Isometric Embeddings of Black Holes: Numerical Horizons in Euclidean Space

Iago Mendes

Honors Thesis presented to Oberlin College's
Department of Physics and Astronomy

Advisor: Robert Owen

April 5, 2024

I dedicate this work to my mother, Karla Janara Mendes,
and my father, Gilmar Braz da Cruz.

Contents

Abstract

Isometric embeddings consist in describing surfaces in a desired space such that infinitesimal distances from a given geometry are preserved. The isometric embedding of black hole apparent horizons in flat geometry is useful both for visualizing the horizon structure and for computing quasilocal quantities such as mass, energy, and angular momentum. However, finding these embeddings requires solving a system of nonlinear partial differential equations for which there is no generally established algorithm. In this context, we have developed two novel, robust numerical methods for finding isometric embeddings. Both of them were developed in the Spectral Einstein Code (`SpEC`), where they were tested and applied to binary black hole merger simulations. From the embedding results, we gained insight into the intrinsic shape of such horizons and how their embeddability possibly behaves.

Acknowledgments

First, I would like to express my sincere gratitude to my advisor, Rob Owen, who introduced me to the astonishing world of Numerical Relativity. Your passion for the subject has been truly contagious, and your guidance throughout this project and other endeavors has been invaluable in many aspects.

I would also like to thank my family for supporting my aspirations even when they lead to long distances and limited visits. Without your love and support, I would not be where I am today.

Finally, I am deeply grateful to Oberlin College for giving me, and many other students who are underrepresented in STEM, opportunities to grow both academically and professionally.

Chapter 1

Introduction

In the early twentieth century, Einstein revolutionized the study of gravity by connecting spacetime geometry with physical dynamics. As John Wheeler says, "Spacetime tells matter how to move; matter tells spacetime how to curve" [1]. Being a highly complex theory, many problems of interest only have analytic solutions in special cases with symmetry. In this context, Numerical Relativity emerged as an essential field to solve these problems numerically, allowing us to explore general cases that can be found in the universe. Specifically, simulations of Binary Black Holes (BBH) became very important as gravitational wave detectors were developed, needing to use numerical results to identify and characterize signals in their data [2].

Since the foundations of differential geometry and general relativity, finding isometric embeddings has been a classic problem that involves constructing a surface in flat space described by a metric tensor. The results from this problem have a long history for visualization, but are also relevant for calculating quantities like black hole mass and energy. That said, in general scenarios, this problem requires a solver capable of handling a system of strongly nonlinear and nonstandard partial differential equations, for which there is no generally established algorithm.

To address that issue, we have developed novel numerical methods capable of solving the isometric embedding problem and applied them to BBH simulations. The purpose of this thesis is to present how this work was done and share our results.

In Chapter 2, I present the main background knowledge needed to understand this thesis. In Chapter 3, I describe the theory behind the numerical methods we have developed. In Chapter 4, I go over some implementation details and our code tests. In Chapter 5, I describe the application of our methods in BBH mergers. Finally, in Chapter 6, I summarize this work and discuss future projects.

Chapter 2

Theoretical Background

Most current research in numerical relativity involves so much theoretical background that it might seem overwhelming at first sight. The precise description of this field requires not only complex physical concepts, but also elaborate mathematical formalism. Taking this into consideration, this chapter aims at covering some necessary topics for the understanding of the remainder of this thesis.

2.1 Notations and Conventions

A common source of confusion for someone getting into General Relativity for the first time is the meaning of equations. For example, consider the geodesic equation:

$$\frac{d^2x^\alpha}{d\tau^2} = -\Gamma^\alpha_{\beta\gamma}\frac{dx^\beta}{d\tau}\frac{dx^\gamma}{d\tau}, \tag{2.1}$$

which is analogous to the "straight-line motion" in Newtonian mechanics. Without knowing what the indices α, β and γ mean, it is hard to interpret equation (2.1), which actually represents 4 equations each with a double sum over β and γ.

First, let's describe the **index notation**. We use indices (α, β, γ, ...) to describe components of a vector, a matrix or, more generally, a tensor. To understand it better, consider a 4-dimensional spacetime. Any index used to describe this space has four possible values: $\alpha \in \{0, 1, 2, 3\}$. Let $\vec{x}$ be a 4-dimensional vector in this geometry with $\vec{x} = (x^0, x^1, x^2, x^3) = (t, x, y, z)$. Then, we can use x^α to represent any component of $\vec{x}$. For example, if we write

$$x^\alpha = \text{constant}, \tag{2.2}$$

we actually mean

$$t = \text{constant}, \tag{2.3a}$$
$$x = \text{constant}, \tag{2.3b}$$
$$y = \text{constant}, \tag{2.3c}$$
$$z = \text{constant}. \tag{2.3d}$$

Note that while (2.2) might seem like one equation, it actually represents *four equations* as shown in (2.3). We will encounter this idea of "one formula" representing multiple equations many times in the remainder of this thesis.

Another important concept is the **summation convention**, also known as the Einstein notation. In General Relativity, we use raised and lowered indices to distinguish between vectors and co-vectors, but this difference will not be relevant here. The important thing to know is that any pair of raised and lowered indices actually represents a sum over those indices. For example,

$$a_\alpha x^\alpha = \sum_\alpha a_\alpha x^\alpha \tag{2.4a}$$
$$= a_0 x^0 + a_1 x^1 + a_2 x^2 + a_3 x^3 \tag{2.4b}$$

and

$$b_{\alpha\beta} x^\alpha x^\beta = \sum_\alpha \sum_\beta b_{\alpha\beta} x^\alpha x^\beta \tag{2.5a}$$
$$\begin{aligned} &= b_{00}x^0x^0 + b_{01}x^0x^1 + b_{02}x^0x^2 + b_{03}x^0x^3 \\ &\quad + b_{10}x^1x^0 + b_{11}x^1x^1 + b_{12}x^1x^2 + b_{13}x^1x^3 \\ &\quad + b_{20}x^2x^0 + b_{21}x^2x^1 + b_{22}x^2x^2 + b_{23}x^2x^3 \\ &\quad + b_{30}x^3x^0 + b_{31}x^3x^1 + b_{32}x^3x^2 + b_{33}x^3x^3. \end{aligned} \tag{2.5b}$$

From these examples, it is clear why we need this convention. It reduces large sums over multiple indices into a concise form, making our equations much easier to work with.

Now that we are familiar with the index notation and the summation convention, let's interpret what equation (2.1) is telling us. First, let's write the sums over β and γ explicitly:

$$\frac{d^2x^\alpha}{d\tau^2} = \sum_\beta \sum_\gamma -\Gamma^\alpha_{\beta\gamma} \frac{dx^\beta}{d\tau} \frac{dx^\gamma}{d\tau}. \tag{2.6}$$

Now, we have one free index α, which means that we have four equations:

$$\frac{d^2t}{d\tau^2} = \sum_\beta \sum_\gamma -\Gamma^0_{\beta\gamma} \frac{dx^\beta}{d\tau} \frac{dx^\gamma}{d\tau}, \tag{2.7a}$$

$$\frac{d^2x}{d\tau^2} = \sum_\beta \sum_\gamma -\Gamma^1_{\beta\gamma} \frac{dx^\beta}{d\tau} \frac{dx^\gamma}{d\tau}, \tag{2.7b}$$

$$\frac{d^2y}{d\tau^2} = \sum_\beta \sum_\gamma -\Gamma^2_{\beta\gamma} \frac{dx^\beta}{d\tau} \frac{dx^\gamma}{d\tau}, \tag{2.7c}$$

$$\frac{d^2z}{d\tau^2} = \sum_\beta \sum_\gamma -\Gamma^3_{\beta\gamma} \frac{dx^\beta}{d\tau} \frac{dx^\gamma}{d\tau}. \tag{2.7d}$$

Going forward, both the index notation and the summation convention will be used, except when the explicit forms bring some insight into the equations. Additionally, I will be using a common convention that Greek letters $(\alpha, \beta, \gamma, ...)$ represent four-dimensional indices, Latin letters near the middle of the alphabet $(i, j, k, ...)$ represent three-dimensional (spatial) indices, and capital Latin letters near the start of the alphabet $(A, B, C, ...)$ represent two-dimensional indices.

2.2 Differential Geometry

Einstein changed the world's understanding of gravity by describing it as a manifestation of the *geometry* of spacetime, not as a force. Hence, it is essential to understand how we can specify geometries mathematically, which is the goal of this section.

In order to distinguish and compare geometries, we need to find an intrinsic property that can be used to find any relevant quantity of a geometry. It turns out that this intrinsic property is the *distance between nearby points*. Specifically, if we know how to compute infinitesimal distances ds, then it is straightforward to integrate it in some way to find any length, area, volume, etc. The equation that defines ds in a geometry is called the **line element**, which is written in a general form as

$$ds^2 = g_{AB}\, dx^A\, dx^B, \tag{2.8}$$

where g_{AB} is the **metric tensor**, usually referred to as the "metric". Note that dx^A and dx^B could be switched in (2.8) without affecting the line element, which implies that the metric is symmetric. That is,

$$g_{AB} = g_{BA}. \tag{2.9}$$

In the context of the index notation introduced in the previous section, it is also worth

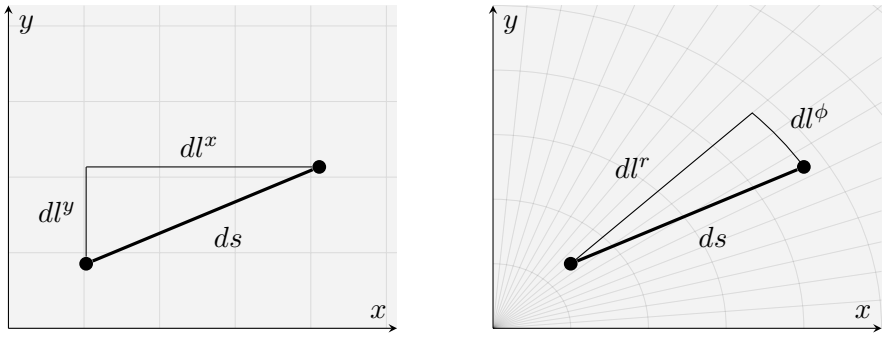


Figure 2.1: Euclidean plane described by rectangular (left) and polar (right) coordinates. The infinitesimal distances due to changes in x, y, r and ϕ are respectively shown as dl^x, dl^y, dl^r and dl^ϕ.

mentioning that the metric g_{AB} can be used to lower indices:

$$g_{AB}x^A = x_B. \tag{2.10}$$

Similarly, the inverse metric, denoted by g^{AB}, can be used to raise indices:

$$g^{AB}x_A = x^B. \tag{2.11}$$

The inverse metric is defined as the matrix-inverse of the metric, so

$$g^{AC}g_{CB} = \delta^A_B = \begin{cases} 1 & \text{if } A = B \\ 0 & \text{if } A \neq B \end{cases}, \tag{2.12}$$

where δ^B_A is the Kronecker delta. With this, note that the lowering (2.10) and raising (2.11) of indices are inverse operations because

$$g^{AC}(g_{CB}x^B) = (g^{AC}g_{CB})x^B = \delta^A_B x^B = x^A. \tag{2.13}$$

For a more rigorous treatment of vectors and co-vectors, refer to [3, chapter 20].

With these definitions, we are ready to describe any geometry. To clarify how we actually make use of (2.8), let's look at a few examples.

2.2.1 Euclidean Plane

First, let's look at a very familiar geometry: the Euclidean plane. We know that it can be described by rectangular coordinates (x and y) and polar coordinates (r and ϕ), as shown in Figure 2.1.

In rectangular coordinates, we know that the infinitesimal distances along each coordinate are simply given by $dl^x = dx$ and $dl^y = dy$. From the Pythagorean theorem, this results in an infinitesimal displacement ds such that

$$ds^2 = (dl^x)^2 + (dl^y)^2 \tag{2.14a}$$
$$= dx^2 + dy^2. \tag{2.14b}$$

In polar coordinates, the infinitesimal distance along r is also simple: $dl^r = dr$. However, note that dl^ϕ is more subtle as it is actually the arc length of an angle $d\phi$ at a radius r. Then, $dl^\phi = r\,d\phi$. In the limit of infinitesimal distances, the Pythagorean theorem still holds, so

$$ds^2 = (dl^r)^2 + (dl^\phi)^2 \tag{2.15a}$$
$$= dr^2 + r^2 d\phi^2. \tag{2.15b}$$

Note that (2.14) and (2.15) were developed independently. However, these coordinate systems are related by

$$x = r\cos\phi \quad \Rightarrow \quad dx = \cos\phi\, dr - r\sin\phi\, d\phi \tag{2.16}$$

and

$$y = r\sin\phi \quad \Rightarrow \quad dy = \sin\phi\, dr + r\cos\phi\, d\phi. \tag{2.17}$$

Plugging (2.16) and (2.17) into (2.14), we get the same result as in (2.15). Therefore, the line elements (2.14) and (2.15) represent the same geometry (the Euclidean plane), being related by a coordinate transformation.

Comparing the line elements (2.14) and (2.15) with the general form in (2.8), we can express the metric of this geometry in rectangular and polar coordinates by

$$(g_{AB}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad (g_{AB}) = \begin{pmatrix} 1 & 0 \\ 0 & r^2 \end{pmatrix}, \tag{2.18}$$

respectively. Note that while the metric can have different representations depending on your coordinate choice, the infinitesimal distances ds should be the same. Another way of saying this is the line element is *invariant* under coordinate transformations.

2.2.2 Surface of a sphere

Now, consider the surface of a sphere with radius R. We can use the spherical coordinate angles θ and ϕ to describe the geometry of this surface. Here, we will use the physics

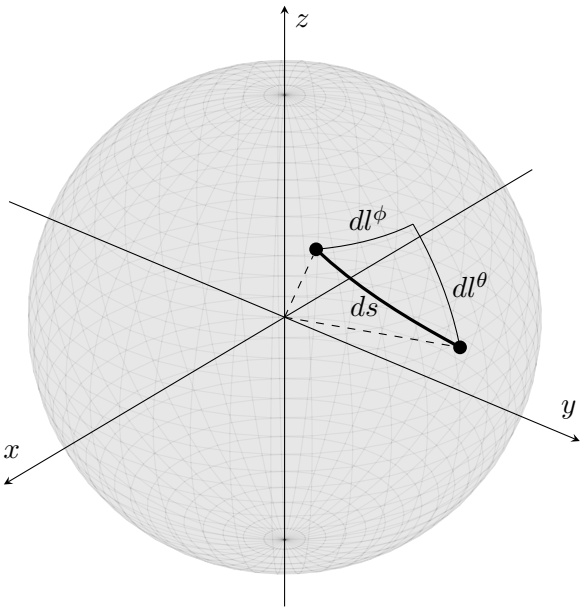


Figure 2.2: Surface of a sphere embedded in 3-dimensional Euclidean space. The arc lengths dl^θ and dl^ϕ correspond to infinitesimal changes in the spherical coordinate angles θ (polar) and ϕ (azimuthal). The shortest distance between the marked points is denoted by ds.

convention, in which θ is the polar angle ($0 \leq \theta \leq \pi$) and ϕ is the azimuthal angle ($0 \leq \phi \leq 2\pi$).

After some careful analysis (see Figure 2.2), you can convince yourself that the infinitesimal changes $d\theta$ and $d\phi$ correspond to arc lengths $dl^\theta = R\,d\theta$ and $dl^\phi = R\sin\theta\,d\phi$. Similar to the previous example, we can use the Pythagorean theorem in the limit of infinitesimal changes, so

$$ds^2 = (dl^\theta)^2 + (dl^\phi)^2 \tag{2.19a}$$

$$= R^2\,d\theta^2 + R^2\sin^2\theta\,d\phi^2, \tag{2.19b}$$

which corresponds to a metric given by

$$(g_{AB}) = \begin{pmatrix} R^2 & 0 \\ 0 & R^2\sin^2\theta \end{pmatrix}. \tag{2.20}$$

To see how this result can be used, suppose that we are interested in computing the surface area of this sphere. From the arc lengths dl^θ and dl^ϕ, we know that the area element is given by

$$dA = dl^\theta dl^\phi = R^2\sin\theta\,d\theta d\phi, \tag{2.21}$$

which is a well-known result from introductory multivariable calculus courses. With this, the surface area of the sphere is given by

$$A = \int dA = \int_0^{2\pi} \int_0^{\pi} R^2 \sin\theta \, d\theta d\phi = 4\pi R^2, \tag{2.22}$$

as expected.

2.2.3 Kerr Horizon

Since we have only derived results that we already knew before, the metric might not seem very helpful. Its usefulness becomes evident when we consider more complex geometries, so let's consider one here. Later on this thesis, we will describe the surface of a rotating black hole, called the Kerr Horizon. For now, all we need is its line element, which is given by

$$ds^2 = \sigma \, d\theta^2 + \frac{4\rho^2}{\sigma} \sin^2\theta \, d\phi^2, \tag{2.23}$$

where

$$\rho := 1 + \sqrt{1 - \chi^2} \quad \text{and} \quad \sigma := 2\rho - \chi^2 \sin^2\theta. \tag{2.24}$$

In (2.23), we are using units such that the constants $c = G = 1$ (geometric units) and letting the mass of the black hole $M = 1$. Also, χ is the dimensionless spin of the black hole, ranging from 0 (no rotation) to 1 (maximum rotation). From (2.23), we know that the Kerr metric is given by

$$(g_{AB}) = \begin{pmatrix} \sigma & 0 \\ 0 & \frac{4\rho^2}{\sigma} \sin^2\theta \end{pmatrix} \quad \Rightarrow \quad \begin{cases} g_{\theta\theta} = \sigma \\ g_{\phi\phi} = \frac{4\rho^2}{\sigma} \sin^2\theta \\ g_{\theta\phi} = 0 \end{cases} . \tag{2.25}$$

Using this information about the Kerr geometry, let's find the area of the surface. Since we have a diagonal metric, we can find our distances along curves of constant θ ($d\theta = 0$) and constant ϕ ($d\phi = 0$):

$$dl^\phi = \sqrt{g_{\phi\phi}} \, d\phi = \frac{2\rho}{\sqrt{\sigma}} \sin\theta \, d\phi \quad \text{and} \quad dl^\theta = \sqrt{g_{\theta\theta}} \, d\theta = \sqrt{\sigma} \, d\theta. \tag{2.26}$$

Then, our area element is given by

$$dA = dl^\theta dl^\phi = 2\rho \sin\theta \, d\theta d\phi. \tag{2.27}$$

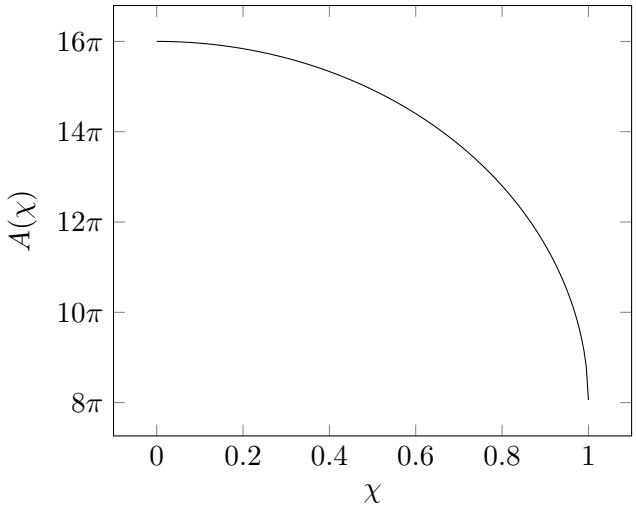


Figure 2.3: Surface area of the Kerr horizon as a function of the dimensionless spin χ.

Finally, we have a surface area of

$$A = \int dA = 2\rho \int_0^{2\pi} d\phi \int_0^{\pi} \sin\theta \, d\theta = 8\pi \left(1 + \sqrt{1-\chi^2}\right). \tag{2.28}$$

A graph of (2.28) is shown in Figure 2.3. When $\chi = 0$, we have a stationary horizon, which should have a radius of $2M$ according to the Schwarzschild geometry. Indeed, Figure 2.3 shows that $A(0) = 4\pi(2)^2 = 16\pi$. Additionally, note that its surface area decreases as the black hole speeds up, which is a result of its polar components shrinking while its equatorial components remain unchanged.

Note how far we have come! With the description of the black hole surface area, we were able to build an intuition to how this complex object behaves as it rotates. This was all derived from the line element (2.23), which we could have explored even more. As we have been discussing in this section, the metric contains all the intrinsic properties that we need to know about a geometry.

Before moving on, it is worth noting a pattern in the area elements (2.21) and (2.27). Let the determinant of the Kerr metric (2.25) be denoted by

$$g = \det \begin{pmatrix} \sigma & 0 \\ 0 & \frac{4\rho^2}{\sigma} \sin^2\theta \end{pmatrix} = 4\rho^2 \sin^2\theta. \tag{2.29}$$

Comparing (2.27) with (2.29), we note that

$$dA = \sqrt{g}\, d\theta d\phi. \tag{2.30}$$

The same argument holds in the sphere surface example. In a general coordinate system, where the metric is not diagonal, a more elaborate argument can be used to show that

$$dA = \sqrt{|g|} dx^A dx^B, \tag{2.31}$$

and similar for volume elements in higher dimensions [4].

2.3 Isometric Embedding

One of the easiest ways of understanding a 2-dimensional geometry is to visualize a surface in 3-dimensional Euclidean space that would have the same geometry as the given geometry (if such a surface exists). To do so, we need to find embedding functions $\vec{x} = (x, y, z)$ in terms of the spherical angle coordinates θ and ϕ that describe a given surface.

The problem of isometric embeddings has been core to the mathematics of differential geometry since its foundation in the mid-nineteenth century. Since these early years, it has been known to be difficult to solve in general. From the early to mid-twentieth century, a lot of mathematical formalism and existence proofs were constructed by Darboux [5], Weyl [6], and Nash [7], but even these proofs have not provided a robust way of constructing the embeddings.

As we will see, this problem involves solving a system of three nonlinear partial differential equations (PDEs). Let's try to develop these PDEs using what we have discussed so far. Consider a generic 2-dimensional geometry described by the metric g_{AB} with $A, B \in \{\theta, \phi\}$. Remembering that all metrics are symmetric, its line element is given by

$$ds^2 = g_{\theta\theta}\, d\theta^2 + g_{\phi\phi}\, d\phi^2 + 2g_{\theta\phi}\, d\theta d\phi. \tag{2.32}$$

Additionally, we know that the 3-dimensional Euclidean space is described by the line element

$$ds^2 = dx^2 + dy^2 + dz^2, \tag{2.33}$$

which is analogous to what we had in (2.14).

Our goal is to find embedding functions $x(\theta, \phi)$, $y(\theta, \phi)$, and $z(\theta, \phi)$ such that the line elements (2.32) and (2.33) are equal:

$$dx^2 + dy^2 + dz^2 = g_{\theta\theta}\, d\theta^2 + g_{\phi\phi}\, d\phi^2 + 2g_{\theta\phi}\, d\theta d\phi. \tag{2.34}$$

That is, we are trying to find a representation of the surface in Euclidean space that preserves

the infinitesimal lengths ds. Since x, y, and z are functions of θ and ϕ, we know that

$$dx = \frac{\partial x}{\partial \theta}\, d\theta + \frac{\partial x}{\partial \phi}\, d\phi = (\partial_\theta x)\, d\theta + (\partial_\phi x)\, d\phi, \tag{2.35a}$$

$$dy = \frac{\partial y}{\partial \theta}\, d\theta + \frac{\partial y}{\partial \phi}\, d\phi = (\partial_\theta y)\, d\theta + (\partial_\phi y)\, d\phi, \tag{2.35b}$$

$$dz = \frac{\partial z}{\partial \theta}\, d\theta + \frac{\partial z}{\partial \phi}\, d\phi = (\partial_\theta z)\, d\theta + (\partial_\phi z)\, d\phi, \tag{2.35c}$$

where I have used $\partial_A x$ to denote the partial derivative of x relative to A. Using (2.35) in (2.34), we have

$$\begin{aligned} g_{\theta\theta}\, d\theta^2 + g_{\phi\phi}\, d\phi^2 + 2g_{\theta\phi}\, d\theta d\phi = {} & (\partial_\theta x)^2\, d\theta^2 + (\partial_\phi x)^2\, d\phi^2 + 2(\partial_\theta x)(\partial_\phi x)\, d\theta d\phi \\ & + (\partial_\theta y)^2\, d\theta^2 + (\partial_\phi y)^2\, d\phi^2 + 2(\partial_\theta y)(\partial_\phi y)\, d\theta d\phi \\ & + (\partial_\theta z)^2\, d\theta^2 + (\partial_\phi z)^2\, d\phi^2 + 2(\partial_\theta z)(\partial_\phi z)\, d\theta d\phi. \end{aligned} \tag{2.36}$$

Rearranging the right side of (2.36), we have

$$\begin{aligned} \Big\{g_{\theta\theta}\Big\}\, d\theta^2 + \Big\{g_{\phi\phi}\Big\}\, d\phi^2 + 2\Big\{g_{\theta\phi}\Big\}\, d\theta d\phi = {} & \Big\{(\partial_\theta x)^2 + (\partial_\theta y)^2 + (\partial_\theta z)^2\Big\}\, d\theta^2 \\ & + \Big\{(\partial_\phi x)^2 + (\partial_\phi y)^2 + (\partial_\phi z)^2\Big\}\, d\phi^2 \\ & + 2\Big\{(\partial_\theta x)(\partial_\phi x) + (\partial_\theta y)(\partial_\phi y) + (\partial_\theta z)(\partial_\phi z)\Big\}\, d\theta d\phi. \end{aligned} \tag{2.37}$$

Making the $d\theta^2$ terms from one side equal to the $d\theta^2$ terms of the other (and similarly for $d\phi^2$ and $d\theta d\phi$), we arrive at the **embedding equations**:

$$(\partial_\theta x)^2 + (\partial_\theta y)^2 + (\partial_\theta z)^2 = g_{\theta\theta}, \tag{2.38a}$$

$$(\partial_\theta x)(\partial_\phi x) + (\partial_\theta y)(\partial_\phi y) + (\partial_\theta z)(\partial_\phi z) = g_{\theta\phi}, \tag{2.38b}$$

$$(\partial_\phi x)^2 + (\partial_\phi y)^2 + (\partial_\phi z)^2 = g_{\phi\phi}. \tag{2.38c}$$

Making use of our index notation, (2.38) can be written as

$$(\partial_A x)(\partial_B x) + (\partial_A y)(\partial_B y) + (\partial_A z)(\partial_B z) = g_{AB}. \tag{2.39}$$

Therefore, if we want to embed any 2-dimensional surface described by a metric g_{AB} into 3-dimensional Euclidean space, we need to solve the system of PDEs (2.39) for the three unknown functions $x(\theta, \phi)$, $y(\theta, \phi)$, and $z(\theta, \phi)$. Because of its nonlinear character (specifically the fact that it is quadratic in highest-order derivatives), this system cannot be solved with usual PDE algorithms, requiring novel numerical methods. One of the main purposes of this thesis is to present the methods that we have developed of approaching

this problem, which is the focus of the next chapter.

In fact, it is not always possible to find embedding functions that satisfy (2.39). Not because it is hard to find them, but because they do not exist. A famous example of this is the Kerr metric (2.25), which describes a black hole of mass M rotating with a certain spin χ. If

$$\chi > \sqrt{3}/2 \approx 0.866, \tag{2.40}$$

then (2.39) has no solution [8].

As an exercise, let's use the results of a familiar embedding (the surface of a sphere) and check if we get what is predicted by (2.39). From the conversion of spherical coordinates to rectangular coordinates, we know that

$$x(\theta,\phi) = R\sin\theta\cos\phi \quad \Rightarrow \quad \partial_\theta x = R\cos\theta\cos\phi \quad \text{and} \quad \partial_\phi x = -R\sin\theta\sin\phi, \tag{2.41a}$$
$$y(\theta,\phi) = R\sin\theta\sin\phi \quad \Rightarrow \quad \partial_\theta y = R\cos\theta\sin\phi \quad \text{and} \quad \partial_\phi y = R\sin\theta\cos\phi, \tag{2.41b}$$
$$z(\theta,\phi) = R\cos\theta \quad \Rightarrow \quad \partial_\theta z = -R\sin\theta \quad \text{and} \quad \partial_\phi z = 0. \tag{2.41c}$$

Using (2.41) in (2.38), we have

$$(R\cos\theta\cos\phi)^2 + (R\cos\theta\sin\phi)^2 + (-R\sin\theta)^2 = R^2\cos^2\theta + R^2\sin^2\theta = R^2, \tag{2.42a}$$
$$(R\cos\theta\cos\phi)(-R\sin\theta\sin\phi) + (R\cos\theta\sin\phi)(R\sin\theta\cos\phi) + (-R\sin\theta)(0) = 0, \tag{2.42b}$$
$$(-R\sin\theta\sin\phi)^2 + (R\sin\theta\cos\phi)^2 + (0)^2 = R^2\sin^2\theta. \tag{2.42c}$$

This indicates that our the metric should be

$$(g_{AB}) = \begin{pmatrix} R^2 & 0 \\ 0 & R^2\sin^2\theta \end{pmatrix}, \tag{2.43}$$

which agrees with what we had in (2.20).

2.4 Black Hole Horizons

In the last section, we talked about the key idea of this thesis: embedding 2-dimensional surfaces into 3-dimensional Euclidean space. However, we have not defined which *surfaces* we are embedding and how they are related to black holes. Addressing those points is the goal of this section.

Here, we will talk about black hole horizons. Specifically, we will focus on two types of horizons: the event horizon (more commonly known, but not useful here) and the apparent horizon (less commonly known, but extremely useful here).

2.4.1 Event Horizon

Let's start by defining the one that is more commonly known: the event horizon. In simple terms, it is the *boundary of no return*. That is, it separates a region from which nothing can escape (inside) and a region from which light can escape to a distant observer (outside) [9, 10]. Despite its simple definition, the event horizon is more subtle than one might expect at first sight. To find this surface boundary, we need trace light paths and check which ones fail to reach infinity [9].

The main subtlety with the event horizon is that it is defined in terms of what will happen in the future (whether light will reach infinity or not). This means that it is a *globally* defined concept [9]. Because of its global nature, the event horizon is not a good description of the location of black holes in numerical relativity codes as it would require knowledge of spacetime in the future [11]. Be that as it may, there are elaborate methods to find them numerically [11, 12].

2.4.2 Apparent Horizon

In order to describe the location of black holes in numerical relativity codes, we require a local notion of horizons. That is, we need a surface that is specified by what is currently happening, not by what will happen. The apparent horizon is such a surface, but we need to discuss the concept of trapped surfaces before defining it.

As an analogy, consider a spherical surface with light being radiated to all directions inwards and outwards. Figure 2.4 shows a cross-section of this scenario. In Euclidean space, we expect that the inward light converges (gaining energy density), while the outward light diverges (losing energy density). However, with a strong curvature of spacetime, it is possible that both the inward and outward light converge [13, 14]. In such cases, we say that the surface is trapped. More rigorously, we define trapped surfaces to be regions with non-negative expansion of the outward light paths that are orthogonal to it [10].

With this, we can define the apparent horizon as the *outermost trapped surface* of a black hole [14]. Note that this definition does not require us to trace light paths all the way to infinity, only up to a finite amount. Then, we say that apparent horizons are defined *quasi-locally* [9]. As proved in [10], if an apparent horizon exists, it must be inside an event horizon. Therefore, by removing the inner region of the apparent horizon from simulations, we can avoid the singularity in our numerical domains, allowing for stable spacetime computations [11].

As apparent horizons are the standard black hole boundaries used in numerical relativity, they are the surfaces that we want to embed in Euclidean space. From now on, we will focus exclusively on the apparent horizon, which will often be referred to as simply the "horizon".

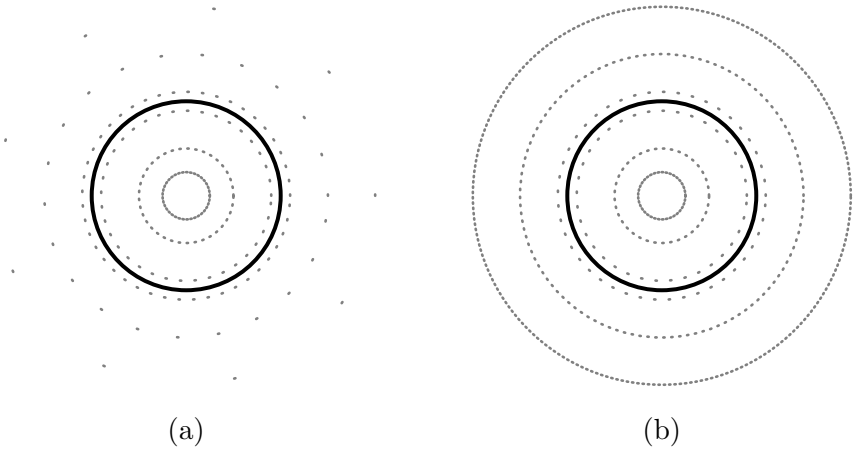


Figure 2.4: Visualization of analogy used to explain trapped surfaces. In both sides, the thick black circle represents a surface that we are analyzing, and the gray dots represent light being emitted from this surface inwards and outwards. The spacing between each dot represents the concentration of light in that region. In (a), the inward light gets concentrated, while the outward light diverges. In (b) both the inward and outward light gets concentrated. Therefore, only the surface shown in (b) is trapped.

Chapter 3

Numerical Methods

So far, we have talked about the problem we are trying to solve and its motivation. We have also discussed how difficult it is to find solutions to the embedding equations, which is why we need good numerical methods. To that end, the main goal of this research project is the development of efficient, accurate approaches to the embedding problem. This chapter focuses on presenting the theory behind these methods.

Here, we focus on two approaches that will be referred to as "relaxation" and "matrix" methods. The relaxation method was idealized by my advisor, Robert Owen, and developed by the two of us during my time at Oberlin. Since this is a radically new method and the one with which I have worked the most, I go over its derivation in precise detail in section 3.1. The matrix method was adapted from the work in [15], having significant improvements made by Hengrui Zhu during his time at Oberlin working with Robert Owen. Since we will see results from this method in the next chapters, I go over its high-level description in section 3.2.

3.1 Relaxation Method

The key idea of the relaxation method starts at its foundations. Instead of finding the embedding functions $\vec{x} = (x, y, z)$, we try to find the **dyad vectors** $\vec{e}_A \in \{\vec{e}_\theta, \vec{e}_\phi\}$, which are the basis vectors of the surface coordinates θ and ϕ. These dyad vectors are related to the embedding functions by

$$\vec{e}_A = \partial_A \vec{x}. \tag{3.1}$$

Figure 3.1 shows an example of the dyad vectors for the embedding of a sphere.

The main idea of relaxation algorithms is to continuously change a field so that it better satisfies the PDEs we need to solve. For this, let's introduce a non-physical "time" parameter t over which these continuous changes occur. In our case, we start with initial

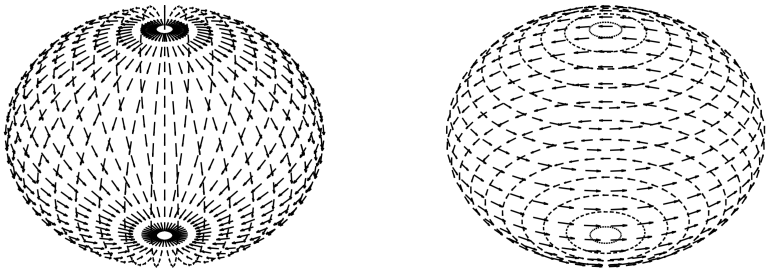


Figure 3.1: Dyad vectors $\vec{e}_\theta$ (left) and $\vec{e}_\phi$ (right) for a sphere embedded in Euclidean space. Note that $\vec{e}_\theta$ points along curves of constant ϕ values and $\vec{e}_\phi$ points along curves of constant θ values, corresponding to equation (3.1).

guesses for $\vec{e}_\theta$ and $\vec{e}_\phi$ and update them via a relaxation scheme:

$$\vec{e}_A|_{\text{new}} = \vec{e}_A|_{\text{old}} + \delta t\, \dot{\vec{e}}_A|_{\text{old}}, \tag{3.2}$$

where δt is our relaxation time step and $\dot{\vec{e}}_A = \partial_t \vec{e}_A$. Hence, we need to define how we are evolving the dyad vectors by finding expressions for $\dot{\vec{e}}_A$ that will give us solutions to the embedding problem.

In order for the dyad vectors to be a valid coordinate basis, $\vec{e}_\theta$ and $\vec{e}_\phi$ need to commute. To measure this, we define a **commutator** quantity

$$[\vec{e}_\theta, \vec{e}_\phi] := \partial_\theta \vec{e}_\phi - \partial_\phi \vec{e}_\theta, \tag{3.3}$$

which should be equal to zero if the dyad vectors commute. Figure 3.2 compares scenarios in which the dyad vectors commute and in which they do not. As described in subsection 3.1.1, we rotate our initial dyad vectors in a specific way so that we get $[\vec{e}_\theta, \vec{e}_\phi] = 0$ up to some target accuracy.

In addition to making sure that the dyad vectors represent a coordinate basis, we need to confirm that they actually give us a solution to the embedding problem. Using their definition (3.1) in the embedding equations (2.39), we have

$$(e^x_A)(e^x_B) + (e^y_A)(e^y_B) + (e^z_A)(e^z_B) = g_{AB}, \tag{3.4}$$

which can be rewritten in terms of dot products as

$$\vec{e}_A \cdot \vec{e}_B = g_{AB}. \tag{3.5}$$

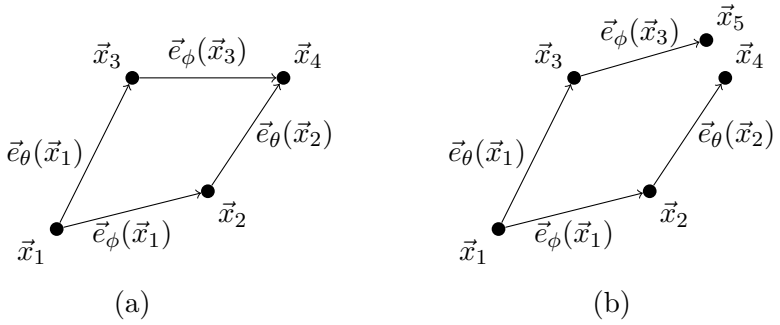


Figure 3.2: (a) A scenario in which the dyad vectors commute, that is $[\vec{e}_\theta, \vec{e}_\phi] = 0$. (b) Another scenario in which the dyad vectors do not commute, that is $[\vec{e}_\theta, \vec{e}_\phi] \neq 0$.

Therefore, we have to make sure that our dyad vectors satisfy (3.5). As described in subsection 3.1.2, we achieve this by treating (3.5) as a constraint and damping any violations to it.

After we have found proper values for the dyad vectors $\vec{e}_A$, we need to solve for the embedding functions $\vec{x}$ in (3.1). We approach this by transforming equations (3.1) into Poisson equations for each embedding function, which is described in further detail in subsection 3.1.3.

3.1.1 Dyad Rotation

Here, let's assume that all we need is for $\vec{e}_\theta$ and $\vec{e}_\phi$ to commute. In order to determine how we will evolve our initial guess, we need to define an "error norm" E that quantifies what we want to drive to zero. As mentioned before, we want to have $[\vec{e}_\theta, \vec{e}_\phi] = 0$, but this commutator has three components and is evaluated at each surface point. To combine the three components into a positive norm at each surface point, we can look at $|[\vec{e}_\theta, \vec{e}_\phi]|^2$. Finally, to get a scalar quantity, we can integrate this norm over the entire surface:

$$E = \frac{1}{2} \int |[\vec{e}_\theta, \vec{e}_\phi]|^2 dA, \tag{3.6}$$

where the factor of $1/2$ was added for convenience in the upcoming derivation.

Before continuing, we need to define the surface we are integrating over. A natural choice is the apparent horizon surface that we are trying to find. However, we only need *some* average measure of our commutator. For simplicity, we can define this error norm

over the unit sphere, which has an area element of $dA = \sin\theta\, d\theta d\phi$. Using this, we have

$$E = \frac{1}{2}\iint |[\vec{e}_\theta, \vec{e}_\phi]|^2 \sin\theta\, d\theta d\phi \tag{3.7a}$$

$$= \frac{1}{2}\iint [\vec{e}_\theta, \vec{e}_\phi] \cdot [\vec{e}_\theta, \vec{e}_\phi] \sin\theta\, d\theta d\phi. \tag{3.7b}$$

The rate of change of this error norm is

$$\dot{E} = \partial_t E = \frac{1}{2}\iint \partial_t\Big([\vec{e}_\theta, \vec{e}_\phi] \cdot [\vec{e}_\theta, \vec{e}_\phi]\Big) \sin\theta\, d\theta d\phi \tag{3.8a}$$

$$= \frac{1}{2}\iint \Big(\partial_t[\vec{e}_\theta, \vec{e}_\phi] \cdot [\vec{e}_\theta, \vec{e}_\phi] + [\vec{e}_\theta, \vec{e}_\phi] \cdot \partial_t[\vec{e}_\theta, \vec{e}_\phi]\Big) \sin\theta\, d\theta d\phi \tag{3.8b}$$

$$= \iint [\vec{e}_\theta, \vec{e}_\phi] \cdot \partial_t[\vec{e}_\theta, \vec{e}_\phi] \sin\theta\, d\theta d\phi \tag{3.8c}$$

$$= \iint \Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \partial_t[\vec{e}_\theta, \vec{e}_\phi]\, d\theta d\phi \tag{3.8d}$$

$$= \iint \Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \Big(\partial_\theta \dot{\vec{e}}_\phi - \partial_\phi \dot{\vec{e}}_\theta\Big)\, d\theta d\phi \tag{3.8e}$$

$$= \iint \Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \partial_\theta\Big(\dot{\vec{e}}_\phi\Big)\, d\theta d\phi - \iint \Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \partial_\phi\Big(\dot{\vec{e}}_\theta\Big)\, d\theta d\phi. \tag{3.8f}$$

Knowing that the area element vanishes near the boundaries of θ (0 and π) and that the boundaries of ϕ are periodic, we can ignore the boundary terms when integrating by parts both on θ and ϕ. Doing so, we have

$$\dot{E} = -\iint \partial_\theta\Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \Big(\dot{\vec{e}}_\phi\Big)\, d\theta d\phi + \iint \partial_\phi\Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \Big(\dot{\vec{e}}_\theta\Big)\, d\theta d\phi \tag{3.9a}$$

$$= -\iint \Big\{\partial_\theta\Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \dot{\vec{e}}_\phi - \partial_\phi\Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \dot{\vec{e}}_\theta\Big\}\, d\theta d\phi \tag{3.9b}$$

$$= -\iint \Big\{\frac{1}{\sin\theta}\partial_\theta\Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \dot{\vec{e}}_\phi - \frac{1}{\sin\theta}\partial_\phi\Big(\sin\theta[\vec{e}_\theta, \vec{e}_\phi]\Big) \cdot \dot{\vec{e}}_\theta\Big\} \sin\theta\, d\theta d\phi \tag{3.9c}$$

$$= -\int \Big\{D_\theta[\vec{e}_\theta, \vec{e}_\phi] \cdot \dot{\vec{e}}_\phi - D_\phi[\vec{e}_\theta, \vec{e}_\phi] \cdot \dot{\vec{e}}_\theta\Big\}\, dA, \tag{3.9d}$$

where we have defined the operators D_A as

$$D_A\vec{V} := \frac{1}{\sin\theta}\partial_A\Big(\sin\theta\, \vec{V}\Big). \tag{3.10}$$

Since we want to evolve the dyad vectors by rotating them, we can assume that their rates of change are given by

$$\dot{\vec{e}}_A = \vec{\omega} \times \vec{e}_A, \tag{3.11}$$

where $\vec{\omega}$ is some angular velocity. Note that (3.11) are our temporary evolution equations.

Using them in (3.9d), we have

$$\dot{E} = -\int \left\{ D_\theta[\vec{e}_\theta, \vec{e}_\phi] \cdot \vec{\omega} \times \vec{e}_\phi - D_\phi[\vec{e}_\theta, \vec{e}_\phi] \cdot \vec{\omega} \times \vec{e}_\theta \right\} dA \tag{3.12a}$$

$$= -\int \left\{ \vec{\omega} \cdot \vec{e}_\phi \times D_\theta[\vec{e}_\theta, \vec{e}_\phi] - \vec{\omega} \cdot \vec{e}_\theta \times D_\phi[\vec{e}_\theta, \vec{e}_\phi] \right\} dA \tag{3.12b}$$

$$= -\int \vec{\omega} \cdot \left\{ \vec{e}_\phi \times D_\theta[\vec{e}_\theta, \vec{e}_\phi] - \vec{e}_\theta \times D_\phi[\vec{e}_\theta, \vec{e}_\phi] \right\} dA, \tag{3.12c}$$

where we have used the cyclic property of scalar triple products. Now, if we define our angular velocity to be

$$\vec{\omega} := \vec{e}_\phi \times D_\theta[\vec{e}_\theta, \vec{e}_\phi] - \vec{e}_\theta \times D_\phi[\vec{e}_\theta, \vec{e}_\phi], \tag{3.13}$$

then (3.12c) becomes

$$\dot{E} = -\int \vec{\omega} \cdot \vec{\omega}\, dA \tag{3.14a}$$

$$= -\int |\vec{\omega}|^2\, dA. \tag{3.14b}$$

Therefore, if we rotate our dyad vectors according to (3.11) and (3.13), then (3.14b) tells us that $\dot{E} \leq 0$. This means that our error norm (as well as the surface-averaged value of the commutator) will be driven to zero as the relaxation goes on. At the worst case ($\dot{E} = 0$), these values will remain constant.

3.1.2 Constraint Damping

Having the rotation of the dyad vectors figured out, we now have to make sure that these vectors give us solutions to the embedding problem. Based on (3.5), we can define the following constraint measures:

$$\mathcal{C}_{AB} = \vec{e}_A \cdot \vec{e}_B - g_{AB}. \tag{3.15}$$

If the embedding equations (3.5) are satisfied, then $\mathcal{C}_{AB} = 0$. With the initial guess for the dyad vectors, this will not be true in most cases, so we must modify our relaxation scheme in a way that drives $\mathcal{C}_{AB}$ to zero.

Using the previous evolution equations (3.11), the rate of change of these constraints is

$$\dot{\mathcal{C}}_{AB} = \partial_t\Big(\vec{e}_A \cdot \vec{e}_B - g_{AB}\Big) \tag{3.16a}$$
$$= \dot{\vec{e}}_A \cdot \vec{e}_B + \vec{e}_A \cdot \dot{\vec{e}}_B \tag{3.16b}$$
$$= (\vec{\omega} \times \vec{e}_A) \cdot \vec{e}_B + \vec{e}_A \cdot (\vec{\omega} \times \vec{e}_B) \tag{3.16c}$$
$$= \vec{e}_B \cdot (\vec{\omega} \times \vec{e}_A) + \vec{e}_A \cdot (\vec{\omega} \times \vec{e}_B) \tag{3.16d}$$
$$= \vec{\omega} \cdot (\vec{e}_A \times \vec{e}_B) + \vec{\omega} \cdot (\vec{e}_B \times \vec{e}_A) \tag{3.16e}$$
$$= \vec{\omega} \cdot (\vec{e}_A \times \vec{e}_B) - \vec{\omega} \cdot (\vec{e}_A \times \vec{e}_B) \tag{3.16f}$$
$$= 0, \tag{3.16g}$$

where we have used the cyclic property of scalar triple products and the antisymmetry of cross products. Therefore, (3.16) tells us that the previous evolution equations keep the constraints $\mathcal{C}_{AB}$ constant.

In order to drive $\mathcal{C}_{AB}$ to zero, we introduce a constraint damping procedure, which follows a derivation first done by Robert Owen in [16]. The idea is to add extra terms to the evolution equations (3.11) that are proportional to the constraints. Because of this proportionality, if the constraints are satisfied ($\mathcal{C}_{AB} = 0$), then the evolution equations remain unchanged. Additionally, as we will show, this has the effect of forcing any constraint violations to zero.

In coming up with new evolution equations, we need to worry about the indices. Note that $\mathcal{C}_{AC}$ has two lowered indices, where the index C is being used to avoid confusions in the upcoming equations. For a given dyad vector $\vec{e}_A$ (where A could either be θ or ϕ), there are actually two constraints because the other index (C) can also be either θ or ϕ. If we want to combine these two components, we need a quantity with a raised C index. Additionally, we want to apply changes parallel to either dyad vector, so we need to multiply our extra term by $\vec{e}_D$. Because of this new lowered D index, we also need a quantity with a raised D index. Therefore, let's introduce a matrix m^{CD} to take care of our need of raised indices. Finally, it is a good idea to add a constant γ to give us some control of how the damping occurs. Combining all of these factors, we have the following evolution equations:

$$\dot{\vec{e}}_A = \vec{\omega} \times \vec{e}_A - \gamma m^{CD} \mathcal{C}_{AC} \vec{e}_D. \tag{3.17}$$

The previous justification for the new indices becomes clearer once we expand the implicit summations:

$$\begin{aligned} \dot{\vec{e}}_A = \vec{\omega} \times \vec{e}_A &- \gamma m^{\theta\theta} \mathcal{C}_{A\theta} \vec{e}_\theta - \gamma m^{\theta\phi} \mathcal{C}_{A\theta} \vec{e}_\phi \\ &- \gamma m^{\phi\theta} \mathcal{C}_{A\phi} \vec{e}_\theta - \gamma m^{\phi\phi} \mathcal{C}_{A\phi} \vec{e}_\phi. \end{aligned} \tag{3.18}$$

Note that we have not defined our choice of the matrix m^{CD}, which will impact the rate

of change of our constraints. In our implementations of the relaxation method, we have experimented with different choices for m^{CD}. For now, I introduce the choice that provides the clearest analytic convergence. At any moment in the relaxation, define the dot products of the dyad vectors as

$$m_{CD} = \vec{e}_C \cdot \vec{e}_D. \tag{3.19}$$

This tensor would equal the surface metric *only if* the dyad vectors already had a vanishing commutator. Otherwise, m_{CD} is simply a positive-definite symmetric 2-by-2 matrix at each point on the surface. Since this matrix is positive-definite, we can define its *inverse matrix* at each point to be the tensor m^{CD} referred to in equation (3.17). The fact that it is an inverse matrix means that

$$m^{CD} m_{AD} = \delta^C_A, \tag{3.20}$$

as in equation (2.12).

With this, we are ready to find the new rate of change of the constraints. From (3.16), we know that the $\vec{\omega} \times \vec{e}_A$ term vanishes in $\dot{\mathcal{C}}_{AB}$, so we can ignore it here. Focusing on the extra terms, we have

$$\begin{aligned}
\dot{\mathcal{C}}_{AB} &= \dot{\vec{e}}_A \cdot \vec{e}_B + \vec{e}_A \cdot \dot{\vec{e}}_B && \text{(3.21a)}\\
&= \Big\{ -\gamma m^{CD} \mathcal{C}_{AC} \vec{e}_D \Big\} \cdot \vec{e}_B + \vec{e}_A \cdot \Big\{ -\gamma m^{CD} \mathcal{C}_{BC} \vec{e}_D \Big\} && \text{(3.21b)}\\
&= -\gamma m^{CD} (\vec{e}_B \cdot \vec{e}_D) \mathcal{C}_{AC} - \gamma m^{CD} (\vec{e}_A \cdot \vec{e}_D) \mathcal{C}_{BC} && \text{(3.21c)}\\
&= -\gamma m^{CD} m_{BD} \mathcal{C}_{AC} - \gamma m^{CD} m_{AD} \mathcal{C}_{BC} && \text{(3.21d)}\\
&= -\gamma \delta^C_B \mathcal{C}_{AC} - \gamma \delta^C_A \mathcal{C}_{BC} && \text{(3.21e)}\\
&= -\gamma \mathcal{C}_{AB} - \gamma \mathcal{C}_{BA} && \text{(3.21f)}\\
&= -2\gamma \mathcal{C}_{AB}. && \text{(3.21g)}
\end{aligned}$$

From (3.21f) to (3.21g), we have used the symmetry of the constraint (due the symmetry of the metric) to say that $\mathcal{C}_{AB} = \mathcal{C}_{BA}$. Note that equation (3.21g) implies that

$$\mathcal{C}_{AB}(t) \propto \exp(-2\gamma t). \tag{3.22}$$

In order words, as long as we apply the evolution equations (3.17) with (3.20) as the choice for m^{CD}, $\mathcal{C}_{AB}$ will be driven to zero exponentially. When $\mathcal{C}_{AB}$ becomes less than our desired accuracy, we can stop our relaxation.

Again, the convergence rate (3.22) is only true if we choose m^{CD} to be the inverse matrix of $m_{CD} = \vec{e}_C \cdot \vec{e}_D$. However, finding this inverse matrix at every time step can be computationally expensive, and we have not seen significant improvements in this choice for m^{CD}. In contrast, if we chose m^{CD} to be the inverse matrix of the already-known surface

metric g_{CD}, we would only need to compute the inverse matrix $m^{CD} = g^{CD}$ once. In this case, the constraints should behave more like (3.22) as the dyad vectors become solutions to the embedding equations. That is, as the relaxation evolves, $m_{CD} = g_{CD} \approx \vec{e}_C \cdot \vec{e}_D$ becomes more and more accurate. This choice for m^{CD} can be represented simply by rewriting (3.18) as

$$\begin{aligned}\dot{\vec{e}}_A = \vec{\omega} \times \vec{e}_A &- \gamma g^{\theta\theta} \mathcal{C}_{A\theta} \vec{e}_\theta - \gamma g^{\theta\phi} \mathcal{C}_{A\theta} \vec{e}_\phi \\ &- \gamma g^{\phi\theta} \mathcal{C}_{A\phi} \vec{e}_\theta - \gamma g^{\phi\phi} \mathcal{C}_{A\phi} \vec{e}_\phi.\end{aligned} \tag{3.23}$$

Another possible choice for the matrix m^{CD} is the inverse metric of the unit sphere:

$$(u^{CD}) = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\sin^2\theta} \end{pmatrix}. \tag{3.24}$$

While not accurate with the convergence rate (3.22), this choice allows us to reduce the evolution equations (3.18) to

$$\dot{\vec{e}}_A = \vec{\omega} \times \vec{e}_A - \gamma \mathcal{C}_{A\theta} \vec{e}_\theta - \gamma \frac{1}{\sin^2\theta} \mathcal{C}_{A\phi} \vec{e}_\phi, \tag{3.25}$$

minimizing our matrix manipulations.

3.1.3 Surface Construction

From the previous subsections, it is clear how to find dyad vectors $\vec{e}_\theta$ and $\vec{e}_\phi$ that are solutions to the embedding equations (3.5). However, our initial embedding equations (2.39) were in terms of the embedding functions $\vec{x}$. This means that the last step of our relaxation method requires us to solve equations (3.1). This can be approached in different ways, but the technique we use here is to turn them into Poisson equations. Before doing so, we have to once again choose an appropriate metric for describing distances on the horizon surface. Here, we do not see any compelling reason to prefer the metric of the physical horizon. If we instead choose the metric of a unit sphere, solving Poisson equations turns out to be much easier, so we use that choice below.

From the definition of our dyad vectors, we know that the angular gradient of the embedding functions is

$$\nabla \vec{x} = \left(\partial_\theta \vec{x}, \frac{1}{\sin\theta} \partial_\phi \vec{x} \right) = \left(\vec{e}_\theta, \frac{1}{\sin\theta} \vec{e}_\phi \right). \tag{3.26}$$

Taking the angular divergence of (3.26), we have

$$\nabla^2\vec{x} = \nabla\cdot\nabla\vec{x} \tag{3.27a}$$
$$= \frac{1}{\sin\theta}\partial_\theta[\sin\theta\,(\nabla\vec{x})_\theta] + \frac{1}{\sin\theta}\partial_\phi[(\nabla\vec{x})_\phi] \tag{3.27b}$$
$$= \frac{1}{\sin\theta}\partial_\theta(\sin\theta\,\vec{e}_\theta) + \frac{1}{\sin\theta}\partial_\phi\left(\frac{1}{\sin\theta}\vec{e}_\phi\right) \tag{3.27c}$$
$$= \frac{1}{\sin\theta}\partial_\theta(\sin\theta\,\vec{e}_\theta) + \frac{1}{\sin^2\theta}\partial_\phi(\vec{e}_\phi). \tag{3.27d}$$

The vector derivatives ∇ used above are referring to *angular* vectors in θ and ϕ. In contrast, we are using $\vec{}$ to indicate *position* vectors in x, y, and z. We have been using this convention all along, but it is easy to mix these two kinds of vectors on the steps above.

Note that the right side of (3.27d) involves only derivatives of the already-known dyad vectors. Since we want to solve for $\vec{x}$, we are now faced with three Poisson equations, which can be solved in various ways. In a finite-difference code, we can solve these equations using a simple relaxation scheme. In a spectral code, we can analytically apply an inverse unit-sphere Laplacian on the right side of (3.27d). We will go over these details in the next chapter when discussing different code implementations.

3.1.4 Summary

The relaxation method is summarized in Figure 3.3. Given a surface metric g_{AB}, we rotate our dyad vectors with a constraint damping scheme to get $\vec{e}_\theta$ and $\vec{e}_\phi$ that satisfy $\vec{e}_A\cdot\vec{e}_B = g_{AB}$. With this, we solve Poisson equations to find the embedding functions $\vec{x}$.

Before using the relaxation method, we need to specify m^{CD} as the inverse matrix of some matrix m_{CD}. In subsection 3.1.2, we discussed three possible choices for the m_{CD}:

1. $m_{CD} = \vec{e}_C\cdot\vec{e}_D$ ("metric"[1] of the current surface), leading to equations (3.18);
2. $m_{CD} = g_{CD}$ (metric of the surface we want to find), leading to equations (3.23); and
3. $m_{CD} = u_{CD}$ (metric of a unit sphere), leading to equations (3.25).

In the next chapter, I go over the choices that we have used in our implementations.

3.2 Matrix Method

Before introducing our second algorithm, it is important to highlight that this approach was developed and implemented in the `SpEC` codebase (introduced in the next chapter) by

[1] As mentioned before, m_{CD} is equal to a surface metric only if $[\vec{e}_\theta, \vec{e}_\phi] = 0$.

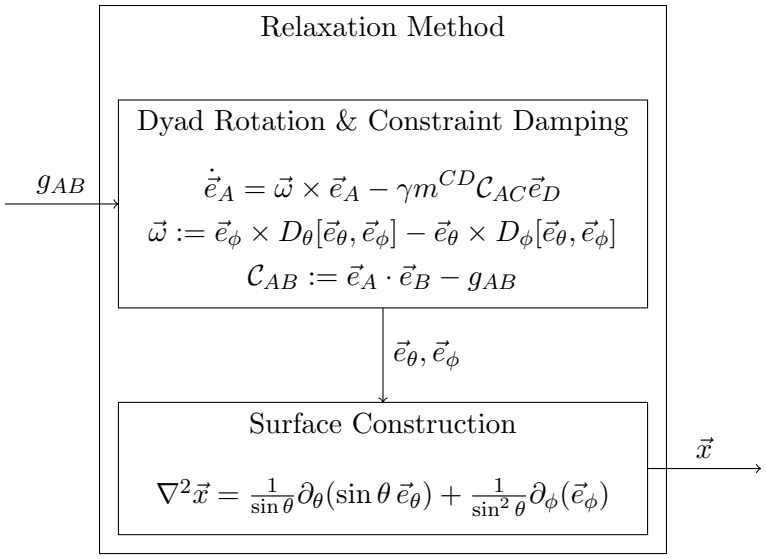


Figure 3.3: Summary of the relaxation method.

Hengrui Zhu, who adapted it from Tichy et al. [15]. The discussion below, included for completeness, is adapted from Zhu's description in our in-preparation paper [17].

The main idea of the matrix method is to convert our embedding PDEs into algebraic equations that can be solved using standard routines. Like most PDE methods in `SpEC`, this approach uses the method of *pseudospectral collocation*. This is a framework where any unknown function is written as a sum of *basis functions*, and the equations are then evaluated on a special grid of *collocation points*, carefully chosen to maximize the accuracy of the discretized versions of the PDEs.

To start, we represent our embedding functions $x^i \in \{x, y, z\}$ as an expansion of basis functions:

$$x^i(\theta, \phi) = \sum_{n=0}^{n_{\max}} d^{in} D_n(\theta, \phi) = d^{in} D_n(\theta, \phi), \tag{3.28}$$

where $d^{in} \in \{d^{xn}, d^{yn}, d^{zn}\}$ are the coefficients for a given basis function D_n. Here, our basis functions are the conventional spherical harmonics $Y_{\ell,m}(\theta, \phi)$, with some appropriate translation of spherical-harmonic indices ℓ, m into the overall index n.

In order to avoid singularity at the poles ($\theta = 0$ and $\theta = \pi$), we choose a special grid of collocation points where the θ values are determined by roots of Legendre polynomials. This technique is called Gauss-Legendre Collocation [18]. Hence, we define our angle coordinates

to be

$$\theta_j = \arccos(p_j^N) \qquad \text{for } 0 \le j < N, \tag{3.29a}$$

$$\phi_k = 2\pi \frac{k}{2N} \qquad \text{for } 0 \le k < 2N, \tag{3.29b}$$

where $N = L + 1$ (with L being the highest ℓ value used in the set of spherical-harmonic basis functions) and p_j^N is the jth root of the Nth degree Legendre polynomial $P_N(\cos\theta)$. We can also re-parameterize the j, k indices with

$$q := j \times 2N + k \tag{3.30}$$

such that $0 \le q < N_q$, where $N_q = 2N^2$ is the total number of grid points.

Having a complete basis, the partial derivatives of our embedding functions relative to $A \in \{\theta, \phi\}$ become

$$\partial_A x^i(\theta, \phi) = d^{in}\, \partial_A D_n(\theta, \phi). \tag{3.31}$$

Note that (3.31) allows us to transform partial derivatives into simple multiplications since we know $\partial_A D_n(\theta, \phi)$ analytically. Using (3.31) in the embedding equations (2.39), we have

$$(d^{xn}\partial_A D_n)(d^{xm}\partial_B D_m) + (d^{yn}\partial_A D_n)(d^{ym}\partial_B D_m) + (d^{zn}\partial_A D_n)(d^{zm}\partial_B D_m) = g_{AB}, \tag{3.32}$$

which can be rearranged as

$$\partial_A D_n \partial_B D_m\ (d^{xn}d^{xm} + d^{yn}d^{ym} + d^{zn}d^{zm}) = g_{AB}. \tag{3.33}$$

In order to solve equations (3.33), we can use a Newton-Raphson method. To that end, let

$$F_{AB}(d^{x0}, d^{x1}, ..., d^{zn_{\max}}) = \partial_A D_n \partial_B D_m\ (d^{xn}d^{xm} + d^{yn}d^{ym} + d^{zn}d^{zm}) - g_{AB}. \tag{3.34}$$

Therefore, we want to find the roots $\{d^{ip}\} = \{d^{x0}, d^{x1}, ..., d^{zn_{\max}}\}$ such that $F_{AB}(\{d^{ip}\}) = 0$. Note that, if we keep A and B fixed, (3.34) actually represents N_q equations (one for each grid point). To make this clearer, let

$$f_{ABq}(\{d^{ip}\}) = F_{AB}(\{d^{ip}\})|_{\theta_j, \phi_k}. \tag{3.35}$$

Assuming that our current values for $\{d^{ip}\}$ make $f_{ABq}(\{d^{ip}\}) \approx 0$, we need to find changes $\{\delta d^{ip}\}$ that will drive it closer to zero:

$$f_{ABq}(\{d^{ip} + \delta d^{ip}\}) = 0. \tag{3.36}$$

Expanding $f_{ABq}(\{d^{ip} + \delta d^{ip}\})$ in a Taylor series and neglecting the terms of order $(\delta d^{ip})^2$ and higher, we have

$$f_{ABq}(\{d^{ip}\}) + \sum_{i\in\{x,y,z\}} \sum_{p=0}^{n_{\max}} \frac{\partial f_{ABq}}{\partial d^{ip}} \delta d^{ip} \approx 0. \tag{3.37}$$

Expanding the indices q and ip, equation (3.37) takes a matrix form:

$$\begin{pmatrix} f_{AB0} \\ f_{AB1} \\ \vdots \\ f_{AB(N_q-1)} \end{pmatrix} + \begin{pmatrix} \frac{\partial f_{AB0}}{\partial d^{x0}} & \frac{\partial f_{AB0}}{\partial d^{x1}} & \cdots & \frac{\partial f_{AB0}}{\partial d^{zn_{\max}}} \\ \frac{\partial f_{AB1}}{\partial d^{x0}} & \frac{\partial f_{AB1}}{\partial d^{x1}} & \cdots & \frac{\partial f_{AB1}}{\partial d^{zn_{\max}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{AB(N_q-1)}}{\partial d^{x0}} & \frac{\partial f_{AB(N_q-1)}}{\partial d^{x1}} & \cdots & \frac{\partial f_{AB(N_q-1)}}{\partial d^{zn_{\max}}} \end{pmatrix} \begin{pmatrix} \delta d^{x0} \\ \delta d^{x1} \\ \vdots \\ \delta d^{zn_{\max}} \end{pmatrix} = 0. \tag{3.38}$$

Now, we can use any standard linear algebra package to solve equations (3.38) for $\{\delta d^{ip}\}$. In our implementation, we use LAPACK's `dgelsy` routine. Once the result is found, we can update the spectral coefficients:

$$d^{ip}|_{\text{new}} = d^{ip}|_{\text{old}} + \delta d^{ip}. \tag{3.39}$$

We repeat this process until $F_{AB} = 0$ up to some target accuracy. Note that (3.38) represents N_q algebraic equations for fixed values of A and B. Since $g_{AB} \in \{g_{\theta\theta}, g_{\phi\phi}, g_{\theta\phi}\}$, we must solve 3 matrix problems like (3.38), resulting in a total of $3N_q = 6N^2$ algebraic equations.

Chapter 4

Code Implementations

In the previous chapter, we discussed the theory behind the numerical methods, but how can we implement them? After all, the only way we can actually use them is if they are implemented in some code. To this end, I devote this chapter to discussing some details that we have to worry about when implementing these methods.

In sections 4.1 and 4.2, I present the two main codes used in this project. Their fundamental difference relates to how they compute derivatives numerically (finite-difference or spectral approaches). In both sections, I focus exclusively on the relaxation method, as it is the one I have worked on the implementation of. In section 4.3, I discuss the test cases we used for analyzing both methods, including the matrix method implementation developed by Hengrui Zhu.

4.1 Finite-Difference Embedding Code (`FiDEC`)

After the idealization of the relaxation method, we had to test it in an isolated environment to make sure that it worked. To this end, I developed the Finite-Difference Embedding Code (`FiDEC`), which is open-source and available on github.com/iago-mendes/fidec. Along the way, we discovered issues with the initial method and improved it.

To start our discussion of `FiDEC`, let's talk about its θ and ϕ coordinate grid. We use a uniform grid distribution:

$$\theta_i = \frac{\pi}{N_\theta}\left(i + \frac{1}{2}\right) \qquad \text{for } 0 \leq i < N_\theta, \tag{4.1}$$

$$\phi_j = \frac{2\pi}{N_\phi} j \qquad \text{for } 0 \leq j < N_\phi, \tag{4.2}$$

where N_θ and N_ϕ are specified by the user. The $1/2$ factor in (4.1) avoids placing any grid

points at the pole singularities. Note that (4.1) and (4.2) lead to uniform grid divisions

$$\delta\theta = \frac{\pi}{N_\theta} \quad \text{and} \quad \delta\phi = \frac{2\pi}{N_\phi}, \tag{4.3}$$

respectively.

As the name suggests, `FiDEC` uses finite-difference approximations to compute numerical derivatives, taking weight coefficients from [19] and [20]. For θ derivatives, we use the centered and one-sided (forward or backward) formulas for interior and boundary points, respectively:

$$\partial_\theta f(\theta_i, \phi_j) = \frac{1}{\delta\theta} \begin{cases} -\frac{1}{2} f(\theta_{i-1}, \phi_j) + \frac{1}{2} f(\theta_{i+1}, \phi_j) & \text{if } 0 < i < N_\theta - 1 \\ -\frac{3}{2} f(\theta_0, \phi_j) + 2 f(\theta_1, \phi_j) - \frac{1}{2} f(\theta_2, \phi_j) & \text{if } i = 0 \\ \frac{3}{2} f(\theta_{N_\theta - 1}, \phi_j) - 2 f(\theta_{N_\theta - 2}, \phi_j) + \frac{1}{2} f(\theta_{N_\theta - 3}, \phi_j) & \text{if } i = N_\theta - 1 \end{cases}. \tag{4.4}$$

For ϕ derivatives, we used the centered formula, wrapping around the grid for boundary points:

$$\partial_\phi f(\theta_i, \phi_j) = \frac{1}{\delta\phi} \begin{cases} -\frac{1}{2} f(\theta_i, \phi_{j-1}) + \frac{1}{2} f(\theta_i, \phi_{j+1}) & \text{if } 0 < j < N_\phi - 1 \\ -\frac{1}{2} f(\theta_i, \phi_{N_\phi - 1}) + \frac{1}{2} f(\theta_i, \phi_1) & \text{if } j = 0 \\ -\frac{1}{2} f(\theta_i, \phi_{N_\phi - 2}) + \frac{1}{2} f(\theta_i, \phi_0) & \text{if } j = N_\phi - 1 \end{cases}. \tag{4.5}$$

The same logic is used for second-order derivatives $\partial_\theta \partial_\theta f$ and $\partial_\phi \partial_\phi f$, which are omitted here for simplicity.

As for m_{CD}, we chose to start with the simplest case, allowing us to focus on the other details of the implementation. Therefore, `FiDEC` uses $m_{CD} = u_{CD}$[1], making the evolution equations $\dot{\vec{e}}_A$ simply (3.25). Additionally, we use the unit sphere as the initial guess for most relaxation solves.

As a last note, it is worth talking about how we approach the last step of the relaxation method: surface construction. As discussed in subsection 3.1.3, we need to solve three Poisson equations given by

$$\nabla^2 \vec{x} = \text{AngularDivergence}(\vec{e}_\theta, \vec{e}_\phi), \tag{4.6}$$

where I am using the notation $\text{AngularDivergence}(\vec{e}_\theta, \vec{e}_\phi)$ to represent the right side of (3.27d). `FiDEC` approaches these equations by choosing an initial guess and applying a simple relaxation scheme:

$$\vec{x}|_{\text{new}} = \vec{x}|_{\text{old}} + \delta t\, \dot{\vec{x}}|_{\text{old}}, \tag{4.7}$$

[1] Metric of a unit sphere.

where

$$\dot{\vec{x}} = \nabla^2 \vec{x} - \text{AngularDivergence}(\vec{e}_\theta, \vec{e}_\phi). \tag{4.8}$$

We have found that running the relaxations (4.7) and (3.2) together is convenient.

4.2 Spectral Einstein Code (SpEC)

After testing the relaxation method with FiDEC, we had enough confidence in the method and decided to implement it in a more useful codebase. This was when we started working with the Spectral Einstein Code (SpEC) [21], which is the main code from the SXS collaboration for simulating binary black hole mergers. To get a sense of its relevance in the gravitational physics community, SpEC was used in the first detection of gravitational waves by LIGO [2].

When it was being created, SpEC differed from the existing codes (that used mostly finite differencing) because of its use of spectral methods, which turned out to work better for its purposes [22]. As discussed in section 3.2, this approach allows us to write any functions as an expansion of spherical harmonics:

$$F(\theta,\phi) = \sum_{\ell,m} f^{\ell,m} Y_{\ell,m}(\theta,\phi) = f^{\ell,m} Y_{\ell,m}(\theta,\phi) \tag{4.9}$$

$$\Rightarrow \quad \partial_A F(\theta,\phi) = f^{\ell,m} \partial_A Y_{\ell,m}(\theta,\phi), \tag{4.10}$$

where $F(\theta,\phi)$ is a generic function and $f^{\ell,m}$ is its spectral coefficient for the spherical harmonic $Y_{\ell,m}(\theta,\phi)$. Since we know $\partial_A Y_{\ell,m}(\theta,\phi)$ analytically, this approach allows us to compute derivatives exactly up to the expansion truncation, which is determined by $\ell \leq L$ for some L that sets our resolution.

In its implementation of spectral methods, SpEC uses SPHEREPACK [23], a software package for spherical harmonic transforms. The reason why this is relevant for us here is that SPHEREPACK uses a non-coordinate basis such that its derivatives are given by

$$\tilde{\partial}_\theta := \partial_\theta, \tag{4.11}$$

$$\tilde{\partial}_\phi := \frac{1}{\sin\theta} \partial_\phi, \tag{4.12}$$

where we use $\tilde{\partial}_A$ and ∂_A to indicate non-coordinate and coordinate derivatives, respectively. Hence, we need to rearrange our equations in terms of $\tilde{\partial}_\theta$ and $\tilde{\partial}_\phi$, as these are the derivatives we are able to compute in SpEC.

In this process, we also have to resolve a common problem when working with spectral methods: derivatives of non-smooth functions. Different than finite-differencing, spectral methods are unforgiving when we attempt to differentiate a non-smooth function. In our

equations, for example, $\vec{e}_\theta = \partial_\theta \vec{x}$ (or any other quantity that results from θ-derivatives) is not smooth at the coordinate singularities $\theta = 0$ and $\theta = \pi$, but we need to differentiate it in our relaxation method equations. While addressing this issue, we also choose to use second-derivative approximations instead of applying first-derivative approximations twice, increasing our numerical precision.

Consider a generic function $f(\theta, \phi)$ that we assume to be non-smooth. Our goal is to make it smooth, which can be done by multiplying it by $\sin\theta$, so that we can take its derivative. Note that

$$\partial_\theta(\sin\theta\, f) = \cos\theta\, f + \sin\theta\, \partial_\theta f \tag{4.13a}$$

$$\Rightarrow \quad \partial_\theta f = \frac{1}{\sin\theta}\partial_\theta(\sin\theta\, f) - \frac{\cos\theta}{\sin\theta} f \tag{4.13b}$$

and

$$\partial_\phi(\sin\theta\, f) = \sin\theta\, \partial_\phi f \tag{4.14a}$$

$$\Rightarrow \quad \partial_\phi f = \frac{1}{\sin\theta}\partial_\phi(\sin\theta\, f). \tag{4.14b}$$

With this, we can replace any derivative of non-smooth functions with quantities that we can compute numerically.

In the relaxation method, angular derivatives appear mostly inside the operator $D_A[\vec{e}_\theta, \vec{e}_\phi]$, used in the definition of the angular velocity $\vec{\omega}$ (3.13). Then, we need to rewrite $D_A[\vec{e}_\theta, \vec{e}_\phi]$ using (4.13b) and (4.14b). For $A = \theta$, we have

$$D_\theta[\vec{e}_\theta, \vec{e}_\phi] = \frac{1}{\sin\theta}\partial_\theta\Big\{ \sin\theta\, [\vec{e}_\theta, \vec{e}_\phi]\Big\} \tag{4.15a}$$

$$= \frac{1}{\sin\theta}\partial_\theta\Big\{ \sin\theta\, (\partial_\theta\vec{e}_\phi - \partial_\phi\vec{e}_\theta)\Big\} \tag{4.15b}$$

$$= \frac{1}{\sin\theta}\partial_\theta\Big\{ \sin\theta\, \partial_\theta\vec{e}_\phi - \sin\theta\, \partial_\phi\vec{e}_\theta\Big\} \tag{4.15c}$$

$$= \frac{1}{\sin\theta}\partial_\theta\Big\{ \sin\theta\, \partial_\theta\vec{e}_\phi - \partial_\phi(\sin\theta\, \vec{e}_\theta)\Big\} \tag{4.15d}$$

$$= \frac{1}{\sin\theta}\Big\{ \cos\theta\, \partial_\theta\vec{e}_\phi + \sin\theta\, \partial_\theta\partial_\theta\vec{e}_\phi - \partial_\theta\partial_\phi(\sin\theta\, \vec{e}_\theta)\Big\} \tag{4.15e}$$

$$= \frac{\cos\theta}{\sin\theta}\partial_\theta\vec{e}_\phi + \partial_\theta\partial_\theta\vec{e}_\phi - \frac{1}{\sin\theta}\partial_\theta\partial_\phi(\sin\theta\, \vec{e}_\theta). \tag{4.15f}$$

Similarly, for $A = \phi$, we have

$$D_\phi[\vec{e}_\theta, \vec{e}_\phi] = \frac{1}{\sin\theta}\partial_\phi\Big\{ \sin\theta\,[\vec{e}_\theta, \vec{e}_\phi]\Big\} \tag{4.16a}$$
$$= \partial_\phi\Big\{[\vec{e}_\theta, \vec{e}_\phi]\Big\} \tag{4.16b}$$
$$= \partial_\phi\Big\{\partial_\theta\vec{e}_\phi - \partial_\phi\vec{e}_\theta\Big\} \tag{4.16c}$$
$$= \partial_\phi\partial_\theta\vec{e}_\phi - \partial_\phi\partial_\phi\vec{e}_\theta \tag{4.16d}$$
$$= \partial_\phi\partial_\theta\vec{e}_\phi - \frac{1}{\sin\theta}\partial_\phi\partial_\phi(\sin\theta\,\vec{e}_\theta). \tag{4.16e}$$

With this, we have successfully avoided derivatives of any non-smooth function.

However, equations (4.15f) and (4.16e) are still in terms of coordinate derivatives. Using non-coordinate derivatives in (4.15f), we have

$$D_\theta[\vec{e}_\theta, \vec{e}_\phi] = \frac{\cos\theta}{\sin\theta}\tilde{\partial}_\theta\vec{e}_\phi + \tilde{\partial}_\theta\tilde{\partial}_\theta\vec{e}_\phi - \frac{1}{\sin\theta}\tilde{\partial}_\theta\Big\{ \sin\theta\,\tilde{\partial}_\phi(\sin\theta\,\vec{e}_\theta)\Big\} \tag{4.17a}$$
$$= \frac{\cos\theta}{\sin\theta}\tilde{\partial}_\theta\vec{e}_\phi + \tilde{\partial}_\theta\tilde{\partial}_\theta\vec{e}_\phi - \frac{\cos\theta}{\sin\theta}\tilde{\partial}_\phi(\sin\theta\,\vec{e}_\theta) - \tilde{\partial}_\theta\tilde{\partial}_\phi(\sin\theta\,\vec{e}_\theta) \tag{4.17b}$$
$$= \frac{\cos\theta}{\sin\theta}\Big\{\tilde{\partial}_\theta\vec{e}_\phi - \tilde{\partial}_\phi(\sin\theta\,\vec{e}_\theta)\Big\} + \tilde{\partial}_\theta\tilde{\partial}_\theta\vec{e}_\phi - \tilde{\partial}_\theta\tilde{\partial}_\phi(\sin\theta\,\vec{e}_\theta). \tag{4.17c}$$

Doing the same with (4.16e), we have

$$D_\phi[\vec{e}_\theta, \vec{e}_\phi] = \sin\theta\,\tilde{\partial}_\phi\tilde{\partial}_\theta\vec{e}_\phi - \sin\theta\,\tilde{\partial}_\phi\tilde{\partial}_\phi(\sin\theta\,\vec{e}_\theta). \tag{4.18a}$$

This makes the implementation of the operators $D_A[\vec{e}_\theta, \vec{e}_\phi]$ straightforward and safe against non-smooth functions.

For m_{CD}, we have focused on $\vec{e}_C \cdot \vec{e}_D$[2] and g_{CD}[3]. In our tests, we have found that the second choice is faster and performs well, so we chose to use it for all results shown here. Moreover, as expected, the relaxation method depends a lot on the initial guess. If it is close enough to the desired solution, the method should be very efficient. However, the opposite is true for an initial guess that is much different.

SpEC has an apparent horizon finder that outputs a *coordinate shape* of this surface. This shape depends on the choice of coordinates (commonly called *gauge conditions* in the field), but the details of how this is done are beyond the scope of this thesis. That said, we can often use this surface as a decent initial guess for the relaxation method. For the test cases presented in the next section, we use the coordinate shape for a Kerr horizon with $\chi = 0.7$. This can be improved quite a lot for a simulation of binary black holes. Since the apparent horizons do not have sudden changes in a short timescale, the embedding at the

[2] "Metric" of the current surface.
[3] Metric of the desired surface.

previous time is an excellent initial guess for the relaxation method. For the next chapter, this choice of initial guess is used.

Finally, because we are working with spectral methods, we can solve the Poisson equations (4.6) analytically. To do so, we have to apply an inverse unit sphere Laplacian on the right side of (4.6):

$$\vec{x} = \text{InverseLaplacian}\Big\{\text{AngularDivergence}(\vec{e}_\theta, \vec{e}_\phi)\Big\}. \tag{4.19}$$

This approach requires no iteration and can be done at any time in the dyad relaxation. At all moments, equation (4.19) gives us the surface represented by the current values of the dyad vectors. In order to speed up the commutation of $\vec{e}_\theta$ and $\vec{e}_\phi$, we choose to periodically update these dyad vectors using on the surface reconstructed by (4.19).

4.3 Test Cases

Here, I present some test cases used to study both the relaxation and matrix methods. All of these results were achieved from the `SpEC` implementation, but it is possible to achieve similar ones in `FiDEC`. Over the course of this project, we have used many test cases, the most important of which are shown in Table 4.1.

As usual, we started with a trivial test case: a round sphere. Once we got it working, we incrementally increased the complexity of our surfaces to test specific features. The dented sphere was useful for testing if we were able to embed surfaces with slightly negative curvatures. The motivation behind the ellipsoid was a twofold: it gave us a well-known analytic solution to compare against the embedding results and allowed us to test for non-axisymmetric (not symmetric relative to the z-axis) surfaces. The Kerr horizon was a particularly important case because our end goal was to apply the embedding methods to binary black hole (BBH) simulations, which often start with individual rotating black holes. With the z-peanut, we were able to test surfaces with significant negative curvatures. Finally, the x-peanut was important to combine non-axisymmetry with negative curvatures, as well as approximating surfaces commonly found in BBH simulations right after merger.

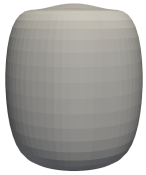
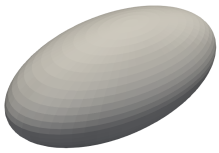
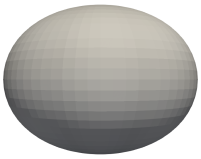
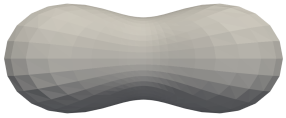
Now, let's delve into specific results from these test cases.

4.3.1 Convergence

A common practice when developing new methods is to analyze how errors change as we increase resolution. This is usually referred to as "convergence tests." To quantify such errors, we define a *residual* by taking the difference of the sides of the embedding equations (2.39),

$$\mathcal{R}_{AB} := (\partial_A \vec{x}) \cdot (\partial_B \vec{x}) - g_{AB}, \tag{4.25}$$

Table 4.1: Surfaces used as test cases for the relaxation and matrix methods.

Name	Embedding	Description	Metric
Dented Sphere		Sphere with dents near the poles.	$g_{\theta\theta} = \cos^2\theta + \sin^2\theta \times (3\cos(2\theta) - 1)^2/4$ $g_{\phi\phi} = \sin^2\theta$ $g_{\theta\phi} = 0$
Ellipsoid		Ellipsoid with semi-axes $a = 2$, $b = 1$, and $c = 1$ in the x, y, and z directions, respectively.	$g_{\theta\theta} = (a^2\cos^2\phi + b^2\sin^2\phi) \times \cos^2\theta + c^2\sin^2\theta$ $g_{\phi\phi} = (b^2\cos^2\phi + a^2\sin^2\phi) \times \sin^2\theta$ $g_{\theta\phi} = (b^2 - a^2)\cos\theta\sin\theta \times \cos\phi\sin\phi$
Kerr Horizon		Horizon of a rotating black hole with a dimensionless spin $\chi = 0.7$.	$g_{\theta\theta} = \sigma$ $g_{\phi\phi} = \frac{4\rho^2}{\sigma}\sin^2\theta$ $g_{\theta\phi} = 0$ $\sigma := 2\rho - \chi^2\sin^2\theta$ $\rho := 1 + \sqrt{1 - \chi^2}$
Z-Peanut		Peanut-shaped surface aligned with the z-axis. The thickness parameter a is set to 0.7.	$g_{\theta\theta} = 1$ $g_{\phi\phi} = (1 - a\sin^2\theta)^2\sin^2\theta$ $g_{\theta\phi} = 0$
X-Peanut		Peanut-shaped surface aligned with the x-axis. The thickness parameter s_0 is set to 1.1.	$g_{\theta\theta} = (\partial_\theta r)^2 + r^2$ $g_{\phi\phi} = (\partial_\phi r)^2 + r^2\sin^2\theta$ $g_{\theta\phi} = (\partial_\theta r)(\partial_\phi r)$ $r := \sqrt{\frac{-b + \sqrt{b^2 - 4 + 4s_0^4}}{2}}$ $b := 2 - 4\sin^2\theta\cos^2\phi$

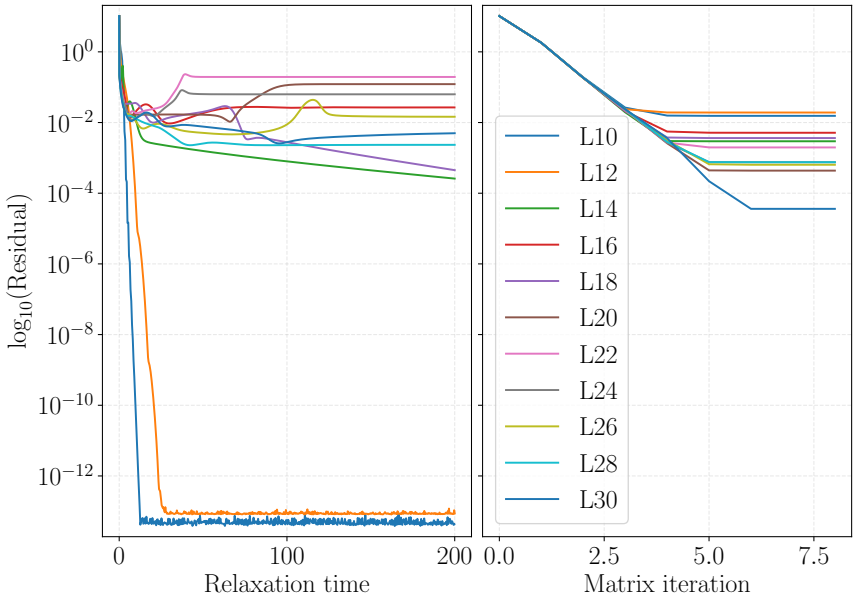


Figure 4.1: Residual evolution for the dented sphere. No consistent pattern is seen.

and averaging its norm over the surface:

$$R := \sqrt{\int |\mathcal{R}|^2 \, dA}, \tag{4.26}$$

where $|\mathcal{R}|^2$ is the inner product of $\mathcal{R}_{AB}$ with itself for $A, B \in \{\theta, \phi\}$.

Looking at how the residuals R change within the relaxation and matrix methods for different resolutions, we have the plots shown in Figures 4.1–4.5. "Relaxation time" refers to the number of iterations multiplied by the time step δt in the relaxation method. "Matrix iteration" refers to the number of Newton-Raphson iterations taken in the matrix method. These plots highlight one of the main differences between the two approaches: while the relaxation method takes many quick steps, the matrix method takes very few long steps.

Additionally, we can see that the relaxation residuals reach *asymptotes* that get smaller as resolution increases. This is a feature that appears in many test cases, and we believe it is related to a slower relaxation rate near the coordinate poles. Figure 4.6 shows how the final residuals are typically distributed over the θ and ϕ grid, from which it is evident that the residuals are larger near the coordinate poles by one order of magnitude, supporting our hypothesis. From Figures 4.4 and 4.5, we see that the residuals sometimes grow after reaching some minimum value, which could be associated with some numerical error that gets accumulated once the relaxation asymptotes.

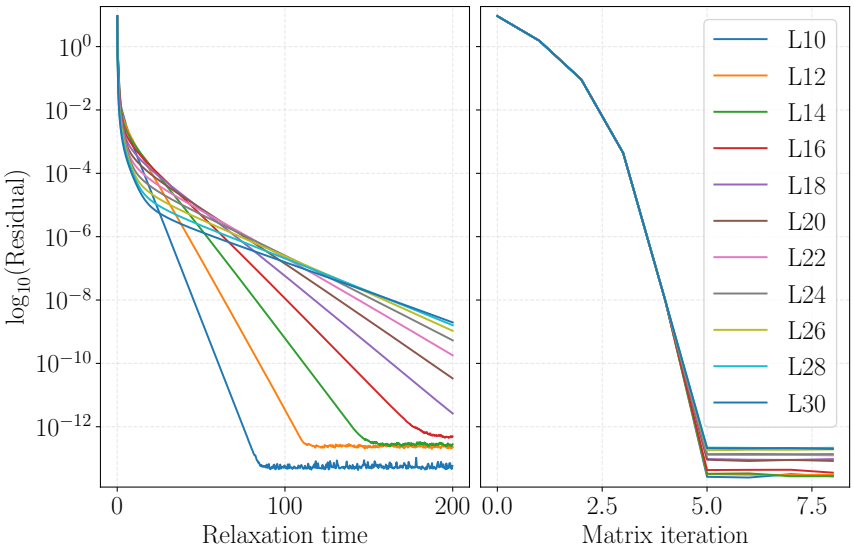


Figure 4.2: Residual evolution for the ellipsoid. Due to computational costs, higher-resolution tests were stopped before they reached the asymptote.

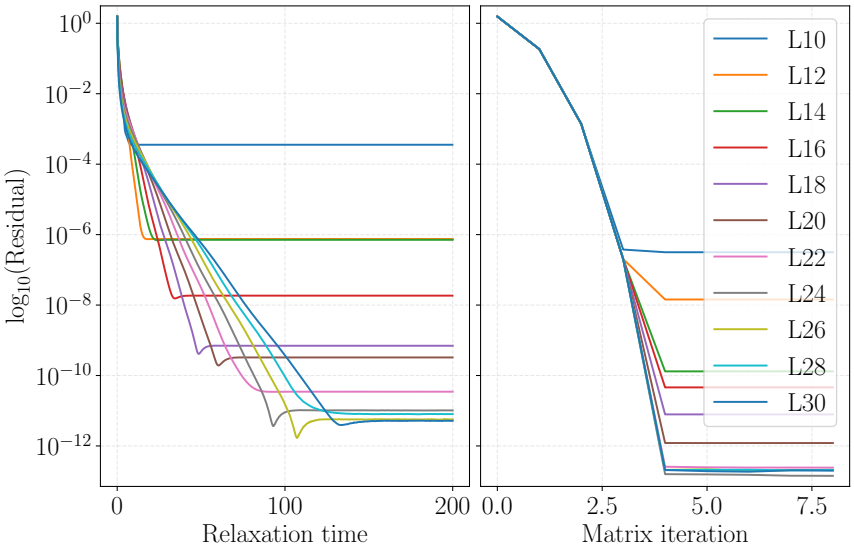


Figure 4.3: Residual evolution for the Kerr horizon.

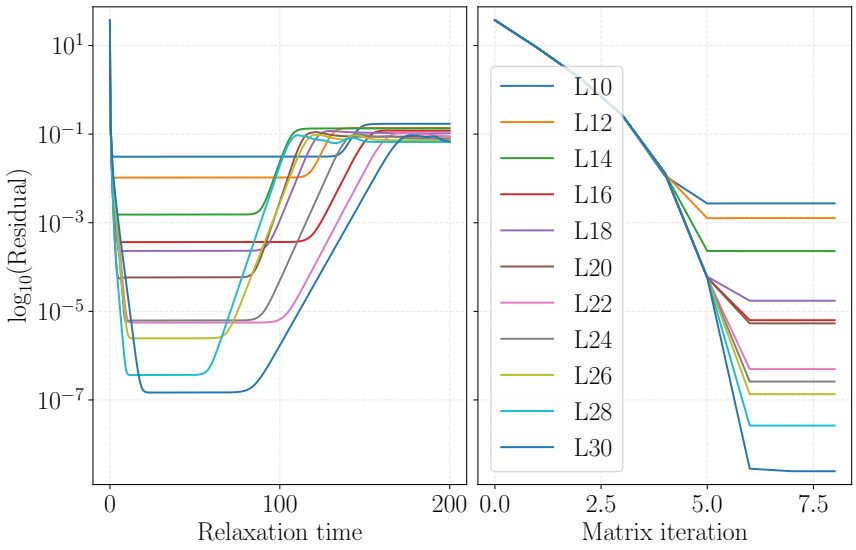


Figure 4.4: Residual evolution for the z-peanut.

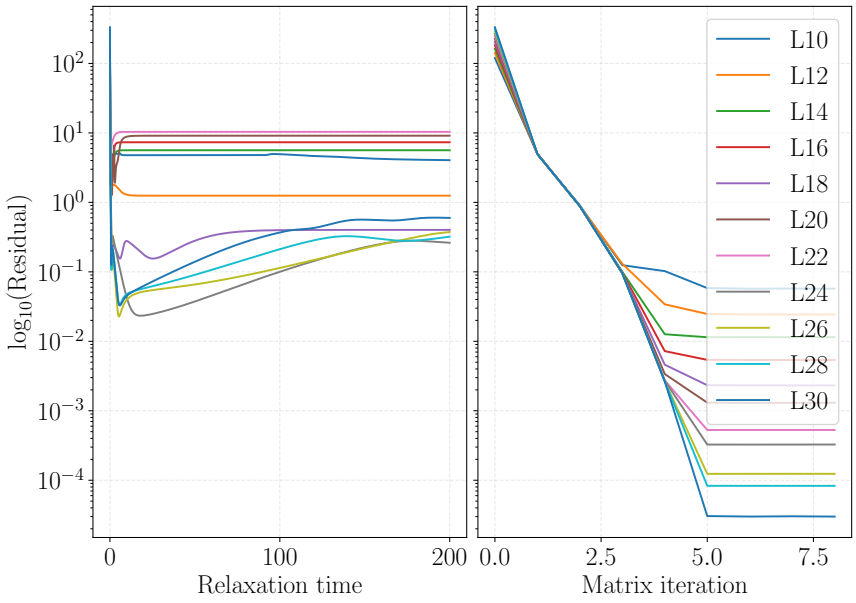


Figure 4.5: Residual evolution for the x-peanut.

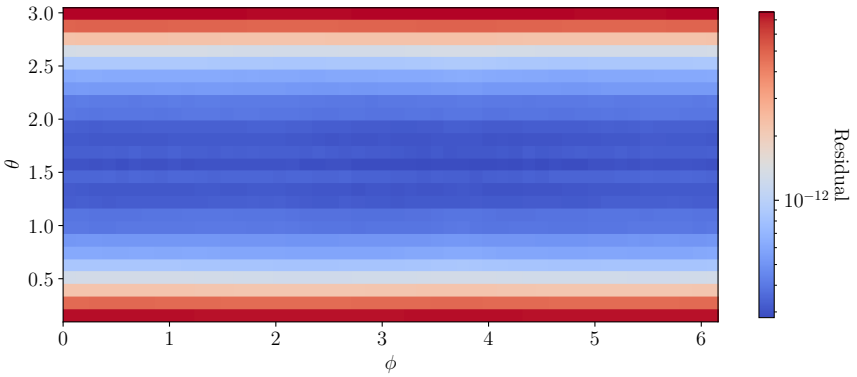


Figure 4.6: Final residual distribution over the θ and ϕ grid for the relaxation method with a Kerr horizon with $\chi = 0.7$ at a resolution of $L = 24$.

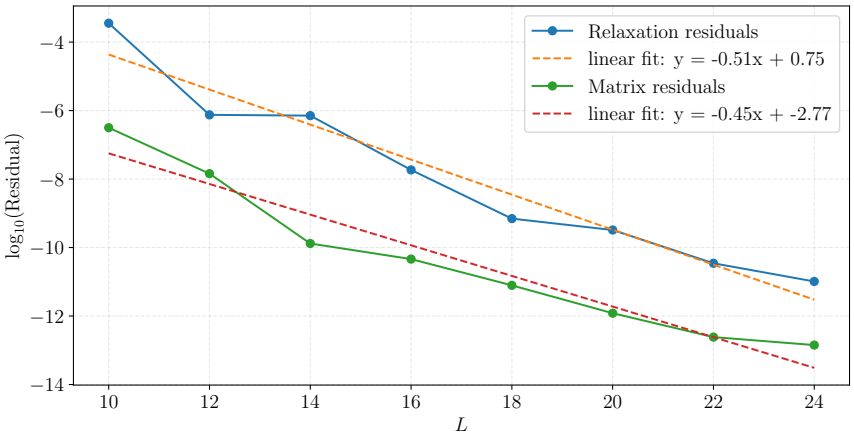


Figure 4.7: Convergence plot for the Kerr horizon.

If we plot the best residuals[4] versus resolution, we can analyze the convergence of both methods. This is done for selected test cases in Figures 4.7–4.9, in which I use a semi-log plot to indicate any exponential decay as a straight line. From the linear fits shown, it is evident that both methods converge exponentially as we increase resolution, as expected for spectral methods [18].

4.3.2 Time Scaling

We can also analyze how the wall time (how long the algorithm runs) scales with resolution. This is done for all test cases in Figure 4.10, in which I use a log-log plot to indicate power relations as straight lines. Again, from the linear fits shown, we see that the relaxation method scales as $\sim O(L^4)$, whereas the matrix method scales as $\sim O(L^6)$.

[4]By best residual, I mean the minimum residual as shown in Figures **??**–4.5.

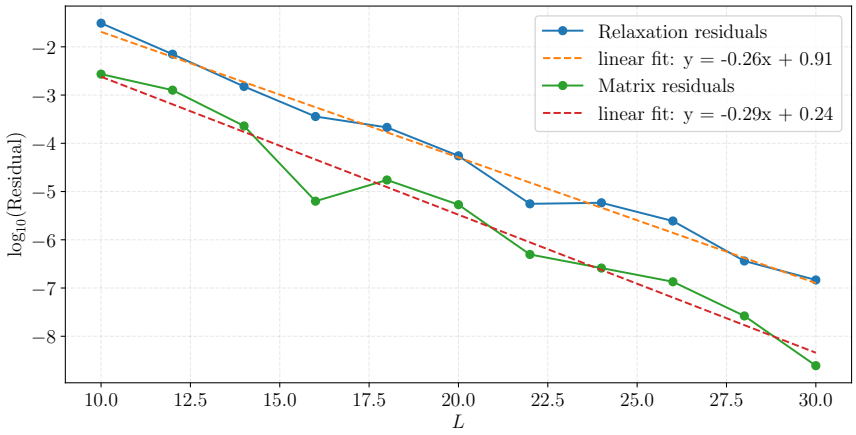


Figure 4.8: Convergence plot for the z-peanut.

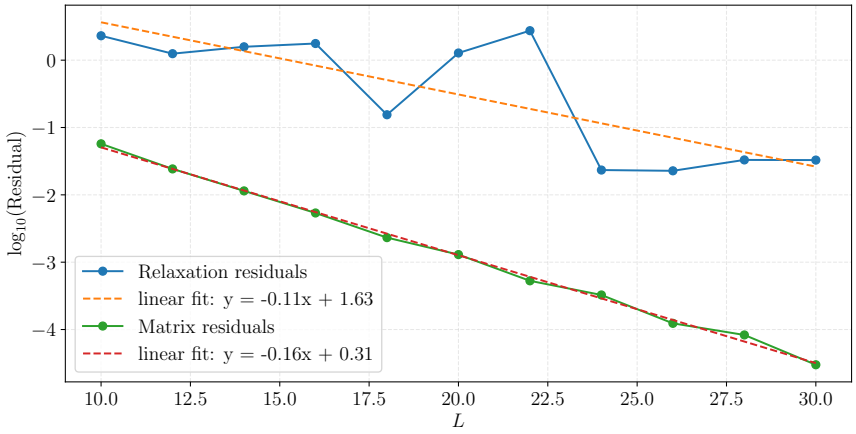


Figure 4.9: Convergence plot for the x-peanut.

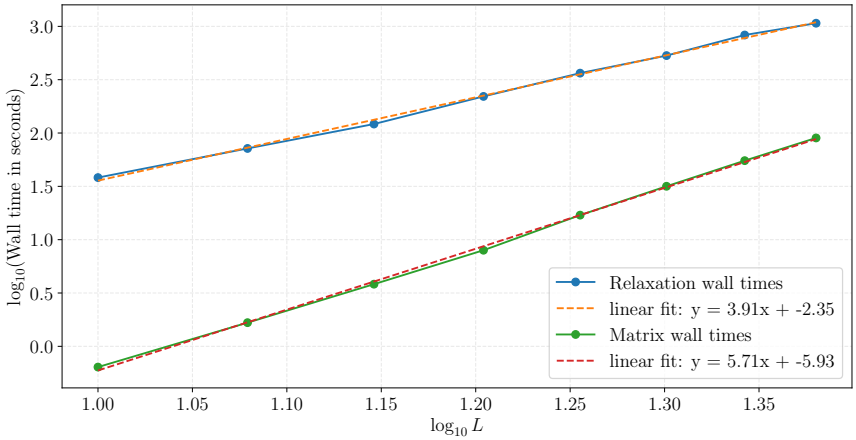


Figure 4.10: Time-scaling plot for the Kerr horizon. All other test cases have time-scaling plots consistent with this one.

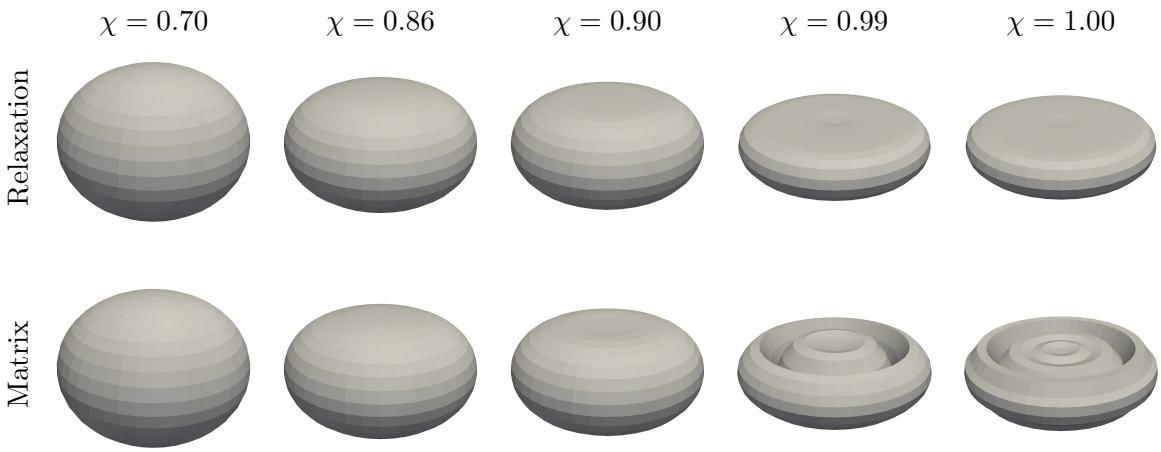


Figure 4.11: Comparison between embedding results from the relaxation and matrix methods as we approach and exceed the embeddability limit for a Kerr horizon.

It is worth mentioning that the times shown in Figure 4.10 are not representative of what we experience in practice. Because of the asymptotic behavior mentioned above, there is no point in continuing the relaxation method once we are close enough to the best achievable residual. To account for this, we often choose to stop the relaxation steps once the relative change in the residuals are less than 0.1%. This choice is used for the remainder of this chapter and in the next one. The reason why we had a fixed final time up to this point is that we would not be able to compare the wall times between different resolutions otherwise.

4.3.3 Kerr Embeddability Limit

Finally, as discussed in subsection 2.2.3, the Kerr horizon is known for not being embeddable if $\chi > \sqrt{3}/2 \approx 0.866$. Knowing this, we can improve our understanding of how the embedding methods behave by studying what happens as that limit is approached and exceeded. For different values of χ, Figure 4.11 shows the embedding results and Figure 4.12 shows the embedding residuals.

From Figure 4.11, we can understand that each method deals with non-embeddability differently. While inaccurate, the relaxation method still outputs a roughly round surface, similar to what one might expect from the Kerr embeddings with $\chi < \sqrt{3}/2$. In contrast, we see that the relaxation method outputs a surface with negative curvature near the poles.

From Figure 4.12, we see that both methods fail (meaning that they have high final residuals) as we get closer to the embeddability limit. Specifically, it appears that the residuals from both methods have a jump near the embeddability limit. Comparing the residuals for $L = 20$ and $L = 30$, we see that this jump gets sharper at higher resolutions.

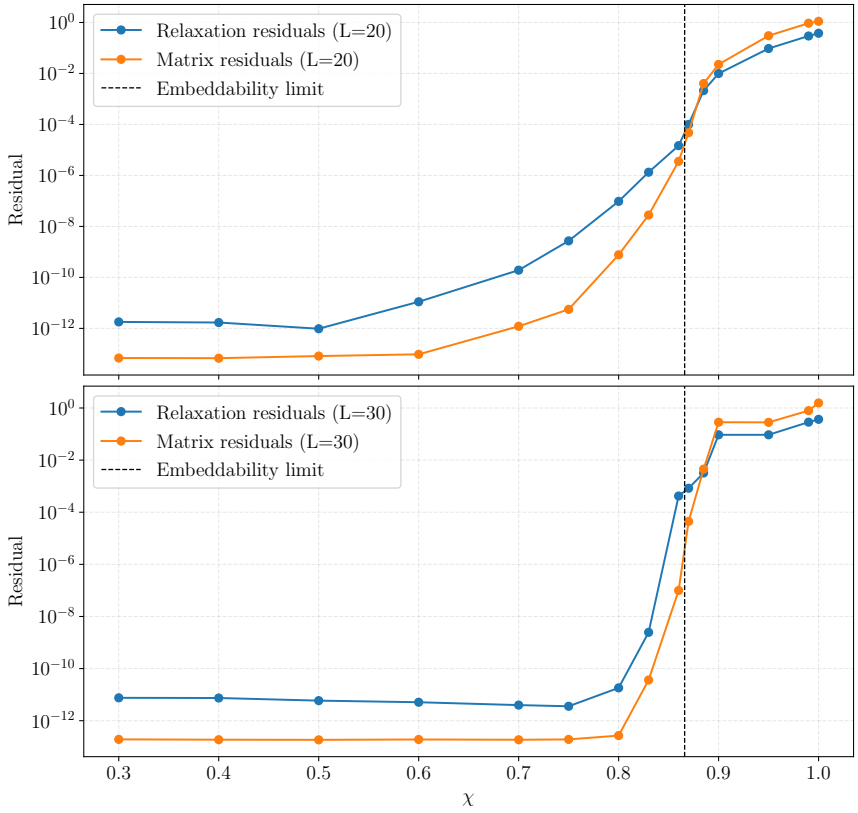


Figure 4.12: Embedding residuals of a Kerr horizon for different χ values.

Chapter 5

Binary Black Hole Merger Simulations

Having implementations and tests of our numerical methods, we are now ready to apply them to simulations of black holes. Specifically, we are interested in Binary Black Hole (BBH) mergers, which can be simulated in `SpEC`. Table 5.1 lists the five runs we will analyze here. For each run, there are three initial parameters of interest: the mass ratio q and the spins of each black hole χ_A and χ_B[1].

Our motivation for the chosen parameters is similar to the one for our test cases. We start with a simple scenario (`EN`), and then analyze increasingly more complex cases. In particular, the `Ca` and `Aa` runs are useful to understand how the tidal structures formed in the horizons depend on the spins being aligned or opposite with the orbital angular momentum. The `q4` run is the most challenging one, as higher mass ratios are in general harder for Numerical Relativity to model. Finally, the `Ge` run is useful to understand how our methods work in a generic scenario. In addition, the spins in `Ge` lead to a precessing orbit, which is also an interesting scenario to simulate.

Once these and other parameters are in the initial data, `SpEC` evolves the binary system until merger, when a common horizon[2] is formed, and beyond. At multiple times during that evolution, `SpEC` runs a horizon finder that outputs the *coordinate shape* of the apparent horizons of the three black holes. These horizons are usually denoted by "Ah" (Apparent horizon) followed by the black hole "letter". In our case, we will refer to the apparent horizon of black holes A, B, and C as AhA, AhB, and AhC, respectively.

Since there are no preferred coordinate system in *General* Relativity, the coordinate shapes of these horizons is arbitrary. That is, we can choose the coordinate system in a specific way to give us any desired shape. Because of this, these surfaces many times do

[1] If $q \neq 1$, `SpEC` defines black hole A to be the more massive object.
[2] The common horizon is usually referred to as black hole C.

Table 5.1: List of BBH merger simulations.

Code	Description	Parameters
`EN`	Equal-mass non-spinning inspiral.	$q = 1$ $\chi_A = (0, 0, 0)$ $\chi_B = (0, 0, 0)$
`Ca`	Co-aligned spins (aligned with the orbit) inspiral.	$q = 1$ $\chi_A = (0, 0, 0.6)$ $\chi_B = (0, 0, 0.6)$
`Aa`	Anti-aligned spins (opposite to the orbit) inspiral.	$q = 1$ $\chi_A = (0, 0, -0.6)$ $\chi_B = (0, 0, -0.6)$
`q4`	Nontrivial mass ratio inspiral.	$q = 4$ $\chi_A = (0, 0, 0)$ $\chi_B = (0, 0, 0)$
`Ge`	Generic configuration inspiral.	$q = 2$ $\chi_A = (0.3, -0.4, 0.2)$ $\chi_B = (0.2, 0.1, -0.3)$

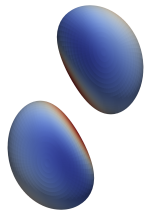
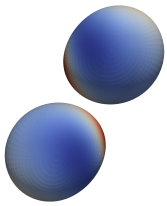
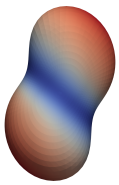
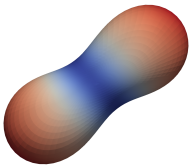
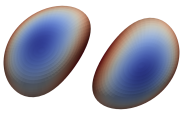
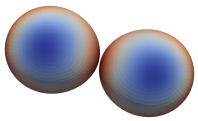
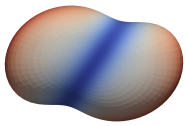
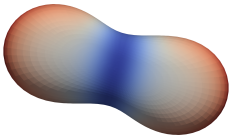
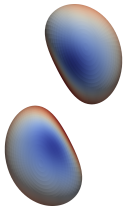
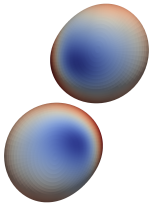
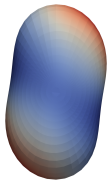
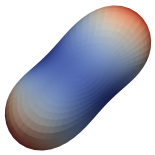
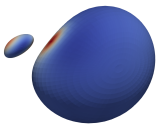
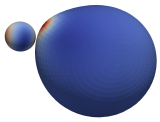
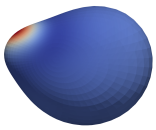
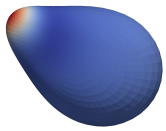
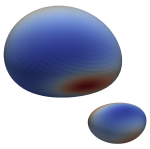
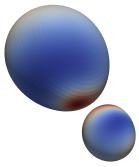
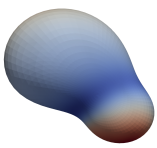
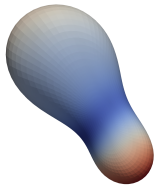
not represent the intrinsic geometry of the apparent horizon.

In contrast, there exists a preferred coordinate system in 3D Euclidean space: rectangular coordinates. Therefore, by embedding these apparent horizon surfaces into flat space, we can visualize its intrinsic geometry without arbitrariness. Table 5.2 compares the coordinate shapes with the isometric embeddings for our five simulations before and after merger. These moments correspond to when the Ricci scalars[3] for the apparent horizons are the most extreme.

From Table 5.2, we can clearly see that the coordinate shapes have *non-physical distortions*. For example, all pre-merger coordinate shapes have a "pancake" shape. Instead of

[3]Simply put, Ricci scalars are measures of curvature.

Table 5.2: Coordinate shapes and isometric embeddings of BBH merger simulations during pre-merger and post-merger. The color maps indicate the dimensionless Ricci scalars for each surface.

Code	Pre-merger Coordinate	Pre-merger Embedding	Post-merger Coordinate	Post-merger Embedding
EN				
Ca				
Aa				
q4				
Ge				

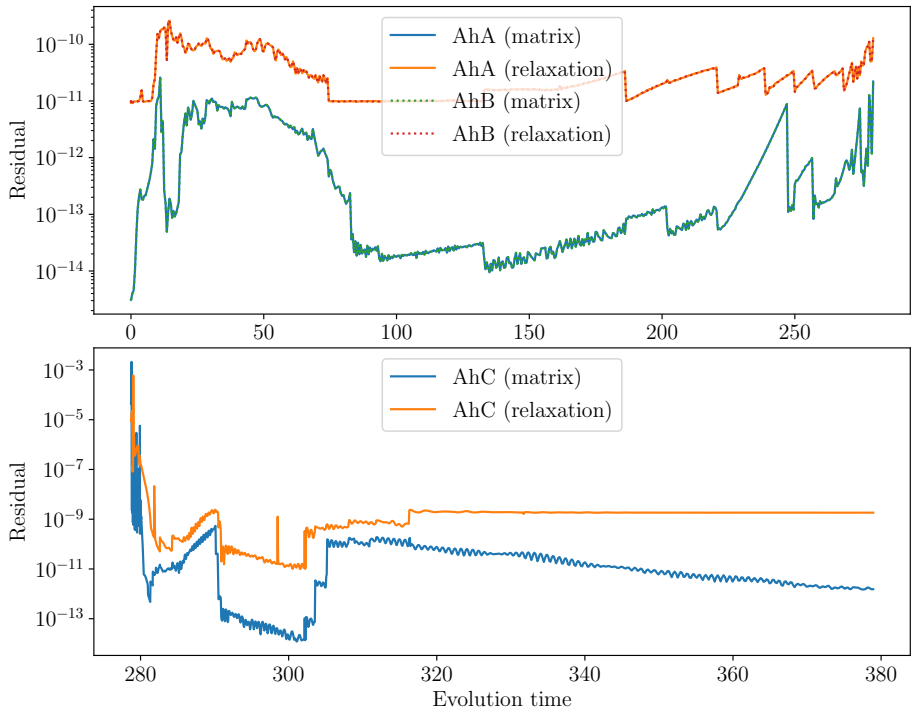


Figure 5.1: Residuals during the `EN` run.

this, we would expect bulges being formed as the apparent horizons approach each other, which is observed in the pre-merger isometric embeddings. Similar distortions are seen in the post-merger horizons.

For animations of the isometric embeddings during the evolution of these five simulations, visit iagomendes.com/embedding-videos.

5.1 Residual & Time Analysis

In addition to looking at visualizations of the isometric embeddings, we can plot the residuals for each method as the BBH evolves. Such plots are shown for all five simulations in Figures 5.1–5.5, where the pre-merger and post-merger phases are shown at the top and bottom, respectively.

Of course, for $q = 1$ runs, the AhA and AhB curves match. For $q \neq 1$ runs (`q4` and `Ge`), we see that AhB generally is embedded with slightly better residuals. More significantly, it is clear that the matrix method is capable of reaching lower residuals than the relaxation method, which agrees with the residuals observed in subsection 4.3.1. Additionally, we see much higher residuals immediately before and after merger, indicating that our methods could be failing at those times.

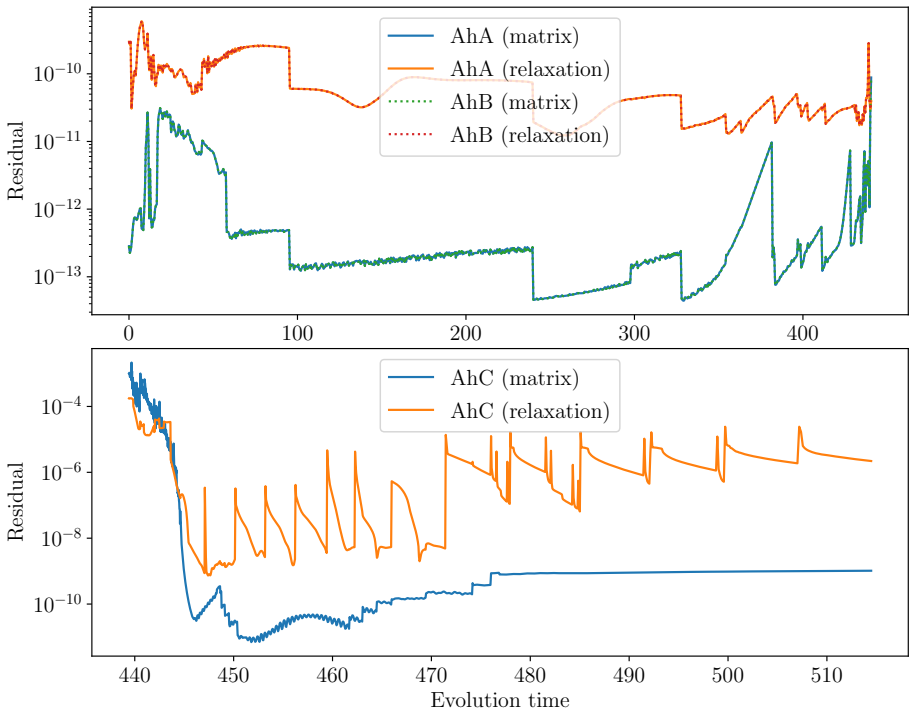


Figure 5.2: Residuals during the `Ca` run.

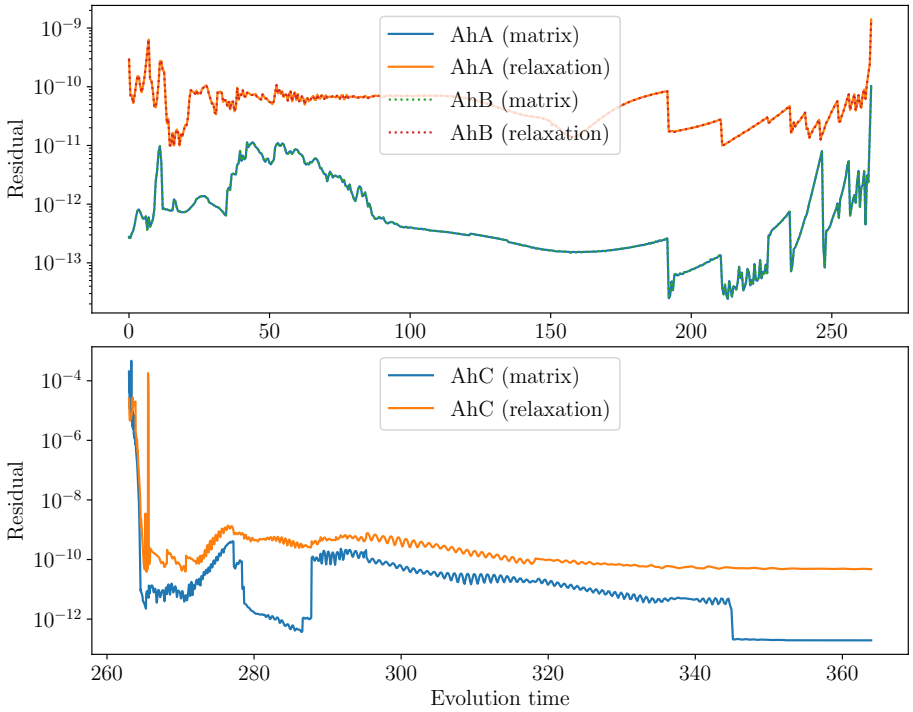


Figure 5.3: Residuals during the `Aa` run.

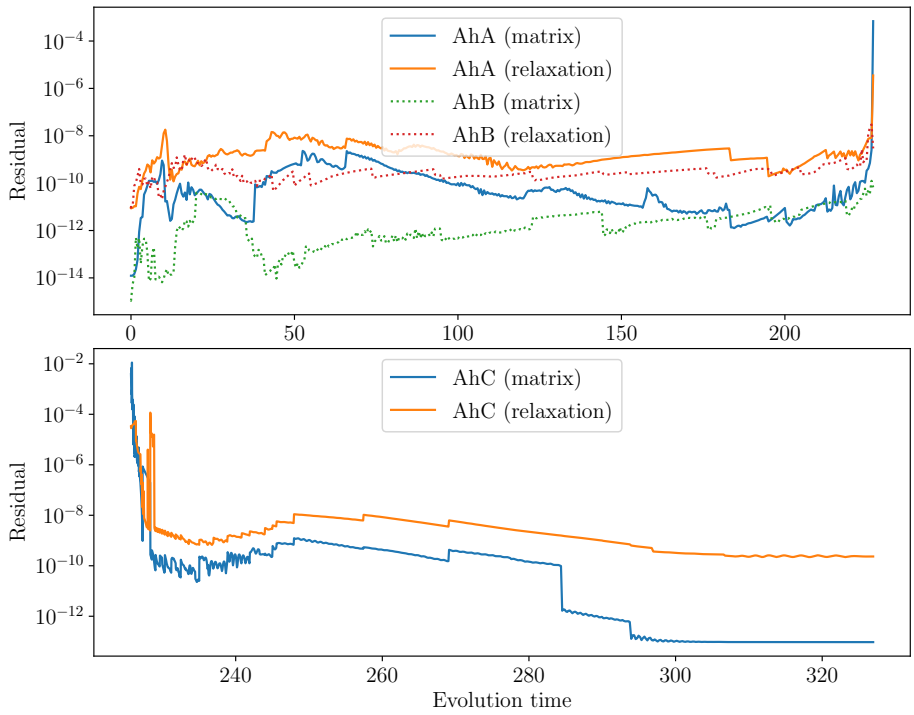


Figure 5.4: Residuals during the `q4` run.

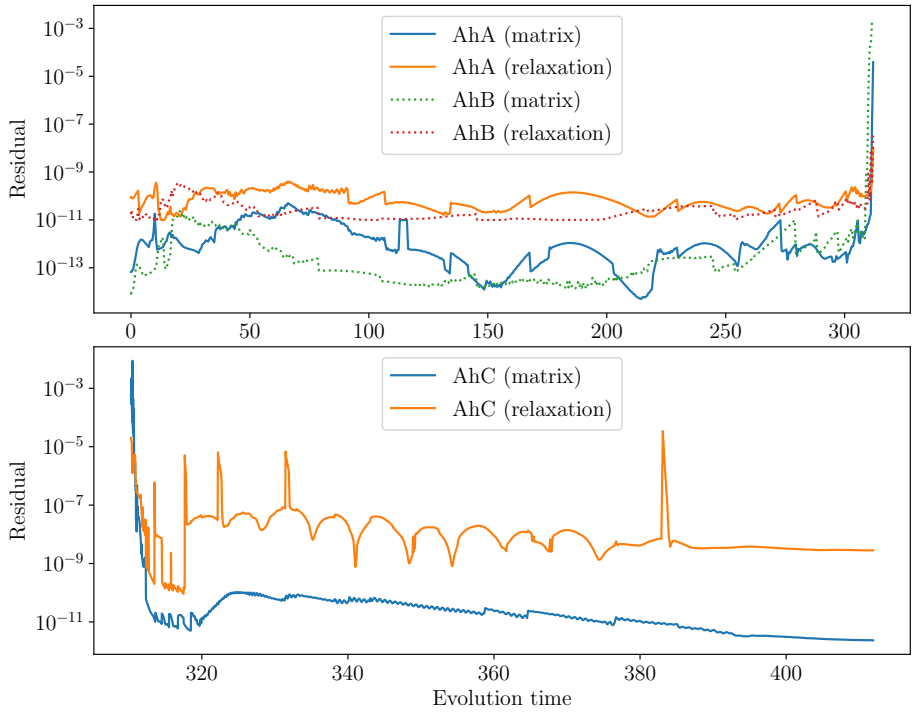


Figure 5.5: Residuals during the `Ge` run.

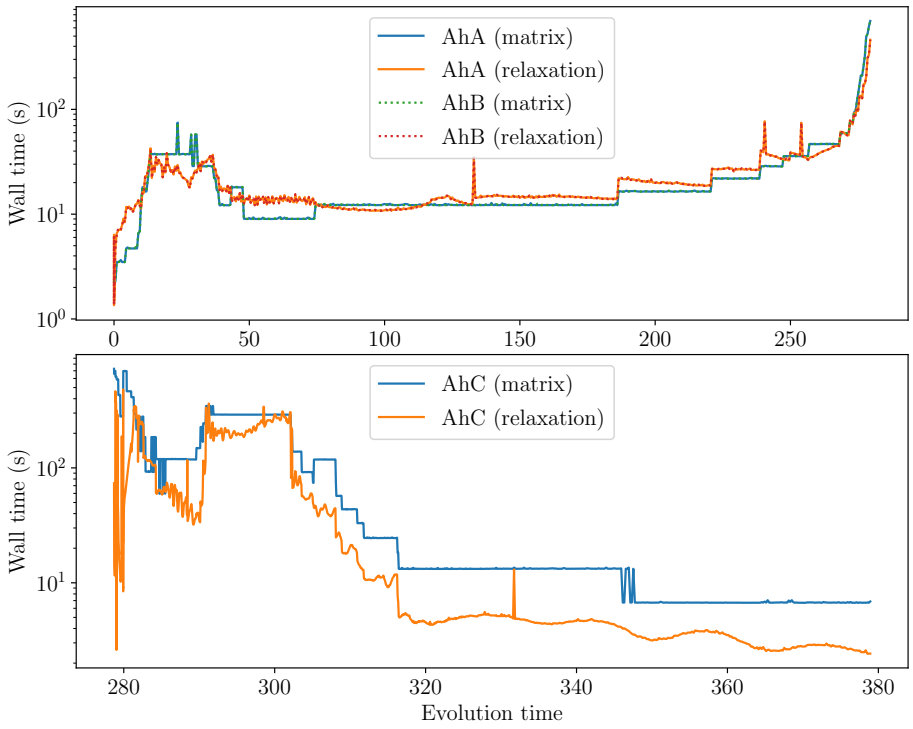


Figure 5.6: Wall times during the `EN` run.

Similarly, we can plot the wall times for each method during the evolution. This is done in Figures 5.6–5.10, which follow a similar layout as before. From these plots, we note that the relaxation method takes roughly the same time to run as the matrix method for the pre-merger phase, but the relaxation approach is generally faster during the post-merger phase. This advantage is a result of using the previous embedding as an initial guess for the relaxation method, as mentioned in section 4.2. Additionally, from subsection 4.3.2, we know that the relaxation method tends to become relatively faster for higher resolutions.

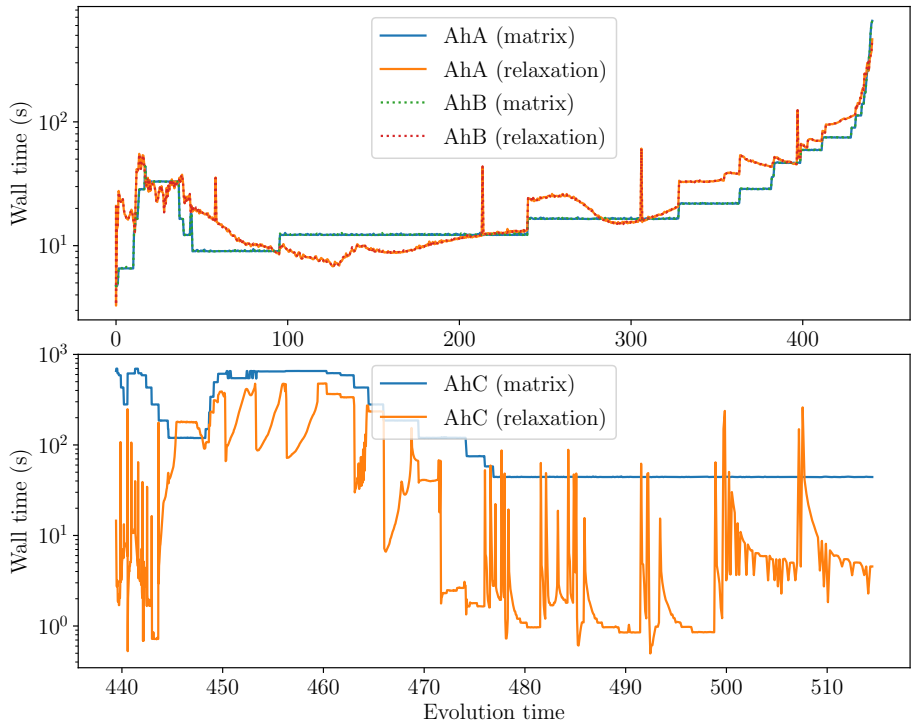


Figure 5.7: Wall times during the `Ca` run.

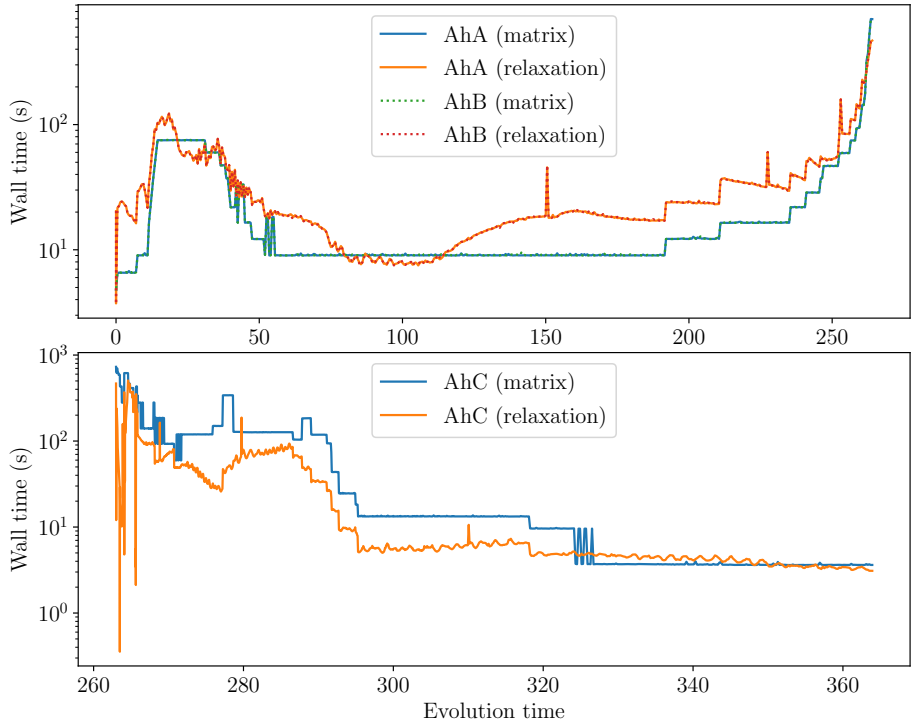


Figure 5.8: Wall times during the `Aa` run.

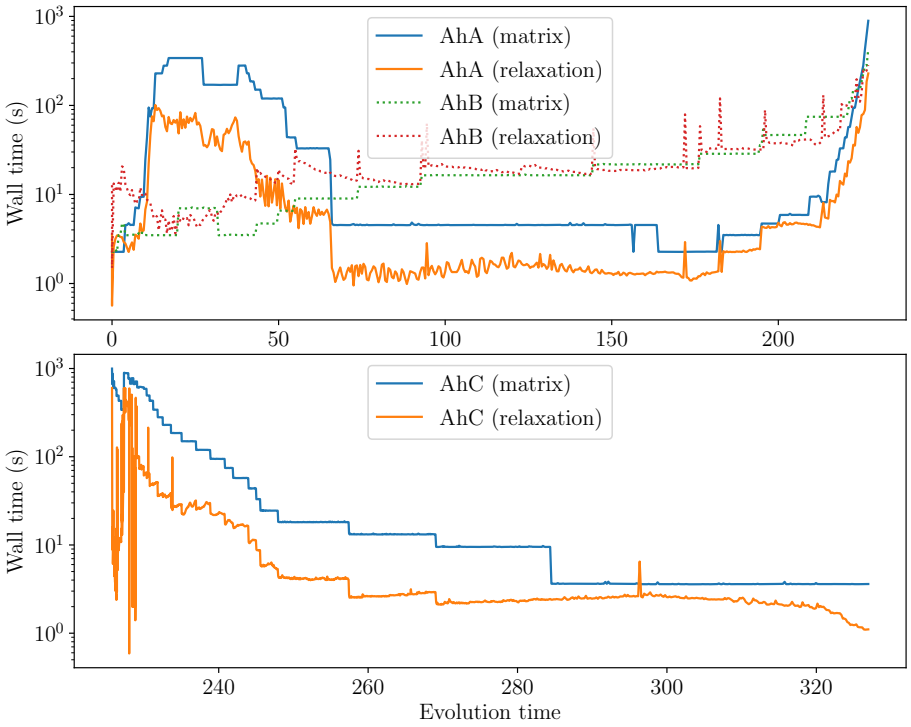


Figure 5.9: Wall times during the `q4` run.

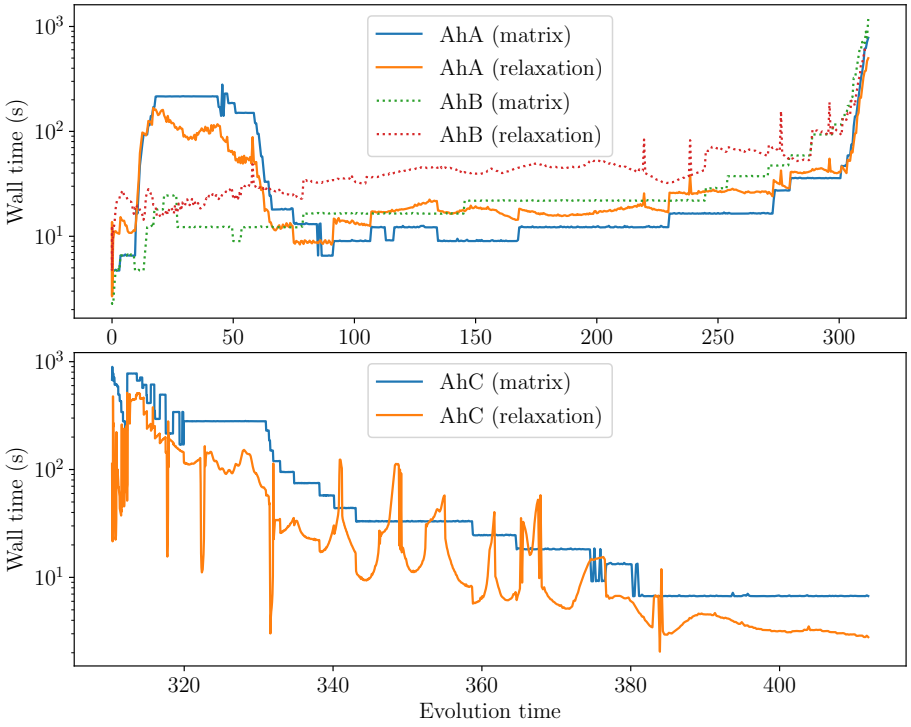
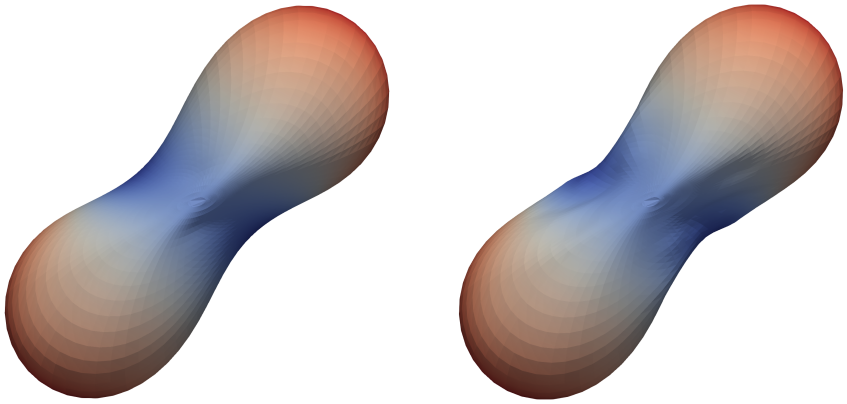
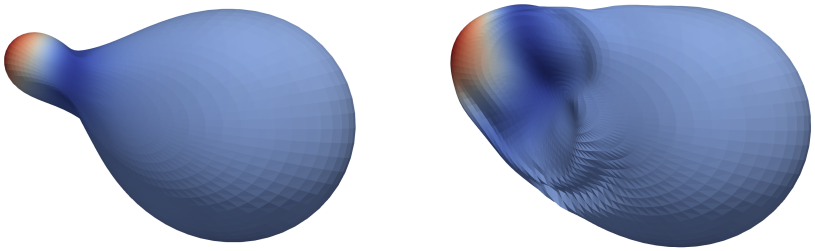


Figure 5.10: Wall times during the `Ge` run.



Relaxation Matrix

Figure 5.11: Failed embedding results for the `Aa` run.



Relaxation Matrix

Figure 5.12: Failed embedding results for the `q4` run.

5.2 Non-embeddability?

As suggested by the residual plots in the previous section, there are moments during the evolutions of BBH simulations in which our embedding methods fail. This is reflected not only in high residuals, but also in unexpected deformations in the resulting surfaces. Figures 5.11 and 5.12 show examples of such deformed surfaces. We believe that these moments can indicate non-embeddable surfaces, but there has never been a description of how numerical horizons become embeddable or non-embeddable. Tackling this problem is one of our next goals.

Chapter 6

Conclusion

6.1 Discussion

In summary, the isometric embedding problem consists in describing a surface in 3D Euclidean space that has the same metric as the desired geometry, requiring us to solve a system of nonlinear partial differential equations (PDEs). We have developed two novel numerical methods for solving this problem in the Spectral Einstein Code (`SpEC`). The matrix method uses spectral methods to transform the embedding PDEs into an algebraic problem. The relaxation method applies continuous rotations to coordinate basis vectors with a constraint damping scheme.

With several test cases, we confirmed that our methods work and studied their unique behaviors. We applied these methods to binary black hole merger simulations in `SpEC`. From the embedding results, we have a better representation of the intrinsic geometry of the apparent horizons compared to the coordinate shapes. During these simulations, we have found that the matrix method usually leads to better residuals, while the relaxation method can lead to faster results. Finally, we have explored our methods at the breakdown of embeddability for Kerr horizons and seen hints of non-embeddability in numerical horizons.

6.2 Future Work

From the conclusion of this project, we have many follow-up projects to tackle. I describe below the most relevant ones.

6.2.1 Critical Embedding

As discussed before, not all surfaces can be embedded into flat space. We have explored a famous example of this (the Kerr horizon), but it is not currently well understood how

numerical horizons in generic simulations transition from an embeddable surface to a non-embeddable one. We have found cases in which both of our methods fail to embed, which we believe to indicate a non-embeddability.

The study of this "critical embedding" would considerably benefit from our work on the embedding problem. For this project, we would run several black hole simulations analyzing which conditions could be associated with the embeddability of apparent horizons and how this transition occurs. One of our hypothesis is that the breakdown of embeddability behaves in an analogous way to different processes that are currently well understood. Possible analogies include the relativistic collapse during black hole formation (in which the energy density is the determining factor) and phase transitions in Statistical Mechanics.

6.2.2 Wang-Yau Quasilocal Mass

In 2009, Mu-Tao Wang and Shing-Tung Yau proposed a new notion of the quasilocal mass for horizons [24]. This definition relies on embedding the apparent horizon into Minkowski space (a 4-dimensional analogue of the 3D Euclidean space). In 2023, Yau and collaborators studied this quasilocal mass on head-on simulations of binary non-spinning black holes [25]. Due to limitations on their embedding method, they were not able to study more complex simulations.

With the embedding methods that we have developed, `SpEC` is now capable of finding the isometric embedding for many generic horizons. The way it is currently implemented, we are only able to embed horizons into Euclidean space, but it should be possible to expand our approach to include the time embedding function so that we can embed horizons into Minkowski space.

6.2.3 Inspecting Gauge Conditions

In numerical relativity, spatial coordinates evolve over time via a gauge condition. In SpEC, this condition is chosen to provide good numerical stability, but does not guarantee a minimization of nonphysical distortions. As we saw in Table 5.2, these distortions can occur especially near merger, when the simulation runs slower due to a need for higher resolution.

A reasonable hypothesis is that the merger phase would run more efficiently if the spatial coordinates were less distorted. The argument is that, with fewer distortions, we could lower the resolution in order to speed up the merger phase. To accomplish this, we would need to choose gauge conditions that minimize such distortions, which would only be possible if we have a way to measure distorted coordinates.

Now that we have developed a method for finding horizon embeddings, we have a possible representation of distortions. That is, the difference between the isometric embedding and

the coordinate shape for a given horizon could potentially be used as a distortion measure. From Table 5.2, it is clear that there are cases in which the intrinsic geometry is more round than the coordinate shape indicates, possibly suggesting that the gauge conditions could be optimized.

Bibliography

[1] John Wheeler and Kenneth Ford. "Geons, Black Holes and Quantum Foam: A Life in Physics". In: *American Journal of Physics* 68 (June 2000). DOI: 10.1119/1.19497.

[2] B. P. Abbott et al. "Observation of Gravitational Waves from a Binary Black Hole Merger". In: *Phys. Rev. Lett.* 116 (6 Feb. 2016), p. 061102. DOI: 10.1103/PhysRevLett.116.061102.

[3] James B. Hartle. *Gravity: An Introduction to Einstein's General Relativity*. Cambridge University Press, 2021. DOI: 10.1017/9781009042604.

[4] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation*. W. H. Freeman, 1973. ISBN: 978-0-7167-0344-0, 978-0-691-17779-3. URL: http://adsabs.harvard.edu/abs/1973grav.book.....M.

[5] M. Spivak. *A Comprehensive Introduction to Differential Geometry*. A Comprehensive Introduction to Differential Geometry v. 5. Publish or Perish, Incorporated, 1975. ISBN: 9780914098041. URL: https://books.google.com/books?id=3skPAQAAMAAJ.

[6] H. Weyl. "Uber die Bestimmung einer geschlossenen konvexen Flache durch ihr Linienelement". In: *Vierteljahrsschrift der naturforschenden Gesellschaft in Zurich* 61 (1916), pp. 40–72. URL: https://www.biodiversitylibrary.org/page/32612443.

[7] John Nash. "The Imbedding Problem for Riemannian Manifolds". In: *Annals of Mathematics* 63.1 (1956), pp. 20–63. ISSN: 0003486X. DOI: 10.2307/1969989.

[8] Mihai Bondarescu, Miguel Alcubierre, and Edward Seidel. "Isometric embeddings of black-hole horizons in three-dimensional flat space". In: *Classical and Quantum Gravity* 19.2 (Jan. 2002), p. 375. DOI: 10.1088/0264-9381/19/2/311.

[9] Valerio Faraoni. *Cosmological and Black Hole Apparent Horizons*. Vol. 907. 2015. ISBN: 978-3-319-19239-0, 978-3-319-19240-6. DOI: 10.1007/978-3-319-19240-6.

[10] S. W. Hawking and G. F. R. Ellis. *The Large Scale Structure of Space-Time*. Cambridge Monographs on Mathematical Physics. Cambridge University Press, 1973. DOI: 10.1017/CBO9780511524646.

[11] Thomas W. Baumgarte and Stuart L. Shapiro. "Numerical relativity and compact binaries". In: *Physics Reports* 376.2 (2003), pp. 41–131. ISSN: 0370-1573. DOI: 10.1016/S0370-1573(02)00537-9.

[12] Andy Bohn, Lawrence E. Kidder, and Saul A. Teukolsky. "Parallel adaptive event horizon finder for numerical relativity". In: *Phys. Rev. D* 94 (6 Sept. 2016), p. 064008. DOI: 10.1103/PhysRevD.94.064008.

[13] Roger Penrose. "Gravitational Collapse and Space-Time Singularities". In: *Phys. Rev. Lett.* 14 (3 Jan. 1965), pp. 57–59. DOI: 10.1103/PhysRevLett.14.57.

[14] Ivan Booth. "Black hole boundaries". In: *Canadian Journal of Physics* 83.11 (Nov. 2005), pp. 1073–1099. ISSN: 1208-6045. DOI: 10.1139/p05-063.

[15] Wolfgang Tichy, Jonathan R McDonald, and Warner A Miller. "New efficient algorithm for the isometric embedding of 2-surface metrics in three dimensional Euclidean space". In: *Classical and Quantum Gravity* 32.1 (Dec. 2014), p. 015002. DOI: 10.1088/0264-9381/32/1/015002.

[16] Robert Owen. "Constraint damping in first-order evolution systems for numerical relativity". In: *Phys. Rev. D* 76 (4 Aug. 2007), p. 044019. DOI: 10.1103/PhysRevD.76.044019.

[17] Iago Mendes, Hengrui Zhu, and Robert Owen. "Isometric Embedding of Numerical Horizons". In preparation. 2024.

[18] John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Books on Mathematics. Dover Publications, 2001. ISBN: 9780486411835. URL: https://store.doverpublications.com/products/9780486411835.

[19] Bengt Fornberg. "Generation of Finite Difference Formulas on Arbitrarily Spaced Grids". In: *Mathematics of Computation* 51.184 (1988), pp. 699–706. DOI: 10.2307/2008770.

[20] Cameron R. Taylor. *Finite Difference Coefficients Calculator*. 2016. URL: https://web.media.mit.edu/~crtaylor/calculator.html.

[21] Lawrence Kidder, Harald Pfeiffer, Mark Scheel, et al. *Spectral Einstein Code (SpEC)*. URL: https://black-holes.org/code/SpEC.

[22] Lawrence E. Kidder et al. "Black hole evolution by spectral methods". In: *Phys. Rev. D* 62 (8 Sept. 2000), p. 084032. DOI: 10.1103/PhysRevD.62.084032.

[23] John C. Adams and Paul N. Swarztrauber. "SPHEREPACK 3.0: A Model Development Facility". In: *Monthly Weather Review* 127.8 (1999), pp. 1872–1878. DOI: 10.1175/1520-0493(1999)127<1872:SAMDF>2.0.CO;2.

[24] Mu-Tao Wang and Shing-Tung Yau. "Quasilocal Mass in General Relativity". In: *Phys. Rev. Lett.* 102 (2 Jan. 2009), p. 021101. DOI: 10.1103/PhysRevLett.102.021101. URL: https://link.aps.org/doi/10.1103/PhysRevLett.102.021101.

[25] Daniel Pook-Kolb et al. *Properties of Quasi-local mass in binary black hole mergers.* https://arxiv.org/abs/2308.10906. 2023. arXiv: 2308.10906 [gr-qc].