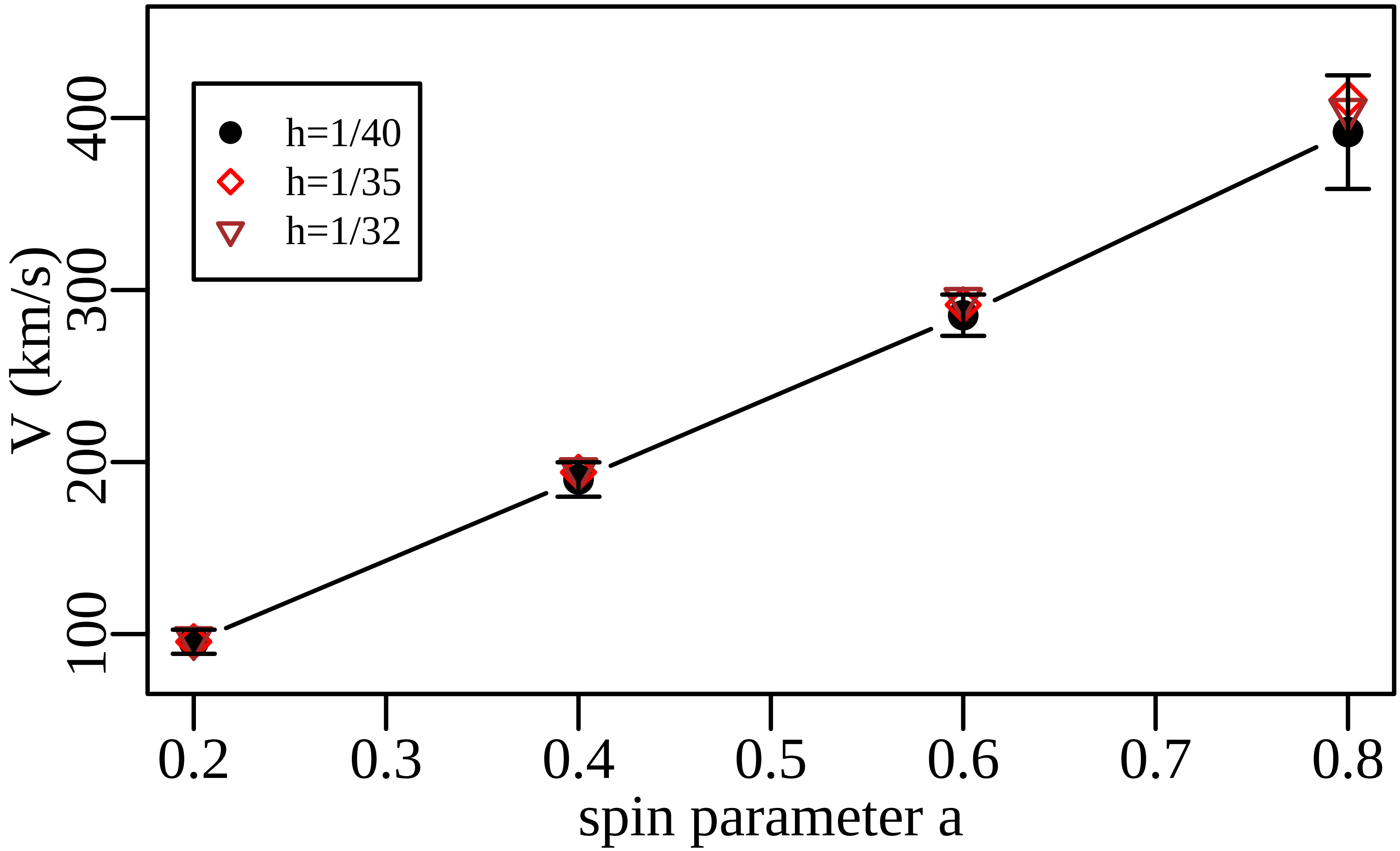
little recoil in inspiral (radiated P^i rotates around)

equal-mass spin recoil can get large

- 4000 km/s for circular inspiral Campanelli et al, Hannam et al 2007
- > 10.000 km/s for eccentric mergers (so far largest found), Healy et al 2008

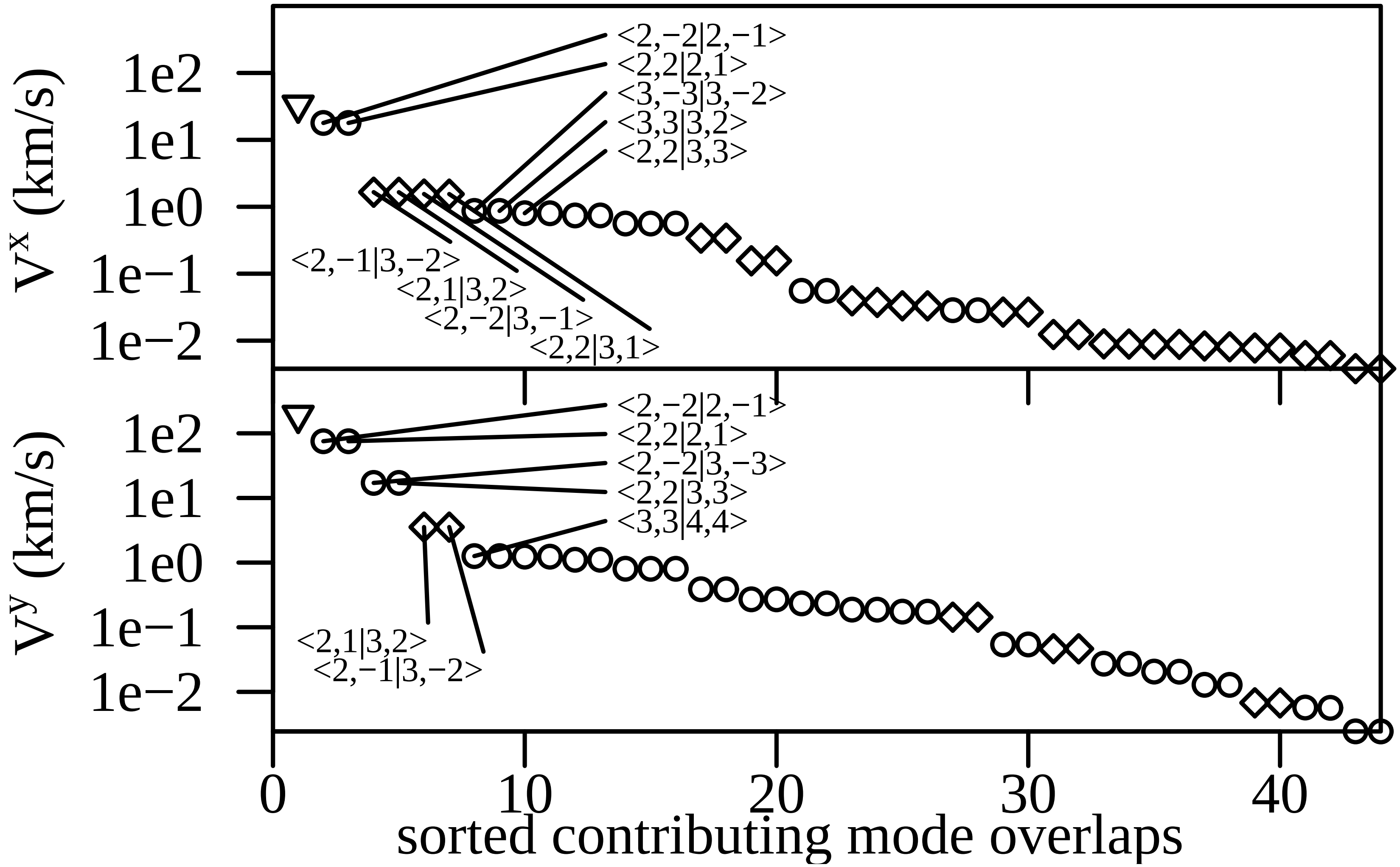
supermassive BH could get kicked out of galaxy (or at least displaced)

Recoil velocity vs. a/m



linear scaling in a/M (as predicted by PN (Kidder 1995)).
maximum recoil ≈ 475 km/s in-plane (FH et al 2007)

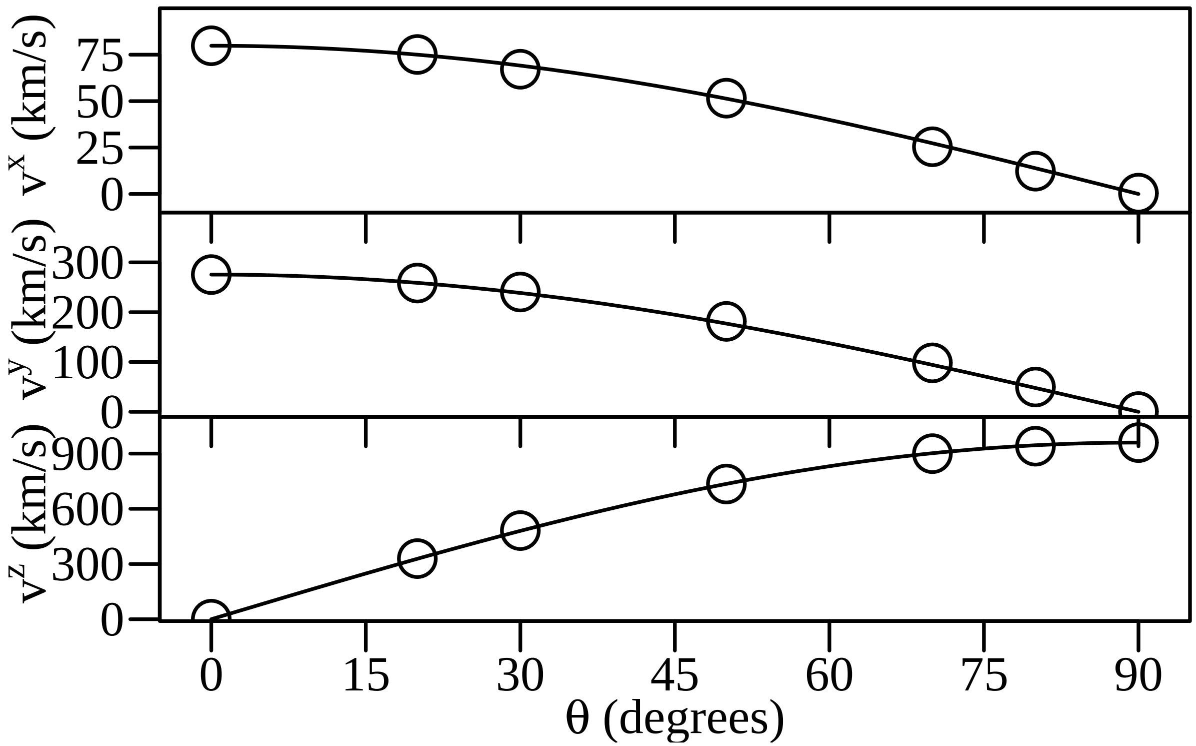
Individual Mode contributions



Mode Overlaps contribute, some are negative.

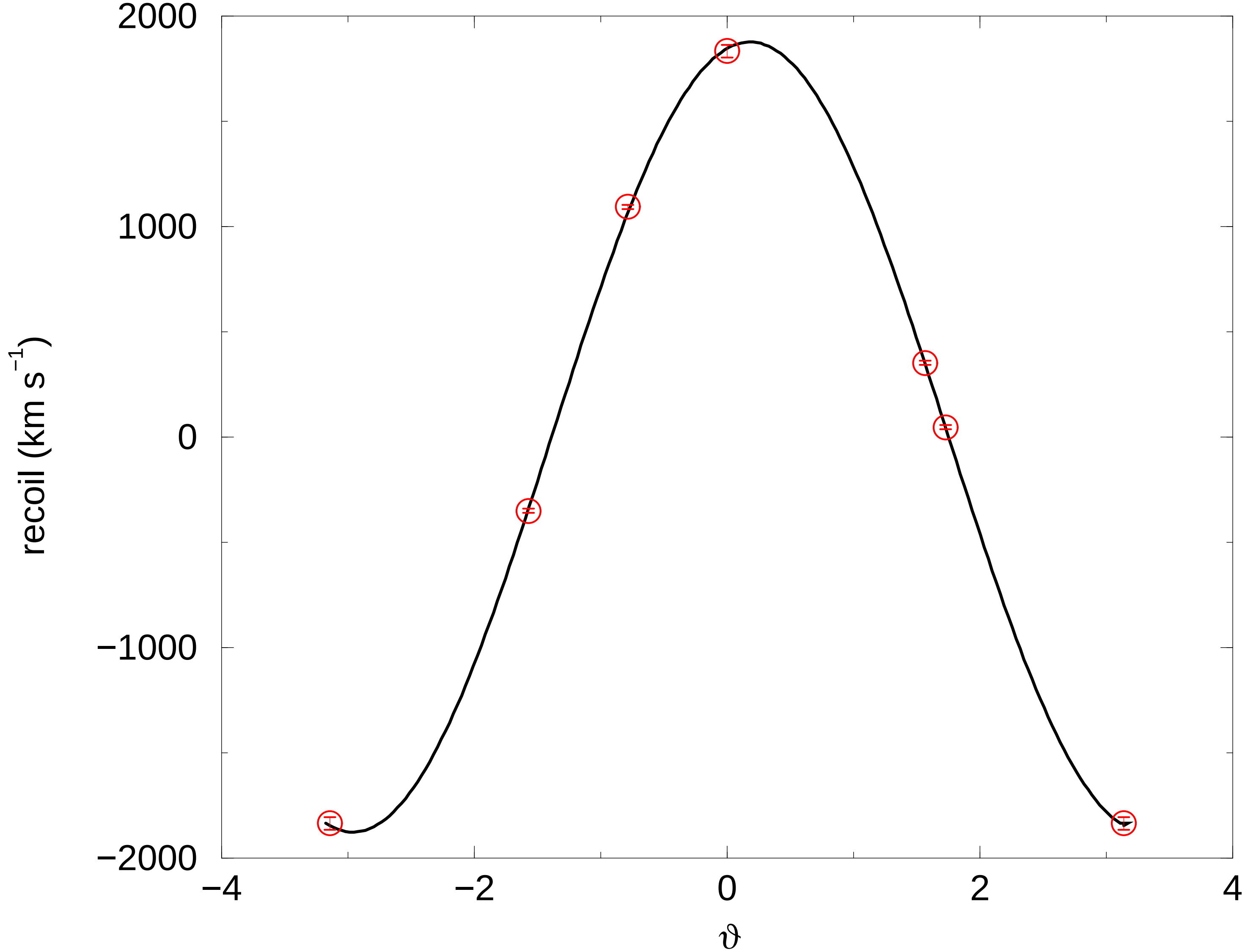
Superkick configuration

off-plane angle (FH et al 2007)

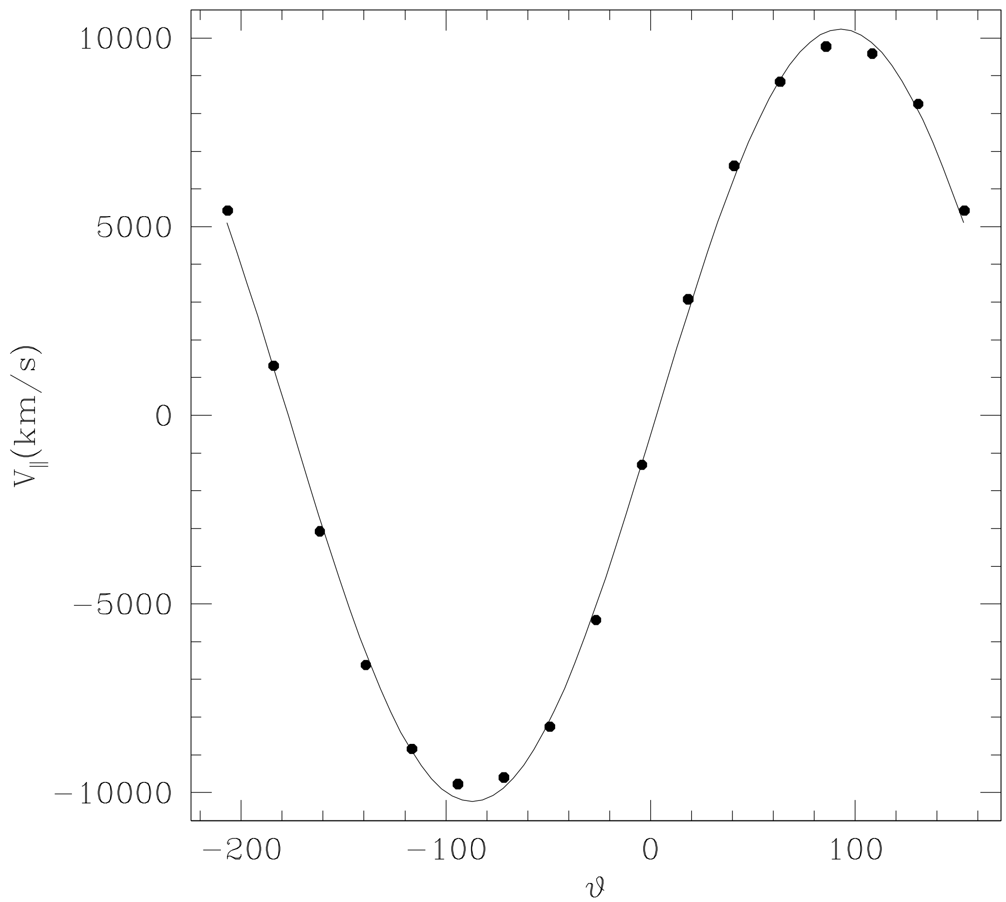


Superkick configuration

S^i in-plane (different angle) (Campanelli et al, Hannam et al 2007)

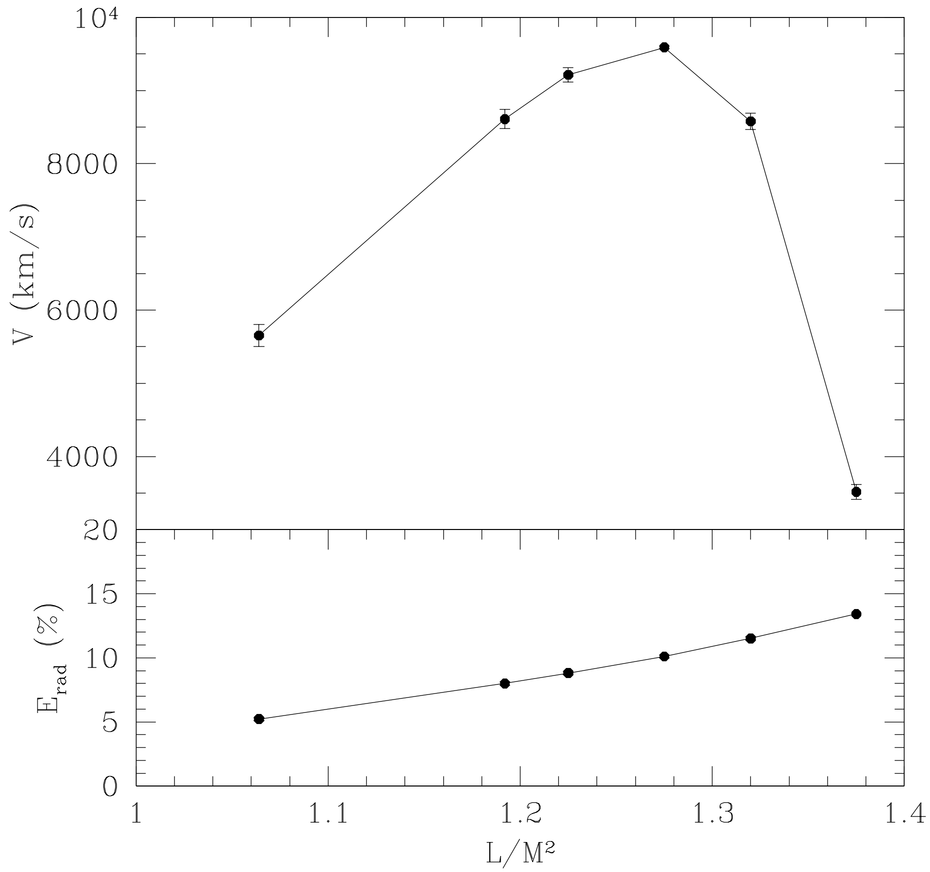


Recoil from hyperbolic mergers



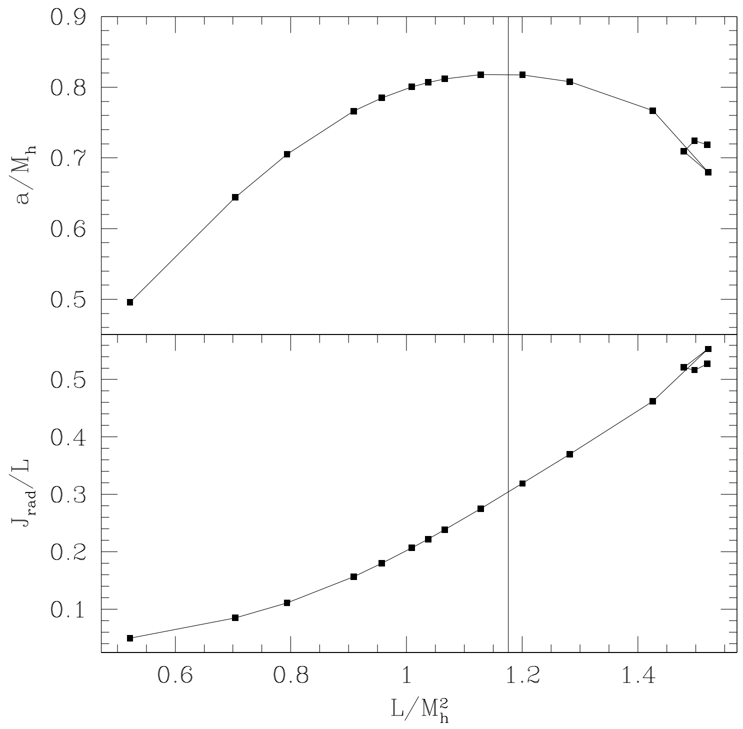
- also sin dependence on angle (Healy et al 2008)

Recoil from hyperbolic mergers



note that $v_{\mathtt{max}}$ and $\mathtt{E}_{\mathtt{max}}$ do not coincide (Healy et al 2008)

How easy is it to produce high-spin BH in merger



- surprisingly difficult due to enormous angular momentum radiation (up to 55%) (Washik et al 2008)
- upper limit of $a/M \approx 0.82$ for this series ($x = \pm 5$, $P \in [0.1, 0.3]$)
- gas accretion does not excite low ℓ modes which carry angular momentum away

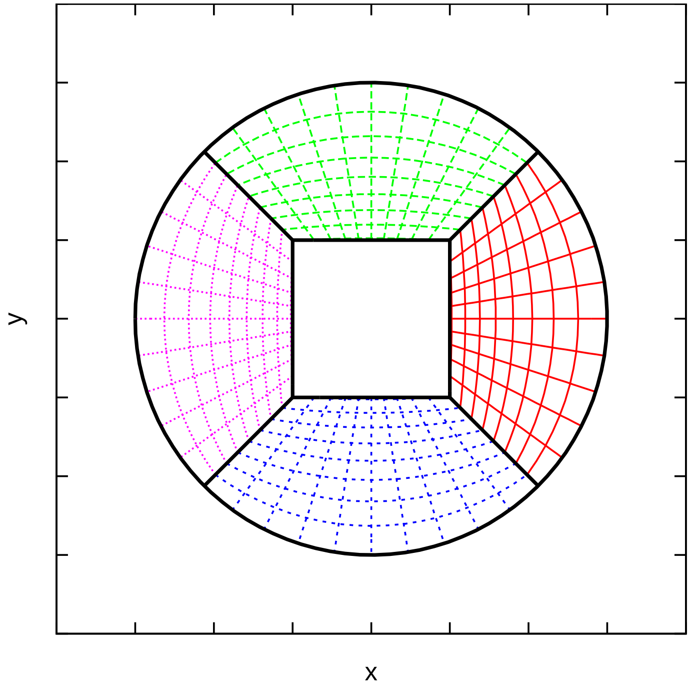
Multi-Patch Finite Difference

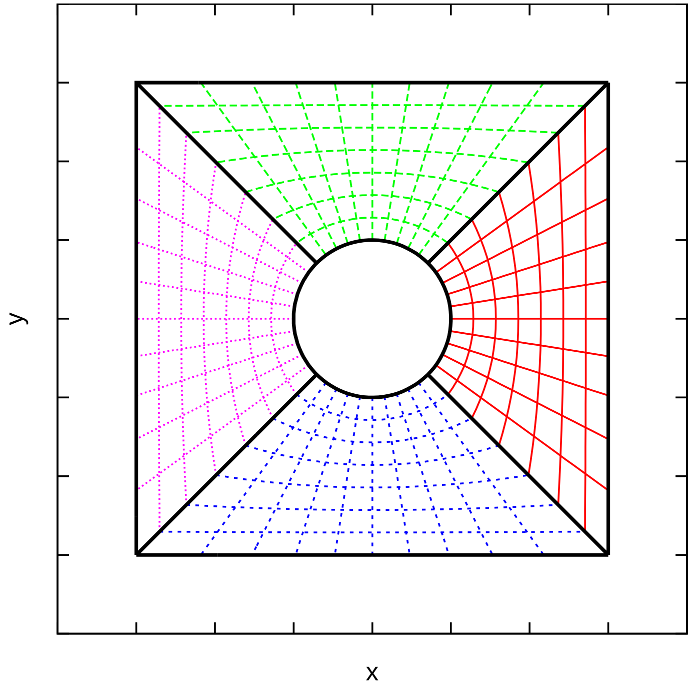
binary BH inspiral using SpEC infrastructure (Pazos et al 2009)

instead of pseudospectral (PS) use high-order finite difference (FD)

- excellent scaling due to box splitting method & communication through boundaries
  - ▷ $Y_{\ell m}$ basis for PS hard to parallelize
- memory issues
  - ▷ PS: very memory efficient, so store everything in memory
  - ▷ FD: much larger memory requirements

Blocks



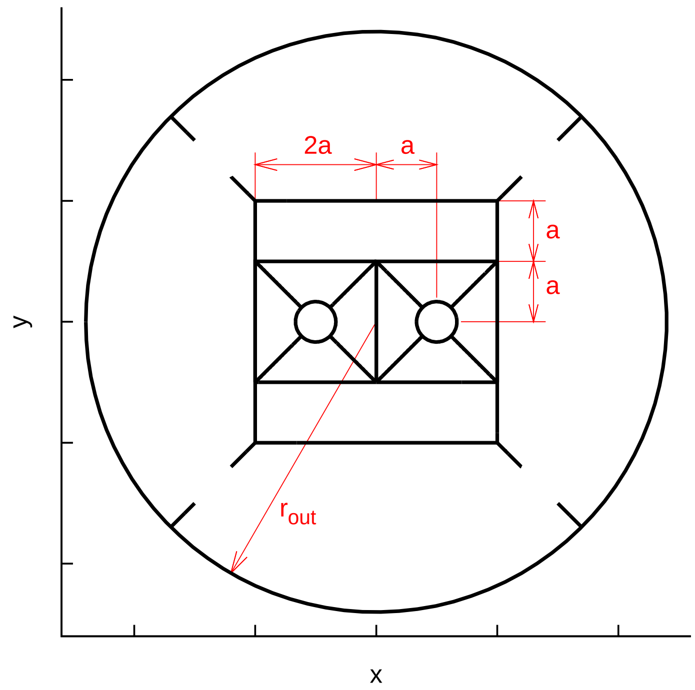


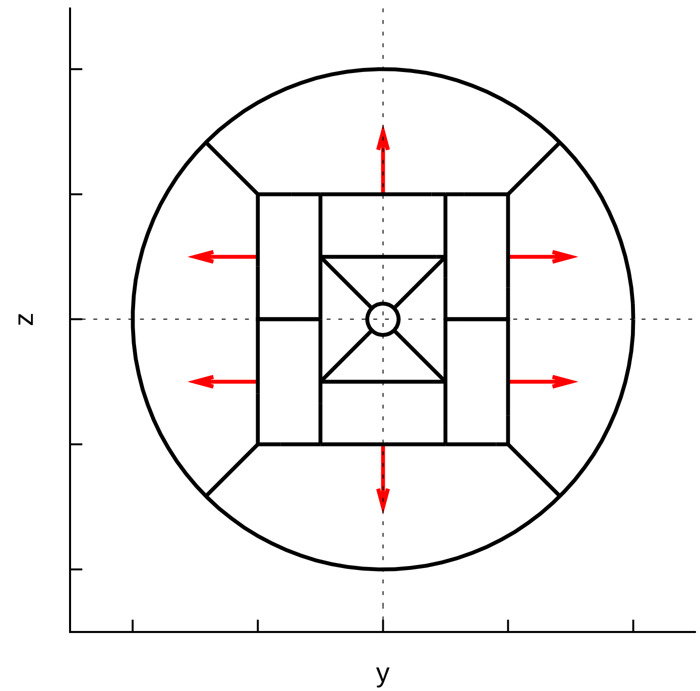
left shows inner juggling ball, right shows outer juggling ball

- minimum 6 blocks

for more CPUs: just split blocks

Blocks



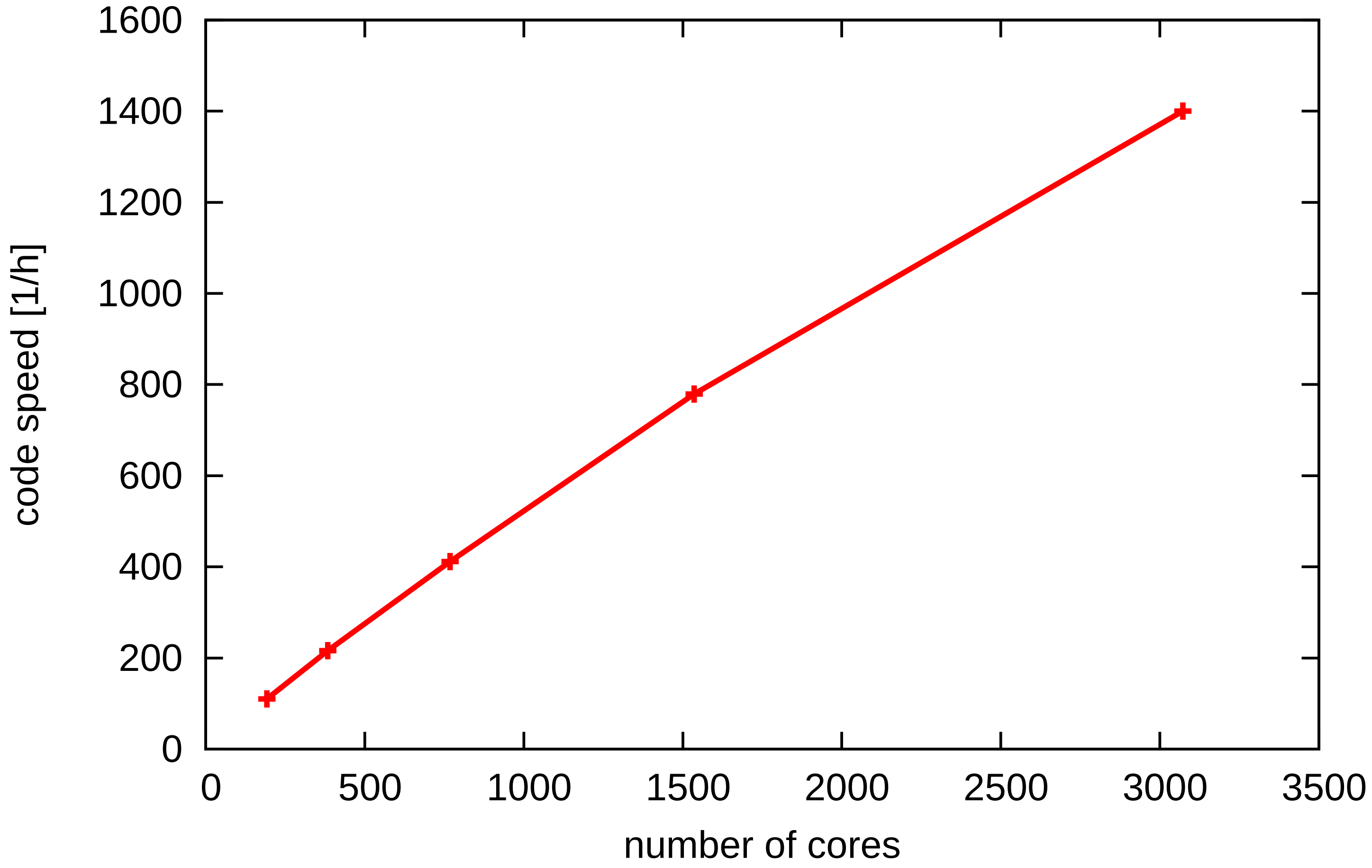


building up domain from multiple blocks

- $O(N)$ scaling of points in radial direction

24 basic blocks: actually used 192 & 384

Strong Scaling



single bh is evolved on different number of CPUs (Pazos et al 2009)

speed grows almost linearly

Multi-Patch BH-NS Inspirals

work done mainly by Matt Duez & Enrique Pazos

BH-NS system

- one of the (very) few systems where $T_{\mu\nu} \neq 0$ “for real”
- initial studies have been performed
- Duez et al 2009 used PS for $g_{\mu\nu}$ and FD for $T_{\mu\nu}$
  - ▷ technique pioneered by Dimmelmeier et al
  - ▷ best of all worlds (except for interpolation)

switch to full finite difference, i.e. $g_{\mu\nu}$ too

- avoid interpolation
- currently able to run

Graphics Processing Units - GPU

with Harald Pfeiffer & John Silberholz

GPU is chip on the graphics cards

GPU is a collection (240) of (fairly) slow compute units

aggregate performance over all is impressive

- 1 TFlops for GPU
- compare: currently typically 8 GFlops for single CPU core!

problems

- single-precision
  - ▷ paradox: care more about numerical method, but libraries often assume exact opposite
- fairly low memory per card $\longrightarrow$ good for pseudospectral

started to work on implementation

Conclusions

numerical relativity can provide accurate simulations

template building can be guided by these results

- aim for model, example EOB
- some of these models work extremely well, in-particular if extra parameters are introduced and fit by comparison to NR

recoil

- very large recoils now actively searched for

rich parameter space still needs much further exploration

matter simulations can profit from BH advances

- electro-magnetic fields in merger region. Palenzuela et al 2009
- gas flows in merger region (geodesics). van Meter et al 2009