2nd-order Gravitational Self-Force in Schwarzschild: Mode Decomposition of the 1st-Order Puncture

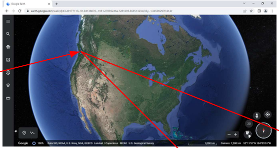
Jonathan Thornburg

Department of Astronomy and Center for Spacetime Symmetries

Indiana University

and

currently on a small island off the west coast of Canada



Work done as part of **2SF group**:
Patrick Bourg, Leanne Durkan, Conor Dyson, Benjamin Leather, Rodrigo Panosso Macedo, Zachary Nasipak, Adam Pound, Andrew Spiers, Jonathan Thornburg, Samuel Upton, Maarten van de Meent, Niels Warburton, Barry Wardell

2nd-order GSF in Schwarzschild: the Big Picture

Puncture scheme near the particle:

1st order puncture:
$^{(1)}h_{ab}^{(\text{puncture})}$

Terminology:

- Penrose abstract-index notation
- ab are tensor indices
- $^{(n)}h_{ab}$ is the nth-order metric perturbation

2nd-order GSF in Schwarzschild: the Big Picture

Puncture scheme near the particle, with mode-sum decomposition:

1st order puncture:
${}^{(1)}h_{ab}^{(\text{puncture})}$

$\Big\downarrow$ compute mode decomposition of 1st-order puncture

Barack-Lousto-Sago modes of the 1st-order puncture:
$\left({}^{(1)}h_{ab}^{(\text{puncture})}\right)^{\mathrm{I}\ell m}$

Terminology:

- Penrose abstract-index notation
- ab are tensor indices
- ${}^{(n)}h_{ab}$ is the n th-order metric perturbation
- I labels the Barack-Lousto-Sago mode
- ℓm label the spherical harmonic

2nd-order GSF in Schwarzschild: the Big Picture

Puncture scheme near the particle, with mode-sum decomposition:

1st order puncture:
${}^{(1)}h_{ab}^{\text{(puncture)}}$

$\Big\downarrow$ compute mode decomposition of 1st-order puncture

Barack-Lousto-Sago modes of the 1st-order puncture:
$\left({}^{(1)}h_{ab}^{\text{(puncture)}}\right)^{\mathrm{I}\ell m}$

$\Big\downarrow$ compute 1st-order metric perturbation (mode-by-mode)

Barack-Lousto-Sago modes of the 1st-order metric perturbation: $\left({}^{(1)}h_{ab}\right)^{\mathrm{I}\ell m}$

Terminology:

- Penrose abstract-index notation
- ab are tensor indices
- ${}^{(n)}h_{ab}$ is the nth-order metric perturbation
- I labels the Barack-Lousto-Sago mode
- ℓm label the spherical harmonic

2nd-order GSF in Schwarzschild: the Big Picture

Puncture scheme near the particle, with mode-sum decomposition:

1st order puncture:
$^{(1)}h_{ab}^{\text{(puncture)}}$

compute mode decomposition of 1st-order puncture

Barack-Lousto-Sago modes of the 1st-order puncture: $\left({}^{(1)}h_{ab}^{\text{(puncture)}}\right)^{\mathrm{I}\ell m}$

compute 1st-order metric perturbation (mode-by-mode)

Barack-Lousto-Sago modes of the 1st-order metric perturbation: $\left({}^{(1)}h_{ab}\right)^{\mathrm{I}\ell m}$

compute 2nd-order metric perturbation (mode-by-mode)

Barack-Lousto-Sago modes of the 2nd-order metric perturbation: $\left({}^{(2)}h_{ab}\right)^{\mathrm{I}\ell m}$

Terminology:

- Penrose abstract-index notation
- ab are tensor indices
- $^{(n)}h_{ab}$ is the n th-order metric perturbation
- I labels the Barack-Lousto-Sago mode
- ℓm label the spherical harmonic

2nd-order GSF in Schwarzschild: the Big Picture

Puncture scheme near the particle, with mode-sum decomposition:

1st order puncture:
${}^{(1)}h_{ab}^{(\text{puncture})}$

compute mode decomposition of 1st-order puncture

Barack-Lousto-Sago modes of the 1st-order puncture: $\left({}^{(1)}h_{ab}^{(\text{puncture})}\right)^{\mathrm{I}\ell m}$

compute 1st-order metric perturbation (mode-by-mode)

Barack-Lousto-Sago modes of the 1st-order metric perturbation: $\left({}^{(1)}h_{ab}\right)^{\mathrm{I}\ell m}$

compute 2nd-order metric perturbation (mode-by-mode)

Barack-Lousto-Sago modes of the 2nd-order metric perturbation: $\left({}^{(2)}h_{ab}\right)^{\mathrm{I}\ell m}$

Terminology:

- Penrose abstract-index notation
- ab are tensor indices
- ${}^{(n)}h_{ab}$ is the n th-order metric perturbation
- I labels the Barack-Lousto-Sago mode
- ℓm label the spherical harmonic

Mode decomposition of the 1st-order puncture

Conceptually, computing the mode decomposition is easy:
the Barack-Lousto-Sago modes $Y_{ab}^{\mathrm{I}\ell m}$ are orthogonal, so

$$\left({}^{(1)}h_{ab}^{\text{(puncture)}}\right)^{\mathrm{I}\ell m} = \int {}^{(1)}h_{ab}^{\text{(puncture)}} \; Y_{ab}^{\mathrm{I}\ell m} \, d\Omega \qquad \text{for each } \mathrm{I}, \ell, m$$

Mode decomposition of the 1st-order puncture

Conceptually, computing the mode decomposition is easy:
the Barack-Lousto-Sago modes $Y_{ab}^{\mathrm{I}\ell m}$ are orthogonal, so

$$\left({}^{(1)}h_{ab}^{\text{(puncture)}}\right)^{\mathrm{I}\ell m} = \boxed{\int {}^{(1)}h_{ab}^{\text{(puncture)}} \, Y_{ab}^{\mathrm{I}\ell m} \, d\Omega} \quad \text{for each } \mathrm{I}, \ell, m$$

The problem is doing the integrals.

Mode decomposition of the 1st-order puncture

Conceptually, computing the mode decomposition is easy:
the Barack-Lousto-Sago modes $Y_{ab}^{\mathrm{I}\ell m}$ are orthogonal, so

$$\left({}^{(1)}h_{ab}^{(\text{puncture})}\right)^{\mathrm{I}\ell m} = \boxed{\int {}^{(1)}h_{ab}^{(\text{puncture})} \, Y_{ab}^{\mathrm{I}\ell m} \, d\Omega} \quad \text{for each } \mathrm{I}, \ell, m$$

The problem is doing the integrals.

For technical reasons, we switch from the usual polar spherical coordinates (θ, ϕ) to (time-dependent) rotated coordinates (α, β) chosen so that the particle's instantaneous position is at the rotated north pole $\alpha = 0$.

Mode decomposition of the 1st-order puncture

Conceptually, computing the mode decomposition is easy:
the Barack-Lousto-Sago modes $Y_{ab}^{\mathrm{I}\ell m}$ are orthogonal, so

$$\left({}^{(1)}h_{ab}^{\text{(puncture)}}\right)^{\mathrm{I}\ell m} = \boxed{\int {}^{(1)}h_{ab}^{\text{(puncture)}} \; Y_{ab}^{\mathrm{I}\ell m} \, d\Omega} \quad \text{for each } \mathrm{I}, \ell, m$$

The problem is doing the integrals.

For technical reasons, we switch from the usual polar spherical coordinates (θ, ϕ) to (time-dependent) rotated coordinates (α, β) chosen so that the particle's instantaneous position is at the rotated north pole $\alpha = 0$.

The α integral ($\int_0^\pi d\alpha$) is easy to do analytically.

Mode decomposition of the 1st-order puncture

Conceptually, computing the mode decomposition is easy:
the Barack-Lousto-Sago modes $Y_{ab}^{\mathrm{I}\ell m}$ are orthogonal, so

$$\left({}^{(1)}h_{ab}^{\text{(puncture)}}\right)^{\mathrm{I}\ell m} = \boxed{\int {}^{(1)}h_{ab}^{\text{(puncture)}} \; Y_{ab}^{\mathrm{I}\ell m} \, d\Omega} \quad \text{for each } \mathrm{I}, \ell, m$$

The problem is doing the integrals.

For technical reasons, we switch from the usual polar spherical coordinates (θ, ϕ) to (time-dependent) rotated coordinates (α, β) chosen so that the particle's instantaneous position is at the rotated north pole $\alpha = 0$.

The α integral ($\int_0^\pi d\alpha$) is easy to do analytically.

The β integral ($\int_0^{2\pi} d\beta$) is harder:

- Compute it numerically $\Rightarrow$ straightforward but **slow**

Mode decomposition of the 1st-order puncture

Conceptually, computing the mode decomposition is easy:
the Barack-Lousto-Sago modes $Y^{\mathrm{I}\ell m}_{ab}$ are orthogonal, so

$$\left({}^{(1)}h^{\text{(puncture)}}_{ab}\right)^{\mathrm{I}\ell m} = \boxed{\int {}^{(1)}h^{\text{(puncture)}}_{ab} \; Y^{\mathrm{I}\ell m}_{ab} \, d\Omega} \quad \text{for each } \mathrm{I}, \ell, m$$

The problem is doing the integrals.

For technical reasons, we switch from the usual polar spherical coordinates (θ, ϕ) to (time-dependent) rotated coordinates (α, β) chosen so that the particle's instantaneous position is at the rotated north pole $\alpha = 0$.

The α integral ($\int_0^\pi d\alpha$) is easy to do analytically.

The β integral ($\int_0^{2\pi} d\beta$) is harder:

- Compute it numerically $\Rightarrow$ straightforward but **slow**
- Compute it analytically $\Rightarrow$ **this talk**

How many β integrals are there?

Each "β integral" is actually a set of integrals,
one for each Barack-Lousto-Sago tensor mode and (ℓ, m):

- there are 10 Barack-Lousto-Sago tensor modes $(1 \leq \mathrm{I} \leq 10)$

How many β integrals are there?

Each "β integral" is actually a set of integrals,
one for each Barack-Lousto-Sago tensor mode and (ℓ, m):

- there are 10 Barack-Lousto-Sago tensor modes ($1 \le \mathrm{I} \le 10$)
- we typically compute for (ℓ, m) in the range $0 \le \ell \lesssim 50$, $0 \le m \lesssim 10$
- integrals vanish by symmetry for about $\frac{1}{2}$ of (ℓ, m)

$\Rightarrow$ There are about 2500 individual integrals in each "β integral" set.

How many β integrals are there?

Each "β integral" is actually a set of integrals,
one for each Barack-Lousto-Sago tensor mode and (ℓ, m):

- there are 10 Barack-Lousto-Sago tensor modes ($1 \le \mathrm{I} \le 10$)
- we typically compute for (ℓ, m) in the range $0 \le \ell \lesssim 50$, $0 \le m \lesssim 10$
- integrals vanish by symmetry for about $\frac{1}{2}$ of (ℓ, m)

$\Rightarrow$ There are about 2500 individual integrals in each "β integral" set.

Each β integral depends on 2 parameters:

$$r_0 = \text{particle orbit radius (assume circular orbit for now)}$$
$$\Delta r = r \text{ of field (evaluation) point} - r_0$$

How many β integrals are there?

Each "β integral" is actually a set of integrals,
one for each Barack-Lousto-Sago tensor mode and (ℓ, m):

- there are 10 Barack-Lousto-Sago tensor modes $(1 \le \mathrm{I} \le 10)$
- we typically compute for (ℓ, m) in the range $0 \le \ell \lesssim 50$, $0 \le m \lesssim 10$
- integrals vanish by symmetry for about $\frac{1}{2}$ of (ℓ, m)

$\Rightarrow$ There are about 2500 individual integrals in each "β integral" set.

Each β integral depends on 2 parameters:

r_0 = particle orbit radius (assume circular orbit for now)

$\Delta r = r$ of field (evaluation) point $- r_0$

We want to compute the 2nd-order self-force for at least 10–100 r_0 values.

How many β integrals are there?

Each "β integral" is actually a set of integrals,
one for each Barack-Lousto-Sago tensor mode and (ℓ, m):

- there are 10 Barack-Lousto-Sago tensor modes ($1 \le \mathrm{I} \le 10$)
- we typically compute for (ℓ, m) in the range $0 \le \ell \lesssim 50$, $0 \le m \lesssim 10$
- integrals vanish by symmetry for about $\frac{1}{2}$ of (ℓ, m)

$\Rightarrow$ There are about 2500 individual integrals in each "β integral" set.

Each β integral depends on 2 parameters:

$$r_0 = \text{particle orbit radius (assume circular orbit for now)}$$
$$\Delta r = r \text{ of field (evaluation) point} - r_0$$

We want to compute the 2nd-order self-force for at least 10–100 r_0 values.

Each self-force computation requires numerically evaluating
the "β integral" set on a grid of 100–1000 Δr values.
$\Rightarrow$ Need 10^3–10^5 numerical evaluations of each of the $\sim$2500 individual integrals

Typical form of an individual integrand

For the I=1 Barack-Lousto-Sago mode, the $\ell = 0$, $m = 0$ integrand is:

$$
\begin{aligned}
I_{1,00} = & \frac{P_3(\sin^2\beta)\ P_6^{(1)}\!\left(1 - \dfrac{M\sin^2\beta}{r_0 - 2M}\right)}{\left(r_0 - 2M - M\sin^2\beta\right)^{5/2}} \\
& \times \left[P_6^{(2)}\!\left(\frac{1}{1 - \dfrac{M\sin^2\beta}{r_0 - 2M}}\right) \left(\frac{K_1}{\left(1 - \dfrac{M\sin^2\beta}{r_0 - 2M}\right)^{3/2}} + \left(K_2 + \frac{K_3}{1 - \dfrac{M\sin^2\beta}{r_0 - 2M}} \right)^{3/2} \right) \right]
\end{aligned}
$$

where each K_i is a "constant" and each P_k or $P_k^{(i)}$ is a polynomial of degree k. The "constants" K_i and the polynomial coefficients all depend on the parameters r_0, and Δr.

Typical form of an individual integrand

For the I=1 Barack-Lousto-Sago mode, the $\ell = 0$, $m = 0$ integrand is:

$$I_{1,00} = \frac{P_3(\sin^2\beta)\ P_6^{(1)}\!\left(1 - \dfrac{M\sin^2\beta}{r_0 - 2M}\right)}{\left(r_0 - 2M - M\sin^2\beta\right)^{5/2}} \times \left[P_6^{(2)}\!\left(\frac{1}{1 - \dfrac{M\sin^2\beta}{r_0 - 2M}}\right) \left(\frac{K_1}{\left(1 - \dfrac{M\sin^2\beta}{r_0 - 2M}\right)^{3/2}} + \left(K_2 + \frac{K_3}{1 - \dfrac{M\sin^2\beta}{r_0 - 2M}}\right)^{3/2} \right) + \text{7 more terms} \right]$$

where each K_i is a "constant" and each P_k or $P_k^{(i)}$ is a polynomial of degree k. The "constants" K_i and the polynomial coefficients all depend on the parameters r_0, and Δr.

Typical form of an individual integrand

For the I=1 Barack-Lousto-Sago mode, the $\ell = 0$, $m = 0$ integrand is:

$$I_{1,00} = \frac{P_3(\sin^2\beta)\; P_6^{(1)}\!\left(1 - \dfrac{M\sin^2\beta}{r_0 - 2M}\right)}{\left(r_0 - 2M - M\sin^2\beta\right)^{5/2}} \times \left[P_6^{(2)}\!\left(\frac{1}{1 - \dfrac{M\sin^2\beta}{r_0 - 2M}}\right) \left(\frac{K_1}{\left(1 - \dfrac{M\sin^2\beta}{r_0 - 2M}\right)^{3/2}} + \left(K_2 + \frac{K_3}{1 - \dfrac{M\sin^2\beta}{r_0 - 2M}}\right)^{3/2} \right) + \text{7 more terms} \right] + \text{4 more terms}$$

where each K_i is a "constant" and each P_k or $P_k^{(i)}$ is a polynomial of degree k. The "constants" K_i and the polynomial coefficients all depend on the parameters r_0, and Δr.

Typical form of an individual integrand

For the I=1 Barack-Lousto-Sago mode, the $\ell = 0$, $m = 0$ integrand is:

$$
\begin{aligned}
I_{1,00} = & \frac{P_3(\sin^2\beta)\; P_6^{(1)}\!\left(1 - \dfrac{M\sin^2\beta}{r_0 - 2M}\right)}{\left(r_0 - 2M - M\sin^2\beta\right)^{5/2}} \\
& \times \left[P_6^{(2)}\!\left(\frac{1}{1 - \dfrac{M\sin^2\beta}{r_0 - 2M}}\right) \left(\frac{K_1}{\left(1 - \dfrac{M\sin^2\beta}{r_0 - 2M}\right)^{3/2}} + \left(K_2 + \frac{K_3}{1 - \dfrac{M\sin^2\beta}{r_0 - 2M}}\right)^{3/2} \right) + \text{7 more terms} \right] \\
& + \text{4 more terms}
\end{aligned}
$$

where each K_i is a "constant" and each P_k or $P_k^{(i)}$ is a polynomial of degree k. The "constants" K_i and the polynomial coefficients all depend on the parameters r_0, and Δr.

This is the simplest of the integrands; the integrands rapidly become more complicated with increasing ℓ and/or m.

Overall strategy for doing the β integrals

None of the symbolic algebra systems I tried (Mathematica, Mathematica with the RUBI rules-based-integration package, Maple, Sage) could do the $I = 1$, $\ell = 0$, $m = 0$ integral directly.

Overall strategy for doing the β integrals

None of the symbolic algebra systems I tried (Mathematica, Mathematica with the RUBI rules-based-integration package, Maple, Sage) could do the $I = 1$, $\ell = 0$, $m = 0$ integral directly.

Instead, use a divide-and-conquer strategy:

- "flatten" the integrand into a single linear combination $K + \sum_k c_k X_k$, where the coefficients K and $\{c_k\}$ depend on r_0 and Δr, but not on β:

Overall strategy for doing the β integrals

None of the symbolic algebra systems I tried (Mathematica, Mathematica with the RUBI rules-based-integration package, Maple, Sage) could do the $l = 1$, $\ell = 0$, $m = 0$ integral directly.

Instead, use a divide-and-conquer strategy:

- "flatten" the integrand into a single linear combination $K + \sum_k c_k X_k$, where the coefficients K and $\{c_k\}$ depend on r_0 and Δr, but not on β:
  - expand product-of-sums into top-level sum
  - merge nested sums into top-level sum
  - move factors that don't depend on β into coefficients $\{c_k\}$
  - do this recursively throughout the integrand's expression structure

Overall strategy for doing the β integrals

None of the symbolic algebra systems I tried (Mathematica, Mathematica with the RUBI rules-based-integration package, Maple, Sage) could do the $l = 1$, $\ell = 0$, $m = 0$ integral directly.

Instead, use a divide-and-conquer strategy:

- "flatten" the integrand into a single linear combination $K + \sum_k c_k X_k$, where the coefficients K and $\{c_k\}$ depend on r_0 and Δr, but not on β:
  - expand product-of-sums into top-level sum
  - merge nested sums into top-level sum
  - move factors that don't depend on β into coefficients $\{c_k\}$
  - do this recursively throughout the integrand's expression structure
- integrate each X_k
  - terminology: X_k is a "component"

Overall strategy for doing the β integrals

None of the symbolic algebra systems I tried (Mathematica, Mathematica with the RUBI rules-based-integration package, Maple, Sage) could do the $l = 1$, $\ell = 0$, $m = 0$ integral directly.

Instead, use a divide-and-conquer strategy:

- "flatten" the integrand into a single linear combination $K + \sum_k c_k X_k$, where the coefficients K and $\{c_k\}$ depend on r_0 and Δr, but not on β:
  - expand product-of-sums into top-level sum
  - merge nested sums into top-level sum
  - move factors that don't depend on β into coefficients $\{c_k\}$
  - do this recursively throughout the integrand's expression structure
- integrate each X_k
  - terminology: X_k is a "component"
- assemble the final result from K, c_k, and the X_k integrals

Example of flattening into a linear combination

For the I=1 Barack-Lousto-Sago mode, the $\ell = 0$, $m = 0$ integrand is a linear combination of 251 components. Some examples:

$$X_1 = \frac{1}{\left(M\cos^2\beta + r_0 - 3M\right)^4}$$

$$X_{10} = \frac{\left[(4Mr_0^2 - 8M^2 r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2 r_0\right]^{3/2}}{\left(M\cos^2\beta + r_0 - 3M\right)^7}$$

$$X_{100} = \frac{\left[(4Mr_0^2 - 8M^2 r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2 r_0\right]^{3/2}\sin^6\beta}{\left(M\cos^2\beta + r_0 - 3M\right)^{10}}$$

$$X_{200} = \frac{1}{\left(M\cos^2\beta + r_0 - 3M\right)^8\left[(4Mr_0^2 - 8M^2 r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2 r_0\right]^{3/2}}$$

$$X_{251} = \frac{\sin^6\beta}{\left[(4Mr_0^2 - 8M^2 r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2 r_0\right]^{1/2}\left(M\cos^2\beta + r_0 - 3M\right)^6}$$

Example of flattening into a linear combination

For the I=1 Barack-Lousto-Sago mode, the $\ell = 0$, $m = 0$ integrand is a linear combination of 251 components. Some examples:

$$X_1 = \frac{1}{\left(M\cos^2\beta + r_0 - 3M\right)^4}$$

$$X_{10} = \frac{\left[(4Mr_0^2 - 8M^2r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2r_0\right]^{3/2}}{\left(M\cos^2\beta + r_0 - 3M\right)^7}$$

$$X_{100} = \frac{\left[(4Mr_0^2 - 8M^2r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2r_0\right]^{3/2}\sin^6\beta}{\left(M\cos^2\beta + r_0 - 3M\right)^{10}}$$

$$X_{200} = \frac{1}{\left(M\cos^2\beta + r_0 - 3M\right)^8\left[(4Mr_0^2 - 8M^2r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2r_0\right]^{3/2}}$$

$$X_{251} = \frac{\sin^6\beta}{\left[(4Mr_0^2 - 8M^2r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2r_0\right]^{1/2}\left(M\cos^2\beta + r_0 - 3M\right)^6}$$

Substituting $x = \sin\beta$ converts each of our component integrals $\int_0^{2\pi} X_k \, d\beta$ into an elliptic integral.

Elliptic integrals

Formally, an elliptic integral is an integral

$$\int_a^b R\left(x, \sqrt{P_{3|4}(x)}\right)\, dx$$

where R is a rational function and $P_{3|4}$ is a polynomial of degree 3 or 4.

Elliptic integrals

Formally, an elliptic integral is an integral

$$\int_a^b R\left(x, \sqrt{P_{3|4}(x)}\right)\, dx$$

where R is a rational function and $P_{3|4}$ is a polynomial of degree 3 or 4.

- any elliptic integral can be written in terms of the 3 Legendre elliptic integrals E, K, and Π
- numerical computation of E, K, and Π is (can be) very efficient

Elliptic integrals

Formally, an elliptic integral is an integral

$$\int_a^b R\left(x, \sqrt{P_{3|4}(x)}\right)\, dx$$

where R is a rational function and $P_{3|4}$ is a polynomial of degree 3 or 4.

- any elliptic integral can be written in terms of the 3 Legendre elliptic integrals E, K, and Π
- numerical computation of E, K, and Π is (can be) very efficient
- but the (symbolic) reduction of an arbitrary elliptic integral to Legendre form can be very complicated

Elliptic integrals

Formally, an elliptic integral is an integral

$$\int_a^b R\left(x, \sqrt{P_{3|4}(x)}\right)\, dx$$

where R is a rational function and $P_{3|4}$ is a polynomial of degree 3 or 4.

- any elliptic integral can be written in terms of the 3 Legendre elliptic integrals E, K, and Π
- numerical computation of E, K, and Π is (can be) very efficient
- but the (symbolic) reduction of an arbitrary elliptic integral to Legendre form can be very complicated
- Maple has excellent code built-in to do this reduction (better than Mathematica or Mathematica/RUBI; alas Sage is very poor at this) $\Rightarrow$ do the elliptic integrals in Maple

Counting the β integrals for multiple (ℓ, m)

How many elliptic integrals do we need to do?

For $I = 1$, $\ell \in \{0, 2, 4, \ldots, 48\}$, $m = 0$, we have:

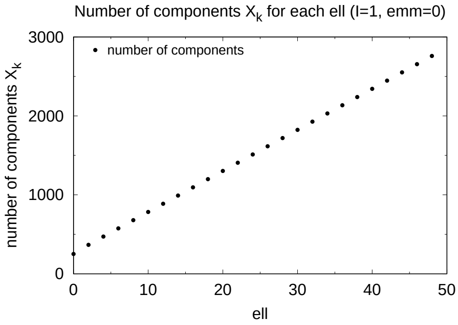
ℓ	number of components X_k
0	251

Counting the β integrals for multiple (ℓ, m)

How many elliptic integrals do we need to do?

For $I = 1$, $\ell \in \{0, 2, 4, \ldots, 48\}$, $m = 0$, we have:

ℓ	number of components X_k
0	251
2	367
4	471
8	679
16	1095
24	1511
36	2135
48	2759

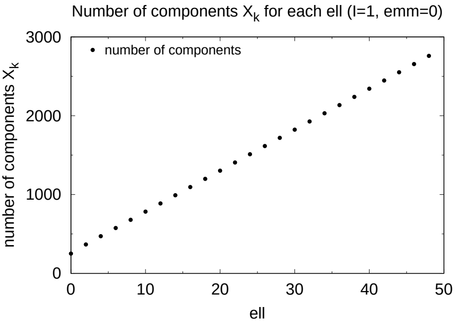


Counting the β integrals for multiple (ℓ, m)

How many elliptic integrals do we need to do?

For $I = 1$, $\ell \in \{0, 2, 4, \ldots, 48\}$, $m = 0$, we have:

ℓ	number of components X_k
0	251
2	367
4	471
8	679
16	1095
24	1511
36	2135
48	2759



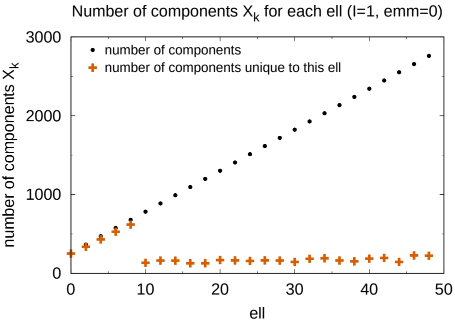
In total, to do all of $\ell \in \{0, 2, 4, \ldots, 48\}$ (again just for $I = 1$, $m = 0$) requires 37,763 elliptic integrals.

Counting the β integrals for multiple (ℓ, m)

How many elliptic integrals do we need to do?

For $I = 1$, $\ell \in \{0, 2, 4, \ldots, 48\}$, $m = 0$, we have:

ℓ	number of components X_k total	unique to this ℓ
0	251	251
2	367	337
4	471	432
8	679	618
16	1095	128
24	1511	158
36	2135	163
48	2759	224



In total, to do all of $\ell \in \{0, 2, 4, \ldots, 48\}$ (again just for $I = 1$, $m = 0$) requires 37,763 elliptic integrals.

Fortunately, many integrands are common to multiple ℓ; for this same set of ℓ we "only" need to integrate 4518 unique integrands X_k.

Cost of computing β integrals for multiple (ℓ, m)

Test computation:

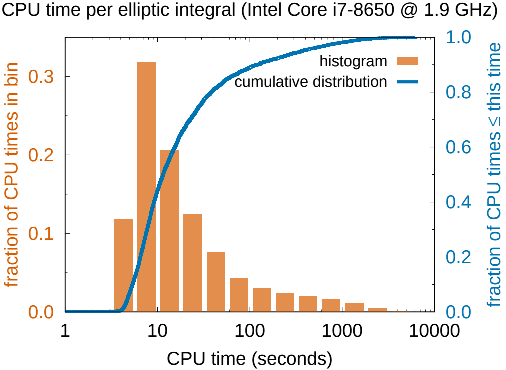
$I = 1$, $\ell \in \{0, 2, 4, 6, 8, 10, 12, 16, 20\}$, $m = 0$
(each ℓ computed independently; duplicate integrals *not* removed)
$\Rightarrow$ 4960 $\int_0^{2\pi} X_k \, d\beta$ integrals

Cost of computing β integrals for multiple (ℓ, m)

Test computation:

$l = 1$, $\ell \in \{0, 2, 4, 6, 8, 10, 12, 16, 20\}$, $m = 0$
(each ℓ computed independently; duplicate integrals *not* removed)
$\Rightarrow$ 4960 $\int_0^{2\pi} X_k \, d\beta$ integrals

The CPU time per $\int_0^{2\pi} X_k \, d\beta$ integral has a very wide distribution.



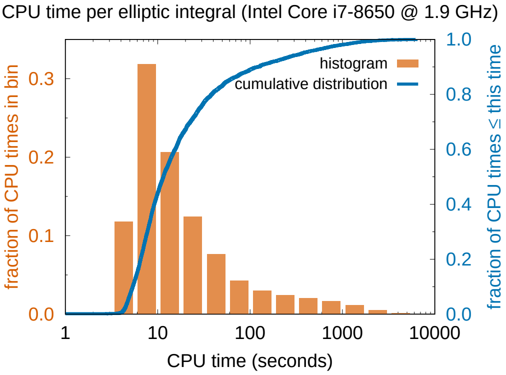
Cost of computing $\boldsymbol{\beta}$ integrals for multiple $(\boldsymbol{\ell}, \boldsymbol{m})$

Test computation:

$I = 1$, $\ell \in \{0, 2, 4, 6, 8, 10, 12, 16, 20\}$, $m = 0$
(each ℓ computed independently; duplicate integrals *not* removed)
$\Rightarrow$ 4960 $\int_0^{2\pi} X_k \, d\beta$ integrals

The CPU time per $\int_0^{2\pi} X_k \, d\beta$ integral has a very wide distribution.

The median CPU time is about 10 seconds per integral. But, some of the integrals are very expensive and take a lot of memory. The maximum for this set is 6300 seconds.



Cost of computing β integrals for multiple (ℓ, m)

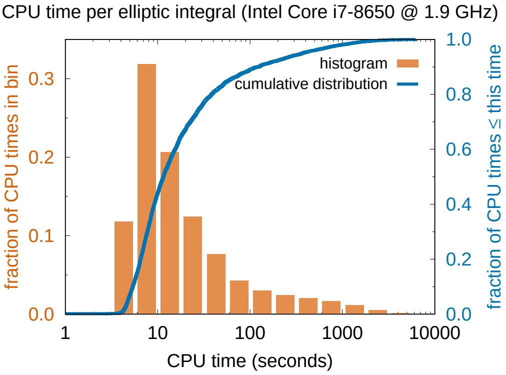
Test computation:

$l = 1$, $\ell \in \{0, 2, 4, 6, 8, 10, 12, 16, 20\}$, $m = 0$
(each ℓ computed independently; duplicate integrals *not* removed)
$\Rightarrow$ 4960 $\int_0^{2\pi} X_k \, d\beta$ integrals

The CPU time per $\int_0^{2\pi} X_k \, d\beta$ integral has a very wide distribution.

The median CPU time is about 10 seconds per integral. But, some of the integrals are very expensive and take a lot of memory. The maximum for this set is 6300 seconds.

The total CPU time for this set of (ℓ, m) is 4.7 days.



An extended divide-and-conquer strategy

To make the computation more efficient, and extend to larger sets of I, ℓ, m, we extend our divide-and-conquer strategy to keep a database of components and integrals:

- "flatten" each I, ℓ, m integrand into a linear combination of components $K + \sum_k c_k X_k$, as described before

An extended divide-and-conquer strategy

To make the computation more efficient, and extend to larger sets of I, ℓ, m, we extend our divide-and-conquer strategy to keep a database of components and integrals:

- "flatten" each I, ℓ, m integrand into a linear combination of components $K + \sum_k c_k X_k$, as described before
- construct a database of all the **unique** components, with the following fields:
  - identifier
  - status (e.g., "TODO", "DONE", or "FAIL")
  - component integrand X_k
  - component integral $\int_0^{2\pi} X_k \, d\beta$

An extended divide-and-conquer strategy

To make the computation more efficient, and extend to larger sets of I, ℓ, m, we extend our divide-and-conquer strategy to keep a database of components and integrals:

- "flatten" each I, ℓ, m integrand into a linear combination of components $K + \sum_k c_k X_k$, as described before
- construct a database of all the **unique** components, with the following fields:
  - identifier
  - status (e.g., "TODO", "DONE", or "FAIL")
  - component integrand X_k
  - component integral $\int_0^{2\pi} X_k \, d\beta$
- repeat until all components are done:
  - extract some not-yet-done components from the database
  - integrate those components (in parallel on a cluster)
  - update the database with the results of the integrations

An extended divide-and-conquer strategy

To make the computation more efficient, and extend to larger sets of I, ℓ, m, we extend our divide-and-conquer strategy to keep a database of components and integrals:

- "flatten" each I, ℓ, m integrand into a linear combination of components $K + \sum_k c_k X_k$, as described before
- construct a database of all the **unique** components, with the following fields:
  - identifier
  - status (e.g., "TODO", "DONE", or "FAIL")
  - component integrand X_k
  - component integral $\int_0^{2\pi} X_k \, d\beta$
- repeat until all components are done:
  - extract some not-yet-done components from the database
  - integrate those components (in parallel on a cluster)
  - update the database with the results of the integrations
- assemble each (I, ℓ, m)'s β integral from the K, c_k coefficients and the component integrals in the database

Current status

Working on $I = 1$, $\ell = 0, 2, 4, \ldots, 48$, $m = 0$ (Maple technical limitation prevents doing $\ell = 50$)

- database has 4518 unique components X_k

Current status

Working on $I = 1$, $\ell = 0, 2, 4, \ldots, 48$, $m = 0$ (Maple technical limitation prevents doing $\ell = 50$)

- database has 4518 unique components X_k
- integrals have been running on Adam Pound's cluster* for about 6 weeks
- currently 4171 components have status DONE, 347 not yet done

* Funding: Royal Society

Current status

Working on $l = 1$, $\ell = 0, 2, 4, \ldots, 48$, $m = 0$ (Maple technical limitation prevents doing $\ell = 50$)

- database has 4518 unique components X_k
- integrals have been running on Adam Pound's cluster* for about 6 weeks
- currently 4171 components have status DONE, 347 not yet done
- remaining not-yet-done components are those which didn't succeed within ~ 1 day CPU time limit per integration; currently retrying them with ~ 3-day CPU time limit per integration (Intel Xeon 5320 @ 2.2 GHz)

* Funding: Royal Society

Current status

Working on $I = 1$, $\ell = 0, 2, 4, \ldots, 48$, $m = 0$ (Maple technical limitation prevents doing $\ell = 50$)

- database has 4518 unique components X_k
- integrals have been running on Adam Pound's cluster* for about 6 weeks
- currently 4171 components have status DONE, 347 not yet done
- remaining not-yet-done components are those which didn't succeed within ~ 1 day CPU time limit per integration; currently retrying them with ~ 3-day CPU time limit per integration (Intel Xeon 5320 @ 2.2 GHz)
- some integrals take a **lot** of memory: largest so far was 315 GB memory (!)

* Funding: Royal Society

Current status

Working on $I = 1$, $\ell = 0, 2, 4, \ldots, 48$, $m = 0$ (Maple technical limitation prevents doing $\ell = 50$)

- database has 4518 unique components X_k
- integrals have been running on Adam Pound's cluster* for about 6 weeks
- currently 4171 components have status DONE, 347 not yet done
- remaining not-yet-done components are those which didn't succeed within ~ 1 day CPU time limit per integration; currently retrying them with $\sim$ 3-day CPU time limit per integration (Intel Xeon 5320 @ 2.2 GHz)
- some integrals take a **lot** of memory: largest so far was 315 GB memory (!)
- many integrals are very large: median size (printed out) is 116,000 characters, maximum size 6.7 million characters
- database size is currently 3.1 GB

* Funding: Royal Society

Largest integral completed so far

$\#3985$ in the database:

$$X_k = \frac{\left[(4Mr_0^2 - 8M^2r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2r_0\right]^{1/2}\sin^2\beta}{\left(M\cos^2\beta + r_0 - 3M\right)^{57}}$$

Largest integral completed so far

#3985 in the database:

$$X_k = \frac{\left[(4Mr_0^2 - 8M^2 r_0)\cos^2\beta + (r_0 - 3M)(\Delta r)^2 + 4r_0^3 - 20Mr_0^2 + 24M^2 r_0\right]^{1/2} \sin^2\beta}{\left(M\cos^2\beta + r_0 - 3M\right)^{57}}$$

Integral took about 2 hours CPU time:

$$\int_0^{2\pi} X_k \, d\beta =$$

```
-1/832843192699583869114919332328039753766781580004556800*r0*(380417909501120488
660120403803606453941776977204388033820862948746338473543360809363671890098185415
12828038160580608000*M^110*r0^55-20922985022561626876306622209198354966797733746
2413418601474621810486160448848445150019539554001978320554209883193344000*M^109*
r0^56-428988086019994434842261158723485871226392233468251032022762117278936734309
21803808212235313059899641759446702489600*M^109*r0^54*Delta_r^2+570151341864804
3323793554552006551728452382445850765656890183444335747872231120130338032452846
55390923510221931701862400*M^108*r0^57+231653566450796994814821025710682370462251
80607285555729229154333062583652697774056434607069052345806550101219344384000*M^
108*r0^55*Delta_r^2-930677522987342195516434903727549826799107354752412894535838
... skip 83967 lines
89301315799478656544276480*M*r0^71*Delta_r^96-856728981658246558707531894321315844
*M*r0^69*Delta_r^98+2725955850730784504978510572840550400*M*r0^67*Delta_r^100-9086
519502435948349928368576135168*M*r0^65*Delta_r^102+324518553658426726783156020576
2560*M*r0^63*Delta_r^104-1298074214633706907132624082305024*M*r0^61*Delta_r^106
+6490371073168534535663120411525124*M*r0^59*Delta_r^108-649037107316853453566312004
1152512*M*r0^57*Delta_r^110-32451855365842672678315602057625600*M*r0^55*Delta_r^1
12+228385074666955709648703689949483806835610217836387053939248332800*r0^168)/M/(2
*M-r0)^56/Delta_r^110/(-r0+3*M)/(9*M^2-6*M*r0+r0^2)^27/(16*M^2*r0-16*M*r0^2-3*M*
Delta_r^2+4*r0^3+r0*Delta_r^2)^(1/2)*EllipticPi(-M/(2*M-r0),2*(1/(16*M^2*r0-16*M
*r0^2-3*M*Delta_r^2+4*r0^3+r0*Delta_r^2)*M*r0*(-2*M+r0))^(1/2))
```

Cost of evaluating the result

Given the complexity of many of the integrals, it's natural to wonder:

1. How expensive will it be to numerically evaluate our final result for a given set of $(r_0, \Delta r)$?

Cost of evaluating the result

Given the complexity of many of the integrals, it's natural to wonder:

1. How expensive will it be to numerically evaluate our final result for a given set of $(r_0, \Delta r)$?
2. How will this compare to the cost of doing the β integrals numerically?

Cost of evaluating the result

Given the complexity of many of the integrals, it's natural to wonder:

1. How expensive will it be to numerically evaluate our final result for a given set of $(r_0, \Delta r)$?
2. How will this compare to the cost of doing the β integrals numerically?

I don't know yet:

- How much do the final expressions simplify?
- What common subexpressions are there?

Cost of evaluating the result

Given the complexity of many of the integrals, it's natural to wonder:

1. How expensive will it be to numerically evaluate our final result for a given set of $(r_0, \Delta r)$?
2. How will this compare to the cost of doing the β integrals numerically?

I don't know yet:

- How much do the final expressions simplify?
- What common subexpressions are there?
- Can the huge-integer coefficients be rounded to double-precision floating-point without introducing significant errors?

Cost of evaluating the result

Given the complexity of many of the integrals, it's natural to wonder:

1. How expensive will it be to numerically evaluate our final result for a given set of $(r_0, \Delta r)$?
2. How will this compare to the cost of doing the β integrals numerically?

I don't know yet:

- How much do the final expressions simplify?
- What common subexpressions are there?
- Can the huge-integer coefficients be rounded to double-precision floating-point without introducing significant errors?
- For a given r_0, many subexpressions are just polynomials in Δr. What other precomputation can/should be done for a given r_0, so as to make evaluation for each Δr cheaper?

Cost of evaluating the result

Given the complexity of many of the integrals, it's natural to wonder:

1. How expensive will it be to numerically evaluate our final result for a given set of $(r_0, \Delta r)$?
2. How will this compare to the cost of doing the β integrals numerically?

I don't know yet:

- How much do the final expressions simplify?
- What common subexpressions are there?
- Can the huge-integer coefficients be rounded to double-precision floating-point without introducing significant errors?
- For a given r_0, many subexpressions are just polynomials in Δr. What other precomputation can/should be done for a given r_0, so as to make evaluation for each Δr cheaper?
- What will the expressions for $\mathrm{I} \neq 1$ and/or $m \neq 0$ look like?

Cost of evaluating the result

Given the complexity of many of the integrals, it's natural to wonder:

1. How expensive will it be to numerically evaluate our final result for a given set of $(r_0, \Delta r)$?
2. How will this compare to the cost of doing the β integrals numerically?

I don't know yet:

- How much do the final expressions simplify?
- What common subexpressions are there?
- Can the huge-integer coefficients be rounded to double-precision floating-point without introducing significant errors?
- For a given r_0, many subexpressions are just polynomials in Δr. What other precomputation can/should be done for a given r_0, so as to make evaluation for each Δr cheaper?
- What will the expressions for $\mathrm{I} \neq 1$ and/or $m \neq 0$ look like?

Still lots to explore here!

Conclusions

Several parts of our 2nd-order self-force calculation require computing the Barack-Lousto-Sago tensor-spherical-harmonic modes of the 1st-order puncture. The main difficulty in doing this is the β integrals.

Conclusions

Several parts of our 2nd-order self-force calculation require computing the Barack-Lousto-Sago tensor-spherical-harmonic modes of the 1st-order puncture. The main difficulty in doing this is the β integrals.

- each β integral depends on the parameters r_0 and Δr

Conclusions

Several parts of our 2nd-order self-force calculation require computing the Barack-Lousto-Sago tensor-spherical-harmonic modes of the 1st-order puncture. The main difficulty in doing this is the β integrals.

- each β integral depends on the parameters r_0 and Δr
- there are about 2500 β integrals, each of which we'd like to numerically evaluate for about 10^3–10^5 distinct $(r_0, \Delta r)$ parameters

Conclusions

Several parts of our 2nd-order self-force calculation require computing the Barack-Lousto-Sago tensor-spherical-harmonic modes of the 1st-order puncture. The main difficulty in doing this is the β integrals.

- each β integral depends on the parameters r_0 and Δr
- there are about 2500 β integrals, each of which we'd like to numerically evaluate for about 10^3–10^5 distinct $(r_0, \Delta r)$ parameters
- divide-and-conquer algorithm:
  - "flatten" each β integral into a linear combination of "component" integrals, where the linear-combination coefficients depend on the parameters r_0 and Δr, but not on β
  - integrate the (unique) components in parallel on a cluster
  - keep a database of all the unique components and their integrals
  - assemble the final results from the component integrals

Conclusions

Several parts of our 2nd-order self-force calculation require computing the Barack-Lousto-Sago tensor-spherical-harmonic modes of the 1st-order puncture. The main difficulty in doing this is the β integrals.

- each β integral depends on the parameters r_0 and Δr
- there are about 2500 β integrals, each of which we'd like to numerically evaluate for about 10^3–10^5 distinct $(r_0, \Delta r)$ parameters
- divide-and-conquer algorithm:
    - "flatten" each β integral into a linear combination of "component" integrals, where the linear-combination coefficients depend on the parameters r_0 and Δr, but not on β
    - integrate the (unique) components in parallel on a cluster
    - keep a database of all the unique components and their integrals
    - assemble the final results from the component integrals
- some integrals take a lot of CPU/memory to compute, and are very large

Conclusions

Several parts of our 2nd-order self-force calculation require computing the Barack-Lousto-Sago tensor-spherical-harmonic modes of the 1st-order puncture. The main difficulty in doing this is the β integrals.

- each β integral depends on the parameters r_0 and Δr
- there are about 2500 β integrals, each of which we'd like to numerically evaluate for about 10^3–10^5 distinct $(r_0, \Delta r)$ parameters
- divide-and-conquer algorithm:
  - "flatten" each β integral into a linear combination of "component" integrals, where the linear-combination coefficients depend on the parameters r_0 and Δr, but not on β
  - integrate the (unique) components in parallel on a cluster
  - keep a database of all the unique components and their integrals
  - assemble the final results from the component integrals
- some integrals take a lot of CPU/memory to compute, and are very large
- I don't yet know how expensive it will be to evaluate the result expressions, or how this will compare to the cost of doing the integrals numerically