UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

ESTUDIO DE AGUJEROS NEGROS BINARIOS VIA EL EINSTEIN TOOLKIT

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

ENZO GIOVANNI IUBINI CORTEZ

PROFESOR GUÍA:
Claudio Muñoz Cerón

MIEMBROS DE LA COMISIÓN:
Tomas Andrade Weber
Jaime Ortega Palma
Paola Rioseco Yañez

Este trabajo ha sido parcialmente financiado por:
CMM ANID BASAL FB210005, Fondecyt 1231250, Fondecyt Exploración 13220060.

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR AL GRADO
DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN MATEMÁTICAS APLICADAS Y
MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO
POR: ENZO GIOVANNI IUBINI CORTEZ
FECHA: 2023
PROF. GUÍA: CLAUDIO MUÑOZ CERÓN

ESTUDIO DE AGUJEROS NEGROS BINARIOS VIA EL EINSTEIN TOOLKIT

En el contexto de la invención de la Relatividad General por Albert Einstein en 1915, la reciente detección de ondas gravitacionales por el Observatorio de Ondas Gravitacionales por Interferometría Láser (LIGO) ha proporcionado una ventana hacia la dinámica de sistemas binarios de agujeros negros. Al utilizar técnicas computacionales avanzadas, específicamente el software Einstein Toolkit, estudiamos la generación, propagación y detección de ondas gravitacionales. En el Capítulo 2, ofrecemos una introducción sencilla a los esquemas numéricos como soluciones a ecuaciones en derivadas parciales, en particular, el esquema de diferencias finitas. En los Capítulos 3 y 4, exploramos las ecuaciones de calor y onda, examinando su comportamiento cuando se resuelven mediante técnicas numéricas. El Capítulo 5 ofrece una visión rápida de los temas más importantes sobre la Relatividad Especial y la Relatividad General, utilizando estos para resolver la ecuación de onda en un espacio-tiempo definido como la solución de las ecuaciones de Einstein en presencia de un agujero negro, a saber, el espacio-tiempo de Schwarzschild. En los últimos capítulos, exploramos el ámbito de las técnicas numéricas para resolver las ecuaciones de Einstein. En el Capítulo 7, presentamos la Relatividad Numérica como una forma de resolver las ecuaciones de campo computacionalmente. Concluimos esta tesis estudiando el software Einstein Toolkit en el Capítulo 8, y aplicando este software en el Capítulo 9 para simular la famosa fusión de agujeros negros binarios GW150914, estudiando las trayectorias de los agujeros negros y las ondas gravitacionales emitidas.

RESUMEN DE LA TESIS PARA OPTAR AL GRADO
DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN MATEMÁTICAS APLICADAS Y
MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO
POR: ENZO GIOVANNI IUBINI CORTEZ
FECHA: 2023
PROF. GUÍA: CLAUDIO MUÑOZ CERÓN

ESTUDIO DE AGUJEROS NEGROS BINARIOS VIA EL EINSTEIN TOOLKIT

In the wake of the invention of General Relativity by Albert Einstein in 1915, recent detection of gravitational waves by the Laser Interferometer Gravitational-Wave Observatory center offered a window into the dynamics of binary black hole systems. By employing advanced computational techniques, namely the Einstein Toolkit software, we study the generation, propagation, and detection of gravitational waves. In Chapter 2, we offer a simple introduction on numerical schemes as solutions to partial differential equations, in particular, the finite differences scheme. In Chapters 3 and 4, we explore the heat and wave equations, examining their behavior when solved using numerical techniques. Chapter 5 offers a quick overview on the most important topics on Special Relativity and General Relativity, using these to solve the wave equation on a spacetime defined as the solution of the Einstein's equations in the presence of a black hole, namely, the Schwarszchild spacetime. In the last chapters, we explore the realm of numerical techniques to solve Einstein's equation. In Chapter 7, we introduce Numerical Relativity as a way to solve the field equations computationally. We end this thesis by studying the Einstein Toolkit software in Chapter 8, and applying this software in Chapter 9 to simulate the famous binary black hole merger GW150914, studying the trajectories of the black holes and the gravitational waves emitted.

To my family, my friends...
to my dogs and to me.

Thank you all

Agradecimientos

I must thank my proffessor, Claudio Muñoz, for giving me the opportunity to research one of the most fascinating fields I have ever encountered in my mathematical carreer.

I must thank my co-proffessor, Tomas Andrade, for guiding me through the world of computing and solving all my questions about black hole simulations, even allowing me to learn how to use a supercomputer cluster.

I must thank the National Laboratory of High Performance Computing in Chile, for giving me the opportunity to work in their multiple clusters, and allowing me to finish my thesis.

I must thank my family, who always wanted me close even when I could not leave my lonely appartment. They, who called every single day to check up on me. They, who always encourage me to follow my dreams, whatever they be and wherever they would take me. And they, who trusted in me and sent me to this big city to live on my own and pursue happiness, wherever I find it.

I must thank my friends, who were always there when I needed them; when I needed to vent, when I needed to be loved and heard, when I needed advice, and when I needed a reason to not give up.

Finally, I must thank my dogs for always being in my mind. Giulia and Duna, giving me the peace and love I needed in my darkest times.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introduction

Since Albert Einstein introduced Special Relativity in 1905, it has revolutionized our understanding of various physics phenomena by the introduction of its two famous postulates; the invariancy of laws of physics on all inertial frames and the uniqueness of the speed of light in vacuum for all observers. This quickly posed new questions regarding gravity: Did it work instantaneously on all bodies, as Newton had predicted? How did it affect massless particles? Soon enough, Einstein introduced the new theory of General Relativity along with his famous Einstein's Field Equations. These equations explained how the presence of matter and energy affected the curvature of spacetime, giving rise to a new perspective on gravitation. These equations predicted the existence of black holes, and the gravitational waves that binary black hole systems emitted. All this was mere theory and inference, until the Laser Interferometer Gravitational-Wave Observatory (LIGO), a key player in gravitational wave detection, made the first direct observation of gravitational waves on 14 September 2015, a signal appropriately named GW150914. In the concluding chapters, we will analyze specific simulations of black hole binary mergings and draw insightful conclusions from our findings.

Throughout this paper, we will study numerical schemes and the relativistic physics to understand the gravitational wave sygnal from the merger of a black hole binary. In particular, we use powerful computing tools to reproduce and understand the GW150914 data obtained by the two LIGO observatories. This accounts to understanding where the idea of black holes came from, how their evolution is governed by the Einstein Field Equations, and how these are solved using approximate methods for gravitational-wave calculation; the weak field approach and numerical relativity.

The Einstein Field Equations are a complex system of non-linear coupled wave equations, for which we need numerical methods to solve. In Chapter 2, we give a brief description of a fundamental Partial Differential Equation (PDE) numerical solving scheme, the Finite Difference Scheme. We describe its formulation, how to apply it to a generic PDE, and how to study its stability. These methods become particularly useful when we encounter PDEs with no apparent analytical solution.

In Chapter 3 and Chapter 4, we apply finite difference schemes to two famous PDEs, the heat equation and the wave equation. We focus on the different parameters these schemes have, studying conditions for which our numerical methods will solve these PDEs as accurately as we want. We will finish these chapters showing Python generated plots for each

relevant scheme, thus giving solid proof of the usefulness of these numerical approximations, later to be used to solve much harder equations.

Chapter 5 provides a quick overview of Special Relativity and General Relativity to establish the theoretical foundation for understanding black hole binaries. We begin by explaining the postulates and results of Special Relativity, followed by a focus on the new physics language used to describe flat and curved spacetime—tensors and covariant derivatives. The most important result of this chapter is the derivation of the Einstein Field Equations, which can be further solved assuming special symmetries along 3-dimensional axes to get the Schwarszchild metric, a solution describing black holes. This knowledge will be useful when solving a project involving a scalar pulse defined by the wave equation in our Schwarszchild spacetime in Chapter 7. We not only derive a first order system to solve the wave equation on this curved spacetime, but explain how to use our finite differences numerical schemes to study its solutions, along with new Python codes to detect masses of bodies on their way to be engorged by our black hole.

Our spacetime equations combine all space and time indices to get a large number of second-order partial differential equations for the *metric*, while our numerical schemes involve a time derivative defining the change of our space at different time intervals. This problem can be fixed by defining spacelike hypersurfaces and a timelike (or null) vector piercing them describing the *flow of time* along these surfaces. The ambiguity on choosing different families of hypersurfaces and different time directions is what we explain how to solve on Chapter 6 by the use of Numerical Relativity. This branch of general relativity focuses on numerical methods and algorithms to solve astrophysical problems, which are later applied on supercomputers to study black holes, gravitational waves, and neutron stars. To this end, we finish this chapter making the link between numerical relativity and Einstein's Field Equations; this means defining a family of hypersurfaces and projecting our Equations on each of them, and evolving our equations on time steps described by a time vector.

To solve these equations and study the gravitational wave emitted by black hole binaries, we make use of a powerful computational tool; the Einstein Toolkit. This community-driven software platform employs computational tools to solve equation systems involving astrophysics and gravitational physics. In Chapter 8, we provide a full description of this software, along with a brief tutorial on solving a simple example on the platform's website: the Multipatch Wave Equation.

Using all the tools on our disposal, we finally have the means to study gravitational waves emitted by a black hole binary. To solve our complex system of equations, we were supported by the National Laboratory of High Performance Computing (NLHPC) in Chile, who allowed us to work on their supercomputers and get the data necessary to completely simulate black hole binaries. We end our thesis by running specific codes to simulate two black hole binary mergings; a quasi-circular collision with equal mass black holes, and a simulation of the GW150914 signal. We further analyze the plots of the black hole trajectories and the gravitational waves emitted, the latter by the decomposition of the Ψ_4 scalar in spherical harmonics.

Capítulo 2

Finite Difference schemes for PDEs

2.1. Introduction

Partial Differential Equations (PDEs) play a fundamental role in describing numerous physical phenomena and natural processes, ranging from heat conduction and fluid dynamics to electromagnetic fields, quantum mechanics, and space-time evolution models of our universe. Analytically solving these equations and obtaining a closed-form solution is often challenging, if not impossible, so many numerical methods have emerged as tools for approximating solutions to these PDEs.

Among these various numerical techniques, ***finite difference schemes*** stand out as a widely used and highly versatile approach to solving PDEs numerically. This method employs the straightforward concept of discretizing the continuous domain of the PDE into a finite set of grid points, thereby transforming the original differential equation into a set of mixed algebraic equations that can be solved computationally.

The aim of this chapter is to explore finite difference schemes for PDEs comprehensively. We will present the underlying principles of finite difference approximations, analyze various schemes used for different PDEs, and discuss their strengths and limitations. We will focus on the discretization of both temporal and spatial domains, showing how these schemes can handle parabolic, elliptic, and hyperbolic PDEs.

2.2. Discretizing our domain

Solving a partial differential equation often involves the differential of a function on various of its independent variables. Let's focus on a function u depending on time and space, that is, $u \equiv u(x,t)$, where $x \in [0,1]$ and $0 \leq t \leq T$. In a specific PDE, when its partial derivative is called, the equations require us to compute the derivative for each point in our domain, which is often an open interval with infinite points. Asking our machine to solve this, with no previous knowledge of differential equations and just a simple definition of a limit will prove impossible due to the number of points. Therefore, our first intuition is to partition our domain into a **finite** set of points. This means defining $\{x_0, x_1, ..., x_{n-1}, x_n\}$ such that,

$$0 = x_0 < x_1 < ... < x_{n-1} < x_n = 1.$$

If we define the **mesh** of our partition as $\max |x_i - x_{i-1}| : i \in \{1, ..., n\}$, then we will show that as this mesh tends to zero, our approximation converges to the real solution of the PDE. This concept of mesh represents the maximum distance between adjacent points in our partition.

In theory, we could define a grid with a varying mesh size (smaller where higher numerical accuracy is needed and larger elsewhere). For simplicity, we will use a uniform partition, where the intervals all have the same length h:

$$x_i = x_0 + i \cdot h = x_0 + i \cdot \frac{x_n - x_0}{N} \quad \text{for} \quad i = 1, ..., N,$$

where N is the number of intervals we divide our domain into.

Therefore, a common discretization of our domain $(0,1) \times \mathbb{R}^+$ arises from introducing our spatial step $\Delta x = 1/N$ and a temporal step $\Delta t > 0$ as nodes of a regular grid:

$$(t_n, x_j) = (n\Delta t, j\Delta x) \quad \text{for} \quad n \geq 0 \text{ and } j \in \{0, 1, ..., N\}.$$

If $u \equiv u(t,x)$ solves a particular PDE, we will denote as u_j^n the discrete approximated solution at the points (t_n, x_j), and by $u(t,x)$ as the exact solution.

2.3. Back and forward derivatives

Let u_0 be a numerical approximation of some function u. We define the **order** of this approximation as the highest integer n such that $\lim_{h \to 0} \frac{u_0 - u}{h^n} = 0$, and we will say the approximation is $\mathcal{O}(h^n)$.

Let u be a n-times differentiable function. If we apply a Taylor series expansion around a point x_i, we get:

$$u(x_i + h) = u(x_i) + \frac{u'(x_i)}{1!}h + \frac{u^{(2)}(x_i)}{2!}h^2 + ... + \frac{u^{(n)}(x_i)}{n!}h^n + R_n(h),$$

where $R_n(h)$ is a remainder term such that $\lim_{h \to 0} \frac{R_n(h)}{h^n} = 0$.

Let us derive an approximation for the first derivative of u by truncating the Taylor polynomial.

$$u(x_i + h) = u(x_i) + u'(x_i)h + R_1(h).$$

Dividing across by h yields:

$$u'(x_i) = \frac{u(x_i + h) - u(x_i) - R_1(h)}{h} \approx \frac{u(x_i + h) - u(x_i)}{h}.$$

This is known as the ***first order approximation forward first derivative (FD1u')*** of u. It is certainly useful when u is defined in a neighborhood to the right of x_i, but when it is not, one can make use of a ***first order approximation backward first derivative (BD1u')***:

$$u'(x_i) = \frac{u(x_i) - u(x_i - h) - R_1(x)}{h} \approx \frac{u(x_i) - u(x_i - h)}{h}.$$

These both approximations are of order $\mathcal{O}(h)$, since the difference between the approximations and the real derivative (as seen in the function's Taylor series expansion) $R_1(h)$ has the property:

$$\lim_{h \to 0} \frac{R_1(h)}{h^1} = 0.$$

2.4. Creating new approximations

Let us derive approximations for u'. Let $u \in C^2$ with bounded second derivative, $0 \leq h \leq h_0$, and consider the expansion given by the Taylor's theorem:

$$u(x \pm h) = u(x) \pm hu'(x) + \frac{h^2}{2}u''(\xi_\pm).$$

Then one has

$$(\text{FD1}u') \left\{ \left| \tfrac{u(x+h)-u(x)}{h} - u'(x) \right| \leq \mathcal{C}h \quad , \quad \mathcal{C} = \max_{\xi \in [x, x+h]} \tfrac{|u''(\xi)|}{2}. \right.$$

Additionally,

$$(\text{BD1}u') \left\{ \left| \tfrac{u(x)-u(x-h)}{h} - u'(x) \right| \leq \mathcal{C}h \quad , \quad \mathcal{C} = \max_{\xi \in [x, x+h]} \tfrac{|u''(\xi)|}{2}. \right.$$

Now, if $u \in C^3$, we can get higher order terms from our Taylor expansion,

$$\begin{cases} u(x + h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \frac{h^3}{3!}u'''(\xi_+), \\ u(x - h) = u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{3!}u'''(\xi_-), \end{cases}$$

to derive a ***second order approximation centered first derivative (CD2u')*** of u:

$$(\text{CD2}u') : \quad \left| \frac{u(x + h) - u(x - h)}{2h} - u'(x) \right| \leq \mathcal{C}h^2 \quad , \quad \mathcal{C} = \max_{\xi \in [x, x+h]} \frac{|u'''(\xi)|}{3!}.$$

Similarly, for $u \in C^4$, we further expand our Taylor approximation,

$$\begin{cases} u(x + h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \frac{h^3}{3!}u'''(x) + \frac{h^4}{4!}u^{iv}(\xi_+), \\ u(x - h) = u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{3!}u'''(x) + \frac{h^4}{4!}u^{iv}(\xi_-), \end{cases}$$

to derive a ***second order approximation centered second derivative (CD2u″)***

$$(\text{CD2}u'')\ \left\{\left|\tfrac{u(x+h)-2f(x)+u(x-h)}{h^2}-u''(x)\right|\le \mathcal{C}h^2 \quad , \quad \mathcal{C}=\max_{\xi\in[x,x+h]}\tfrac{|u^{iv}(\xi)|}{4!}.\right.$$

Let us try to generalize this with an example. Suppose we need to find an approximation of u' using five points: $u(x-2h), u(x-h), u(x), u(x+h), u(x+2h)$. This amounts to finding the values γ_k for which:

$$u'(x)\sim \gamma_{-2}u(x-2h)+\gamma_{-1}u(x-h)+\gamma_0 u(x)+\gamma_1 u(x+h)+\gamma_2 u(x+2h).$$

where we define this approximation as

$$S=\sum_{k=-m}^{m}\gamma_k u(x+kh).$$

Using, again, our Taylor series expansion, we get:

$$\begin{cases}\gamma_0 u(x)=\gamma_0 u(x),\\ \gamma_{\pm1}u(x\pm h)=\gamma_{\pm1}u(x)\pm\gamma_{\pm1}hu'(x)+\gamma_{\pm1}\frac{h^2}{2}u''(x)\pm\gamma_{\pm1}\frac{h^3}{3!}u'''(x)\\ \qquad\qquad +\gamma_{\pm1}\frac{h^4}{4!}u^{iv}(x)+\mathcal{O}(h^4),\\ \gamma_{\pm2}u(x\pm 2h)=\gamma_{\pm2}u(x)\pm\gamma_{\pm2}2hu'(x)+\gamma_{\pm2}\frac{(2h)^2}{2}u''(x)\pm\gamma_{\pm2}\frac{(2h)^3}{3!}u'''(x)\\ \qquad\qquad +\gamma_{\pm2}\frac{(2h)^4}{4!}u^{iv}(x)+\mathcal{O}(h^4).\end{cases}$$

Therefore, since we are looking for an approximation of $u'(x)$, then the following must hold:

$$\begin{aligned}u'(x)&=\underbrace{\sum_{k=-2}^{2}\gamma_k}_{=0}u(x)\pm\underbrace{\sum_{k=-2}^{2}kh\gamma_k}_{=1}u'(x)+\underbrace{\sum_{k=-2}^{2}\frac{(kh)^2}{2}\gamma_k}_{=0}u''(x)\\ &=\pm\underbrace{\sum_{k=-2}^{2}\frac{(kh)^3}{3!}\gamma_k}_{=0}u'''(x)+\underbrace{\sum_{k=-2}^{2}\frac{(kh)^4}{4!}\gamma_k}_{=0}u^{iv}(x)+\mathcal{O}(h^4).\end{aligned}$$

This is a linear system of equations, whose solution satisfies $|S-f'(x)|\le\mathcal{C}h^4$ and can be expressed in matrix notation:

$$\begin{pmatrix}1 & 1 & 1 & 1 & 1\\ -2h & -h & 0 & h & 2h\\ \frac{(-2h)^2}{2!} & \frac{(-h)^2}{2!} & 0 & \frac{h^2}{2!} & \frac{(2h)^2}{2!}\\ \frac{(-2h)^3}{3!} & \frac{(-h)^3}{3!} & 0 & \frac{h^3}{3!} & \frac{(2h)^3}{3!}\\ \frac{(-2h)^4}{4!} & \frac{(-h)^4}{4!} & 0 & \frac{h^4}{4!} & \frac{(2h)^4}{4!}\end{pmatrix}\begin{pmatrix}\gamma_{-2}\\ \gamma_{-1}\\ \gamma_0\\ \gamma_1\\ \gamma_2\end{pmatrix}=\begin{pmatrix}0\\ 1\\ 0\\ 0\\ 0\end{pmatrix}.$$

Solving this system yields our final solution for the desired approximation:

$$\gamma_{-2} = \frac{1}{12h}\ ,\ \gamma_{-1} = -\frac{8}{12h}\ ,\ \gamma_0 = 0\ ,\ \gamma_1 = \frac{8}{12h}\ ,\ \gamma_2 = -\frac{1}{12h};$$
$$\implies u'(x) \sim \frac{u(x-2h) - 8u(x-h) + 8u(x+h) - u(x+2h)}{12h}.$$

2.5. Derivatives in time and space

The same logic applied on derivatives for multivalued functions is used for finite difference approximation schemes.

This means, considering forward approximations for derivatives in time and space, the equations read:

$$\frac{\partial}{\partial t}u(t,x) \approx \frac{u(t+\Delta t, x) - u(t,x)}{\Delta t},$$
$$\frac{\partial}{\partial x}u(t,x) \approx \frac{u(t, x+\Delta x) - u(t,x)}{\Delta x}.$$

Similarly, for the centered second derivatives, the equations are:

$$\frac{\partial^2}{\partial t^2}u(t,x) \approx \frac{u(t+\Delta t, x) - 2u(t,x) + u(t-\Delta t, x)}{\Delta t^2},$$
$$\frac{\partial^2}{\partial x^2}u(t,x) \approx \frac{u(t, x+\Delta x) - 2u(t,x) + u(t, x-\Delta x)}{\Delta x^2}.$$

Here each approximation is made with respect to each variable, while keeping others constant.

2.6. Consistency and accuracy

A finite difference scheme is defined, for all possible indices n, j, by the following formula:

$$F_{\Delta t, \Delta x}\left(\{u_{j+k}^{n+m}\}_{m^- \le m \le m^+, k^- \le k \le k^+}\right) = 0,$$

where integers m^-, m^+, k^-, k^+ define the length of our stencil.

Our finite difference scheme is said to be consistent with our PDE $F(u) = 0$, if for every solution $u(t,x)$ sufficiently regular, the **truncation error** of the scheme, defined by

$$F_{\Delta t, \Delta x}\left(\{u(t+m\Delta t, x+k\Delta x)\}_{m^- \le m \le m^+, k^- \le k \le k^+}\right),$$

tends to zero, uniformly with respect to (t,x), when Δt and Δx tend to zero independently.

Also, the scheme is said to be accurate to order p in space and order q in time if the truncation error tends to zero with $\mathcal{O}\left((\Delta x)^p + (\Delta t)^q\right)$ as Δt and Δx tend to zero.

2.7. Stability

Usually, not every finite difference scheme works in a desired manner. Some schemes may not actually represent our equation, and thus they will not give back a consistent solution, for which we refer to the consistency and accuracy of the scheme. But sometimes our scheme is actually consistent, but our solution begins oscillating in an unbounded manner, for which we say our scheme is ***unstable***. We give a mathematical definition for stability, beginning with the definition of a norm. Let us define the following norm for a numerical solution $u^n = (u_j^n)_{1 \leq j \leq N}$.

$$\|u^n\|_p = \left(\sum_{j=1}^{N} \Delta x |u_j^n|^p \right)^{1/p} \quad \text{for} \quad 1 \leq p \leq \infty.$$

A finite difference scheme is said to be **stable** if there exists a constant $K > 0$, that does not depend on Δt or Δx, such that

$$\|u^n\| \leq K \|u^0\| \quad \text{for all} \quad n \geq 0,$$

for every initial condition u^0.

If this condition is not met for every step size Δt and Δx, but is blinded to particular values of these step sizes, then the scheme is said to be **conditionally stable**.

We will be working with linear schemes, for which our PDE representing formula,

$$F_{\Delta t, \Delta x} \left(\{u_{j+k}^{n+m}\}_{m^- \leq m \leq m^+, k^- \leq k \leq k^+} \right) = 0,$$

is linear with respect to its arguments u_{j+k}^{n+m}. Stability for a linear scheme is easy to interpret, since we can write these schemes as a matrix equation,

$$u^{n+1} = Au^n,$$

where A is a linear operator of $\mathbb{R}^N$. By iterating backwards, one gets that $u^n = A^n u^0$, so stability of our scheme is equivalent to:

$$\|A^n u^0\| \leq K \|u^0\| \qquad \forall n \geq 0, \forall u^0 \in \mathbb{R}^N.$$

If we define our linear operator norm,

$$\|M\| = \sup_{u \in \mathbb{R}^N, u \neq 0} \frac{\|Mu\|}{\|u\|},$$

then the stability of our scheme is equivalent to,

$$\|A^n\| \leq K \qquad \forall n \geq 0,$$

which means that we ask that all powers of our matrix be bounded by some constant.

Capítulo 3

Numerical methods for the Heat Equation

3.1. Introduction

We will begin by presenting an elementary discussion on an important partial differential equation: the 1-D Heat equation. The Heat equation describes the distribution of heat in a one-dimensional rod over time and is mathematically expressed as,

$$\frac{\partial u}{\partial t}(x,t) = \kappa \frac{\partial^2 u}{\partial x^2}(x,t), \quad \text{for } x \in [0, L].$$

Here, L represents the length of the rod, and κ is a parameter reflecting the material properties affecting heat conduction.

As is customary with PDEs, solutions to this equation require initial conditions (the heat distribution at the starting time) and boundary conditions (constraints on the heat at the boundaries, either fixed values or constraints on the heat flow).

In this chapter, our focus will be on numerical schemes that allow us to compute approximated solutions, demonstrating their convergence to analytical solutions. The 1-D heat equation is often the starting point for studying finite difference schemes due to its straightforward formulation, and its stability is relatively easy to analyze.

3.2. Derivation

We begin by deriving the equation for the temperature $u(x,t)$ of a rod of length L, with a diffusion coefficient κ and an external heat source $H(x,t)$, at the point x and time t. We need to make some assumptions to correctly pose this problem:

- The rod is made of homogeneous material.
- The rod is insulated perpendicular to it, so heat flows only in the x-direction.
- The rod is sufficiently thin, so the temperature within any cross-section is constant.

The first assumption, concerning the homogeneity of the material, is not necessary for the original problem, but it simplifies our solution techniques.

From the principle of conservation of energy, the heat within a segment of the rod $[x, x+\partial x]$ satisfies the following:

$$\text{Net change inside } [x, x+\partial x] = \text{Net inward flux across boundaries at } x \text{ and } x+\partial x + \text{total heat generated inside } [x, x+\partial x].$$

The total amount of heat in any segment $[a, b]$ is given by:

$$\int_a^b c\rho A u(s,t)\, ds,$$

where c is the thermal capacity of the rod (specific heat), ρ is the density of the rod, and A is the cross-sectional area of the rod—constants under our assumptions. The conservation relation reads:

$$\underbrace{c\rho A \int_x^{x+\partial x} u_t(s,t)\, ds}_{\text{Net change}} = \underbrace{KA[u_x(x+\partial x, t) - u_x(x,t)]}_{\text{Net inward flux across boundaries}} + \underbrace{A \int_x^{x+\partial x} H(s,t)\, ds}_{\text{Total heat generated inside}},$$

where $H(x,t)$ describes the heat generated by an external source per unit length and per unit of time.

Dividing both sides by A and applying the FTC to show that $u_x(b,t) - u_x(a,t) = \int_a^b u_{xx}(s,t)\, ds$, we get:

$$c\rho \int_x^{x+\partial x} u_t(s,t)\, ds = \int_x^{x+\partial x} \left(K u_{xx}(s,t) + H(s,t)\right)\, ds.$$

Rearranging yields:

$$\int_x^{x+\partial x} u_t(s,t)\, ds = \int_x^{x+\partial x} \left(\kappa u_{xx}(s,t) + h(s,t)\right)\, ds,$$

where $\kappa = \frac{K}{c\rho}$ and $h(x,t) = \frac{1}{c\rho} H(x,t)$ are the diffusivity of the rod and the heat source density, respectively.

Since this equation holds on any arbitrary segment of the rod, it follows that the integrands must be the same everywhere on the rod. Therefore, we finally obtain the 1-D heat equation:

$$u_t = \kappa u_{xx} + h(x,t).$$

3.3. Analytical solution of the heat equation

We again consider a conductive rod of length L as seen in 3.1. To model this object, we assume this rod is insulated along $(0, L)$ and exchanging heat from its boundary point $x = 0$ and/or $x = L$. We also model an external heat source as $h(x)$, this time not depending on t.

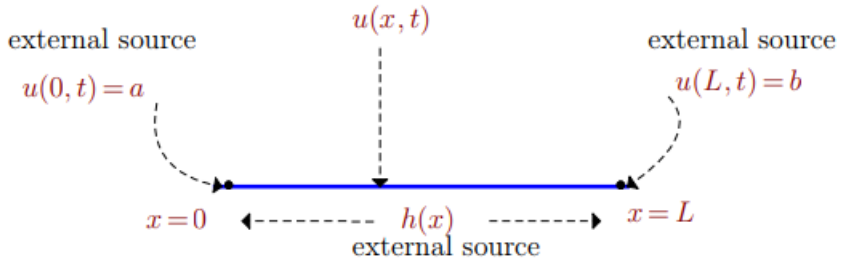

Figura 3.1: Visual representation of a conductive rod, for which we find an equation to solve for $u(x, t)$.

Our goal is to find the equation that defines $u(x, t)$ being the temperature of the rod at the point x and time t. Note that u is a function of two independent variables x, t, where $x \in [0, L]$ and $t \geq 0$. The heat equation that describes u is:

$$u_t(x, t) = \kappa u_{xx}(x, t) + h(x),$$

where $\kappa > 0$ is a positive constant that depends on the heat capacity, the density and other properties of the rod.

For this problem to have a unique solution, we will need one initial condition and two boundary conditions. The initial condition is usually described by an arbitrary function of x.

Boundary conditions can vary according to the physical problem at hand, and these can be of the following types:

1. Dirichlet boundary conditions (D)
2. Neumann boundary conditions (N)
3. Mixed boundary conditions (M)

The following are three examples, one of each boundary condition:

$$\text{(D)} \begin{cases} u_t(x,t) = \kappa u_{xx}(x,t) + h(x) \\ u(x,0) = f(x) \\ u(0,t) = a,\ u(L,t) = b, \end{cases}$$

$$\text{(N)} \begin{cases} u_t(x,t) = \kappa u_{xx}(x,t) + h(x) \\ u(x,0) = f(x) \\ u_x(0,t) = c_1,\ u_x(L,t) = c_2, \end{cases}$$

$$\text{(M)} \begin{cases} u_t(x,t) = \kappa u_{xx}(x,t) + h(x) \\ u(x,0) = f(x) \\ a_1 u(0,t) + b_1 u_x(0,t) = c_1 \\ a_2 u(L,t) + b_2 u_x(L,t) = c_2. \end{cases}$$

We derive an analytical solution for the heat equation with Dirichlet boundary conditions.

To solve this equation, we need to understand the concept of the ***steady state solution*** to a heat equation. This is a solution that remains unchanged with respect to t, that is, a solution $v(x)$ that satisfies the heat equation. This is equivalent to asking that

$$\begin{cases} 0 = \kappa v''(x) + h(x) \\ v(0) = a,\ v(L) = b. \end{cases}$$

The above equation is simply solved by integration, and we can prove that, regardless of the initial heat distribution $u(x,0) = f(x)$,

$$\lim_{t\to\infty} u(x,t) = v(x).$$

As seen in any basic ODE course, to find the particular solution to the original intial-boundary value problem, we need to sum two solutions: the homogeneous and the particular solution.

First, let $w(x,t)$ satisfy the homogeneous problem:

$$\begin{cases} w_t(x,t) = \kappa w_{xx}(x,t) \\ w(x,0) = g(x) \\ w(0,t) = 0,\ w(L,t) = 0. \end{cases}$$

Next, we need a steady state solution satisfying the original problem. Let $v(x)$ be this steady state solution satisfying,

$$\begin{cases} 0 = \kappa v''(x) + h(x) \\ v(0) = a,\ v(L) = b. \end{cases}$$

Now we consider $u(x,t) = v(x) + w(x,t)$. We first note that,

$$\begin{aligned} u_t - \kappa u_{xx} &= (v+w)_t - \kappa(v+w)_{xx} \\ &= \underbrace{v_t}_{=0} + w_t - \underbrace{\kappa v_{xx}}_{=-h(x)} - \kappa w_{xx} \\ &= \underbrace{w_t - \kappa w_{xx}}_{=0} + h(x) \\ &= h(x), \end{aligned}$$

so $u(x,t)$ satisfies the nonhomogeneous heat equation. The boundary conditions are also satisfied:

$$\begin{cases} u(0,t) = v(0) + w(0,t) = a \\ u(L,t) = v(L) + w(L,t) = b. \end{cases}$$

Since the initial condition equation implies

$$f(x) = u(x,0) = v(x) + w(x,0) = v(x) + g(x),$$

then, if we set $g(x) = f(x) - v(x)$, we can first solve our ODE for $v(x)$ to get $g(x)$ and solve our homogeneous PDE for $w(x,t)$.

3.3.1. Homogeneous heat equation

Since our non-homogeneous heat equation can be split into a simple ODE and an homogeneous heat equation, we focus our attention on the latter. Let $w(x,t)$ be a solution for the homogeneous heat equation with Dirichlet boundary conditions, that is,

$$\begin{cases} w_t(x,t) = \kappa w_{xx}(x,t) \\ w(x,0) = g(x) \\ w(0,t) = 0,\ w(L,t) = 0. \end{cases}$$

We will use the separation of variables technique to solve this. We first assume a solution takes the form of a product of a purely spatial function and a purely temporal function. This usually leaves us with a free parameter solution, which determines an unique solution when setting the initial condition. Let $w(x,t)$ be our solution, that is,

$$w(x,t) = X(x)T(t).$$

We can easily compute partial derivatives on this solution to get,

$$w_t(x,t) = X(x)T'(t) \qquad \text{and} \qquad w_{xx}(x,t) = X''(x)T(t),$$

so our equation reads as

$$\begin{cases} X(x)T'(t) = \kappa X''(x)T(t) \\ X(x)T(0) = g(x) \\ X(0)T(t) = 0,\ X(L)T(t) = 0. \end{cases}$$

Diving both sides of our heat equation by $\kappa X(x)T(t)$ we get,

$$\frac{1}{\kappa}\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)}.$$

Since both sides depend on different independent variables, we impose they must be equal to a constant $-\lambda^2$, where the negative sign is there just to make our calculations easier, and doesn't change the final solution. This gives us two equations:

$$\begin{cases} T'(t) = -\kappa\lambda^2 T(t) \\ X''(x) = -\lambda^2 X(x),\ X(0) = 0,\ X(L) = 0. \end{cases}$$

We must now consider three cases for $-\lambda^2$ to be a real constant. If a case leads to a zero solution, we will discard it as trivial.

Case 1: λ purely imaginary

In this case, we consider $\lambda = i\lambda'$, with λ' a real constant. Our equation for $X(x)$ would yield a solution,

$$X(x) = c_1 \cosh(\lambda' x) + c_2 \sinh(\lambda' x).$$

Applying both boundary conditions yield,

$$0 = X(x) = c_1, \qquad 0 = X(L) = c_2 \sinh(\lambda' x)$$

which lead to $c_1 = c_2 = 0$, so then $X(x) \equiv 0$. This leads to a trivial solution, so we discard this case.

Case 2: $\lambda = 0$

In this case, our equation for $X(x)$ reduces to,

$$X''(x) = 0.$$

This is a second order ODE, easily solved by integration, and yields,

$$X(x) = c_1 x + c_2.$$

Applying both boundary conditions force c_1 and c_2 to be zero, yielding another trivial solution, so this case also does not work.

Case 3: λ purely real

For Case 3, the equation for $X(x)$ produces the following solution,

$$X(x) = c_1 \cos(\lambda x) + c_2 \sin(\lambda x).$$

Applying both boundary conditions yield,

$$0 = X(x) = c_1, \qquad 0 = X(L) = c_2 \sin(\lambda x)$$

Since we are looking for non trivial solutions, we must have that $c_2 \neq 0$, so then,

$$\sin(L\lambda) = 0 \quad \Longrightarrow \quad L\lambda = n\pi,\ n = 1, 2, 3, ..$$

which is an equation of λ for each possible value of n. Therefore, we can define $\lambda_n = \frac{n\pi}{L}$ and get our equation for $X_n(x)$

$$\lambda_n = \frac{n\pi}{L} \quad X_n(x) = \sin\left(\frac{n\pi x}{L}\right),\ n = 1, 2, 3, ...$$

Plugging this λ_n to our equation for $T(t)$ for each value of n, yields a family of solutions $T_n(t)$,

$$T_n(t) = c_n e^{-\kappa\lambda^2 t},$$

Due to the antisymmetry of the sine function, it suffices to only consider positive values of n, therefore our infinite solutions will be of the form:

$$u_n(x,t) = B_n \sin\left(\frac{n\pi x}{L}\right) e^{-\kappa\left(\frac{n\pi}{L}\right)^2 t} \quad n = 1, 2, 3, ...$$

Our full solution will be a particular sum of these eigenfunctions, where the specific choice of the B_n values will depend of our initial solution $g(x)$, in particular:

$$u(x,t) = \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi x}{L}\right) e^{-\kappa\left(\frac{n\pi}{L}\right)^2 t}.$$

Where B_n are to be determined. To get an unique solution, that is, determine the values of B_n, we set $t = 0$ and multiply both sides by a sine function of the n-th frequency. When integrating, that sets every other sine product equal to zero and leaves us with an equation easily integrable for B_n, that is,

$$B_n = \frac{2}{L}\int_0^L g(x) \sin\left(\frac{n\pi x}{L}\right) dx,$$

which are values that determine an unique solution for our homogeneous heat equation.

3.4. Finite Differences for the Heat Equation

For some initial conditions, the homogeneous heat equation is actually pretty easy to solve, and may have few non-zero constant terms, leaving us with a neat solution. However, it may happen that all B_n are non-zero, where the infinite sum determining the final solution has no closed form, and this is only considering the homogeneous heat equation where there's no extra terms, but physics admits a great variety of equations that may arise, and we should be prepared to solve, or at least visualize, each one of them. This is where numerical schemes con in handy, allowing us to visualize solutions to every equation, even if it admits no closed form.

To delve further, we must start from the beginning, so let us use numerical schemes to solve the homogeneous heat equation. Let $u(x,t)$ solve this PDE with an initial condition $u_0(x)$, that is,

$$\begin{cases} \frac{\partial u}{\partial t}(x,t) - \kappa \frac{\partial^2 u}{\partial x^2}(x,t) = 0 & \text{for } (x,t) \in (0,1) \times \mathbb{R}_*^+ \\ u(x,0) = u_0(x) & \text{for } x \in (0,1). \end{cases}$$

As mentioned before, our domain grid will be discretized by,

$$(x_j, t_n) = (j\Delta x, n\Delta t) \qquad \text{for } n \geq 0, \ , j \in \{0, 1, ..., N\}.$$

where $\Delta x = 1/N$, with N being a positive integer, and $\Delta t > 0$.

Remembering our previous notation, we denote u_j^n as the approximation of $u(x,t)$ at the points (x_j, t_n), with our initial approximation vector being $u_j^0 = u_0(x_j)$. Let us figure our how to get these values and study them.

The finite difference schemes are not unique to an equation, this may have been obvious by the fact that derivatives can be approximated to different orders with different schemes, and those expressions would undoubtedly lead up to different schemes. It is important to know the order to which our approximations will be computed, but when considering spatial derivatives, it is ever more important to choose at *which* time these derivatives will be computed.

Whenever we are studying a first derivative in time, we usually do a forward derivative $\partial_t u = \frac{u_j^{n+1} - u_j^n}{\Delta t}$, and so we are implicitly saying that we are 'standing' at the n-th time. So one may ask, are we computing the a spatial derivative at the n-th time, at the $(n-1)$-th time, a combination of both, or any other possible time? The answer is that one may choose any possible time step, and study the scheme accordingly. We will talk about the most used schemes as follows.

Whenever we are at the n-th time, and compute our spatial derivatives in this time step, we call our equations an ***explicit scheme***. For the homogeneous heat equation, it becomes,

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} - \kappa \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2} = 0,$$

where the second spatial derivative is taken with respect to the current position $n\Delta t$, that

is, given our first stage by u_0, the next one can be computed with the formula:

$$\begin{aligned} u_j^{n+1} &= u_j^n + cu_{j-1}^n - 2cu_j^n + cu_{j+1}^n \\ &= cu_{j-1}^n + (1-2c)u_j^n + cu_{j+1}^n, \end{aligned}$$

where $c = \frac{\kappa \Delta t}{\Delta x^2}$ is called our ***CFL number***.

This means, beginning with our initial condition we can compute the next time step by multiplying our vector by a matrix given by:

$$\begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} 1-2c & c & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ c & 1-2c & c & \ddots & & & & \vdots \\ 0 & c & 1-2c & c & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & c & 1-2c & c & 0 \\ \vdots & & & & \ddots & c & 1-2c & c \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & c & 1-2c \end{pmatrix} \begin{pmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_{N-2}^n \\ u_{N-1}^n \\ u_N^n \end{pmatrix}.$$

On the other hand, the ***implicit scheme***, takes values at the next time step of the discretization and forces them to solve the equations of the previous one. The scheme reads as follows:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} - \kappa \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{\Delta x^2} = 0,$$

and it solves the equation

$$\begin{aligned} & u_j^{n+1} - cu_{j-1}^{n+1} + 2cu_j^{n+1} - cu_{j+1}^{n+1} = u_j^n \\ \implies & -cu_{j-1}^{n+1} + (1+2c)u_j^{n+1} - cu_{j+1}^{n+1} = u_j^n. \end{aligned}$$

We can check this scheme is well posed, and one can indeed compute values of u_j^{n+1} given the values of u_j^n. In fact, the linear system can be written as

$$\begin{pmatrix} 1+2c & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -c & 1+2c & -c & \ddots & & & & \vdots \\ 0 & -c & 1+2c & -c & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & -c & 1+2c & -c & 0 \\ \vdots & & & & \ddots & -c & 1+2c & -c \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & -c & 1+2c \end{pmatrix} \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_{N-2}^n \\ u_{N-1}^n \\ u_N^n \end{pmatrix},$$

where this matrix is definite positive, therefore invertible.

These two schemes can be joined together by a convex combination of the two, where for $0 \leq \theta \leq 1$, we can obtain the ***θ-scheme***,

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \theta\kappa \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{\Delta x^2} + (1-\theta)\kappa \frac{u_{j-1}^{n} - 2u_j^{n} + u_{j+1}^{n}}{\Delta x^2}.$$

We will note some important things about this scheme:

- For $\theta = 0$, the scheme is explicit.
- For $\theta = 1$, the scheme is implicit.
- For $\theta = 1/2$, the scheme will be called ***Crank-Nicholson scheme***.
- For other values of θ, the scheme can be given other names, such as the ***six points scheme***, given by $\theta = 1/2 - \frac{\Delta x^2}{12\kappa\Delta t}$.

These previous schemes will be called ***two-level*** schemes, since they only consider at most two consecutive time indices. It is possible to work with multi-level schemes, for example, the ***DuFort-Frankel scheme***:

$$\frac{u_j^{n+1} - u_j^{n-1}}{2\Delta t} + \kappa \frac{-u_{j-1}^{n} + u_j^{n+1} + u_j^{n-1} - u_{j+1}^{n+1}}{\Delta x^2} = 0,$$

and the ***Gear scheme***:

$$\frac{3u_j^{n+1} - 4u_j^{n} + u_j^{n-1}}{2\Delta t} + \kappa \frac{-u_{j-1}^{n+1} + 2u_j^{n+1} - u_{j+1}^{n+1}}{\Delta x^2} = 0.$$

The list is exhaustive and our job is to compare these schemes and use the one that's most efficient and accurate according to our needs.

From now on, we will be focusing on the ***implicit two level scheme*** and the ***explicit two level scheme***.

As we have mentioned in previous subsections, the initial condition is totally represented in the values of u_j^0 that start our iterations. The conditions for our boundaries are, on the other hand, given by the values of our iteration matrix. Each scheme can be used to solve the equation given the requested boundary conditions, which we will study separatedly.

3.4.1. Dirichlet Boundary Conditions

Let us consider our usual heat equations with Dirichlet conditions on both sides, that is

$$u(t,0) = 0 \quad \text{and} \quad u(t,1) = 0.$$

It is important to note that since values at these boundaries must be preserved, then the initial condition must also have these values. This means that whichever value our initial condition has at our spatial domain border, then we can replace for the Dirichlet condition in the corresponding border. It also means that the iteration matrix must be such that the boundary value is preserved at every step. This can be easily done by replacing the corresponding row of the matrices by an identity matrix row.

For the explicit scheme, the system of equations one must solve to solve for Dirichlet boundary conditions are,

$$\begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ c & 1-2c & c & \ddots & & & & \vdots \\ 0 & c & 1-2c & c & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & c & 1-2c & c & 0 \\ \vdots & & & & \ddots & c & 1-2c & c \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_{N-2}^n \\ u_{N-1}^n \\ u_N^n \end{pmatrix},$$

while the implicit schemes solves a different system,

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -c & 1+2c & -c & \ddots & & & & \vdots \\ 0 & -c & 1+2c & -c & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & -c & 1+2c & -c & 0 \\ \vdots & & & & \ddots & -c & 1+2c & -c \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_{N-2}^n \\ u_{N-1}^n \\ u_N^n \end{pmatrix}.$$

These two schemes can be combined into a single $\boldsymbol{\theta}$-scheme, if we consider the following matrices for the Dirichlet conditions on a θ-scheme,

$$A_{dd}(c,\theta) = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -c\theta & 1+2c\theta & -c\theta & \ddots & & & & \vdots \\ 0 & -c\theta & 1+2c\theta & -c\theta & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & -c\theta & 1+2c\theta & -c\theta & 0 \\ \vdots & & & & \ddots & -c\theta & 1+2c\theta & -c\theta \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix},$$

$$B_{dd}(c,\theta) = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ c\theta' & 1-2c\theta' & c\theta' & \ddots & & & & \vdots \\ 0 & c\theta' & 1-2c\theta' & c\theta' & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & c\theta' & 1-c2\theta' & c\theta' & 0 \\ \vdots & & & & \ddots & c\theta' & 1-2c\theta' & c\theta' \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix},$$

where we have used the notation $\theta' = 1 - \theta$. By combining the two schemes as mentioned before, we get the θ-scheme in matrix form as,

$$A_{dd}(c,\theta)\mathcal{U}^{n+1} = B_{dd}(c,\theta)\mathcal{U}^n,$$

where $\mathcal{U}^n = (u_0^n, u_1^n, ..., u_{N-1}^n, u_N^n)^T$ denotes the approximated solution at time n.

3.4.2. Neumann Boundary conditions

Now, we consider the heat equation with Neumann boundary conditions at both of its borders, this means:

$$\frac{\partial u}{\partial x}(t,0) = c_1 \quad \text{and} \quad \frac{\partial u}{\partial x}(t,1) = c_2.$$

For a first order approximation, we could easily use forward spatial derivatives at both points u_0^n and u_N^n:

$$\frac{u_1^n - u_0^n}{\Delta x} = c_1 \quad \text{and} \quad \frac{u_N^n - u_{N-1}^n}{\Delta x} = c_2.$$

This allows us to solve for our respective values of u_0^n and u_N^n. Sadly, this approximation is only of first order wheras our scheme has a second order at the center points, and will make us lose accuracy around the borders. To fix this, we can apply a second order approximation at the border to get:

$$\frac{u_1^n - u_{-1}^n}{2\Delta x} = c_1 \quad \text{and} \quad \frac{u_{N+1}^n - u_{N-1}^n}{2\Delta x} = c_2.$$

This better approximation gives us accuracy, but adds cost to our code by introducing two "ghost" points u_{-1}^n and u_{N+1}^n. The equation for these ghost points are, respectively:

$$u_{-1}^n = u_1^n - 2c_1\Delta x \quad \text{and} \quad u_{N+1}^n = u_{N-1}^n + 2c_2\Delta x.$$

These equations are used to compute the centered derivative for the boundary points. To show this, we will consider the explicit scheme equation for the left boundary point:

$$\begin{aligned}
&\frac{u_0^{n+1} - u_0^n}{\Delta t} = \kappa \frac{u_1^n - 2u_0^n + u_{-1}^n}{\Delta x^2} \\
&\implies u_0^{n+1} = u_0^n + cu_1^n - 2cu_0^n + cu_{-1}^n \\
&\implies u_0^{n+1} = (1-2c)u_0^n + cu_1^n + c\left(u_1^n - 2c_1\Delta x\right) \\
&\implies u_0^{n+1} = (1-2c)u_0^n + 2cu_1^n - 2cc_1\Delta x.
\end{aligned}$$

Similarly, the equation for the right boundary points would read,

$$u_N^{n+1} = (1-2c)u_N^n + 2cu_{N-1}^n + 2cc_2\Delta x.$$

This translates to the iteration matrix for the **explicit** scheme:

$$\begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} 1-2c & 2c & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ c & 1-2c & c & \ddots & & & & \vdots \\ 0 & c & 1-2c & c & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & c & 1-2c & c & 0 \\ \vdots & & & & \ddots & c & 1-2c & c \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 2c & 1-2c \end{pmatrix} \begin{pmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_{N-2}^n \\ u_{N-1}^n \\ u_N^n \end{pmatrix} + \begin{pmatrix} -2cc_1\Delta x \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \\ 2cc_2\Delta x \end{pmatrix}.$$

The same methods can be applied to the original **implicit** scheme, for which we will show the final matrix form of the equation:

$$\begin{pmatrix} 1+2c & -2c & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -c & 1+2c & -c & \ddots & & & & \vdots \\ 0 & -c & 1+2c & -c & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & -c & 1+2c & -c & 0 \\ \vdots & & & & \ddots & -c & 1+2c & -c \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & -2c & 1+2c \end{pmatrix} \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ \vdots \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} + \begin{pmatrix} 2cc_1\Delta x \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \\ -2cc_2\Delta x \end{pmatrix} = \begin{pmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_{N-2}^n \\ u_{N-1}^n \\ u_N^n \end{pmatrix}.$$

To combine these two into a single ***θ*-scheme**, we will start from the beginning and make our way up to it. We begin posing the original θ-scheme for the left boundary point and expand it accordingly

$$\frac{u_0^{n+1}-u_0^n}{\Delta t}=\theta\kappa\frac{u_1^{n+1}-2u_0^{n+1}+u_{-1}^{n+1}}{\Delta x^2}+\theta'\kappa\frac{u_1^n-2u_0^n+u_{-1}^n}{\Delta x^2}$$
$$\implies (1+2c\theta)u_0^{n+1}-c\theta' u_1^{n+1}-c\theta u_{-1}^{n+1}=(1-2c\theta')u_0^n+c\theta' u_1^n+c\theta' u_{-1}^n$$
$$\implies (1+2c\theta)u_0^{n+1}-c\theta u_1^{n+1}-c\theta(u_1^{n+1}-2c_1\Delta x)=(1-2c\theta')u_0^n+c\theta' u_1^n$$
$$+c\theta'(u_1^n-2c_1\Delta x)$$
$$\implies (1+2c\theta)u_0^{n+1}-2c\theta u_1^{n+1}+2cc_1\theta\Delta x=(1-2c\theta')u_0^n+2c\theta' u_1^n-2cc_1\theta'\Delta x$$
$$\implies (1+2c\theta)u_0^{n+1}-2c\theta u_1^{n+1}=(1-2c\theta')u_0^n+2c\theta' u_1^n-2cc_1(\theta+\theta')\Delta x$$
$$\implies (1+2c\theta)u_0^{n+1}-2c\theta u_1^{n+1}=(1-2c\theta')u_0^n+2c\theta' u_1^n-2cc_1\Delta x.$$

Similarly, for our right boundary, we get:

$$\frac{u_N^{n+1}-u_N^n}{\Delta t}=\theta\kappa\frac{u_{N+1}^{n+1}-2u_N^{n+1}+u_{N-1}^{n+1}}{\Delta x^2}+\theta'\kappa\frac{u_{N+1}^n-2u_N^n+u_{N-1}^n}{\Delta x^2}$$
$$\implies (1+2c\theta)u_N^{n+1}-c\theta' u_{N-1}^{n+1}-c\theta u_{N+1}^{n+1}=(1-2c\theta')u_N^n+c\theta' u_{N-1}^n+c\theta' u_{N+1}^n$$
$$\implies (1+2c\theta)u_N^{n+1}-c\theta u_{N-1}^{n+1}-c\theta(u_{N-1}^{n+1}+2c_2\Delta x)=(1-2c\theta')u_N^n+c\theta' u_{N+1}^n$$
$$+c\theta'(u_{N-1}^n+2c_2\Delta x)$$
$$\implies (1+2c\theta)u_N^{n+1}-2c\theta u_{N-1}^{n+1}-2cc_1\theta\Delta x=(1-2c\theta')u_N^n+2c\theta' u_{N-1}^n+2cc_2\theta'\Delta x$$
$$\implies (1+2c\theta)u_N^{n+1}-2c\theta u_{N-1}^{n+1}=(1-2c\theta')u_N^n+2c\theta' u_{N-1}^n+2cc_1(\theta+\theta')\Delta x$$
$$\implies (1+2c\theta)u_N^{n+1}-2c\theta u_{N-1}^{n+1}=(1-2c\theta')u_N^n+2c\theta' u_{N-1}^n+2cc_1\Delta x.$$

Therefore, our θ-scheme written in matrix notation can be represented by the following two matrices that represent this scheme with Neumann boundary conditions,

$$A_{nn}(c,\theta) = \begin{pmatrix} 1+2c\theta & -2c\theta & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -c\theta & 1+2c\theta & -c\theta & \ddots & & & & \vdots \\ 0 & -c\theta & 1+2c\theta & -c\theta & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & -c\theta & 1+2c\theta & -c\theta & 0 \\ \vdots & & & & \ddots & -c\theta & 1+2c\theta & -c\theta \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & -2c\theta & 1+2c\theta \end{pmatrix}$$

$$B_{nn}(c,\theta) = \begin{pmatrix} 1-2c\theta' & 2c\theta' & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ c\theta' & 1-2c\theta' & c\theta' & \ddots & & & & \vdots \\ 0 & c\theta' & 1-2c\theta' & c\theta' & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & c\theta' & 1-2c\theta' & c\theta' & 0 \\ \vdots & & & & \ddots & c\theta' & 1-2c\theta' & c\theta' \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 2c\theta' & 1-2c\theta' \end{pmatrix}$$

and so our θ-scheme is condensed in the following equation,

$$A_{nn}(c,\theta)\mathcal{U}^{n+1} = B_{nn}(c,\theta)\mathcal{U}^n + C_{nn},$$

where again $\mathcal{U}^n = (u_0^n, u_1^n, ..., u_N^n)$ and $C_{nn} = (-2cc_1\Delta x, 0, ..., 0, 2cc_2\Delta x)$ is a constant vector that deals with the conditions at the boundary.

3.5. Stability analysis

We have previously defined the condition for a linear scheme to be stable. Luckily, the scheme for the heat equation is in fact linear, so we can apply them. Remember the condition is there exist a constant $K > 0$, independent on Δt and Δx, such that,

$$\|u^n\| \leq K\|u^0\| \quad \forall n \geq 0, \forall u^0 \in \mathbb{R}^N.$$

This condition was equivalent to asking that the following condition holds,

$$\|A^n u^0\| \leq K\|u^0\| \quad \forall n \geq 0, \forall u^0 \in \mathbb{R}^N,$$

which reduces to a matrix norm inequality,

$$\|A^n\| \leq K \quad \forall n \geq 0,$$

when considering the following matrix norm,

$$\|M\| = \sup_{u \in \mathbb{R}^N, u \neq 0} \frac{\|Mu\|}{\|u\|}.$$

For simplicity, we will consider the L^2 norm for the study of stability, allowing us to use tools as Fourier analysis and Plancherel's equality.

3.5.1. Stability in L^2

The L^2 norm is easy to study thanks to Fourier analysis. We will be assuming periodic border conditions for our heat equation, since this helps us apply the Fourier transform without problem. These conditions are better written as $u(t, x+1) = u(t, x)$, for all $x \in [0, 1]$ and all t.

Since our vector $u^n = (u_j^n)_{0 \leq j \leq N}$ is discretized, its Fourier transform would always be zero. To fix this, we will "complete" this function by extending its values into the rest of the domain, therefore getting a function $u^n(x)$ defined on all $[0, 1]$. This means,

$$u^n(x) = u_n^j \quad \text{if} \quad x_{j-\frac{1}{2}} < x < x_{j+\frac{1}{2}},$$

where $x_{j+\frac{1}{2}} = (j + \frac{1}{2})\Delta x$ for $0 \leq j \leq N$. Defined this way, our function $u^n(x)$ belongs to $L^2(0, 1)$. Thanks to Fourier analysis, every $L^2(0, 1)$ function can be decomposed into a Fourier sum,

$$u^n(x) = \sum_{k \in \mathbb{Z}} \hat{u}^n(k) \exp\{2\pi i k x\},$$

where $\hat{u}^n(k) = \int_0^1 u^n(x) \exp\{(-2\pi i k x)\}$. We also have Plancherel's theorem applied to our functions,

$$\int_0^1 |u^n(x)|^2 dx = \sum_{k \in \mathbb{Z}} |\hat{u}^n(k)|^2.$$

A property of the Fourier transform is that when translate our function by a fixed value, its Fourier transform can be studied in the same frequency by multiplying by an exponential according to the following formula,

$$v^n(x) = u^n(x + \Delta x) \implies \hat{v}^n(k) = \hat{u}^n(k) \exp\{(2\pi i k \Delta x)\}.$$

Let us begin studying the stability of the **explicit scheme**, defined by its formula:

$$u^{n+1}(x) = \frac{\kappa\Delta t}{\Delta x^2}u^n(x+\Delta x) + \left(1 - 2\frac{\kappa\Delta t}{\Delta x^2}\right)u^n(x) + \frac{\kappa\Delta t}{\Delta x^2}u^n(x-\Delta x).$$

After applying Fourier transform to both sides, we get

$$\begin{aligned}\hat{u}^{n+1}(k) &= \frac{\kappa\Delta t}{\Delta x^2}\hat{u}^n(k)\exp\{(2\pi ik\Delta x)\} + (1 - 2\frac{\kappa\Delta t}{\Delta x^2})\hat{u}^n(k) + \frac{\kappa\Delta t}{\Delta x^2}\hat{u}^n(k)\exp\{(-2\pi ik\Delta x)\} \\ &= \left(1 + \frac{\kappa\Delta t}{\Delta x^2}(\exp\{(2\pi ik\Delta x)\} - 2 + \exp\{(-2\pi ik\Delta x)\})\right)\hat{u}^n(k).\end{aligned}$$

We can show that $e^{2ix} + e^{-2ix} - 2 = -4\sin^2(x)$. In fact, using Euler's formula for complex exponentials, we have $e^{ix} = \cos x + i\sin x$, therefore

$$\begin{aligned}e^{2ix} + e^{-2ix} - 2 &= 2\cos(2x) - 2 \\ &= 2(\cos^2(x) - \sin^2(x)) - 2 \\ &= 2(\cos^2(x) - 1) - 2\sin^2(x) \\ &= -2\sin^2(x) - 2\sin^2(x) \\ &= -4\sin^2(x).\end{aligned}$$

Using the previous property, we finally get our formula for the Fourier transform of $\hat{u}^{n+1}(k)$,

$$\begin{aligned}\hat{u}^{n+1}(k) &= \left(1 + \frac{\kappa\Delta t}{\Delta x^2}(\exp\{(2\pi ik\Delta x)\} - 2 + \exp\{(-2\pi ik\Delta x)\})\right)\hat{u}^n(k) \\ &= \left(1 - 4\frac{\kappa\Delta t}{\Delta x^2}\sin^2(\pi k\Delta x)\right)\hat{u}^n(k).\end{aligned}$$

We see that this formula is clearly linear, in other words

$$\hat{u}^{n+1}(k) = A(k)\hat{u}^n(k) = A(k)^{n+1}\hat{u}^0(k) \quad \text{with} \quad A(k) = 1 - \frac{4\kappa\Delta t}{\Delta x^2}\sin^2(\pi k\Delta x).$$

For $k \in \mathbb{Z}$, the stability condition is $|A(k)| = 1 - \frac{4\kappa\Delta t}{\Delta x^2}\sin^2(\pi k\Delta x) \leq 1$, that is

$$2\kappa\Delta t\sin^2(\pi k\Delta x) \leq \Delta x^2.$$

For this condition to be satisfied for any value of k, we define our ***CFL condition*** to be,

$$2\kappa\Delta t \leq \Delta x^2 \iff \frac{\kappa\Delta t}{\Delta x^2} \leq \frac{1}{2}$$

If this condition is indeed satisfied, then by Plancherel's formula we have,

$$\|u^n\|_2^2 = \int_0^1 |u^n(x)|^2 dx = \sum_{k\in\mathbb{Z}} |\hat{u}^n(k)|^2 \leq \sum_{k\in\mathbb{Z}} |\hat{u}^0(k)|^2 = \int_0^1 |u^0(x)|^2 dx = \|u^0\|_2^2,$$

which means that our explicit scheme is L^2 stable.

On the other hand, if the stability condition is not met, then one could choose Δx small enough, and k_0 big enough, and an initial condition with one non zero Fourier component $\hat{u}^0(k_0) \neq 0$ with $\sin(\pi k_0\Delta x) = 1$, such that $|A(k_0)| > 1$, so the transfored iteration matrix would grow exponentially

in each iteration. This translates in unbounded oscillations of our simulation.

Therefore, we have that the explicit scheme is stable if and only if the CFL condition $\frac{\kappa\Delta t}{\Delta x^2} \leq \frac{1}{2}$ is satisfied.

Let us now study the stability of the **implicit scheme**. Remembering that this scheme is,

$$-\frac{\kappa\Delta t}{\Delta x^2}u^{n+1}(x-\Delta x) + \left(1+2\frac{\kappa\Delta t}{\Delta x^2}\right)u^{n+1}(x) - \frac{\kappa\Delta t}{\Delta x^2}u^{n+1}(x+\Delta x) = u^n(x),$$

we can apply Fourier transform to both sides to get

$$\hat{u}^{n+1}(k)\left(1+\frac{\kappa\Delta t}{\Delta x^2}(-\exp\{(-2\pi ik\Delta x)\} - \exp\{(2\pi ik\Delta x)\} + 2)\right) = \hat{u}^n(k).$$

In other words,

$$\hat{u}^{n+1}(k) = A(k)\hat{u}^n(k) = A(k)^{n+1}\hat{u}^0(k) \quad \text{where} \quad A(k) = \left(1+4\frac{\kappa\Delta t}{\Delta x^2}\sin^2(\pi k\Delta x)\right)^{-1}.$$

Since $|A(k)| \leq 1$ always, then by Plancherel's formula we have that the implicit scheme is L^2 stable for any choice of Δt and Δx, a property we will call **unconditional stability**. This means that this scheme poses no conditions on our grid size for it to be stable, at the cost of being expensive to compute, since it involves inverse matrices of huge dimensions.

3.6. Visualizing numerical schemes for the Heat equation

We will show that these schemes actually work in practice through one example. For this, we will take a heat equation with a known solution, and check the plot of our numerical solution against the real solution.

Let us consider the following heat equation with both Dirichlet boundary conditions,

$$\begin{cases} 4u_t &= u_{xx}, \quad 0 \leq x \leq 2, t > 0 \\ u(t,0) &= 0 \\ u(t,2) &= 0 \\ u(0,x) &= 2\sin\left(\frac{\pi x}{2}\right) - \sin(\pi x) + 4\sin(2\pi x). \end{cases}$$

This equation can be solved via our previous methods to get exact solution,

$$u(x,t) = 2\sin\left(\frac{\pi x}{2}\right)\exp\left\{\left(-\frac{\pi^2}{16}t\right)\right\} - \sin(\pi x)\exp\left\{\left(-\frac{\pi^2}{4}t\right)\right\} + 4\sin(2\pi x)\exp\left\{(-\pi^2 t)\right\}.$$

To study our numerical schemes in a detailed manner, we focus our attention on both explicit and implicit finite difference schemes.

First, let us see how our ***explicit scheme*** finite difference solution is plotted against the real solution of this PDE.

We know the CFL condition for our numerical scheme is that $c = \frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$. Let us check how

our numerical solution behaves against the real solution, so we can easily check it converges.

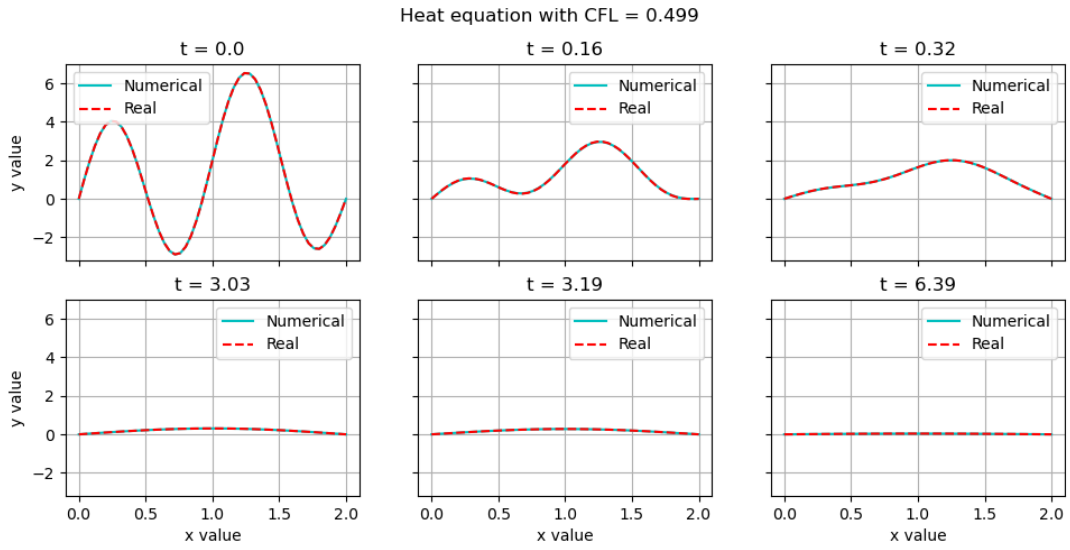

Figura 3.2: A comparison between an explicit numerical scheme solution with a CFL number of 0.499, and real solution.

For a CFL number lower than $\frac{1}{2}$, as is seen in Fig. 3.2 where a CFL number of 0.499 is chosen, we can see that the plot against our numerical solution is equal to the real one. This comes at the cost of having to solve multiple equations, since for a small $(\Delta x)^2$, we must have that $\Delta t \leq \frac{1}{2}(\Delta x)^2$, which bounds our time step by a really small amount.

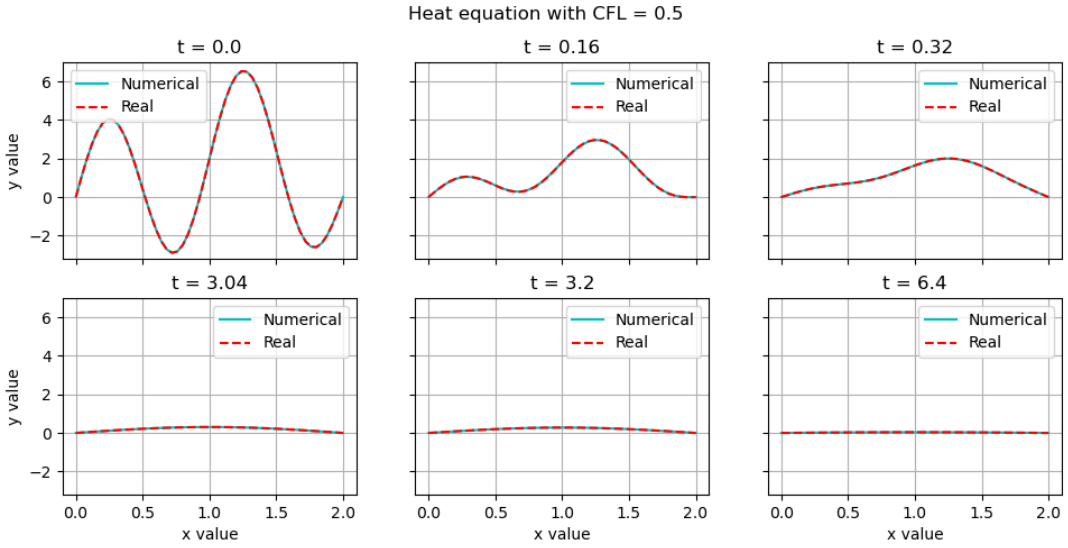

Figura 3.3: A comparison between an explicit numerical scheme solution with a CFL number of 0.5, and its real solution. This is the highest value the CFL number can take before numerical inestabilities are seen.

We can even plot the solution for a CFL value of exactly 0.5 in Fig. 3.3, and check that even for extreme values of the CFL number the explicit scheme has no problem being stable and behaves pretty well, being almost identical to our real solution. The hability to choose a high enough value of

CFL number to study more complex PDEs in a higher time interval, while simultaneously choosing it sufficiently low to not lose stability, is what makes this value special. In this case, choosing a CFL number of 0.1 or 0.5 makes no remarkable difference, since the cost of our simulations is really low, but eventually one may encounter code where even the highest posible value for our CFL number doesn't allow us to solve it in reasonable time, but that's a problem for the future. Let us now study our solution for a slightly higher value of the CFL number.

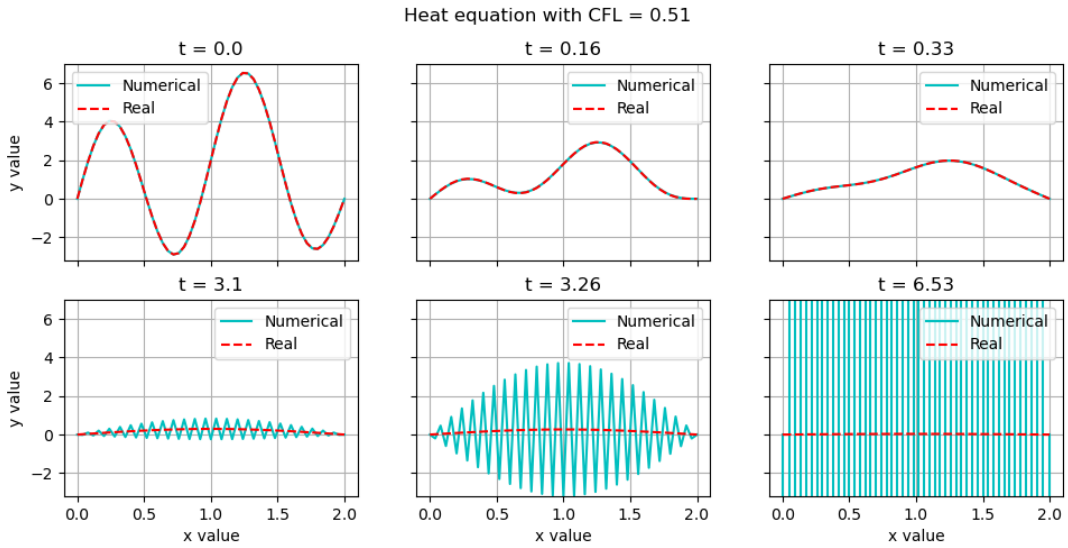

Figura 3.4: A comparison between an explicit numerical scheme solution with a CFL number of 0.51, and its real solution. The CFL condition is not satisfied in this case, and one can see how oscillations beging to grow as time evolves.

To end our analysis, we check our solutions for CFL numbers higher than 0.5. In Fig. 3.4 we choose a value of 0.51, just slightly higher than our extreme value. Even if our value is extremely close, as long as it is higher than 0.5 one can see that soon enough this scheme stops being stable, up to the point where oscillations are so big that they don't even fit our plots. In this case, a CFL number of 0.51 violates our condition, so we expected the iteration matrix to be unbounded by our Fourier analysis. For slightly smaller numbers, as long as it is strictly higher than 0.5, there will always be a time step in which our solution will begin to oscillate unboundedly and diverge.

Let us now focus our attention to the behavior of the ***implicit scheme*** for this same equation, and for different values of our CFL number.

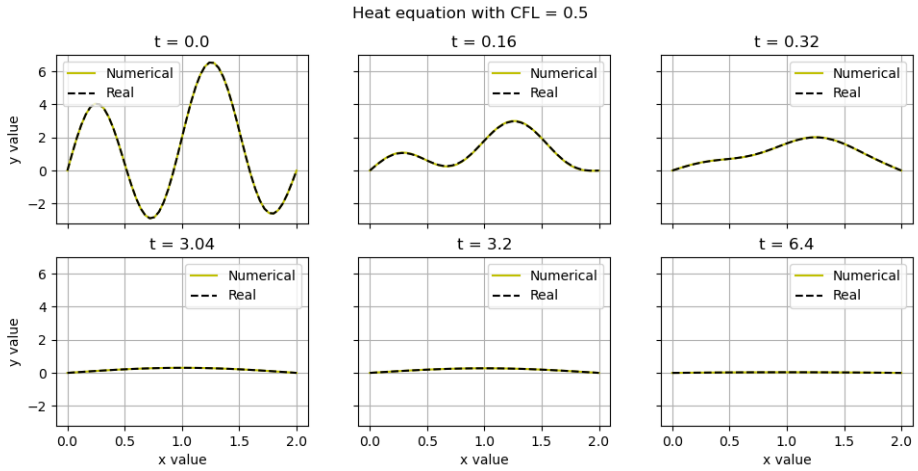

Figura 3.5: A comparison between an implicit numerical scheme solution with a CFL number of 0.5, and its real solution. The CFL condition is satisfied in this case, and one can see the scheme is stable.

Let us now pick the highest value for our CFL number where we know it was stable for the explicit scheme, and plot it as seen in Fig. 3.5. Picking this value of 0.5 shows the scheme is stable and consistent as was the explicit scheme for this value. Our fourier analysis showed that no matter what value for the CFL, the scheme would be stable nonetheless, so let us plot a solution using a bigger value.

For an extreme value of 0.5 for the CFL number, the implicit scheme is very much stable, as we would expect. What is interesting is what happens for higher values of the CFL number.

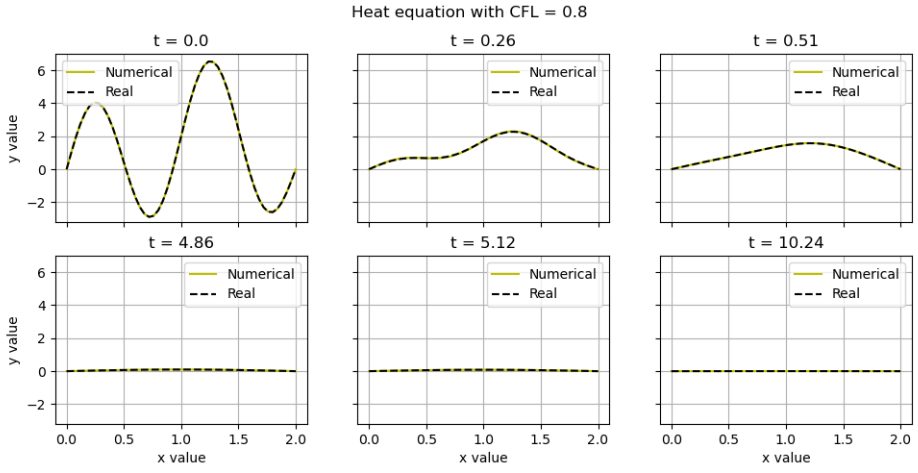

Figura 3.6: A comparison between an implicit numerical scheme solution with a CFL number of 0.8, and its real solution. The CFL condition is not satisfied in this case, yet the solution is stable nevertheless.

Fig. 3.6 shows that for higher values of the CFL number like 0.8, even if the CFL condition is not satisfied by a big amount, one can easily see that the solution doesn't oscillate and stabilizes

itself pretty well. This is what we mean when we say that the implicit scheme is unconditionally stable; one may choose any value for the CFL number as big as one would like, and solutions should never diverge in value. If one checked the time elapsed between these two simulations, however, the implicit scheme is certainly slower. One must find a way to use this knowledge to optimize the time our simulation will run in.

Capítulo 4

Numerical methods for the Wave equation

4.1. Introduction

The 1-D wave equation is a linear second-order PDE which describes the propagation of oscillations at a fixed speed. This equation is a good description of a wide range of phenomena, typically used to model small oscillations about an equilibrium. Solutions to the wave equation play crucial roles in electromagnetism, optics, gravitational physics, and heat transfer.

It is different to consider a one dimensional problem of waves on a string stretched in the x-axis, and radial waves within a sphere. The latter make use of Fourier series, Laplace's spherical harmonics and their generalizations, but we will, for now, limit ourselves to the first case.

The 1-D wave equation has the following form,

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}(x,t) = \frac{\partial^2 u}{\partial x^2}(x,t),$$

where c is the speed at which the wave propagates.

Numerical schemes for this equation come in various forms, by taking different time values at each equation, or even taking fractional time values. Most schemes used are also multi-level schemes, taking into account two time steps in each equation. In this chapter we explore another important one, the linearized wave equation scheme. This allows us to reduce the order of time derivatives, while adding a variable to work with.

4.2. Derivation

We will focus on the context of a piece of string obeying Hooke's law. Letting our string be stretched on the x-axis, we define $u(x,t)$ as the height of our string at the point x and time t. Let us start considering the below diagram (Fig. 4.1) showing a piece of string displaced a small amount about an equilibrium.

We will focus on the forces acting on a small mass element dm contained in a small interval dx, allowing us to approximate the forces. For small displacements about the equilibrium, the horizontal forces acting on this mass element is approximately zero.

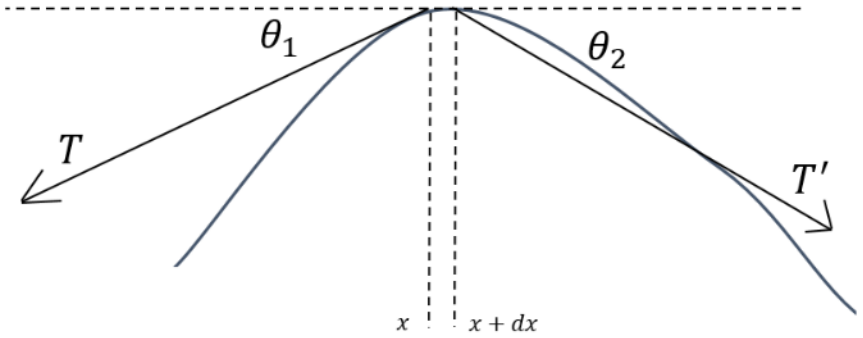

Figura 4.1: A piece of string displaced from its equillibrium. We study its height about the equillibrium at each position and time.

The vertical force acting on this mass element is,

$$\sum F_y = -T' \sin\theta_2 - T\sin\theta_1 = (dm)a = \mu dx \frac{\partial^2 u}{\partial t^2},$$

where μ is the mass density $\mu = \frac{\partial m}{\partial x}$ of the string.

On the other hand, since the horizontal force is approximately zero, we have $T\cos\theta_1 \approx T'\cos\theta_2 \approx T$, therefore,

$$\begin{aligned} -\frac{\mu dx \frac{\partial^2 u}{\partial t^2}}{T} \approx \frac{T'\sin\theta_2 + T\sin\theta_1}{T} &= \frac{T'\sin\theta_2}{T} + \frac{T\sin\theta_1}{T} \\ &\approx \frac{T'\sin\theta_2}{T'\cos\theta_2} + \frac{T + \sin\theta_1}{T\cos\theta_1} \\ &= \tan\theta_1 + \tan\theta_2. \end{aligned}$$

However, for small angles, we have that $\tan\theta_1 + \tan\theta_2 = -\Delta\frac{\partial u}{\partial x}$ (since the tangent is approximately equal to the slope), where this difference is taken between x and $x + dx$.

Diving over dx, we get:

$$-\frac{\mu\frac{\partial^2 u}{\partial t^2}}{T} = \frac{\tan\theta_1 + \tan\theta_2}{dx} = -\frac{\partial\frac{\partial u}{\partial x}}{dx}.$$

Thus giving us the final equation:

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2},$$

where c is the wave velocity $c = \sqrt{\frac{T}{\mu}}$.

4.3. Analytical solution of the wave equation

We begin considering an elastic string with fixed ends at $x = 0$ and $x = L$. The governing equation for $u(x, t)$, that is, the position of the string from its equilibrium, will be,

$$\begin{cases} u_{tt} = c^2 u_{xx} \\ u(t, 0) = u(t, L) = 0 \\ u(0, x) = f(x) \\ u_t(0, x) = g(x) \end{cases}$$

For this, we define new coordinates $a = x - ct$ and $b = x + ct$, representing right and left propagation waves, respectively. By solving for x and t, one gets that their derivatives are rewritten as:

$$\begin{aligned} \frac{\partial}{\partial x} = \frac{1}{2}\left(\frac{\partial}{\partial a} + \frac{\partial}{\partial b}\right) &\implies \frac{\partial^2}{\partial x^2} = \frac{1}{4}\left(\frac{\partial^2}{\partial a^2} + 2\frac{\partial^2}{\partial a \partial b} + \frac{\partial^2}{\partial b^2}\right) \\ \frac{\partial}{\partial t} = \frac{c}{2}\left(\frac{\partial}{\partial b} - \frac{\partial}{\partial a}\right) &\implies \frac{\partial^2}{\partial t^2} = \frac{c^2}{4}\left(\frac{\partial^2}{\partial a^2} - 2\frac{\partial^2}{\partial a \partial b} + \frac{\partial^2}{\partial b^2}\right). \end{aligned}$$

Substituting in for the partial derivatives yields the new equation in coordinates a and b:

$$\frac{\partial^2 u}{\partial a \partial b} = 0.$$

Integrating twice yields the general solution:

$$u(t, x) = F(a) + G(b) = F(x - ct) + G(x + ct),$$

where F and G are two C^2 functions.

To solve for F and G, one uses the initial conditions to get that:

$$\begin{aligned} u(0, x) = f(x) &\implies F(x) + G(x) = f(x) \\ u_t(0, x) = g(x) &\implies cG'(x) - cF'(x) = g(x). \end{aligned}$$

Integrating the last equations yields:

$$cG(x) - cF(x) = \int_{-\inf}^{x} g(\xi)d\xi + c_1.$$

This last equation allows us to solve the system for F and G, whose solution is given by:

$$\begin{aligned} F(x) &= -\frac{1}{2c}\left(-cf(x) + \left(\int_{-\infty}^{x} g(\xi)d\xi + c_1\right)\right) \\ G(x) &= -\frac{1}{2c}\left(-cf(x) - \left(\int_{-\infty}^{x} g(\xi)d\xi + c_1\right)\right). \end{aligned}$$

Reeplacing this on our initial solution,

$$u(t, x) = F(x - ct) + G(x + ct),$$

yields the general solution, often called ***d'Alambert's formula***:

$$u(t, x) = \frac{1}{2}[f(x - ct) + f(x + ct)] + \frac{1}{2c}\int_{x-ct}^{x+ct} g(\xi)d\xi.$$

4.4. Finite differences for the wave equation

For this section, we will focus on a specific form of the wave equation, which we will call the ***linearized first order wave equation***. This is the wave equation we will later use to solve for the light waves on a black hole background.

For this, we consider the usual wave equation with the following boundary conditions:

$$\begin{cases} u_{tt} = u_{xx} \\ u_t = u_x \quad \text{at} \quad x = 0 \\ u_t = -u_x \quad \text{at} \quad x = 1. \end{cases}$$

These new boundary conditions are often called ***Sommerfeld conditions***, where at $x = 0$ and $x = 1$, we demand that there be no radiation coming from the left or right respectively. To solve this, we introduce two new variables $v = \frac{\partial u}{\partial t}$ and $w = \frac{\partial u}{\partial x}$. Using these new variables, the equation can be expressed as:

$$\begin{cases} \frac{\partial}{\partial t}\begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \frac{\partial}{\partial x}\begin{pmatrix} v \\ w \end{pmatrix} \\ v_t = v_x \quad \text{at} \quad x = 0 \\ w_t = w_x \quad \text{at} \quad x = 0 \\ v_t = -v_x \quad \text{at} \quad x = 1 \\ w_t = -w_x \quad \text{at} \quad x = 1. \end{cases} \tag{4.1}$$

To solve this via finite differences, we begin creating a system of equations, using centered derivatives on x and forward derivatives on t:

$$\begin{aligned} \frac{v_j^{n+1} - v_j^n}{\Delta t} &= \theta \frac{w_{j+1}^{n+1} - w_{j-1}^{n+1}}{2\Delta x} + (1-\theta)\frac{w_{j+1}^n - w_{j-1}^n}{2\Delta x} \\ \frac{w_j^{n+1} - w_j^n}{\Delta t} &= \theta \frac{v_{j+1}^{n+1} - v_{j-1}^{n+1}}{2\Delta x} + (1-\theta)\frac{v_{j+1}^n - v_{j-1}^n}{2\Delta x}. \end{aligned}$$

Leting $\varphi = \frac{\Delta t}{2\Delta x}$, this is equivalent to the following system,

$$\begin{aligned} v_j^{n+1} - c^2\theta\varphi w_{j+1}^{n+1} + \theta\varphi w_{j-1}^{n+1} &= v_j^n + c^2(1-\theta)\varphi w_{j+1}^n - (1-\theta)\varphi w_{j-1}^n \\ w_j^{n+1} - \theta\varphi v_{j+1}^{n+1} + \theta\varphi v_{j-1}^{n+1} &= w_j^n + (1-\theta)\varphi v_{j+1}^n - (1-\theta)\varphi v_{j-1}^n. \end{aligned}$$

For the boundaries, we will use second order forward and backward derivatives, which for a random function are expressed as follows:

$$\begin{cases} f'(x)_{\text{forward}} &= \frac{-3f(x)+4f(x+\Delta x)-f(x+2\Delta x)}{2\Delta x} \\ f'(x)_{\text{backward}} &= \frac{3f(x)-4f(x-\Delta x)+f(x-2\Delta x)}{2\Delta x}. \end{cases}$$

At the boundary $x = 0$, we will use the forward derivative for both v and w:

$$\begin{cases} \frac{v_0^{n+1}-v_0^n}{\Delta t} &= \theta\frac{-3v_0^{n+1}+4v_1^{n+1}-v_2^{n+1}}{2\Delta x} + (1-\theta)\frac{-3v_0^n+4v_1^n-v_2^n}{2\Delta x} \\ \frac{w_0^{n+1}-w_0^n}{\Delta t} &= \theta\frac{-3w_0^{n+1}+4w_1^{n+1}-w_2^{n+1}}{2\Delta x} + (1-\theta)\frac{-3w_0^n+4w_1^n-w_2^n}{2\Delta x}. \end{cases}$$

Similarly, at $x = 1$, we use a backward derivative:

$$\begin{cases} \frac{v_N^{n+1}-v_N^n}{\Delta t} & = \theta \cdot -\frac{3v_N^{n+1}-4v_{N-1}^{n+1}+v_{N-2}^{n+1}}{2\Delta x} + (1-\theta) \cdot -\frac{3v_N^n-4v_{N-1}^n+v_{N-2}^n}{2\Delta x} \\ \frac{w_N^{n+1}-w_N^n}{\Delta t} & = \theta \cdot -\frac{3w_N^{n+1}-4w_{N-1}^{n+1}+w_{N-2}^{n+1}}{2\Delta x} + (1-\theta) \cdot -\frac{3w_N^n-4w_{N-1}^n+w_{N-2}^n}{2\Delta x}. \end{cases}$$

All this leads to the following equations for the boundaries:

$$\begin{cases} (1+3\theta\varphi)v_0^{n+1} - 4\theta\varphi v_1^{n+1} + \theta\varphi v_2^{n+1} & = (1-3(1-\theta))\varphi v_0^n \\ & \quad +4(1-\theta)\varphi v_1^n - (1-\theta)\varphi v_2^n \\ (1+3\theta\varphi)w_0^{n+1} - 4\theta\varphi w_1^{n+1} + \theta\varphi w_2^{n+1} & = (1-3(1-\theta))\varphi w_0^n \\ & \quad +4(1-\theta)\varphi w_1^n - (1-\theta)\varphi w_2^n \\ \\ (1+3\theta\varphi)v_N^{n+1} - 4\theta\varphi v_{N-1}^{n+1} + \theta\varphi v_{N-2}^{n+1} & = (1-3(1-\theta)\varphi)v_N^n + 4(1-\theta)\varphi v_{N-1}^n \\ & \quad -(1-\theta)\varphi v_{N-2}^n \\ (1+3\theta\varphi)w_N^{n+1} - 4\theta\varphi w_{N-1}^{n+1} + \theta\varphi w_{N-2}^{n+1} & = (1-3(1-\theta)\varphi)w_N^n \\ & \quad +4(1-\theta)\varphi w_{N-1}^n - (1-\theta)\varphi w_{N-2}^n. \end{cases}$$

4.5. Visualizing numerical schemes for the wave equation

Considering our previous wave linearized system (4.1), we can impose some simple initial conditions for our new variables. In this case, we will consider the following two initial conditions with $v_0 = id \cdot w_0$, where $id = +1, 0, -1$, and

$$w_0(x,t) = e^{-(\frac{x-x_c}{\Delta})^2},$$

that is, w_0 is a simple Gaussian pulse.

We can check that our initial Gaussian amplitude has a value of 1, and we will set a value of $\Delta = 0.05$ for it to be sufficiently smooth throughout our space domain. For our simulation, we will be using a Crank-Nicholson scheme ($\theta = 0.5$) and a value of $\varphi = 0.01$. Let's plot some evolution of these values for our different values of id.

Let's begin with a value of $id = 1$, in figures 4.2 and 4.3. It's easy to see initial conditions are indeed satisfied, and that it sets a perfectly symmetric problem; both variables follow a symmetric system of equations and they have the exact same initial conditions. We can check how it perfectly describes a wave that moves to the left, due to its positive velocity, and that it gets absorbed completely at the left boundary, due to our Sommerfield boundary conditions.

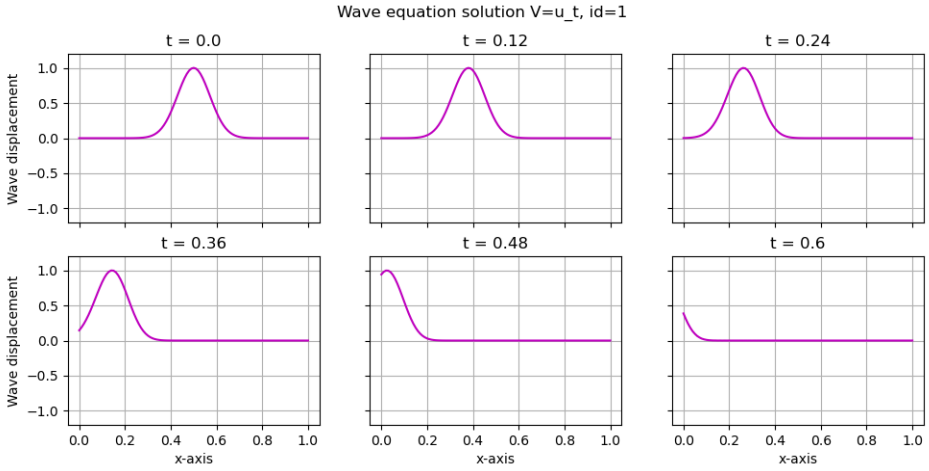

Figura 4.2: Wave equation solution of our $v = u_t$ variable, with $id = 1$.

Our solution for $id = 0$ is a bit more interesting, and we see them in figures 4.4 and 4.5. We notice, again, initial conditions are satisfied, and that waves are formed due to the coupling of our equations. For v, waves are formed with opposite amplitudes and pushed to the sides, while w starts as a Gaussian that splits into two equal smaller waves that are also pushed to the sides. Again, this is the behavior we want for wave equations that get absorbed at the boundaries.

Finally, we plot our solutions for $id = -1$ at figures 4.6 and 4.7. Due to the opposite sign on the initial conditions, it is expected that the waves formed behave the same, but with opposite amplitudes. Again, they show waves moving towards the right boundary, and getting absorbed at it.

While it is the most basic form and numerical scheme of our wave equation that we are studying,

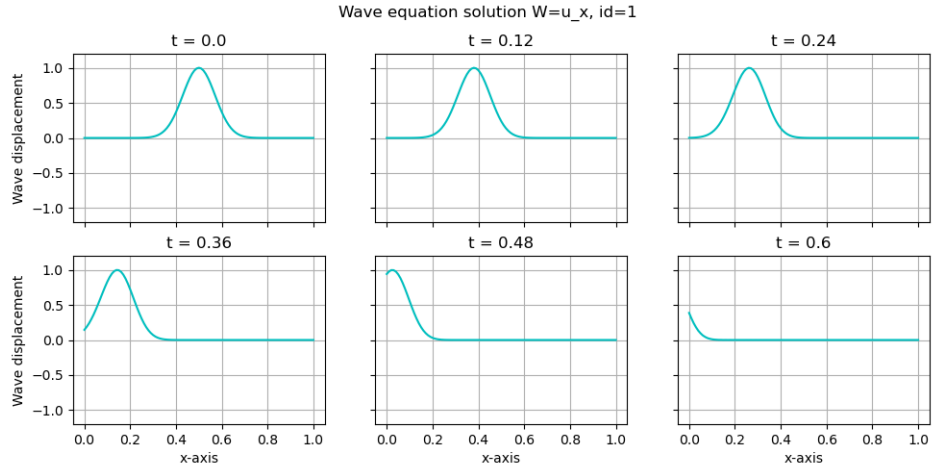

Figura 4.3: Wave equation solution of our $w = u_x$ variable, with $id = 1$.

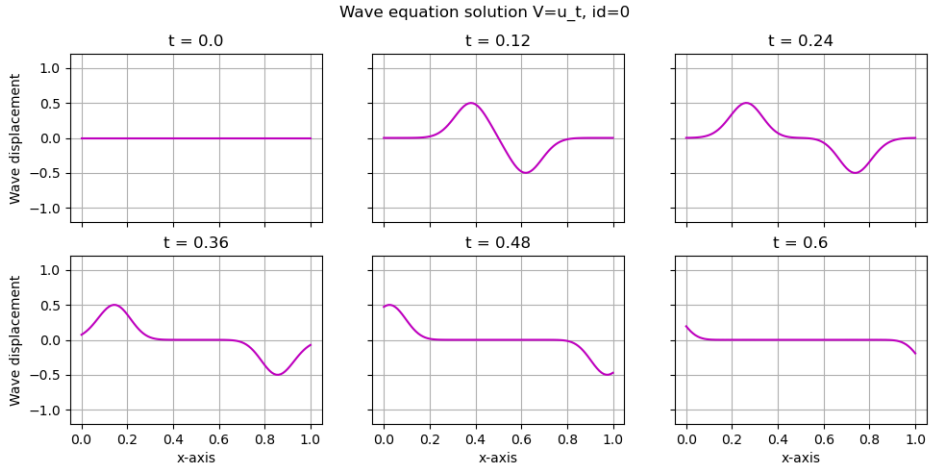

Figura 4.4: Wave equation solution of our $v = u_t$ variable, with $id = 0$.

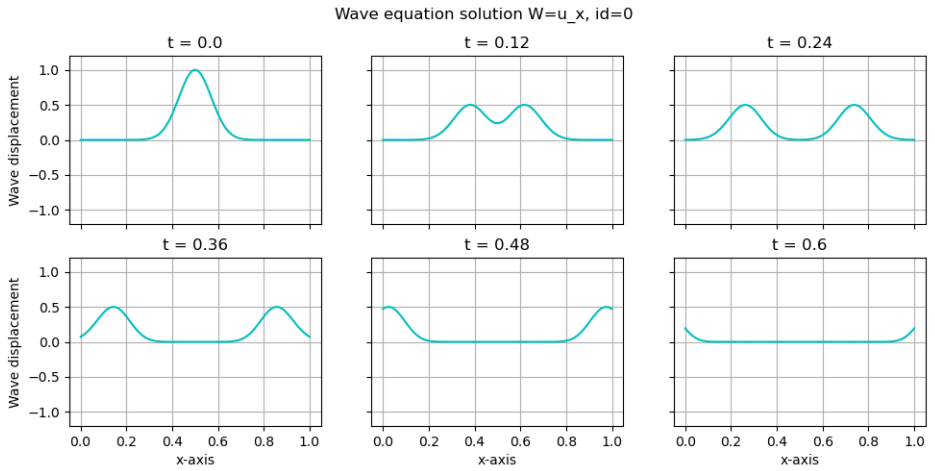

Figura 4.5: Wave equation solution of our $w = u_x$ variable, with $id = 0$.

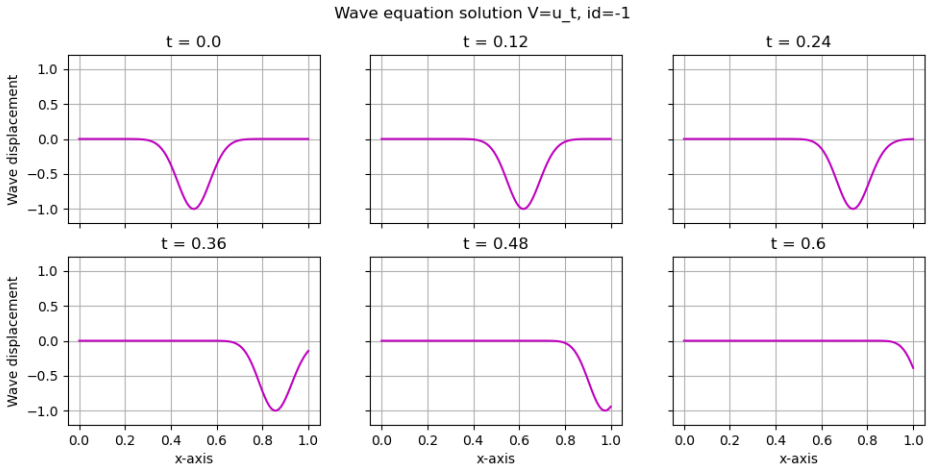

Figura 4.6: Wave equation solution of our $v = u_t$ variable, with $id = -1$.

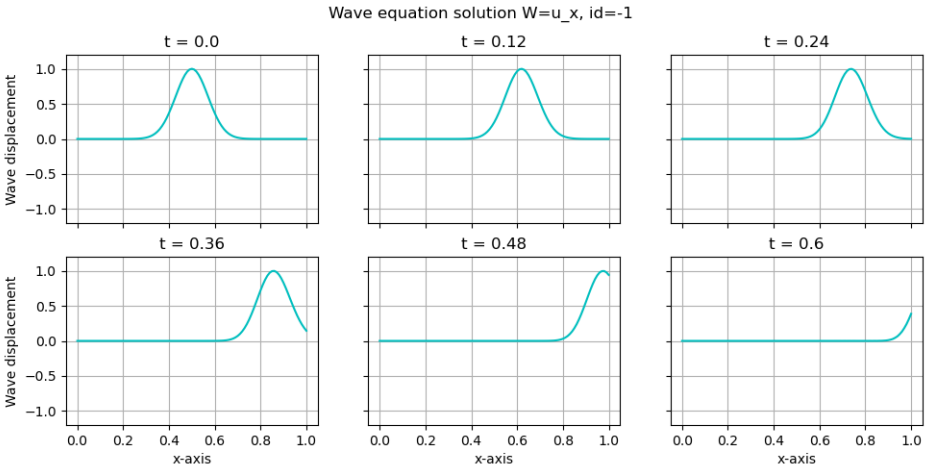

Figura 4.7: Wave equation solution of our $w = u_x$ variable, with $id = -1$.

we will later on use this equation to describe waves in a curved spacetime. Watching these simple wave equations, ones we actually know the behavior they must have throughout the domain, we can deduce our scheme works. Again, since we're working with a Crank-Nicholson scheme, we can be reassured it will be stable by the implicitness of it. Having solved our wave equation, we can save our scheme for when we need it, and continue working our way to the theory of waves in curved spacetime.

Capítulo 5

Relativity and Gravitation

5.1. Introduction

In this section we begin our journey to explain where and how to derive Einstein's field equations. We begin studying the basics of ***Special Relativity***, which is a theory that explains space and time perception when looking at inertial frames of reference. Concepts seen throughout this section are crucial to then look at ***General Relativity***, which amounts to understanding how to fit gravity in the previous theory.

This section wraps up the study of mass, time, space, energy, and gravity when looking at the world from a relativistic point of view. We begin this chapter by looking at how Special relativity started and its key concepts.

5.2. Special Relativity

Special Relativity is a theory of the relationship between space and time, published by Albert Einstein in 1905. This theory heavily describes how time and space are felt different between different observers moving at different velocities. It is described on Einstein's paper "On the Electrodynamics of Moving Bodies", where the incompatibility of Maxwell's equations with Newtonian mechanics proved something was being lost in the study of small velocity moving objects.

This theory is based on two postulates, the two of which require some knowledge about inertial observers. An **inertial reference frame** is a reference frame in which the first law of classical mechanics holds, that is, a mass object on which no external forces act, either remains at rest or move with constant speed. An **inertial observer** is any observer at rest with respect to an inertial reference frame. Having understood this, we list the two postulates:

1. Any inertial observer perceives the same laws of physics.
2. The speed of light in vacuum detected by an inertial observer has always the same value.

This value is a constant called the speed of light and will be denoted by c from this point on. It is intriguing how the notion of relativity and symmetries naturally arise from these two postulates; if all inertial observers perceive the same laws of physics, then there's no preferred reference frame to study the world in. Similarly, the second postulate will affect the perception of time and space on a given reference frame, thus making these notions relative to the inertial observer we pick to make measurements from. Many experiments have yielded that the speed of light value is indeed constant in vacuum, the most famous being the Michelson-Morley experiment. Its accepted value is $c = 299.792.458\ m/s$.

5.2.1. The study of space and time

Let us consider an inertial observer, whose frame of reference's origin is located at $\mathcal{O}$. An *event* is a position in spacetime, and it requires four components: three for the location in space, plus one for the position in time. It is usually represented by a set of coordinates (t, x, y, z) as measured by $\mathcal{O}$. Let us now consider two events labeled by Cartesian coordinates (t_1, x_1, y_1, z_1) and (t_2, x_2, y_2, z_2). We call the lapse of time measured by $\mathcal{O}$ as $\Delta t = t_2 - t_1$ and the space between the two events in the x axis, measured by $\mathcal{O}$, as $\Delta x = x_2 - x_1$. We define Δy and Δz the same way. The *spacetime interval* of these two events will be defined by

$$\Delta s^2 = -(c\Delta t)^2 + \Delta x^2 + \Delta y^2 + \Delta z^2.$$

Let there be another inertial observer $\mathcal{O}'$ moving at constant speed v with respect to $\mathcal{O}$. This observer will measure a new set of coordinates (t'_1, x'_1, y'_1, z'_1) and (t'_2, x'_2, y'_2, z'_2) for each event. If $\mathcal{O}'$ and $\mathcal{O}$ are moving side to side in the x-direction, we can use the second postulate that affirms both observers measure light travelling at the same constant speed c, therefore one can prove their coordinates will be related by

$$ct' = \gamma(ct - v\frac{x}{c}), \quad x' = \gamma(x - vt), \quad y' = y, \quad z' = z.$$

Where $\gamma = \frac{1}{\sqrt{1-\left(\frac{v}{c}\right)^2}}$ is called the **Lorentz factor**. It can be proven that this coordinate transformation preserves our quantity Δs^2. This means that

$$-(\Delta ct)^2 + \Delta x^2 + \Delta y^2 + \Delta z^2 = \Delta s^2 = -(\Delta ct')^2 + \Delta x'^2 + \Delta y'^2 + \Delta z'^2.$$

Coordinate transformations that preserve Δs^2 will be called **Lorentz transformations**.

The spacetime of special relativity is topologically $\mathbb{R}^4$ when endowed with this measure of distance between events, and it will be referred to as a Minkowski space. The invariant interval provides an observer-independent characterization of distances between events. Thanks to this, we can group events based on the sign of their spacetime interval:

- Two events whose separation is $\Delta s^2 < 0$ are said to be *timelike* separated. This implies $\Delta x^2 + \Delta y^2 + \Delta z^2 < (c\Delta t)^2$, indicating that a signal can travel from one event to another—a term often referred to as *causality*.
- Two events whose separation is $\Delta s^2 > 0$ are said to be *spacelike* separated. Due to the previous points, these events cannot have a causal relationship, and no observer can travel from one event to another.
- Two events whose separation is $\Delta s^2 = 0$ are said to be *lightlike* separated. This implies $\frac{\Delta x^2+\Delta y^2+\Delta z^2}{\Delta t^2} = c^2$, meaning that events with null separation are connected by signals travelling at the speed of light. This is, in fact, the spacetime interval of light-separated events.

5.2.2. The Lorentz Group

The spacetime interval was previously defined as a measure of distance in a Minkowski spacetime which is invariant under a Lorentz transformation. Let us try to characterize these transformation and see what their general form is.

Let us sit in a fixed frame $\mathcal{O}$. The coordinates of an event are written in terms of a *four vector* X. Its components will be usually noted as

$$X^\mu = (ct, x, y, z) \quad \mu = 0, 1, 2, 3.$$

The invariant distance between the origin and this event will be

$$\Delta s^2 = -(ct)^2 + x^2 + y^2 + z^2.$$

We can see this as an inner product, defined as

$$X \cdot X \equiv X^T \eta X = X^\mu \eta_{\mu\nu} X^\nu,$$

where the usual Einstein notation is applied. In this space, the matrix η is given by

$$\eta_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

This matrix that defines the inner product is often called a *metric*, and this particular metric is the *Minkowski metric*.

A Lorentz transformation can be thought of as a matrix Λ, that rotates coordinates from a frame $\mathcal{O}$ to coordinates in a frame $\mathcal{O}'$, such that

$$X' = \Lambda X.$$

In index notation, this means $X'^\mu = \Lambda^\mu_\nu X^\nu$. The solutions for these equations can be shown to be *rotations* in 3-dimensions (that is, only in the space dimensions of our four-vector), and *boosts* between time and space.

5.2.3. Proper Time

Time is a quantity that is measured differently between various observers, but there is a way to uniquely determine the time lapse between events that is unique to them, and that is the *proper time*.

Whenever we can describe the trajectory of a particle in an inertial frame we use $x(t)$ and $\frac{dx}{dt}(t)$ for the position and velocity respectively, and thus one could ask what would be the best time (relative to an observer) to parameterize our equations with.

Let us consider a particle at rest in the origin of an inertial frame of reference $\mathcal{O}$, and two events measured by it such that $\Delta \boldsymbol{x} = 0$. The invariant interval is therefore:

$$\Delta s^2 = c^2 \Delta t^2$$

The invariant interval between two events is proportional to the time experienced by the particle. Since it is true for the particle's frame, it must be true for all of them. This time is called the *proper time* and we denote it by τ. In all frames, it is given by

$$\Delta\tau = \left(-\frac{\Delta s^2}{c^2} \right)^{1/2},$$

where one can easily note that this quantity is real as long as the events are *timelike*, that is, the particle does not travel faster than the speed of light. Since all frames agree on the proper time between events, it provides a much better way to parameterize our equations of motion, that is $\boldsymbol{x}(\tau)$,$t(\tau)$ and $\boldsymbol{u}(\tau)$.

This raises the question, how do we use the proper time if we can't always move alongside our test particle? Indeed, the only information we have is the velocity of our particle (its frame of reference), and we need a way to make use of this.

Let us determine the time experienced by a particle moving along a general trajectory. Along a infinitely small segment, it experiences a proper time (as measured by us), such that

$$ds^2 = -(cd\tau)^2 = -(ct)^2 + d\boldsymbol{x}^2.$$

This implies that

$$d\tau = \sqrt{dt^2 - \frac{d\boldsymbol{x}^2}{c^2}} = dt\sqrt{1 - \frac{1}{c^2}\left(\frac{d\boldsymbol{x}}{dt}\right)^2} = dt\sqrt{1 - \frac{\boldsymbol{u}^2}{c^2}},$$

from which we get

$$\frac{dt}{d\tau} = \gamma.$$

From this, the proper time elapsed between two events can be easily computed by standing in any inertial frame, knowing the velocity of the co-moving frame and the time measured between these two events, and summing these infinitesimal small proper times, that is

$$\tau = \int d\tau = \int \frac{1}{\gamma} dt.$$

5.2.4. 4-Velocity

It is no surprise that since there's a special way to parameterize our equations, they get their own names and unique properties. Let us begin considering a general trajectory in spacetime

$$X(\tau) = \begin{pmatrix} ct(\tau) \\ \boldsymbol{x}(\tau) \end{pmatrix}.$$

From this equation, we define the *4-velocity* as,

$$U = \frac{dX}{d\tau} = \begin{pmatrix} c\frac{dt}{d\tau} \\ \frac{d\boldsymbol{x}}{d\tau} \end{pmatrix} = \begin{pmatrix} c\frac{dt}{d\tau} \\ \frac{dt}{d\tau}\frac{d\boldsymbol{x}}{dt} \end{pmatrix}.$$

Using the previous equations for our τ parametrization, and that $\mathbf{u} = \frac{d\boldsymbol{x}}{dt}$, we then get that

$$U = \gamma \begin{pmatrix} c \\ \boldsymbol{u} \end{pmatrix}.$$

Since our proper time $d\tau$ is invariant, this means whenever two set of coordinates are transformed by a Lorentz transformation matrix Λ, that is, $X' = \Lambda X$, then the observer in the $\mathcal{O}'$ frame will measure

$$U' = \Lambda U.$$

This is, the 4-velocity is also transformed by the same Lorentz transformation. Quantities that are transformed only by Lorentz transformations are called *4-vectors*.

Since our 4-velocity is also a 4-vector, we can apply previous analysis to get this vector's norm, and see that it is also invariant,

$$U \cdot U = \gamma \begin{pmatrix} c \\ \mathbf{u} \end{pmatrix} \cdot \gamma \begin{pmatrix} c \\ \mathbf{u} \end{pmatrix} = \gamma^2(c^2 - \mathbf{u}^2) = \gamma^2 \frac{c^2}{\gamma^2} = c^2,$$

a value agreed on by all inertial frames.

5.2.5. 4-momentum

Just as the 4-velocity, the *4-momentum* is defined similarly,

$$P = mU = \begin{pmatrix} m\gamma c \\ m\gamma \boldsymbol{u} \end{pmatrix},$$

where m is the particle's mass. It will turn out that P is a quantity that is conserved. We note that the spatial components show the relativistic generalisation of the 3-momentum,

$$\boldsymbol{p} = m\gamma \boldsymbol{u}.$$

We note that as the particle's speed approaches c, the momentum diverges, because of the formula for γ. Since momentum is conserved, this gives us a bound for a massive particle's speed. Indeed, one can think of $m\gamma$ as the relativistic mass of the particle, and it diverges as the particle approaches the speed of light.

One thing that is usually done is considering the time component of the 4-momentum P^0, and expanding it in a Taylor's series,

$$P^0 = \frac{mc}{\sqrt{1 - u^2/c^2}} = \frac{1}{c}\left(mc^2 + \frac{1}{2}mu^2 + ...\right).$$

The first term is a constant, while the second term actually looks like the non-relativistic kinetic energy of the particle. This suggests that the right interpretation of P^0 is the *energy* of the particle over c, though proving that requires more depth on the subject, but the interpretation will turn out to be correct, so

$$P = \begin{pmatrix} E/c \\ \boldsymbol{p} \end{pmatrix}.$$

The expansion for P^0 shows that,

$$E = cP^0 = m\gamma c^2.$$

Again, we recognize a bound for the particle's speed. As the particle's speed approaches c, its energy (and thus the energy required to make it go faster) gets infinitely bigger.

For a stationary particle, all its energy is contained in its rest mass, giving us Einstein's famous equation

$$E = mc^2.$$

If we place ourselves on a particle's rest frame, we get $P = (mc, 0, 0, 0)$, therefore,

$$P \cdot P = -(mc)^2.$$

Also, for a moving particle, we already know that

$$P \cdot P = -\frac{E^2}{c^2} + \boldsymbol{p}^2.$$

Since the 4-momentum is invariant, we can equate these two equations to get the following equation,

$$E^2 = \boldsymbol{p}^2 c^2 + m^2 c^4,$$

an equation mostly used for particle collisions and general systems where energy is conserved.

5.2.6. Massless Particles

All our previous discussion is built on the notion of a trajectory's proper time, where it was defined for timelike trajectories. But for lightlike trajectories, proper time is zero, and all our notions of velocity, momentum and energy begin to fall apart. Let us focus our attention in massless particles, the ones capable of travelling at the speed of light (thanks to our previous discussion on energy of massive particles), and study their trajectories.

We can sidestep the need for an analysis on proper time by looking at the 4-momentum of a massless particle. Indeed, since $m = 0$, then the 4-momentum must be null for their trajectories, that is,

$$P \cdot P = 0.$$

But particles with zero 4-momentum (and therefore $\Delta s^2 = 0$ and $d\tau = 0$) necessarily follow the trajectory of a light ray.

We know of two types of massless particles: the *photon* and the *graviton*. The photon is heavily studied in various subjects since they are easy to capture and experiment with. Not a big surprise, since it is what allows us to see everything around us. On the other hand, gravitons are the quantum particles that we use to explain the presence of **gravitational waves** in our Universe. These are generated by accelerated masses of an orbital binary system, and were first detected on 14 September 2015 by the Laser Interferometer Gravitational-Wave Observatory (LIGO). On the next chapters we explain the existence of these waves and the physics behind them.

5.3. General Relativity and Differential Geometry

5.3.1. Introduction

In 1915, Albert Einstein finished developing his theory of gravitation, also called General Relativity. According to this theory, the observed gravitational interaction between masses is not due to their induced "gravitational field", but because of the wrapping of space and time which alter the trajectory of *free fall* of moving objects around them. This theory is easily noticed by the sight of bent light ray trajectories around extremely massive objects, on which Newton's law of gravity would not work, since it requires mass for a force to exist.

General relativity also predicts the effects of gravity, such as gravitational waves, gravitational lensing and gravitational time dilation. It provides the foundation for our current understanding on black holes; regions of space where mass is so dense that the gravitational attraction they create is so strong that not even light escapes.

Two physical facts are what inspired Einstein to come up with this theory:

- The **principle of general covariance**, which essentially says that physical phenomena does not depend on a particular frame. Even though physical laws may take different forms in different reference frames, it should be possible to express them in frame-independent formulas. The objects that will represent these formulas will be *tensor fields*.
- The fact that Newtonian gravitational acceleration is the gradient of a scalar field. This fact expresses the Newtonian equivalence of inertial and gravitational mass.

In this theory, gravity can be seen as a tensor field and, in the absense of external forces, motion of test particles is determined by this tensor field, and not on the mass of the particle. The space where these tensors live are *4-dimensional Lorentzian manifolds* (V, g) called the *spacetime*. The trajectory of test particles correspond to the *geodesics* of the metric g.

One always begins explaining this theory with the *weak equivalence principle*. This corresponds to the fact that in general Lorentzian spacetimes, geodesic equations are locally the same as in Minkowski spacetime. This means that in small enough regions of spacetime, the effect of gravity and all laws of motion for freely falling particles are the same as in unaccelerated reference frames. It can be also stated as "At every point in spacetime on an arbitrary gravitational field, it is possible to choose a locally inertial coordinate system such that, within a small enough region, laws of nature take the same form as in unaccelerated Cartesian coordinate systems.

To further understand this theory, specially the wrapping of space and time, we need some tools that will help describe the geometry of curved spaces on higher dimensions. This area of mathematics is called *differential geometry*.

5.3.2. Tensors

Spacetime is defined as a 4-dimensional differential manifold $\mathcal{M}$ endowed with a Lorentzian metric g. The metric is a structure that allows defining distances and angles on a manifold. More precisely, a metric at a point $x \in \mathcal{M}$ is a bilinear form defined on the tangent space at x, that is, a bilinear function that maps pairs of vectors to a real number. We call the pair $(\mathcal{M}, g)$ a psudo-Riemannian manifold.

In these spaces, we can define *tensors*, which are algebraic objects that describe multilinear relationships between objects in this space. They are mappings between different objects such as vectors, scalars and other tensors as well. A crucial property of tensors is that they are frame-independent. This means that one can define a tensor in a specific set of coordinates, and uniquely transform it to a tensor on another set of coordinates, thus forming an equivalence class.

A *covariant p-tensor* can be defined on a point $x \in \mathcal{M}$ as a p multilinear form on the p direct product of the tangent space $T_x\mathcal{M}$, whereas *contravariant p-tensors* are p multilinear forms on p direct products of the cotangent space $T_x^*\mathcal{M}$. For instance, a contravariant 2-tensor T at $x \in \mathcal{M}$ can be thought of as a 2-dimensional matrix that allows for coordinate transformations from $(\mathcal{M}, \phi)$ to $(\mathcal{M}, \phi')$:

$$T_{i'j'} = \frac{\partial x^h}{\partial x^{i'}} \frac{\partial x^k}{\partial x^{j'}} T_{hk},$$

where Einstein's summation convention is applied.

Covariant 2-tensors at x, also called *2-forms*, live on the space denoted as $T_x^* \otimes T_x^*$, whose natural basis associated with local coordinates x^i is denoted by $dx^i \otimes dx^j$. Thus, T is given by the expression

$$T = T_{ij} dx^i \otimes dx^j.$$

As mentioned before, this tensor is a direct product of the tangent space, meaning that it takes two vectors and produces a real number:

$$T(v, w) = T_{ij}(dx^i \otimes dx^j)(v, w) = T_{ij} v^i w^j.$$

The fundamental tensor in general relativity is the *metric tensor*, a covariant 2-tensor. We mentioned that the metric tensor allows us to compute distances between objects in our manifold. Indeed, this tensor induces an inner product inside our manifold that takes the form:

$$g(v, w) = g_{\mu\nu}(dx^\mu \otimes dx^\nu)(v, w) = g_{\mu\nu} v^\mu w^\nu,$$

where it can be easily shown that this tensor is *symmetric*, that is, $g_{\mu\nu} = g_{\nu\mu}$

One can abbreviate the basis $dx^\mu \otimes dx^\nu$ as $dx^\mu dx^\nu$ whenever it causes no confusion, and therefore

define the *infinitesimal line element* as:

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu.$$

This interval imparts information about the causal structure of our spacetime:

- Whenever $ds^2 < 0$, the interval is *timelike*, and one has the formula for the incremental proper time
$$d\tau = \sqrt{-ds^2}.$$
- If $ds^2 = 0$, the interval is *lightlike* and can only be traversed by massless particles moving at the speed c.
- When $ds^2 > 0$, the interval is *spacelike*. Massive objects cannot traverse these intervals of spacetime, and one can define the *proper length* as:
$$dl = \sqrt{ds^2}.$$

The metric tensor plays a key role in the manipulation of indices, the coefficients $g_{\mu\nu}$ of the metric tensor g provide a link between the covariant and contravariant components of other tensors. It has the following properties:

1. $g_{\mu\nu} A^\nu = A^\nu g_{\mu\nu} = A_\mu$.
2. $g^{\mu\nu} A_\nu = A_\nu g^{\mu\nu} = A^\mu$.
3. $g_{\mu\nu} g^{\nu\lambda} = \delta^\lambda_\mu$.

5.3.3. Riemannian connection

The partial derivative operator ∂_μ acting on components of tensor fields are not intrinsic operators on a manifold, since the mapping it provides depends on the coordinate system used. We therefore try to define a *covariant derivative* operator ∇, that performs the same function of the partial derivative, but is independent of coordinates. We require ∇ to be a map from (k, l) tensor fields to $(k, l+1)$ tensor fields, where (k, l) represents the rank or order of the tensor, and to satisfy two properties:

1. Linearity: $\nabla(T + S) = \nabla T + \nabla S$,
2. Leibniz product rule: $\nabla(T \otimes S) = (\nabla T) \otimes S + T \otimes (\nabla S)$.

These conditions actually provide a form for the covariant derivative. If we consider a vector V^ν and get its covariant derivative, we arrive to some coefficients called *connection coefficients* $\Gamma^p_{\mu\sigma}$, which do *not* transform as tensor fields themselves, but help transform a partial derivative into a tensor field. These are used in the general form:

$$\nabla_\mu V^\nu = \partial_\mu V^\nu + \Gamma^\nu_{\mu\lambda} V^\lambda.$$

This gives us idea of how the covariant derivative looks like; a partial derivative plus a correction coefficient, but this doesn't help us see how it would act on different tensor fields, specially on higher order ones. For this purpose, we must introduce two new rules that are crucial for understanding how the covariant derivative acts on different tensor fields:

1. Commutes with contractions: $\nabla_\mu (T^\lambda{}_{\lambda\rho}) = (\nabla T)_\mu{}^\lambda{}_{\lambda\rho}$,
2. Reduces to partial derivatives on scalar functions: $\nabla_\mu \phi = \partial_\mu \phi$.

These two new properties of the covariant derivative (that we have imposed) actually shows us how the covariant derivative looks on any tensor field. Basically, for each upper index we introduce a term with a single $+\Gamma$, and for each lower index a term with a single $-\Gamma$,

$$\begin{aligned}\nabla_\sigma T^{\mu_1\mu_2\ldots\mu_k}{}_{\nu_1\nu_2\ldots\nu_l} =& \partial_\sigma T^{\mu_1\mu_2\ldots\mu_k}{}_{\nu_1\nu_2\ldots\nu_l} \\ &+ \Gamma^{\mu_1}_{\sigma\lambda} T^{\lambda\mu_2\ldots\mu_k}{}_{\nu_1\nu_2\ldots\nu_l} + \Gamma^{\mu_2}_{\sigma\lambda} T^{\mu_1\lambda\ldots\mu_k}{}_{\nu_1\nu_2\ldots\nu_l} + \ldots \\ &- \Gamma^{\lambda}_{\sigma\nu_1} T^{\mu_1\mu_2\ldots\mu_k}{}_{\lambda\nu_2\ldots\nu_l} - \Gamma^{\lambda}_{\sigma\nu_2} T^{\mu_1\mu_2\ldots\mu_k}{}_{\nu_1\lambda\ldots\nu_l} - \ldots.\end{aligned}$$

This gives us how the covariant derivative acts on any tensor, but it does not yet gives us a definite form. One could define a large number of connection and get distinct notions of covariant differentiation. However, this is no concern since we impose the last two conditions to get our connection used in general relativity:

1. Torsion-free connection (symmetric on lower indices): $\Gamma^\lambda_{\mu\nu} = \Gamma^\lambda_{\nu\mu}$,
2. Metric compatibility: $\nabla_p g_{\mu\nu} = 0$.

These conditions let us arrive to an unique connection, called by many names such as *Christoffel connection*, *Levi-Civita connection*, and sometimes the *Riemannian connection*, and has the following form in terms of the *metric*:

$$\Gamma^\sigma_{\mu\nu} = \frac{1}{2} g^{\sigma\rho}(\partial_\mu g_{\nu\rho} + \partial_\nu g_{\rho\mu} - \partial_\rho g_{\mu\nu}).$$

5.3.4. Parallel transport and geodesics

An important thing about manifolds is that vectors are elements of tangent spaces at each individual point. This means that it makes no sense to compare a vector defined at two different points, without observing how tangent spaces are transformed from one point to another. When working with flat spaces, the notion of "moving a vector from one point to another while keeping it constant" is natural to all of us, we have always worked with vectors in that way, but in curved spaces it is not that simple.

The concept of moving a vector along a path while keeping it constant is known as *parallel transport*. This is defined whenever we have a connection. One important difference between flat and curved spaces is that, while flat spaces keep vectors constant when moving them along a path, curved space parallel transport gives different results depending on the path taken.

Let us work how to solve this issue. Suppose we have a curve on our manifold $x^\mu(\lambda)$. The requirement for a tensor T to be constant along this curve in flat space is $\frac{dT}{d\lambda} = \frac{dT}{dx^\mu}\frac{dx^\mu}{d\lambda} = 0$. We can therefore define a covariant derivative along our curve to be given by:

$$\frac{D}{d\lambda} = \frac{dx^\mu}{d\lambda}\nabla_\mu,$$

and define the *parallel transport* of a tensor T along the path $x^\mu(\lambda)$ to be the condition that, along this path,

$$\left(\frac{D}{d\lambda}T\right)^{\mu_1\mu_2\ldots\mu_k}{}_{\nu_1\nu_2\ldots\nu_l} = 0.$$

This is the equation of parallel transport, ensuring that the tensor T remains constant along the path $x^\mu(\lambda)$. For a vector, it takes the form

$$\frac{d}{d\lambda}V^\mu + \Gamma^\mu_{\sigma\rho}\frac{dx^\sigma}{d\lambda}V^\rho = 0.$$

One can clearly note that this equation depends on the connection chosen. Whenever we have a metric compatible connection, we clearly have that the metric is parallel transported along any path (its covariant derivative is everywhere zero), so that we can see that the inner product of parallel

transported vectors is preserved:

$$\frac{D}{d\lambda}(g_{\mu\nu}V^\mu W^\nu) = \underbrace{\left(\frac{D}{d\lambda}g_{\mu\nu}\right)}_{=0} V^\mu W^\nu + g_{\mu\nu}\underbrace{\left(\frac{D}{d\lambda}V^\mu\right)}_{=0} W^\nu + g_{\mu\nu}V^\nu \underbrace{\left(\frac{D}{d\lambda}W^\nu\right)}_{=0} = 0,$$

so then all notions of angles and norms are also preserved.

Once we understand parallel transport, the next step is to get to *geodesics*. Geodesics are the notion of straight lines in curved spaces. They make use of the fact that straight lines are paths that parallel transport their own tangent vectors.

Suppose we have, again, our curve $x^\mu(\lambda)$. The tangent vector to this curve is $\frac{dx^\mu}{d\lambda}$, so the condition for parallel transportation reads,

$$\frac{D}{d\lambda}\frac{dx^\mu}{d\lambda} = \underbrace{\frac{d^2x^\mu}{d\lambda^2} + \Gamma^\mu_{\rho\sigma}\frac{dx^\rho}{d\lambda}\frac{dx^\sigma}{d\lambda} = 0}_{*}.$$

Here, the terms represent the acceleration of the curve $\frac{d^2x^\mu}{d\lambda^2}$ the the influence of the connection coefficients $\Gamma^\mu{}_{\rho\sigma}$ on the curve's tangent vector. The **geodesic equation**, denoted by (*), represents the condition for parallel transporting the tangent vector of a curve along the curve itself. It is a second-order differential equation that characterizes the path of objects in 'free fall' within our manifold.

The preservation of the inner products of vectors (in particular, these tangent vectors) while parallel transported imply that the character of timelight/lightlight/spacelike curves never change, since they are characterized by the inner product of the tangent vector of the path with itself. One can also prove that timelike geodesics are curves that maximize proper time between two events.

5.3.5. The Riemann curvature tensor

While in flat spaces, parallel transport around a closed loop leaves the object unchanged, in curved spaces this does not necessarily happen. How much objects are transported around closed infinitesimal loops is what we will use to characterize the *curvature* of our manifold. For this purpose, we define the *Riemann curvature tensor*. To speak mathematically, curvature will signal the non-commutativity of covariant derivatives.

For this, we consider a commutation $(\nabla_\alpha\nabla_\beta - \nabla_\beta\nabla_\alpha)v^\lambda$ of two covariant derivatives (characterized by the spacetime connection ∇) of a vector v. This is a rank (1,2) tensor which we can prove depends linearly on v, therefore there exist coefficients ${}^{(4)}R^\gamma{}_{\mu\alpha\beta}$ such that

$$(\nabla_\alpha\nabla_\beta - \nabla_\beta\nabla_\alpha)v^\gamma \equiv {}^{(4)}R^\gamma{}_{\mu\alpha\beta}v^\mu.$$

These coefficients form a rank (1,3) tensor called the **Riemann curvature tensor**.

Expanding covariant derivatives with our chosen Riemann connection, we can derive a formula for this tensor, yielding

$${}^{(4)}R^\gamma{}_{\mu\alpha\beta} = \partial_\alpha\Gamma^\gamma_{\beta\mu} - \partial_\beta\Gamma^\gamma_{\alpha\mu} + \Gamma^\gamma_{\alpha\rho}\Gamma^\rho_{\beta\mu} - \Gamma^\gamma_{\beta\rho}\Gamma^\rho_{\alpha\mu}.$$

Clearly when a space is flat, the Riemann tensor vanishes (since Christoffel symbols would be zero everywhere). However, one can also prove that if the Riemann tensor vanishes on a domain of the manifold, then the metric is locally flat in this domain.

Since our goal is to work with Riemann tensor components, we would be talking of a total of $4 \times 4 \times 4 \times 4 = 256$ components, and that would be a lot of different equations to solve. Luckily,

the Riemann tensor has a few symmetries that allow us to get a minimal number of independent components. They are listed as follows:

- Antisymmetry on its last two indices:

$$^{(4)}R^{\gamma}{}_{\mu\alpha\beta} = -^{(4)}R^{\gamma}{}_{\mu\beta\alpha},$$

- Antisymmetry on its first two indices:

$$^{(4)}R^{\gamma}{}_{\mu\alpha\beta} = -^{(4)}R^{\mu}{}_{\gamma\alpha\beta},$$

- Sum of cyclic permutation of lower indices vanishes:

$$^{(4)}R^{\gamma}{}_{\mu\alpha\beta} + {}^{(4)}R^{\gamma}{}_{\alpha\beta\mu} + {}^{(4)}R^{\gamma}{}_{\beta\mu\alpha} = 0,$$

- Symmetry on permutations of the first two indices with the last two indices, when all indices are lowered:

$$^{(4)}R_{\lambda\mu\alpha\beta} = {}^{(4)}R_{\alpha\beta\lambda\mu}.$$

These are all non independent equations (some imply the others), but they are nonetheless all true. Given these relationships, we can prove the Riemann curvature tensor is left with 20 independent components.

In addition to the algebraic symmetries of this tensor, there's a differential identity it obeys. If we consider the covariant derivatives of this tensor we obtain the *Bianchi identities*:

$$\nabla_{\lambda}{}^{(4)}R_{\rho\sigma\mu\nu} + \nabla_{\rho}{}^{(4)}R_{\sigma\lambda\mu\nu} + \nabla_{\sigma}{}^{(4)}R_{\lambda\rho\mu\nu} = 0.$$

We will often need to take contractions of the Riemann tensor, but due to the number of symmetries and anti-symmetries, most of these contractions are zero. The only non zero contraction gives us the *Ricci tensor*:

$$^{(4)}R_{\mu\nu} := {}^{(4)}R^{\lambda}_{\mu\lambda\nu},$$

where the symmetries of the Riemann tensor show this tensor is also symmetric.

Using the metric, we can take one further contraction and form the ***Ricci scalar***:

$$^{(4)}R := g^{\alpha\beta}R_{\alpha\beta} = R^{\alpha}{}_{\alpha}.$$

Let us take our Bianchi identities and contract twice,

$$\begin{aligned}
0 &= g^{\nu\sigma}g^{\mu\lambda}(\nabla_{\lambda}{}^{(4)}R_{\rho\sigma\mu\nu} + \nabla_{\rho}{}^{(4)}R_{\sigma\lambda\mu\nu} + \nabla_{\sigma}{}^{(4)}R_{\lambda\rho\mu\nu}) \\
&= g^{\nu\sigma}g^{\mu\lambda}(\nabla_{\lambda}{}^{(4)}R_{\rho\sigma\mu\nu} + \nabla_{\rho}{}^{(4)}R_{\mu\nu\sigma\lambda} + \nabla_{\sigma}{}^{(4)}R_{\lambda\rho\mu\nu} \\
&= g^{\nu\sigma}(\nabla^{\mu(4)}R_{\rho\sigma\mu\nu} + \nabla_{\rho}{}^{(4)}R^{\lambda}_{\nu\sigma\lambda} + \nabla_{\sigma}{}^{(4)}R^{\mu}_{\rho\mu\nu}) \\
&= g^{\nu\sigma}(\nabla^{\mu(4)}R_{\rho\sigma\mu\nu} - \nabla_{\rho}{}^{(4)}R^{\lambda}_{\nu\lambda\sigma} + \nabla_{\sigma}{}^{(4)}R^{\mu}_{\rho\mu\nu}) \\
&= g^{\nu\sigma}(\nabla^{\mu(4)}R_{\sigma\rho\nu\mu} - \nabla_{\rho}{}^{(4)}R_{\nu\sigma} + \nabla_{\sigma}{}^{(4)}R_{\rho\nu} \\
&= \nabla^{\mu(4)}R^{\nu}_{\rho\nu\mu} - \nabla_{\rho}{}^{(4)}R + \nabla^{\nu(4)}R_{\rho\nu} \\
&= \nabla^{\mu(4)}R_{\rho\mu} - \nabla_{\rho}{}^{(4)}R + \nabla^{\nu(4)}R_{\rho\nu} \\
&= 2\nabla^{\mu(4)}R_{\rho\mu} - \nabla_{\rho}{}^{(4)}R.
\end{aligned}$$

Equivalently,

$$\nabla^{\mu(4)}R_{\rho\mu} = \frac{1}{2}\nabla_{\rho}{}^{(4)}R.$$

Transforming ∇_ρ into $g_{\rho\mu}\nabla^\mu$, we get:

$$\nabla^{\mu\,(4)}R_{\rho\mu} = \frac{1}{2}g_{\rho\mu}\nabla^{\mu\,(4)}R.$$

Linearity of the contravariant derivative leads us to our final equation:

$$\nabla^\mu \left({}^{(4)}R_{\rho\mu} - \frac{1}{2}g_{\rho\mu}{}^{(4)}R \right) = 0$$

We then define the *Einstein tensor* as

$${}^{(4)}G_{\mu\nu} = {}^{(4)}R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}{}^{(4)}R,$$

and so, with a subtle change of indices, the above equation is equivalent to

$$\nabla^{\nu\,(4)}G_{\mu\nu} = 0.$$

This result brings us one step closer to obtaining Einstein's equations.

5.3.6. The Newtonian limit

As we have mentioned earlier, geodesics are the paths that free particles move along in curved spacetime. This way of looking at gravity changes the whole paradigm of how gravity works, but we know that when looking at small bodies (compared to huge stars and black holes), then gravity is a good theory to work with. Thus we would like to know how the geodesic equation related to Newtonian gravity, defining thus the ***Newtonian limit***.

Let us begin remembering the geodesic equation:

$$\frac{d^2x^\mu}{d\lambda^2} + \Gamma^\mu_{\rho\sigma}\frac{dx^\rho}{d\lambda}\frac{dx^\sigma}{d\lambda} = 0.$$

To define the Newtonian limit, we impose three requirements: the particles move slowly (with respect to the speed of light), the gravitational field is weak (considered as a perturbation of flat space), and the field is static.

Taking the proper time τ as an affine parameter, we check that moving slowly means that:

$$\frac{dx^i}{dt} \ll 1 \implies \frac{dx^i}{d\tau} \ll \frac{dt}{d\tau}.$$

The geodesic equation with this affine parameter thus becomes,

$$\frac{d^2x^\mu}{d\tau} + \Gamma^\mu_{00}\left(\frac{dt}{d\tau}\right)^2 = 0.$$

The field being static allow us to not consider partial derivatives of the metric, thus our equations for the Christoffel symbols become,

$$\Gamma^\mu{}_{00} = \frac{1}{2}g^{\mu\lambda}(\partial_0 g_{\lambda 0} + \partial_0 g_{0\lambda} - \partial_\lambda g_{00}) = -\frac{1}{2}g^{\mu\lambda}\partial_\lambda g_{00}.$$

The weakness of the gravitational field also allow us to decompose the metric into its Minkowski form plus a small perturbation around it:

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}, \qquad \|h_{\mu\nu}\| \ll 1,$$

where $\eta_{\mu\nu}$ is the canonical form of the metric and everything is expressed in Cartesian coordinates. To get the inverse metric, we can see that the inverse of $(I + A)$ is $(I - A)$ when A^2 is small, that is, $g^{\mu\nu} = \eta^{\mu\nu} - h^{\mu\nu}$. Therefore we have that,

$$\Gamma^{\mu}{}_{00} = -\frac{1}{2}\eta^{\mu\lambda}\partial_\lambda h_{00},$$

to first order in h.

The geodesic equation can then be rewritten as

$$\frac{d^2x^\mu}{d\tau^2} = \frac{1}{2}\eta^{\mu\lambda}\partial_\lambda\left(\frac{dt}{d\tau}\right)^2.$$

Since the field is static, we have that $\partial_0 h_{00} = 0$, therefore the $\mu = 0$ component of this equation is

$$\frac{d^2t}{d\tau^2} = 0.$$

This just means $\frac{dt}{d\tau}$ is constant. Let now examine spacelike components of the geodesic equation:

$$\frac{d^2x^i}{d\tau^2} = \frac{1}{2}\left(\frac{dt}{d\tau}\right)^2\partial_i h_{00}.$$

Diving both sides by $(\frac{dt}{d\tau})^2$, we get that

$$\frac{d^2x^i}{dt^2} = \frac{1}{2}\partial_i h_{00}.$$

Remembering that in Newton's theory of gravitation we have that there exists a potential such that

$$m\boldsymbol{a}^i = m\frac{d^2x^i}{dt^2} = \partial_i\Phi,$$

we can compare it to the previous equation to get that,

$$h_{00} = -2\Phi,$$

and consecutively,

$$g_{00} = -(1 + 2\Phi).$$

This means that the curvature of spacetime is sufficient to describe gravity in the Newtonian limit as long as the metric takes the previous form, and that for a single gravitating body we can recover the formula

$$\Phi = -\frac{GM}{r}.$$

5.3.7. The Einstein Equations

We will be trying to find an equation that supersedes the Poisson equation for the Newtonian potential:

$$\nabla^2\Phi = 4\pi G\rho,$$

where ∇ is the Laplacian in space and ρ is the mass density. We know that the left-hand side is a second order differential operator acting on the gravitational potential, and the right-hand side is a measure of mass distribution. What we would like to get is a tensor generalization of the above equation. A tensor generalization of the mass density is the energy-momentum tensor $T_{\mu\nu}$. We could guess that the gravitational potential should be therefore replaced by the metric tensor.

Using that for the metric in the Newtonian limit we have that $g_{00} = -(1 + \Phi)$, and that $T_{00} = \rho$, we see that in this limit we are looking for the equation that predicts

$$\nabla^2 h_{00} = -8\pi g T_{00},$$

but with all tensorial elements. We remember we have a known tensorial object that involves second order derivatives of the metric, and that is the Riemann tensor $R^{rho}{}_{\sigma\mu\nu}$. Since it does not have the right number of indices, we contract it to form the Ricci tensor $R_{\mu\nu}$. We could guess that the gravitational field equations are,

$$R_{\mu\nu} = \kappa T_{\mu\nu},$$

for some value of κ, but this does not hold when $\nabla^\mu R_{\mu\nu} \neq 0$, since by the conservation of energy-momentum in curved spacetime we always have $\nabla^\mu T_{\mu\nu} = 0$. On the other hand, we do know another tensorial equation whose divergence vanishes, and that is the Einstein tensor $G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu}$. We can then guess that the equation for the metric is:

$$G_{\mu\nu} = \kappa T_{\mu\nu}.$$

We can check that this equation does reproduce gravity in the Newtonian limit, that is,

$$\nabla^2 h_{00} = -\kappa T_{00},$$

when κ is set to $8\pi G$. Having found our proportionality constant thanks to the Newtonian limit, we arrive to ***Einstein's equations*** for general relativity:

$$R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} = 8\pi G T_{\mu\nu}.$$

Capítulo 6

Numerical Relativity

General relativity states that the Universe is a 4-dimensional curved spacetime, with the Einstein Equations describing its dynamics. To solve these equations numerically, it would be hard to consider a time evolution when there is no a priori concept of "flow of time". To do such, we will need to adopt a different strategy.

Throughout this chapter, we will consider a spacetime $(\mathcal{M}, g)$, where $\mathcal{M}$ is a 4 dimensional real smooth manifold and g is a Lorentzian metric on $\mathcal{M}$. At any given point $p \in \mathcal{M}$, we denote $\mathcal{T}_p(\mathcal{M})$ as the *tangent space of* p. Its dual space of all linear forms at p will be denoted by $\mathcal{T}_p^*(\mathcal{M})$. Naturally we have:

$$\mathcal{T}(\mathcal{M}) \equiv \bigcup_{p\in\mathcal{M}} \mathcal{T}_p(\mathcal{M}) \quad \mathcal{T}^*(\mathcal{M}) \equiv \bigcup_{p\in\mathcal{M}} \mathcal{T}_p^*(\mathcal{M}).$$

Since we will be dealing with manifolds and hypersurfaces, we will deal with indices separately: Greek indices run in $\{0, 1, 2, 3\}$, and lower case Latin indices run in $\{1, 2, 3\}$.

6.1. Fitting a hypersurface into a Manifold

We begin considering a space-time given by a 4-dimensional manifold $\mathcal{M}$ with a metric $g_{\mu\nu}$ defined on it. The coordinates in this manifold will be denoted by x^i. We define the embedding of 3-dimensional hypersurfaces $(\Sigma_t)_{t\in\mathbb{R}}$ as the foliation of $\mathcal{M}$ by this family of hypersurfaces such that:

$$\mathcal{M} = \bigcup_{t\in\mathbb{R}} \Sigma_t.$$

For it to define an embedding, we demand that the hypersurface does not intersect itself. A hypersurface can be defined as a set of points for which a scalar field on $\mathcal{M}$, t for instance, is constant. This means that if Σ_n represents our n-th hypersurface, then:

$$\forall p \in \mathcal{M}, \quad p \in \Sigma_n \iff t(p) = n. \tag{6.1}$$

From now on we will talk about a single hypersurface Σ whenever there's no reason to talk about a specific one.

The metric $g = g_{\mu\nu}$ on $\mathcal{M}$ induces a metric $\gamma = \gamma_{ij}$ on Σ as given by:

$$\gamma_{ij} = g_{ij},$$

representing the values of g which are only in Σ.

6.1.1. Normal vector to our hypersurface

Taking t as our scalar field on $\mathcal{M}$ that defines our hypersurface Σ (6.1), then the gradient 1-form $\boldsymbol{dt}$ is normal to Σ. This means that for every vector v tangent to Σ, we have $(\boldsymbol{dt})_\mu v^\mu = 0$. Using this, we have that the vector $\vec{\nabla} t$ whose components are $\nabla^\mu t = g^{\mu\nu}\nabla_\nu t = g^{\mu\nu}(\boldsymbol{dt})_\nu$, is a vector normal to Σ.

For spacelike hypersurfaces, we will define our unit normal vector $\boldsymbol{n}$ as the vector colinear to $\vec{\nabla} t$ such that:

$$\boldsymbol{n} \cdot \boldsymbol{n} = -1, \tag{6.2}$$

and its components as $(\boldsymbol{n})^\mu = n^\mu$.

6.1.2. Orthogonal projector

When working with our equations, our goal will be to study variables within our hypersurface itself, and later check how they change as the hypersurface changes. For this, we will derive an operator that will take tensors from $\mathcal{T}_p(\mathcal{M})$ to $\mathcal{T}_p(\Sigma)$.

Taking $\boldsymbol{n}$ as the vector normal to our hypersurface, we can decompose each tangent space $\mathcal{T}_p(\mathcal{M})$ into a direct sum as follows:

$$\mathcal{T}_p(\mathcal{M}) = \mathcal{T}_p(\Sigma) \oplus \langle \boldsymbol{n} \rangle,$$

where $\langle \boldsymbol{n} \rangle$ stands for the subspace generated by the vector $\boldsymbol{n}$.

We can define the *orthogonal projector onto* Σ as the operator associated with the previous decomposition, that is:

$$\begin{aligned} \vec{\gamma} \colon \mathcal{T}_p(\mathcal{M}) &\longrightarrow \mathcal{T}_p(\Sigma) \\ v &\longmapsto v + (\boldsymbol{n} \cdot v)\boldsymbol{n}. \end{aligned}$$

By the normalization condition on $\boldsymbol{n}$ (6.2), this projection satisfies

$$\vec{\gamma}(\boldsymbol{n}) = 0 \quad \text{and} \quad \forall v \in \mathcal{T}_p(\Sigma), \quad \vec{\gamma}(v) = v.$$

Since it is clearly a linear operator, this means that it transforms every vector into one whose components lay completely on Σ. If we express our vector v in terms of a basis (e_α) of $\mathcal{T}_p(\mathcal{M})$, that is,

$$v = v^\alpha e_\alpha,$$

then our projection operator acts the following way on the components of v:

$$\begin{aligned} (v^i e_i)^\alpha &= \gamma^\alpha{}_\beta (v^\beta) \\ &= v^\alpha + (v^\beta n_\beta) n^\alpha \\ &= v^\beta \delta^\alpha{}_\beta + v^\beta n^\alpha n_\beta \\ &= v^\beta (\delta^\alpha{}_\beta + n^\alpha n_\beta). \end{aligned}$$

We therefore have the components of $\vec{\gamma}$ with respect to any basis (e_α):

$$\gamma^\alpha{}_\beta = \delta^\alpha{}_\beta + n^\alpha n_\beta.$$

This operator can also be used to induce a metric on $\mathcal{T}_p(\mathcal{M})$ (in contrast to $\gamma_{ij} = g_{ij}$ that only works for vectors in Σ). Indeed, we can lower the index of this orthogonal projector to get:

$$\gamma_{\alpha\beta} = g_{\alpha\mu}\gamma^\mu{}_\beta = g_{\alpha\beta} + n_\alpha n_\beta. \tag{6.3}$$

We can see this metric works by checking that if v and u are vectors tangent to Σ, then:

$$\gamma(u,v) = g(u,v) + \underbrace{n_\alpha u^\alpha}_{=0}\underbrace{n_\beta v^\beta}_{=0} = g(u,v),$$

by the orthogonality of $\boldsymbol{n}$.

On the other hand, if we have that $u = \lambda \boldsymbol{n}$, then:

$$\gamma(u,v) = \lambda g(\boldsymbol{n},v) + \lambda \underbrace{(n_\alpha n^\alpha)}_{=-1}(n_\beta v^\beta) = \lambda(g(\boldsymbol{n},v) - \underbrace{n_\beta v^\beta}_{=g(\boldsymbol{n},v)}) = 0.$$

On a last note, we will use this projection operator to project any tensor on $\mathcal{M}$. Given a rank (p,q) tensor T on $\mathcal{M}$, we denote by $\vec{\gamma}$T another tensor on $\mathcal{M}$ such that its components in any basis (e_α) of $\mathcal{T}_p(\mathcal{M})$ are expressed in terms of those of T by:

$$(\vec{\gamma}T)^{\alpha_1 \dots \alpha_p}{}_{\beta_1 \dots \beta_q} = \gamma^{\alpha_1}{}_{\mu_1} \dots \gamma^{\alpha_p}{}_{\mu_p} \gamma^{\nu_1}{}_{\beta_1} \dots \gamma^{\nu_q}{}_{\beta_q} T^{\mu_1 \dots \mu_p}{}_{\nu_1 \dots \nu_q}.$$

We notice that for any tensor T, $\vec{\gamma}T$ is tangent to Σ.

Using our induced metric $\gamma_{\mu\nu}$ (6.3), we can project our 4D tensors on the Einstein equations straightforwardly onto 3D spatial slices. For instance, the ***3D Christoffel symbols***:

$$\Gamma^\alpha{}_{\beta\delta} = \frac{1}{2}\gamma^{\alpha\mu}(\gamma_{\mu\beta,\delta} + \gamma_{\mu\delta,\beta} - \gamma_{\beta\delta,\mu}),$$

and consequently the ***3D Riemann tensor***

$$R^\alpha{}_{\beta\rho\sigma} = \partial_\rho \Gamma^\alpha{}_{\sigma\beta} - \partial_\sigma \Gamma^\mu{}_{\rho\beta} + \Gamma^\alpha{}_{\rho\lambda}\Gamma^\lambda{}_{\sigma\beta} - \Gamma^\alpha{}_{\sigma\lambda}\Gamma^\lambda{}_{\rho\beta},$$

the ***3D Ricci tensor***

$$R_{\alpha\beta} = R^\delta{}_{\alpha\delta\beta},$$

and the ***3D Ricci scalar***

$$R = R^\delta_\delta.$$

It is important to note that these are all tensors on Σ, in the sense that components that are on $\boldsymbol{n}$ are transformed into 0, but they are still expressed in the 4 dimensional basis, hence the Greek indices.

6.2. Extrinsic curvature

One may wonder where's the information of the 4D Riemann curvature tensor that is not present in our 3D one, that is, how does one explain the curvature of our spacetime manifold just by looking at our hypersurfaces $\{\Sigma_t\}_t$. We therefore need to study the way a hypersurface is embedded in a higher dimentional space.

For this, we define the ***extrinsic curvature tensor*** $K_{\alpha\beta}$, which describes how the hypersurface curves within the spacetime. It contains information about the rate of change of the spatial metric as one moves along the normal direction of the hypersurface, capturing all the bending, stretching, and shearing as it evolves within the spacetime.

Remembering the definition for the spatial projection operator $\gamma^\mu{}_\nu = \delta^\mu{}_\nu + n^\mu n_\nu$, we can project the components of a rank 2 tensor to its spatial projections in the following way:

$$\mathcal{P}T_{\alpha\beta} \coloneqq \gamma^\mu{}_\alpha \gamma^\nu{}_\beta T_{\mu\nu},$$

so the definition of the *extrinsic curvature tensor* is:

$$K_{\alpha\beta} \coloneqq -\mathcal{P}\nabla_\alpha n_\beta.$$

We apply this definition to calculate the value of this tensor.

$$\begin{aligned}
K_{\alpha\beta} = -\mathcal{P}\nabla_\alpha n_\beta &= -\gamma^\mu{}_\alpha h^\nu{}_\beta \nabla_\mu n_\nu \\
&= -[\delta^\mu{}_\alpha + n^\mu n_\alpha][\delta^\nu{}_\beta + n^\nu n_\beta]\nabla_\mu n_\nu \\
&= -[\delta^\mu{}_\alpha \delta^\nu{}_\beta + \delta^\mu{}_\alpha n^\nu n_\beta + n^\mu n_\alpha \delta^\nu{}_\beta + n^\mu n_\alpha n^\nu n_\beta]\nabla_\mu n_\nu \\
&= -[(\nabla_\alpha n_\beta) + n^\nu n_\beta(\nabla_\alpha n_\nu) + n^\mu n_\alpha(\nabla_\mu n_\beta) + n^\mu n^\nu n_\alpha n_\beta(\nabla_\mu n_\nu)] \\
&= -[\nabla_\alpha n_\beta + n^\mu n_\alpha(\nabla_\mu n_\beta) + \underbrace{n^\nu n_\beta(\nabla_\alpha n_\nu) + n^\mu n^\nu n_\alpha n_\beta(\nabla_\mu n_\nu)}_{\text{Let us work out this part}}].
\end{aligned}$$

We notice there's two $n^\nu(\nabla_\lambda n_\nu)$ terms on that this last sum. We can think of this as the normal projection of the covariant derivative of the normal vector. Since this vector is normalized, its derivative will always be tangential to hypersurface, so the normal projection will be zero. We can actually prove this:

Beginning with the fact that the normal vector is normalized, $n^\nu n_\nu = -1$.

$$\begin{aligned}
0 = \nabla_\lambda(n^\nu n_\nu) &= n^\nu \nabla_\lambda n_\nu + n_\nu \nabla_\lambda n^\nu \\
&= n^\nu \nabla_\lambda n_\nu + n_\nu \nabla_\lambda (g^{\mu\nu} n_\mu) \\
&= n^\nu \nabla_\lambda n_\nu + g^{\mu\nu} n_\nu \nabla_\lambda n_\mu \quad \text{(by metric compatibility)} \\
&= n^\nu \nabla_\lambda n_\nu + n^\mu \nabla_\lambda n_\mu \\
&= 2n^\nu \nabla_\lambda n_\nu \qquad\qquad \text{(by equating dummy indices).}
\end{aligned}$$

The above equations finally prove that $n^\nu \nabla_\lambda n_\nu = 0$.

This leads us to finally conclude that:

$$\begin{aligned}
K_{\alpha\beta} = -\mathcal{P}\nabla_\alpha n_\beta &= -h^\mu{}_\alpha h^\nu{}_\beta \nabla_\mu n_\nu \\
&= -[\nabla_\alpha n_\beta + n^\mu n_\alpha(\nabla_\mu n_\beta) + \underbrace{n^\nu n_\beta(\nabla_\alpha n_\nu) + n^\mu n^\nu n_\alpha n_\beta(\nabla_\mu n_\nu)}_{\text{Let us rearrange the relevant terms}}] \\
&= -[\nabla_\alpha n_\beta + n_\alpha n^\mu(\nabla_\mu n_\beta) + n_\beta \underbrace{n^\nu(\nabla_\alpha n_\nu)}_{=0} + n_\alpha n_\beta n^\mu \underbrace{n^\nu(\nabla_\mu n_\nu)}_{=0}] \\
&= -[\nabla_\alpha n_\beta + n_\alpha n^\mu(\nabla_\mu n_\beta)].
\end{aligned}$$

So our equation for the extrinsic curvature is:

$$K_{\alpha\beta} = -[\nabla_\alpha n_\beta + n_\alpha n^\mu(\nabla_\mu n_\beta)]. \tag{6.4}$$

It is clear that $K_{\alpha\beta}$ is a purely spatial tensor, that is, $n^\alpha K_{\alpha\beta} = 0$. It can also be shown to be a symmetric tensor. Combining these two facts, it is clear that

$$K_{00} = K_{0i} = 0.$$

Because of this, we will generally consider only spatial components K_{ij}.

6.2.1. Symmetry of the curvature tensor

We remember our extrinsic curvature tensor equation reads:

$$K_{\alpha\beta} = -\mathcal{P}\nabla_\alpha n_\beta = -(\nabla_\alpha n_\beta + n_\alpha n^\mu \nabla_\mu n_\beta),$$

where ∇_α represents the covariant derivative represented using Christoffel symbols as:

$$\nabla_\alpha n_\beta = \partial_\alpha n_\beta - \Gamma^\lambda{}_{\alpha\beta} n_\lambda.$$

Let us prove K_{ij} is a ***symmetric tensor***

Beginning with our formula for K_{ij}:

$$K_{ij} = -\nabla_i n_j + n_i n^\lambda \nabla_\lambda n_j.$$

We consider two non-zero vector fields $\mathcal{X}$ and $\mathcal{Y}$ tangent to our hypersurface. By definition, we notice that:

$$\begin{aligned}
n_c \mathcal{X}^c = g_{cd} n^c \mathcal{X}^c = g(\boldsymbol{n}, \mathcal{X}) = 0, \\
n_c \mathcal{Y}^c = g_{cd} n^c \mathcal{Y}^c = g(\boldsymbol{n}, \mathcal{Y}) = 0.
\end{aligned}$$

Then let us consider:

$$\begin{aligned}
\mathcal{X}^i \mathcal{Y}^j K_{ij} &= -\mathcal{X}^i \mathcal{Y}^j \nabla_i n_j + \mathcal{X}^i \mathcal{Y}^j n_i n^\lambda \nabla_\lambda n_j \\
&= -\mathcal{X}^i \mathcal{Y}^j \nabla_i n_j + \mathcal{Y}^j \underbrace{\mathcal{X}^i n_i}_{=0} n^\lambda \nabla_\lambda n_j \\
&= -\mathcal{X}^i \mathcal{Y}^j \nabla_i n_j.
\end{aligned}$$

We also use the fact that,

$$0 = \nabla_i(\mathcal{Y}^j n_j) = \mathcal{Y}^j \nabla_i n_j + n_j \nabla_i \mathcal{Y}^j,$$

to conclude that,

$$\mathcal{Y}^j \nabla_i n_j = -n_j \nabla_i \mathcal{Y}^j.$$

So our equation becomes,

$$\begin{aligned}
\mathcal{X}^i \mathcal{Y}^j K_{ij} &= -\mathcal{X}^i \mathcal{Y}^j \nabla_i n_j \\
&= \mathcal{X}^i n_j \nabla_i \mathcal{Y}^j \\
&= n_j \mathcal{X}^i \nabla_i \mathcal{Y}^j \\
&= n_j (\mathcal{X}^i \nabla_i \mathcal{Y})^j \\
&= g_{ij} n^j (\nabla_{\mathcal{X}} \mathcal{Y})^j \\
&= g(\boldsymbol{n}, \nabla_{\mathcal{X}} \mathcal{Y}).
\end{aligned}$$

Where we have used that the derivative of a vector v^i with respect to a vector field $\mathcal{Z}$ is $\nabla_{\mathcal{Z}} v^i = \mathcal{Z}^k \nabla_k v^i$.

Our previous calculations all lead us to our result:

$$\begin{aligned}
\mathcal{X}^i \mathcal{Y}^j \left(K_{ij} - K_{ji}\right) &= \mathcal{X}^i \mathcal{Y}^j K_{ij} - \mathcal{X}^i \mathcal{Y}^j K_{ji} \\
&= \mathcal{X}^i \mathcal{Y}^j K_{ij} - \mathcal{X}^j \mathcal{Y}^i K_{ij} \\
&= g\left(\boldsymbol{n}, \nabla_{\mathcal{X}} \mathcal{Y}\right) - g\left(\boldsymbol{n}, \nabla_{\mathcal{Y}} \mathcal{X}\right) \\
&= g\left(\boldsymbol{n}, \nabla_{\mathcal{X}} \mathcal{Y} - \nabla_{\mathcal{Y}} \mathcal{X}\right) \\
&= g\left(\boldsymbol{n}, [\mathcal{Y}, \mathcal{X}]\right) \\
&= 0,
\end{aligned} \tag{6.5}$$

since the commutator of $\mathcal{X}$ and $\mathcal{Y}$ is also on the hypersurface.

This last result (6.5), called the Frobenius theorem, is easy to establish:

$$\begin{aligned}
\nabla t \cdot [\mathcal{X}, \mathcal{Y}] &= \langle \boldsymbol{d}t, [\mathcal{X}, \mathcal{Y}] \rangle \\
&= \nabla_\mu t [\mathcal{X}^\nu \nabla_\nu \mathcal{Y}^\mu - \mathcal{Y}^\nu \nabla_\nu \mathcal{X}^\mu] \\
&= \nabla_\mu t \mathcal{X}^\nu \nabla_\nu \mathcal{Y}^\mu - \nabla_\mu t \mathcal{Y}^\nu \nabla_\nu \mathcal{X}^\mu \\
&= \mathcal{X}[\nabla_\nu (\underbrace{(\nabla_\mu) t \mathcal{Y}^\mu}_{=0}) - \mathcal{Y}^\mu \nabla_\nu \nabla_\mu t] - \mathcal{Y}^\nu [\nabla_\nu (\underbrace{(\nabla_\mu t) \mathcal{X}^\mu}_{=0}) - \mathcal{X}^\mu \nabla_\nu \nabla_\mu t] \\
&= \mathcal{X}^\mu \mathcal{Y}^\nu (\nabla_\nu \nabla_\mu t - \nabla_\mu \nabla_\nu t) = 0,
\end{aligned}$$

where the last equality is the result of our connection being torsion-free.

Using the fact that these vector fields are non-zero, we can conclude that the $(K_{ij} - K_{ji})$ term vanishes, that is, K_{ij} is symmetric.

6.3. Gauss-Codazzi relations

The next step is to work with our Einstein equation,

$$^{(4)}G_{\mu\nu} \equiv^{(4)} R_{\mu\nu} - \frac{1}{2}{}^{(4)}Rg_{\mu\nu} = 8\pi T_{\mu\nu},$$

for which we will decompose our 4-D Riemann tensor in terms of quantities relative to our spacelike hypersurface, namely the induced metric γ and extrinsic curvature tensor K.

For this, we define our **Gauss-Codazzi relations**, which constitute the basis for the 3+1 formalism of general relativity.

We start by defining the projection of a covariant derivative, which is a tensor field itself, therefore our previous calculations and definitions apply. Given a tensor field in T in Σ, we define its ***covariant derivative with respect to the connection of the metric*** $\boldsymbol{\gamma}$ as DT, as to be distinguished from ∇T which is the ***covariant derivative with respect to the spacetime connection of the metric*** g.

As explained in previous chapters, DT is expressible in terms of ∇T according to the formula:

$$DT = \vec{\gamma}\nabla T,$$

with a component version formula as given by:

$$D_\rho T^{\alpha_1\ldots\alpha_p}{}_{\beta_1\ldots\beta_q} = \gamma^{\alpha_1}{}_{\mu_1}\ldots\gamma^{\alpha_p}{}_{\mu_p}\gamma^{\nu_1}{}_{\beta_1}\ldots\gamma^{\nu_q}{}_{\beta_q}\gamma^\sigma_\rho\nabla_\sigma T^{\mu_1\ldots\mu_p}{}_{\nu_1\ldots\nu_q}.$$

6.3.1. Gauss relation

We begin considering the Ricci identity defining the three-dimensional Riemann tensor, which measures the difference of two covariant derivatives with respect to the three-dimensional connection D, associated with the three-dimensional metric γ on Σ. The four-dimensional version of this identity (where indices range from 0 to 3) is:

$$(D_\alpha D_\beta - D_\beta D_\alpha)v^\gamma \equiv R^\gamma{}_{\mu\alpha\beta}v^\mu.$$

where we will consider v to be a generic vector from the vector field tangent to Σ. Noticing that $D_\beta v^\gamma$ is a tensor itself, we can relate the D derivative to the ∇ derivate with our previous formula to get:

$$D_\alpha D_\beta v^\gamma = D_\alpha(D_\beta v^\gamma) = \gamma^\mu{}_\alpha\gamma^\nu{}_\beta\gamma^\gamma{}_\rho\nabla_\mu(D_\nu v^\rho) = \gamma^\mu{}_\alpha\gamma^\nu{}_\beta\gamma^\gamma{}_\rho\nabla_\mu(\gamma^\sigma{}_\nu\gamma^\rho{}_\lambda\nabla_\sigma v^\lambda).$$

We expand this formula by using that the derivative of the projection operator is,

$$\nabla_\nu\gamma^\sigma{}_\nu = \nabla_\mu(\delta^\sigma{}_\nu + n^\sigma n_\nu) = \nabla_\mu(n^\sigma n_\nu) = (\nabla_\mu n^\sigma)n_\nu + n^\sigma(\nabla_\mu n_\nu).$$

We will use the following facts about the projection operator:

- The projection operator is orthogonal to the normal vector,

$$\gamma^\nu{}_\beta n_\nu = \gamma^\nu{}_\beta n^\beta = 0.$$

- The projection operator is idempotent,

$$\gamma^\gamma{}_\rho\gamma^\rho{}_\lambda = \gamma^\gamma{}_\lambda.$$

- The explicit formula for the extrinsic curvature,

$$\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\nabla_{\mu}n_{\nu} = -K_{\alpha\beta}.$$

Let expand the aforementioned formula:

$$\begin{aligned}
D_{\alpha}D_{\beta}v^{\gamma} &= D_{\alpha}(D_{\beta}v^{\gamma}) \\
&= \gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\nabla_{\mu}(D_{\nu}v^{\rho}) = \gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\nabla_{\mu}(\gamma^{\sigma}{}_{\nu}\gamma^{\rho}{}_{\lambda}\nabla_{\sigma}v^{\lambda}) \\
&= \gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\left((\nabla_{\mu}\gamma^{\sigma}{}_{\nu})\,\gamma^{\rho}{}_{\lambda}\nabla_{\sigma}v^{\lambda} + \gamma^{\sigma}{}_{\nu}\,(\nabla_{\mu}\gamma^{\rho}{}_{\lambda})\,\nabla_{\sigma}v^{\lambda} + \gamma^{\sigma}{}_{\nu}\gamma^{\rho}{}_{\lambda}\left(\nabla_{\mu}\nabla_{\sigma}v^{\lambda}\right)\right) \\
&= \gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\Bigg(\Big[(\nabla_{\mu}n^{\sigma})n_{\nu} + \underbrace{n^{\sigma}(\nabla_{\mu}n_{\nu})}_{\gamma^{\nu}{}_{\beta}n_{\nu}=0}\Big]\gamma^{\rho}{}_{\lambda}\nabla_{\sigma}v^{\lambda} \\
&\qquad + \gamma^{\sigma}{}_{\nu}\Big[\underbrace{(\nabla_{\mu}n^{\rho})n_{\lambda}}_{\gamma^{\gamma}{}_{\rho}n^{\rho}=0} + n^{\rho}(\nabla_{\mu}n_{\lambda})\Big]\nabla_{\sigma}v^{\lambda} + \gamma^{\sigma}{}_{\nu}\gamma^{\rho}{}_{\lambda}\left(\nabla_{\mu}\nabla_{\sigma}v^{\lambda}\right)\Bigg) \\
&= \gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}(n^{\sigma}(\nabla_{\mu}n_{\nu})\gamma^{\rho}{}_{\lambda}\nabla_{\sigma}v^{\lambda} + \gamma^{\sigma}{}_{\nu}(\nabla_{\mu}n^{\rho})\,\underbrace{n_{\lambda}\nabla_{\sigma}v^{\lambda}}_{=-v^{\lambda}\nabla_{\sigma}n_{\lambda}} \\
&\qquad + \gamma^{\sigma}{}_{\nu}\gamma^{\rho}{}_{\lambda}\left(\nabla_{\mu}\nabla_{\sigma}v^{\lambda}\right)) \\
&= \gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\lambda}(\nabla_{\mu}n_{\nu})n^{\sigma}\nabla_{\sigma}v^{\lambda} \\
&\qquad - \gamma^{\mu}{}_{\alpha}\underbrace{\gamma^{\sigma}{}_{\beta}}_{=\gamma^{\sigma}{}_{\lambda}\gamma^{\lambda}{}_{\beta}}\gamma^{\gamma}{}_{\rho}v^{\lambda}(\nabla_{\mu}n^{\rho})\nabla_{\sigma}n_{\lambda} + \gamma^{\mu}{}_{\alpha}\gamma^{\sigma}{}_{\beta}\gamma^{\gamma}{}_{\lambda}\nabla_{\mu}\nabla_{\sigma}v^{\lambda} \\
&= \gamma^{\gamma}{}_{\lambda}\underbrace{\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}(\nabla_{\mu}n_{\nu})}_{-K_{\alpha\beta}}n^{\sigma}\nabla_{\sigma}v^{\lambda} \\
&\qquad - \underbrace{\gamma^{\mu}{}_{\alpha}\gamma^{\gamma}{}_{\rho}\nabla_{\mu}n^{\rho}}_{=-K_{\alpha}{}^{\gamma}}\underbrace{\gamma^{\sigma}{}_{\lambda}\gamma^{\lambda}{}_{\beta}\nabla_{\sigma}n_{\lambda}}_{=-K_{\beta\lambda}}v^{\lambda} + \gamma^{\mu}{}_{\alpha}\gamma^{\sigma}{}_{\beta}\gamma^{\gamma}{}_{\lambda}\nabla_{\mu}\nabla_{\sigma}v^{\lambda} \\
&= -K_{\alpha\beta}\gamma^{\gamma}{}_{\lambda}n^{\sigma}\nabla_{\sigma}v^{\lambda} - K^{\gamma}{}_{\alpha}K_{\beta\lambda}v^{\lambda} + \gamma^{\mu}{}_{\alpha}\gamma^{\sigma}{}_{\beta}\gamma^{\gamma}{}_{\lambda}\nabla_{\mu}\nabla_{\sigma}v^{\lambda}.
\end{aligned}$$

To form $D_{\alpha}D_{\beta}v^{\gamma} - D_{\beta}D_{\alpha}v^{\gamma}$, we can check the first term vanishes by the symmetry of $K_{\alpha\beta}$, so then what remains is,

$$D_{\alpha}D_{\beta}v^{\gamma} - D_{\beta}D_{\alpha}v^{\gamma} = (K_{\alpha\mu}K^{\gamma}{}_{\beta} - K_{\beta\mu}K^{\gamma}{}_{\alpha})\,v^{\mu} + \gamma^{\rho}{}_{\alpha}\gamma^{\sigma}{}_{\beta}\gamma^{\gamma}{}_{\lambda}(\nabla_{\rho}\nabla_{\sigma}v^{\lambda} - \nabla_{\sigma}\nabla_{\rho}v^{\lambda}).$$

Now, we have that for our 4-dimensional connection, the Ricci identity reads:

$$(\nabla_{\rho}\nabla_{\sigma} - \nabla_{\sigma}\nabla_{\rho})v^{\lambda} = {}^{(4)}R^{\lambda}{}_{\mu\rho\sigma}v^{\mu}.$$

Therefore,

$$D_{\alpha}D_{\beta}v^{\gamma} - D_{\beta}D_{\alpha}v^{\gamma} = (K_{\alpha\mu}K^{\gamma}{}_{\beta} - K_{\beta\mu}K^{\gamma}{}_{\alpha})\,v^{\mu} + \gamma^{\rho}{}_{\alpha}\gamma^{\sigma}{}_{\beta}\gamma^{\gamma}{}_{\lambda}({}^{(4)}R^{\lambda}{}_{\mu\rho\sigma}v^{\mu}).$$

Ultimately, what we are looking for is a formula for the projection of ${}^{(4)}R^{\lambda}{}_{\mu\rho\sigma}$ on all its indices. For this, we can use that $v^{\mu} = \gamma^{\mu}{}_{\sigma}v^{\sigma}$ to introduce the projection operator, but this will change the index over v, and we need them all to be equal. Rearranging free indices so all v's have the same upper index, we arrive to the equation:

$$\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\gamma^{\sigma}{}_{\lambda}{}^{(4)}R^{\rho}{}_{\sigma\mu\nu}v^{\lambda} = R^{\gamma}{}_{\lambda\alpha\beta}v^{\lambda} + (K^{\gamma}{}_{\alpha}K_{\lambda\beta} - K^{\gamma}{}_{\beta}K_{\alpha\lambda})v^{\lambda}.$$

Since v can be replaced by any vector on $\mathcal{T}_p(\mathcal{M})$, we arrive to the **Gauss relation**, which relates

the projection of the 4-dimensional Riemann curvature tensor (the one induced by the metric g) to the 3-dimensional Riemann curvature tensor (induced by the metric γ) and the extrinsic curvature:

$$\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\gamma^{\sigma}{}_{\lambda}{}^{(4)}R^{\rho}{}_{\sigma\mu\nu} = R^{\gamma}{}_{\lambda\alpha\beta} + (K^{\gamma}{}_{\alpha}K_{\lambda\beta} - K^{\gamma}{}_{\beta}K_{\alpha\lambda}). \tag{6.6}$$

6.3.2. Codazzi relation

We now derive a formula for the 3 projections in space and 1 projection in time of the 4-dimensional Riemann tensor. For this, we consider the 4-dimensional Ricci identity applied to the normal vector $\boldsymbol{n}$:

$$(\nabla_{\alpha}\nabla_{\beta} - \nabla_{\beta}\nabla_{\alpha})n^{\gamma} = {}^{(4)}R^{\gamma}{}_{\mu\alpha\beta}n^{\mu}.$$

Projecting this relation onto Σ, we get:

$$\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}{}^{(4)}R^{\rho}{}_{\sigma\mu\nu}n^{\sigma} = \gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}(\nabla_{\mu}\nabla_{\nu}n^{\rho} - \nabla_{\nu}\nabla_{\mu}n^{\rho}).$$

We will begin projecting the first term on the right hand side of the equation. For this, we will define $\boldsymbol{a} = \nabla_{\boldsymbol{n}}\boldsymbol{n}$, and use an equation that extends the definition of the extrinsic curvature to vectors defined on $\mathcal{T}_p(\mathcal{M})$ as $K_{\alpha\beta}$ (in contrast to our usual definition, that only works for vectors on $\mathcal{T}_p(\Sigma)$ denoted by K_{ij}:

$$\nabla_{\beta}n_{\alpha} = -K_{\alpha\beta} - a_{\alpha}n_{\beta}.$$

Using this last definition, we get:

$$\begin{aligned}
\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\nabla_{\mu}\nabla_{\nu}n^{\rho} &= \gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\nabla_{\mu}(-K^{\rho}{}_{\nu} - a^{\rho}n_{\nu}) \\
&= -\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}(\nabla_{\mu}K^{\rho}{}_{\nu} + \underbrace{(\nabla_{\mu}a^{\rho})n_{\nu}}_{\gamma^{\nu}{}_{\beta}n_{\nu}=0} + a^{\rho}(\nabla_{\mu}n_{\nu}) \\
&= -D_{\alpha}K^{\gamma}{}_{\beta} + a^{\gamma}K_{\alpha\beta},
\end{aligned}$$

where we have used the definition of K to get $\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\nabla_{\mu}n_{\nu} = -K_{\alpha\beta}$.

By permutating indices α and β, and using the symmetry of K, we get the **Codazzi relation**, which relates the mixed projection of the 4-dimensional Riemann tensor to the 3-derivatives of the extrinsic curvature:

$$\gamma^{\gamma}{}_{\rho}n^{\sigma}\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}{}^{(4)}R^{\rho}{}_{\sigma\mu\nu} = D_{\beta}K^{\gamma}{}_{\alpha} - D_{\alpha}K^{\gamma}{}_{\beta}. \tag{6.7}$$

6.4. Lie Derivatives

In numerical relativity, a useful strategy for simplifying the field equations system is to focus on solutions with some kind of symmetry. By adapting the coordinate system to a given symmetry of the solution, we can reduce the number of relevant coordinates, thus making our model much easier to handle.
For example, in the case of axial symmetry, we can take the azimuthal angle ϕ to be one of our four spacetime coordinates. In this adapted coordinate system ones has

$$\partial_\phi g_{\mu\nu} = 0,$$

thus making all our relevant tensors be also symmetrical with respect to this parameter.

From a group-theoretical point of view, we can identify ϕ with the parameter labelling a continuous group of transformations, usually known as a 'Lie group'. These transformations are interpreted as mappings from a spacetime point P into a continuous set of points. These set of points define an **orbit** of the group.

We recall that the Einstein Equations are tensor equations, thus we need to explore ways to describe the evolution of these tensors along important spacetime curves, in particular, we want to study the change of these tensor fields along orbits of symmetry parameters as described in our spacetime models. This means finding a way of expressing derivatives not along certain parameters, but along curves defined in a way that help us reduce the difficulty of our equations. In this way we start defining the **Lie derivatives**.

6.4.1. Derivation of the Lie derivative formula

We consider a manifold on which some vector field, $u^i(x)$ is defined, as seen in Fig. 6.1. The integral curves of this vector field are given by $x^i(\lambda, \mu)$, such that at all points they have a tangent vector given by:

$$\frac{\partial x^i}{\partial \lambda} = u^i(x).$$

Clearly each point of the manifold has only one curve passing through it, thus we say that the curves form a congruence.

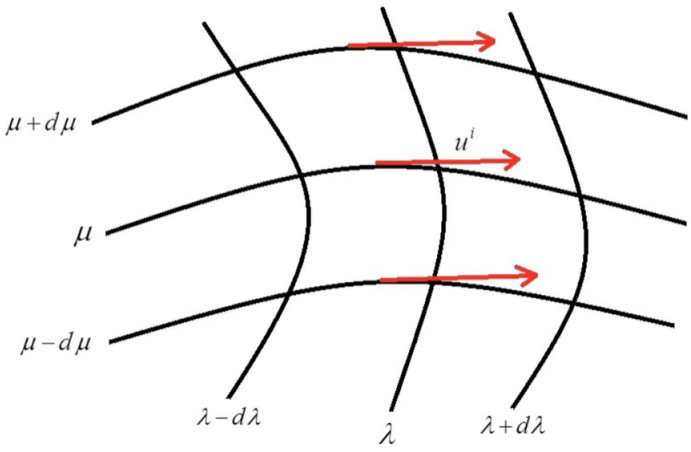

Figura 6.1: Figure illustrating the congruence of our curves, all defined by our tangent vectors.

We now define a curve $\boldsymbol{C}$ with parametrization $x^i(\lambda)$ and tangent vector,

$$\xi^i = \frac{dx^i}{d\lambda},$$

and two points on $\boldsymbol{C}$ that are infinitesimally separated, $\boldsymbol{P}(x)$ and $\boldsymbol{Q}(x+dx)$. Fig. 6.2 illustrates this nicely, where the curve's tangent vectors defined by $u^i(x)$ lie along $x^i(\lambda)$. In this case u^i is treated as a coordinate in our space.

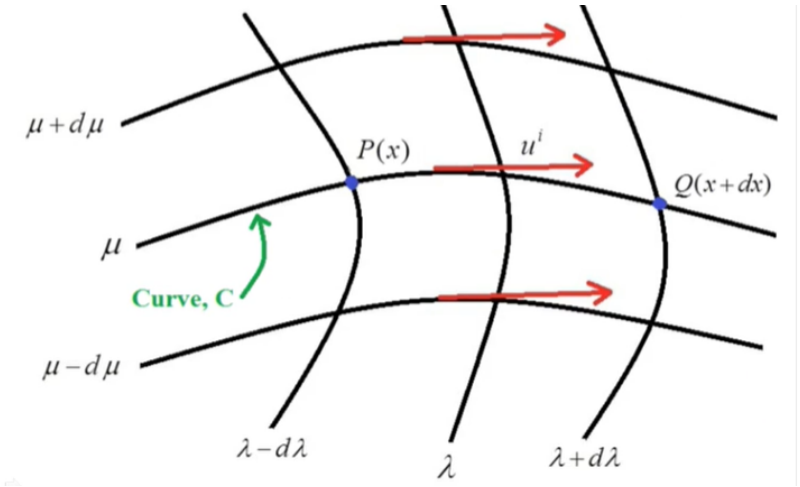

Figura 6.2: Two points along our μ coordinate defining a curve $\boldsymbol{C}$

We can now define another vector field $v^i(x)$ in the region of the curve $\boldsymbol{C}$. We will evaluate how this vector $v^i(x)$ changes as it moves along the curve $\boldsymbol{C}$. An example is given in 6.3, where a specific vector at $\boldsymbol{P}$ is distorted on our whole space when changing its origin to an arbitrary point $\boldsymbol{Q}$. This change is of course studied infinitesimally, hence $\boldsymbol{Q}$ is located at $x + dx$.

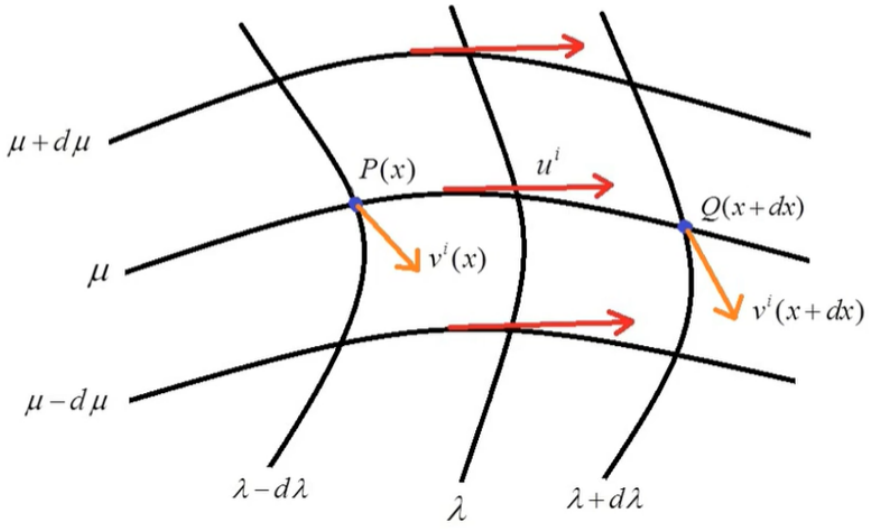

Figura 6.3: We study the way a vector field changes from $\boldsymbol{P}$ to $\boldsymbol{Q}$.

First, we consider what happens to the vector at the point $\boldsymbol{Q}$ in terms of its value at the points

$\boldsymbol{P}$. We do this by considering a new set of primed coordinates x' which represent an infinitesimal coordinate transformation. That is, from $\mathbf{P}(x)$ to $\mathbf{Q}(x+dx) = \mathbf{Q}(x')$, we have:

$$x'^i = x^i + dx^i = x^i + \frac{dx^i}{d\lambda} d\lambda = x^i + \xi^i d\lambda.$$

Now, tensor transformation law forces:

$$v'^i(x') = \frac{\partial x'^i}{\partial x^j} v^j(x).$$

Combining these two results, we get:

$$\begin{aligned} v'^i(x') &= \frac{\partial x'^i}{\partial x^j} v^j(x) \\ &= (\frac{\partial x^i}{\partial x^j} + \frac{\partial \xi^i}{\partial x^j} d\lambda) v^j(x) \\ &= (\delta^i{}_j + \partial_j \xi^i d\lambda) v^j(x) \\ &= v^i(x) + \partial_j \xi^i v^j(x) d\lambda. \end{aligned}$$

Since this new point x'^i represents the point $\boldsymbol{Q}$ on the curve, we can write:

$$v'^i(Q) = v^i(P) + \partial_j \xi^i v^j(P) d\lambda.$$

Now, we can do a first order Taylor expansion to express v^i at $\mathbf{Q}$:

$$\begin{aligned} v^i(Q) = v^i(x+dx) &\approx v^i(x) + dx^j \partial_j v^i(x) \\ &\approx v^i(P) + \xi^j \partial_j v^i(P) d\lambda. \end{aligned}$$

Finally, we define the Lie derivative as the infinitesimal change of v^i as it moves from $\boldsymbol{P}$ to $\boldsymbol{Q}$. That is, the change between $v'^i(Q)$ and $v^i(Q)$.

$$\begin{aligned} \mathcal{L}_{\vec{\boldsymbol{\xi}}} v^i(P) &= \lim_{d\lambda \to 0} \frac{v^i(Q) - v'^i(Q)}{d\lambda} \\ &= \lim_{d\lambda \to 0} \frac{v'^i(Q) = v^i(P) + \partial_j \xi^i v^j(P) d\lambda - (v^i(P) + \xi^j \partial_j v^i(P) d\lambda)}{d\lambda} \\ &= \xi^j \partial_j v^i(P) - v^j(P) \partial_j \xi^i. \end{aligned}$$

Finally, considering the equation for the covariant derivative of a vector reads:

$$\nabla_j w^i = \partial_j w^i - \Gamma^i{}_{jk} w^i \iff \partial_j w^i = \nabla_j w^i + \Gamma^i{}_{jk} w^i.$$

We notice that replacing this in our Lie derivative equation, the Christoffel symbols cancel out thanks to a dummy index rearranging and using the fact that the connection is, in this case,

symmetric on both lower indices:

$$\begin{aligned}\mathcal{L}_{\vec{\boldsymbol{\xi}}} v^i(P) &= \xi^j \partial_j v^i(P) - v^j(P)\partial_j \xi^i \\ &= \xi^j(\nabla_j v^i(P) + \Gamma^i{}_{jk} v^k(P)) - v^j(P)(\nabla_j \xi^i + \Gamma^i{}_{jk}\xi^k) \\ &= \xi^j \nabla_j v^i(P) - v^j(P)\nabla_j \xi^i + (\xi^j v^k(P)\Gamma^i{}_{jk} - \xi^k v^j(P)\Gamma^i{}_{jk}) \\ &= \xi^j \nabla_j v^i(P) - v^j(P)\nabla_j \xi^i + (\xi^j v^k(P)\Gamma^i{}_{jk} - \xi^j v^k(P)\Gamma^i{}_{kj}) \\ &= \xi^j \nabla_j v^i(P) - v^j(P)\nabla_j \xi^i + (\xi^j v^k(P))(\Gamma^i{}_{jk} - \Gamma^i{}_{kj}) \\ &= \xi^j \nabla_j v^i(P) - v^j(P)\nabla_j \xi^i.\end{aligned}$$

With this in mind, one can show that given a vector field $\boldsymbol{\xi}$, we can compute the Lie derivative for any tensor:

- For a scalar function ϕ:
$$\mathcal{L}_{\vec{\boldsymbol{\xi}}}\phi = \xi^\mu \nabla_\mu \phi = \xi^\mu \partial_\mu \phi,$$
since the covariant derivative reduces to a normal derivative on scalars.
- For a V^μ vector field, the Lie derivative is given by the **commutator**:
$$\mathcal{L}_{\vec{\boldsymbol{\xi}}} V^\mu = \xi^\nu \nabla_\nu V^\nu - V^\nu \nabla_\nu \xi^\mu = [\boldsymbol{\xi}, \boldsymbol{V}]^\mu.$$
- For a ω_μ 1-form:
$$\mathcal{L}_{\vec{\boldsymbol{\xi}}}\omega_\mu = \xi^\nu \nabla_\nu \omega_\mu + \omega_\nu \nabla_\mu \xi^\nu.$$
- For a generic $T^\mu{}_\nu$ tensor of (1,1) rank:
$$\mathcal{L}_{\vec{\boldsymbol{\xi}}} T^\mu{}_\nu = \xi^\lambda \nabla_\lambda T^\mu{}_\nu - T^\lambda{}_\nu \nabla_\lambda \xi^\mu + T^\mu{}_\lambda \nabla_\nu \xi^\lambda.$$

In particular, for the metric tensor we have:

$$\mathcal{L}_{\vec{\boldsymbol{\xi}}} g_{\mu\nu} = \xi^\lambda \partial_\lambda g_{\mu\nu} + g_{\mu\lambda}\partial_\nu \xi^\lambda + g_{\lambda\nu}\partial_\mu \xi^\lambda.$$

Likewise, we have:

$$\begin{aligned}\nabla_\mu \xi_\nu + \nabla_\nu \xi_\mu &= \partial_\mu \xi_\nu + \partial_\nu \xi_\mu - 2\Gamma^\lambda{}_{\mu\nu}\xi_\lambda \\ &= \partial_\mu \xi_\nu + \partial_\nu \xi_\mu + \xi^\lambda \partial_\lambda g_{\mu\nu} - \xi^\lambda \partial_\mu g_{\nu\lambda} - \xi^\lambda \partial_\nu g_{\mu\lambda} \\ &= \xi^\lambda \partial_\lambda g_{\mu\nu} + g_{\nu\lambda}\partial_\mu \xi^\lambda + g_{\mu\lambda}\partial_\nu \xi^\lambda,\end{aligned}$$

from which we find that:

$$\mathcal{L}_{\vec{\boldsymbol{\xi}}} g_{\mu\nu} = \nabla_\mu \xi_\nu + \nabla_\nu \xi_\mu.$$

6.5. Lapse function and foliation kinematics

We have previously defined Σ_t as the set of points for which a scalar field t is constant. With this, the unit vector $\boldsymbol{n}$ normal to our hypersurface Σ_t is collinear to $\vec{\nabla}t$, for which we may write:

$$\boldsymbol{n} := -N\vec{\nabla}t,$$

where

$$N := (-\vec{\nabla}t \cdot \vec{\nabla}t)^{-1/2}.$$

The value of N just ensures $\boldsymbol{n}$ is an unit vector, and the sign ensures that $\boldsymbol{n}$ is future-oriented. The scalar field N will be called from now on the **lapse function**.

6.5.1. Normal evolution vector

It is certainly useful to have an expression for an unit vector given any scalar field generating it, but we may also need to adapt this vector so it successfully takes points from one slice to another, as these may be differently separated for different scalar fields. For this we define the ***normal evolution vector***:

$$\boldsymbol{m} := N\boldsymbol{n}.$$

An important fact about this vector is that one has

$$\langle \boldsymbol{d}t, \boldsymbol{m}\rangle = N\langle \boldsymbol{d}t, \boldsymbol{n}\rangle = N^2 \underbrace{(-\langle \boldsymbol{d}t, \vec{\nabla}t\rangle)}_{=N^{-2}} = 1$$

A geometrical consequence of this relation is that one can move between two hypersurfaces Σ_t and $\Sigma_{t+\delta t}$ by a small displacement of $\delta t\boldsymbol{m}$ of each point of Σ_t. Indeed, let us consider a point p of Σ_t and displace it infinitesimally by $\delta t\boldsymbol{m}$ to get the point $p' = p + \delta t\boldsymbol{m}$, then

$$\begin{aligned} t(p') &= t(p + \delta t\boldsymbol{m}) = t(p) + \langle \boldsymbol{d}t, \delta t\boldsymbol{m}\rangle = t(p) + \delta t \underbrace{\langle \boldsymbol{d}t, \boldsymbol{m}\rangle}_{=1} \\ &= t(p) + \delta t, \end{aligned}$$

so then $p' \in \Sigma_{t+\delta t}$.

6.6. The shift vector

We have the illusion of living in a 3D space, where its easy to tell computer to perform step by step simulations. So our recipe involves decomposing our 4 dimensional manifold into space-like 3-dimensional hypersurfaces. After this, what we will need is to transform all tensorial equations (valid for any coordinates) into PDEs easy to solve on our computers, and for this we will need to introduce a coordinate system on our manifold.

Given our spacetime manifold $\mathcal{M}$, described by a 4-metric $g_{\mu\nu}$, we foliate it via our previously described hypersurfaces $\{\Sigma_t\}$, where t is now the parameter that defines $\boldsymbol{\partial}_t = \frac{\partial}{\partial t}$. This vector is now not necessarily normal to our hypersurfaces, but rather a vector tangent to the lines of constant spatial coordinates. We will need these We also define our spatial coordinates $(x^i) = (x^1, x^2, x^3)$ and their corresponding basis vectors $\boldsymbol{\partial}_i = \frac{\partial}{\partial x^i}$. These vectors $(\boldsymbol{\partial}_\alpha) = (\boldsymbol{\partial}_t, \boldsymbol{\partial}_i)$ denote the natural basis of $\mathcal{T}_p(\mathcal{M})$ associated with coordinates (x^α).

One can check that this $\boldsymbol{\partial}_t$ vector is also dual to $\boldsymbol{dt}$, as is the normal evolution vector $\boldsymbol{m}$. This means that $\langle \boldsymbol{dt}, \boldsymbol{\partial}_t \rangle = \langle \boldsymbol{dt}, \boldsymbol{m} \rangle = 1$, so $\boldsymbol{\partial}_t$ also drags the hypersurfaces Σ_t as $\boldsymbol{m}$ does. These vectors, however, often differ and only coincide if the lines $x^i =$ constant are orthogonal to the hypersurfaces Σ_t.

The difference between $\boldsymbol{\partial}_t$ and $\boldsymbol{m}$ is called the ***shift vector*** and will be denoted by $\boldsymbol{\beta}$:

$$\boldsymbol{\partial}_t = \boldsymbol{m} + \boldsymbol{\beta}.$$

Clearly $\langle \boldsymbol{dt}, \boldsymbol{\beta} \rangle = 0$, so then $\boldsymbol{\beta}$ is tangent to Σ_t. Since we are most interested in the normal vector, it is often useful to rewrite our equation in terms of $\boldsymbol{n}$:

$$\boldsymbol{\partial}_t = N\boldsymbol{n} + \boldsymbol{\beta}.$$

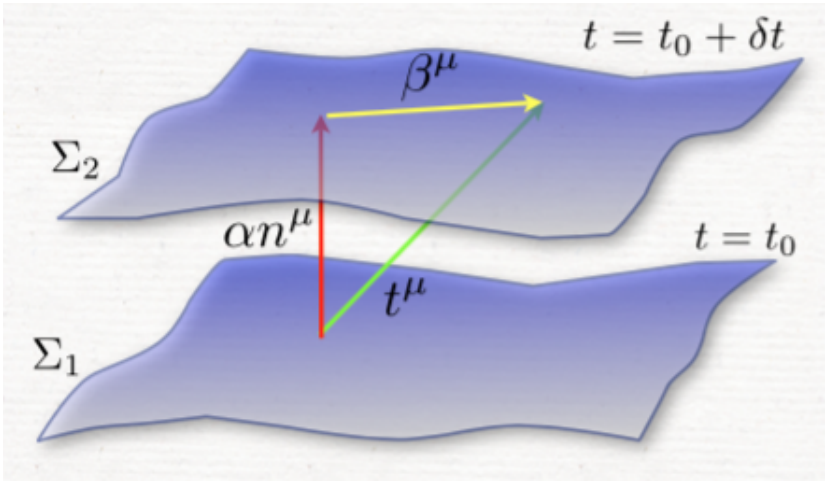

Figura 6.4: Representation of our lapse function and normal coordinate (red), our time coordinate (green), and our shift vector (yellow).

Let us start remembering that ***proper time*** between two events is the time as measured by a clock moving with an observer between these two events. We denote the proper time as τ. The lapse function N now measures proper time between two hypersurfaces by traversing them along the coordinate t.

$$d\tau = N(t, x^i)dt.$$

We now consider a specific foliation and take two adjacent hypersurfaces Σ_t and Σ_{t+dt}. The geometry of the region between these two hypersurfaces can be now determined from three ingredients:

- The three-dimensional metric γ_{ij} which measures proper distances within the hypersurface itself:
$$dl^2 = \gamma_{ij} dx^i dx^j.$$
- The lapse function $N(t, x^i)$ which measures proper time elapsed by an observer moving along the direction normal to the hypersurface (an Eulerian observer):
$$d\tau = N(t, x^i) dt.$$
- The relative velocity vector $\beta^i(t, x^j)$, also called the ***shift vector*** between Eulerian observers and lines that correspond to constant spatial coordinates:
$$x^i_{t+dt} = x^i_t - \beta^i(t, x^j) dt.$$

These elements constitute the ***ADM formalism*** of numerical relativity. Let us see how to write the space-time line element in terms of the new ADM variables. The line element between two events $p(x^i, t)$ and $p'(x^i + dx^i, t + dt)$, can be decomposed into two parts: the square of the distance over the hypersurface containing p, minus the square of the proper time between hypersurfaces:

$$\begin{aligned} ds^2 &= \gamma_{ij}(dx^i + \beta^i dt)(dx^j + \beta^j dt) - (N dt)^2 \\ &= (-N + \beta^i \beta_i) dt^2 + 2\beta_i dx^i dt + \gamma_{ij} dx^i dx^j, \end{aligned}$$

where $\beta_i = \gamma_{ij} \beta^j$.

We can calculate the spacetime metric and its inverse using this line element:

$$\begin{aligned} g_{\mu\nu} &= \begin{bmatrix} -N^2 + \beta_k \beta^k & \beta_i \\ \beta_j & \gamma_{ij} \end{bmatrix}, \\ g^{\mu\nu} &= \begin{bmatrix} -1/N^2 & \beta^i/N^2 \\ \beta^j/N^2 & \gamma^{ij} - \beta^i \beta^j / N^2 \end{bmatrix}. \end{aligned}$$

one can check that our normal vector in the ADM formalism satisfies:

$$\begin{aligned} n^\mu &= \frac{1}{N} t^\mu - \frac{1}{N} \beta^\mu \\ &= \frac{1}{N}(1, \vec{0}) - \frac{1}{N}(0, \beta^i) \\ &= (1/N, -\beta^i/N). \end{aligned}$$

Similarly, lowering our index with our metric $g_{\mu\nu}$, it is easy to show that:

$$n_\mu = (-N, \vec{0}).$$

It is easy to see that this vector is also normalized, so we can summarize its components below:

$$n^\mu = (1/N, -\beta^i/N), \quad n_\mu = (-N, 0), \quad n^\mu n_\mu = -1.$$

6.7. Evolution of the 3D metric

Substituting the explicit form of the normal vector $n^\mu = \left(\frac{1}{N}, -\frac{\beta^i}{N}\right)$ and $n_\mu = \left(-N, \vec{0}\right)$, plus our formula for the covariant derivative in terms of Christoffel symbols in the definition of the extrinsic curvature, we get:

$$\begin{aligned} K_{ij} &= -\left(\nabla_i n_j + n_i n^\mu \nabla_\mu n_j\right) \\ &= -\left(\left[\partial_i n_j - \Gamma^\lambda{}_{ij} n_\lambda\right] + n_i n^\mu \left[\partial_\mu n_j - \Gamma^\lambda{}_{\mu j} n_\lambda\right]\right) \\ &= \Gamma^0{}_{ij} n_0 \\ &= -N\Gamma^0{}_{ij}. \end{aligned}$$

The term $\Gamma^0{}_{ij}$ can be easily computed from the metric:

$$\begin{aligned} \Gamma^0{}_{ij} &= \frac{1}{2} g^{0\mu}\left(\partial_i g_{\mu j} + \partial_j g_{i\mu} - \partial_\mu g_{ij}\right) \\ &= \frac{1}{2}\left[g^{00}\left(\partial_i g_{0j} + \partial_j g_{i0} - \partial_0 g_{ij}\right) + g^{0k}\left(\partial_i g_{kj} + \partial_j g_{ik} - \partial_k g_{ij}\right)\right] \\ &= \frac{1}{2}\left[-\frac{1}{N^2}\left(\partial_i \beta_j + \partial_j \beta_i - \partial_t \gamma_{ij}\right) + \frac{\beta^k}{N^2}\left(\partial_i \gamma_{kj} + \partial_j \gamma_{ik} - \partial_k \gamma_{ij}\right)\right] \\ &= \frac{1}{2N^2}\left[\left(\partial_t \gamma_{ij} - \partial_i \beta_j - \partial_j \beta_i\right) + \beta_l q^{lk}\left(\partial_i \gamma_{kj} + \partial_j \gamma_{ik} - \partial_k \gamma_{ij}\right)\right] \\ &= \frac{1}{2N^2}\left[\left(\partial_t \gamma_{ij} - \partial_i \beta_j - \partial_j \beta_i\right) + 2\beta_l \Gamma^l{}_{ij}\right] \\ &= \frac{1}{2N^2}\left(\partial_t \gamma_{ij} - D_i \beta_j - D_J \beta_i\right). \end{aligned}$$

Therefore, our equation for the extrinsic curvature using the ADM variables is:

$$K_{ij} = \frac{1}{2N}\left(-\partial_t \gamma_{ij} + D_i \beta_j + D_j \beta_i\right).$$

This gives us an evolution equation for the 3-dimensional metric:

$$\partial_t \gamma_{ij} = -2NK_{ij} + \nabla_i \beta_j + \nabla_j \beta_i.$$

Remembering our equation for the Lie derivative of the 4-dimensional metric tensor,

$$\mathcal{L}_{\vec{\boldsymbol{\xi}}} g_{\mu\nu} = \nabla_\mu \xi_\nu - \nabla_\nu \xi_\mu,$$

we can derive the same result for the derivative of the induced 3-dimensional metric tensor:

$$\mathcal{L}_{\vec{\boldsymbol{\xi}}} \gamma_{ij} = D_i \xi_j - D_j \xi_i.$$

This means that our previous equation reads:

$$\begin{aligned} & \partial_t \gamma_{ij} = -2NK_{ij} + \mathcal{L}_{\vec{\boldsymbol{\beta}}} \gamma_{ij}, \\ \Longrightarrow\ & (\partial_t - \mathcal{L}_{\vec{\boldsymbol{\beta}}})\gamma_{ij} = -2NK_{ij}, \\ \Longrightarrow\ & (\mathcal{L}_{\boldsymbol{t}} - \mathcal{L}_{\vec{\boldsymbol{\beta}}})\gamma_{ij} = -2NK_{ij}, \qquad \text{since } \partial_t = \mathcal{L}_{\boldsymbol{t}} \\ \Longrightarrow\ & \mathcal{L}_{N\boldsymbol{n}} \gamma_{ij} = -2NK_{ij}, \qquad \text{since } \boldsymbol{t} - \boldsymbol{\beta} = N\boldsymbol{n} \\ \Longrightarrow\ & \mathcal{L}_{\boldsymbol{m}} \gamma_{ij} = -2NK_{ij}, \qquad \text{since } \boldsymbol{m} = N\boldsymbol{n}. \end{aligned} \tag{6.8}$$

6.8. Ricci equation

We have worked out our projection of the spacetime Riemann tensor fully onto Σ_t, yielding the Gauss equation (6.6). We have also projected three times onto Σ_t and once along $\boldsymbol{n}$, yielding the Codazzi equation (6.7). These decompositions involve only fields tangent to Σ_t and derivatives in directions parallel to Σ_t, so it was natural to define them for a single hypersurface. We will now form a projection of the spacetime Riemann tensor twice onto Σ_t and twice along $\boldsymbol{n}$. This will be the last non trivial projection of the spacetime Riemann tensor.

To begin our analysis, we will first derive some important relations. Taking our previously defined $\boldsymbol{a} = \nabla_{\boldsymbol{n}}\boldsymbol{n}$ vector, let us work out its components.

$$\begin{aligned}
a_\mu &= n^\lambda \nabla_\lambda n_\mu = -n^\lambda \nabla_\lambda (N \nabla_\mu t) \\
&= -n^\lambda (\nabla_\lambda N) \underbrace{(\nabla_\mu t)}_{=-n_\mu/N} - N n^\lambda \underbrace{\nabla_\lambda \nabla_\mu t}_{=\nabla_\mu \nabla_\lambda t} \\
&= \frac{1}{N} n_\mu \nabla^\lambda \nabla_\lambda N + N n^\lambda \nabla_\mu (-\frac{1}{N} n_\lambda) \\
&= \frac{1}{N} n_\mu n^\lambda \nabla_\lambda N + N n^\lambda ((-\frac{1}{N}) \nabla_\mu n_\lambda + n_\lambda \nabla_\mu (-\frac{1}{N})) \\
&= \frac{1}{N} n_\mu n^\lambda \nabla_\lambda N + N n^\lambda (-\frac{1}{N}) \nabla_\mu n_\lambda + n^\lambda n_\lambda \underbrace{N \nabla_\mu (-\frac{1}{N})}_{-\frac{1}{N}\nabla_\mu N} \\
&= \frac{1}{N} n_\mu n^\lambda \nabla_\lambda N - \underbrace{n^\lambda \nabla_\mu n_\lambda}_{=0} - \frac{1}{N} (\nabla_\mu N) \underbrace{n^\lambda n_\lambda}_{=-1} \\
&= \frac{1}{N} (\nabla_\mu N + n^\mu n^\lambda \nabla_\lambda N) \\
&= \frac{1}{N} \gamma^\lambda{}_\mu \nabla_\lambda N \\
&= \frac{1}{N} D_\mu N \\
&= D_\mu \ln N.
\end{aligned}$$

Therefore, using our formula that relates the 4 dimensional extrinsic curvature to the spacetime derivative of the normal vector (6.4), we can replace the **a** vector to get:

$$\begin{aligned}
\nabla_\beta n_\alpha &= -K_{\alpha\beta} - a_\alpha n_\beta \\
&= -K_{\alpha\beta} - D_\alpha \ln N n_\beta.
\end{aligned}$$

Having derived this equation, we can now take the 4 dimensional Ricci identity, and project twice onto Σ_t and once along $\boldsymbol{n}$:

$$\gamma_{\alpha\mu} n^\sigma \gamma^\nu{}_\beta (\nabla_\nu \nabla_\sigma n^\mu - \nabla_\sigma \nabla_\nu n^\mu) = \gamma_{\alpha\mu} n^\sigma \gamma^\nu{}_\beta {}^{(4)}R^\mu{}_{\rho\nu\sigma} n^\rho.$$

We can now use our derived equation to further work with these derivatives:

$$\begin{aligned}
\gamma_{\alpha\mu}n^\sigma\gamma^\nu{}_\beta{}^{(4)}R^\mu{}_{\rho\nu\sigma}n^\rho &= \gamma_{\alpha\mu}n^\sigma\gamma^\nu{}_\beta[\nabla_\nu(-K^\mu{}_\sigma - D^\mu\ln Nn_\sigma) - \nabla_\sigma(-K^\mu{}_\nu - D^\mu\ln Nn_\nu)] \\
&= \gamma_{\alpha\mu}n^\sigma\gamma^\nu{}_\beta[- \underbrace{\nabla_\nu K^\mu{}_\sigma}_{n^\sigma\nabla_\nu K^\mu{}_\sigma = -K^\mu{}_\sigma\nabla_\nu n^\sigma} - (\underbrace{\nabla_\nu n_\sigma}_{n^\sigma\nabla_\nu n_\sigma = 0})D^\mu\ln N \\
&\qquad - \underbrace{n_\sigma}_{n^\sigma n_\sigma = -1}(\nabla_\nu D^\mu\ln N) + \nabla_\sigma K^\mu{}_\nu \\
&\qquad + \underbrace{(\nabla_\sigma n_\nu)}_{n^\sigma\nabla_\sigma n_\nu = D_\nu\ln N} D^\mu\ln N + \underbrace{n_\nu}_{\gamma^\nu{}_\beta n_\nu = 0}(\nabla_\sigma D^\mu\ln N)] \\
&= \gamma_{\alpha\mu}\gamma^\nu{}_\beta[K^\mu{}_\sigma\nabla_\nu n^\sigma + \nabla_\nu D^\mu\ln N + n^\sigma\nabla_\sigma \underbrace{K^\mu{}_\nu}_{\gamma_{\alpha\mu}K^\mu{}_\nu = K_{\alpha\nu} = \gamma^\mu{}_\alpha K_{\mu\nu}} \\
&\qquad + D_\nu\ln ND^\mu\ln N] \\
&= -K_{\alpha\sigma}K^\sigma{}_\beta + D_\beta D_\alpha\ln N + \gamma^\mu{}_\alpha\gamma^\nu{}_\beta n^\sigma\nabla_\sigma K_{\mu\nu} \\
&\qquad + (D_\alpha\ln N)(D_\beta\ln N) \\
&= -K_{\alpha\sigma}K^\sigma{}_\beta + D_\beta(\frac{1}{N}D_\alpha N) + \gamma^\mu{}_\alpha\gamma^\nu{}_\beta n^\sigma\nabla_\sigma K_{\mu\nu} \\
&\qquad + (D_\alpha\ln N)(D_\beta\ln N) \\
&= -K_{\alpha\sigma}K^\sigma{}_\beta + [(D_\alpha N)(D_\beta\frac{1}{N}) + \frac{1}{N}D_\beta D_\alpha N] \\
&\qquad + \gamma^\mu{}_\alpha\gamma^\nu{}_\beta n^\sigma\nabla_\sigma K_{\mu\nu} + (D_\alpha\ln N)(D_\beta\ln N) \\
&= -K_{\alpha\sigma}K^\sigma{}_\beta + [\frac{1}{N}(D_\alpha N)\frac{1}{1/N}(D_\beta\frac{1}{N}) + \frac{1}{N}D_\beta D_\alpha N] \\
&\qquad + \gamma^\mu{}_\alpha\gamma^\nu{}_\beta n^\sigma\nabla_\sigma K_{\mu\nu} + (D_\alpha\ln N)(D_\beta\ln N) \\
&= -K_{\alpha\sigma}K^\sigma{}_\beta + [-(D_\alpha\ln N)(D_\beta\ln N) + \frac{1}{N}D_\beta D_\alpha N] \\
&\qquad + \gamma^\mu{}_\alpha\gamma^\nu{}_\beta n^\sigma\nabla_\sigma K_{\mu\nu} + (D_\alpha\ln N)(D_\beta\ln N) \\
&= -K_{\alpha\sigma}K^\sigma{}_\beta + \frac{1}{N}D_\beta D_\alpha\ln N + \gamma^\mu{}_\alpha\gamma^\nu{}_\beta n^\sigma\nabla_\sigma K_{\mu\nu}.
\end{aligned}$$

We can see that the $\gamma^\mu{}_\alpha\gamma^\nu{}_\beta n^\sigma\nabla_\sigma K_{\mu\nu}$ term is related to $\mathcal{L}_{\boldsymbol{m}}K_{\alpha\beta}$. For this, let us derive an expression for the gradient of $\boldsymbol{m}$. Indeed, $\boldsymbol{\nabla m} = \boldsymbol{\nabla}(N\boldsymbol{n}) = N\boldsymbol{\nabla n} + \boldsymbol{n}\otimes\boldsymbol{\nabla}N$. Since we already have an expression for $\boldsymbol{\nabla n}$, we get:

$$\begin{aligned}
\nabla_\beta m_\alpha &= N\nabla_\beta n_\alpha + n_\alpha\nabla_\beta N \\
&= -NK_{\alpha\beta} - N(D_\alpha\ln N)n_\beta + n_\alpha\nabla_\beta N \\
&= -NK_{\alpha\beta} - (D_\alpha N)n_\beta + n_\alpha\nabla_\beta N.
\end{aligned}$$

We can easily raise indices to get our equation for $\nabla_\beta m^\alpha$:

$$\nabla_\beta m^\alpha = -NK^\alpha{}_\beta - (D^\alpha N)n_\beta + n^\alpha\nabla_\beta N.$$

With this in mind, we get that the Lie derivative of $\boldsymbol{K}$ along $\boldsymbol{m}$ is:

$$\begin{aligned}
\mathcal{L}_{\boldsymbol{m}} K_{\alpha\beta} &= m^\mu \nabla_\mu K_{\alpha\beta} + K_{\mu\beta}\nabla_\alpha m^\mu + K_{\alpha\mu}\nabla_\beta m^\mu \\
&= N n^\mu \nabla_\mu K_{\alpha\beta} + K_{\mu\beta}(-N K^\mu{}_\alpha - (D^\mu N) n_\alpha + n^\mu \nabla_\alpha N) \\
&\qquad + K_{\alpha\mu}(-N K^\mu{}_\beta - (D^\mu N) n_\beta + n^\mu \nabla_\beta N) \\
&= N n^\mu \nabla_\mu K_{\alpha\beta} - \underbrace{(N K_{\mu\beta} K^\mu{}_\alpha + N K_{\alpha\mu} K^\mu{}_\beta)}_{K_{\mu\beta}K^\mu{}_\alpha = \gamma^{\mu\nu} K_{\mu\beta} K_{\nu\alpha} = K_{\alpha\nu} K^\nu{}_\beta} - K_{\mu\beta}(D^\mu N) n_\alpha \\
&\qquad - K_{\alpha\mu}(D^\mu N) n_\beta + \underbrace{(K_{\mu\beta} n^\mu \nabla_\alpha N + K_{\alpha\mu} n^\mu \nabla_\beta N)}_{K_{\mu\beta} n^\mu = K_{\alpha\mu} n^\mu = 0} \\
&= N n^\mu \nabla_\mu K_{\alpha\beta} - 2N K_{\alpha\mu} K^\mu{}_\beta - K_{\mu\beta}(D^\mu N) n_\alpha - K_{\alpha\mu}(D^\mu N) n_\beta.
\end{aligned}$$

We can project this equation onto Σ_t by applying the projection operator $\vec{\gamma}$ on both sides. Using the fact that $\mathcal{L}_{\boldsymbol{m}}\boldsymbol{K}$ lies on Σ_t (that is, $\vec{\gamma}\mathcal{L}_{\boldsymbol{m}}\boldsymbol{K} = \mathcal{L}_{\boldsymbol{m}}\boldsymbol{K}$), we get that

$$\begin{aligned}
\mathcal{L}_{\boldsymbol{m}} K_{\alpha\beta} &= N n^\sigma \nabla_\sigma K_{\alpha\beta} - 2N K_{\alpha\sigma} K^\sigma{}_\beta - K_{\sigma\beta}(D^\sigma N) n_\alpha - K_{\alpha\sigma}(D^\sigma N) n_\beta \\
&= \gamma^\mu{}_\alpha \gamma^\nu{}_\beta (N n^\sigma \nabla_\sigma K_{\mu\nu} - 2N K_{\mu\sigma} K^\sigma{}_\nu - K_{\sigma\nu}(D^\sigma N) n_\mu - K_{\mu\sigma}(D^\sigma N) n_\nu) \\
&= N \gamma^\mu{}_\alpha \gamma^\nu{}_\beta n^\sigma \nabla_\sigma K_{\mu\nu} - 2N K_{\alpha\mu} K^\mu{}_\beta
\end{aligned}$$

So replacing this expression for $\gamma^\mu{}_\alpha \gamma^\nu{}_\beta n^\sigma \nabla_\sigma K_{\mu\nu}$ on our previous projection, we get successively

$$\begin{aligned}
\gamma_{\alpha\mu} n^\sigma \gamma^\nu{}_\beta {}^{(4)}R^\mu{}_{\rho\nu\sigma} n^\rho &= -K_{\alpha\sigma} K^\sigma{}_\beta + \frac{1}{N} D_\beta D_\alpha \ln N + \gamma^\mu{}_\alpha \gamma^\nu{}_\beta n^\sigma \nabla_\sigma K_{\mu\nu} \\
&= -K_{\alpha\sigma} K^\sigma{}_\beta + \frac{1}{N} D_\beta D_\alpha \ln N + \frac{1}{N} \mathcal{L}_{\boldsymbol{m}} K_{\alpha\beta} + 2K_{\alpha\sigma} K^\sigma{}_\beta,
\end{aligned}$$

leading us to our final result,

$$\gamma_{\alpha\mu} n^\rho \gamma^\nu{}_\beta n^\sigma {}^{(4)}R^\mu{}_{\rho\nu\sigma} = \frac{1}{N} \mathcal{L}_{\boldsymbol{m}} K_{\alpha\beta} + \frac{1}{N} D_\alpha D_\beta N + K_{\alpha\mu} K^\mu{}_\beta. \tag{6.9}$$

This relation is called the ***Ricci equation*** and constitutes the last non trivial decomposition of the Riemann tensor. This one, together with the Gauss equation (6.6) and the Codazzi equation (6.7), completes the 3+1 decomposition of the spacetime Riemann tensor

6.9. 3+1 expression for the spacetime Ricci tensor and scalar curvature

Taking our previously defined Gauss relation,

$$\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}\gamma^{\gamma}{}_{\rho}\gamma^{\sigma}{}_{\lambda}{}^{(4)}R^{\rho}{}_{\sigma\mu\nu} = R^{\gamma}{}_{\lambda\alpha\beta} + (K^{\gamma}{}_{\alpha}K_{\lambda\beta} - K^{\gamma}{}_{\beta}K_{\alpha\lambda}),$$

we can contract on the indices γ and α, using that $\gamma^{\mu}{}_{\alpha}\gamma^{\alpha}{}_{\rho} = \gamma^{\mu}{}_{\rho} = \delta^{\mu}{}_{\rho} + n^{\mu}n_{\rho}$, we obtain the following expression, called the ***contracted Gauss relation***:

$$\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}{}^{(4)}R_{\mu\nu} + \gamma_{\alpha\mu}n^{\nu}\gamma^{\rho}{}_{\beta}n^{\sigma}{}^{(4)}R^{\mu}{}_{\nu\rho\sigma} = R_{\alpha\beta} + KK_{\alpha\beta} - K_{\alpha\mu}K^{\mu}{}_{\beta}. \tag{6.10}$$

We notice that we already have an expression for the spacetime Riemann tensor projection term, and replacing it on our contracted Gauss equations (by changing corresponding free indices so they fit) yields an expression for the ***projection of the spacetime Ricci tensor***:

$$\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}{}^{(4)}R_{\mu\nu} = -\frac{1}{N}\mathcal{L}_{\boldsymbol{m}}K_{\alpha\beta} - \frac{1}{N}D_{\alpha}D_{\beta}N + R_{\alpha\beta} + KK_{\alpha\beta} - 2K_{\alpha\mu}K^{\mu}{}_{\beta}.$$

By taking the trace of this equation with respect to the metric γ, which means contracting with $\gamma^{\alpha\beta}$, yields,

$$\begin{aligned}
\gamma^{\alpha\beta}\gamma^{\mu}{}_{\alpha}\gamma^{\nu}{}_{\beta}{}^{(4)}R_{\mu\nu} &= \gamma^{\mu\nu}{}^{(4)}R_{\mu\nu} \\
&= -\frac{1}{N}\gamma^{\alpha\beta}\mathcal{L}_{\boldsymbol{m}}K_{\alpha\beta} - \frac{1}{N}\gamma^{\alpha\beta}D_{\alpha}D_{\beta}N + \gamma^{\alpha\beta}R_{\alpha\beta} \\
&\qquad + K\gamma^{\alpha\beta}K_{\alpha\beta} - 2\gamma^{\alpha\beta}K_{\alpha\mu}K^{\mu}{}_{\beta} \\
&= -\frac{1}{N}\gamma^{ij}\mathcal{L}_{\boldsymbol{m}}K_{ij} - \frac{1}{N}D_iD^iN + R + K^2 - 2K_{ij}K^{ij}.
\end{aligned}$$

By replacing $\gamma^{\mu\nu}{}^{(4)}R_{\mu\nu} = (g^{\mu\nu} + n^{\mu}n^{\nu})^{(4)}R_{\mu\nu} = {}^{(4)}R\text{t}^{(4)}R_{\mu\nu}n^{\mu}n^{\nu}$ and noticing that,

$$\begin{aligned}
-\gamma^{ij}\mathcal{L}_{\boldsymbol{m}}K_{ij} &= -\mathcal{L}_{\boldsymbol{m}}(\gamma^{ij}K_{ij}) + K_{ij}\mathcal{L}_{\boldsymbol{m}}\gamma^{ij} \\
&= -\mathcal{L}_{\boldsymbol{m}}K + K_{ij}\mathcal{L}_{\boldsymbol{m}}\gamma^{ij},
\end{aligned}$$

with an equation for $\mathcal{L}_{\boldsymbol{m}}\gamma^{ij}$ being:

$$\begin{aligned}
0 &= \mathcal{L}_{\boldsymbol{m}}\delta^{j}_{i} \\
&= \mathcal{L}_{\boldsymbol{m}}(\gamma_{ik}\gamma^{kj}) \\
&= \gamma_{ik}\mathcal{L}_{\boldsymbol{m}}\gamma^{kj} + \gamma^{kj}\mathcal{L}_{\boldsymbol{m}}\gamma_{ik} \\
&= \gamma^{il}\gamma_{ik}\mathcal{L}_{\boldsymbol{m}}\gamma^{kj} + \gamma^{il}\gamma^{kj}\mathcal{L}_{\boldsymbol{m}}\gamma_{ik} \\
&= \delta^{l}{}_{k}\mathcal{L}_{\boldsymbol{m}}\gamma^{kj} + \gamma^{il}\gamma^{kj}\mathcal{L}_{\boldsymbol{m}}\gamma_{ik} \\
&= \mathcal{L}_{\boldsymbol{m}}\gamma^{lj} - 2N\gamma^{il}\gamma^{kj}K_{ik}
\end{aligned}$$

Therefore, plugging this into our previous equation yields

$$-\gamma^{ij}\mathcal{L}_{\boldsymbol{m}}K_{ij} = -\mathcal{L}_{\boldsymbol{m}}K + 2NK_{ij}K^{ij}.$$

Leading us to our equation for the spacetime scalar curvature:

$${}^{(4)}R + {}^{(4)}R_{\mu\nu}n^{\mu}n^{\nu} = R + K^2 - \frac{1}{N}\mathcal{L}_{\boldsymbol{m}}K - \frac{1}{N}D_iD^iN.$$

If we used to Gauss relation and contracted it twice, it can be proven yields a result that allows us to get rid of the Ricci tensor term ${}^{(4)}R_{\mu\nu}n^\mu n^\nu$. This gives us a final equation that only involves the spacetime scalar curvature:

$$ {}^{(4)}R = R + K^2 + K_{ij}K^{ij} - \frac{2}{N}\mathcal{L}_{\boldsymbol{m}}K - \frac{2}{N}D_iD^iN. $$

6.10. 3+1 decomposition of the Einstein equation

Let us consider the Einstein tensor equation on a spacetime $(\mathcal{M}, g)$:

$$^{(4)}R_{\mu\nu} - \frac{1}{2}{}^{(4)}Rg_{\mu\nu} = 8\pi T_{\mu\nu}. \tag{6.11}$$

As mentioned in previous chapter, we will consider a globally hyperbolic spacetime and consider a foliation $(\Sigma_t)_{t\in\mathbb{R}}$ of $\mathcal{M}$ by a family of spacelike hypersurfaces.

6.10.1. Projection of the stress-energy tensor

To fully project the Einstein equation, we need to consider projections of the stress-energy tensor $T_{\mu\nu}$. For this, we will define:

- The double timelike projection of the stress-energy tensor, called the ***matter energy density***:

$$E = n^\mu n^\nu T_{\mu\nu}.$$

- The mixed time and spatial projection, called the ***matter momentum density***:

$$p_\alpha = -n^\nu \gamma^\nu{}_\alpha T_{\mu\nu}.$$

- The double spatial projection, called the ***matter stress tensor***:

$$S_{\alpha\beta} = \gamma^\mu{}_\alpha \gamma^\nu{}_\beta T_{\mu\nu}.$$

- The trace of $S_{\alpha\beta}$ with respect to the metric γ, or equivalently with respect to the metric g:

$$S = \gamma^{ij} S_{ij} = g^{\mu\nu} S_{\mu\nu}.$$

An important formula is one that recovers T from E, p and S. A consequence of this formula proves that:

$$T = S - E.$$

6.10.2. Projection of the Einstein equation

The final step in numerical relativity is to use all our previous equations to derive how to Einstein's field equations work when looking at them in their projected form. Since we study all possible non trivial projections, these would give us an equivalent form of the full Einstein's equations. We will consider the Einstein's equations in their equivalent form,

$$^{(4)}R_{\mu\nu} = 8\pi \left(T_{\mu\nu} - \frac{1}{2} T g_{\mu\nu} \right). \tag{6.12}$$

(1) Full projection onto Σ_t

We apply our projection operator to our equivalent Einstein field equations on both sides,

$$\vec{\gamma}\,^{(4)}R_{\mu\nu} = 8\pi \left(\vec{\gamma} T_{\mu\nu} - \frac{1}{2} T \vec{\gamma} g \right).$$

This yields,

$$-\frac{1}{N}\mathcal{L}_{\boldsymbol{m}} K_{\alpha\beta} - \frac{1}{N} D_\alpha D_\beta N + R + K K_{\alpha\beta} - 2 K_{\alpha\mu} K^\mu{}_\beta = 8\pi \left[S_{\alpha\beta} - \frac{1}{2}(S - E)\gamma_{\alpha\beta} \right].$$

Since each term is a tensor field tangent to Σ_t, we can restrict indices to only spatial indices without loss of generality. By rearranging terms we get finally:

$$\mathcal{L}_{\boldsymbol{m}} K_{ij} = -D_i D_j N + N(R_{ij} + KK_{ij} - 2K_{ik}K^k{}_j + 4\pi[(S-E)\gamma_{ij} - 2S_{ij}]). \tag{6.13}$$

(2) Full projection perpendicular to Σ_t

We begin taking our previously contracted Gauss relation (6.10):

$$\gamma^\mu{}_\alpha \gamma^\nu{}_\beta {}^{(4)}R_{\mu\nu} + \gamma_{\alpha\mu} n^\nu \gamma^\rho{}_\beta n^\sigma {}^{(4)}R^\mu{}_{\nu\rho\sigma} = R_{\alpha\beta} + KK_{\alpha\beta} - K_{\alpha\mu}K^\mu{}_\beta.$$

and taking its trace with respect to γ. Noticing that

$$\gamma^{\alpha\beta}\gamma_{\alpha\mu}\gamma^\rho{}_\beta n^\sigma {}^{(4)}R^\mu{}_{\nu\rho\sigma} = \gamma^\rho{}_\mu n^\nu n^\sigma {}^{(4)}R^\mu{}_{\nu\rho\sigma} = \underbrace{{}^{(4)}R^\mu{}_{\nu\mu\sigma}}_{={}^{(4)}R_{\nu\sigma}} n^\nu n^\sigma + \underbrace{{}^{(4)}R^\mu{}_{\nu\rho\sigma}}_{=0} = {}^{(4)}R_{\mu\nu} n^\mu n^\nu,$$

we obtain the scalar Gauss relation,

$${}^{(4)}R + 2\,{}^{(4)}R_{\mu\nu} n^\mu n^\nu = R + K^2 - K_{ij}K^{ij}.$$

We notice the left hand side is just two times the perpendicular projection of the Einstein equation, which is,

$${}^{(4)}R_{\alpha\beta} n^\alpha n^\beta - \frac{1}{2}\,{}^{(4)}R \underbrace{g_{\alpha\beta} n^\alpha n^\beta}_{=-1} = 8\pi \underbrace{T_{\alpha\beta} n^\alpha n^\beta}_{=E}$$

Leading us to the so called ***Hamiltonian constraint***:

$$R + K^2 + K_{ij}K^{ij} = 16\pi E. \tag{6.14}$$

(3) Mixed projection

Now, we take the Codazzi relation,

$$\gamma^\gamma{}_\rho n^\sigma \gamma^\mu{}_\alpha \gamma^\nu{}_\beta {}^{(4)}R^\rho{}_{\sigma\mu\nu} = D_\beta K^\gamma{}_\alpha - D_\alpha K^\gamma{}_\beta,$$

and contract it on the indices α and γ. This yields,

$$\gamma^\mu{}_\rho n^\sigma \gamma^\nu{}_\beta {}^{(4)}R^\rho{}_{\sigma\mu\nu} = D_\beta K - D_\mu K^\mu{}_\beta.$$

Let us work out the LHS term:

$$\begin{aligned}
\gamma^\mu{}_\rho n^\sigma \gamma^\nu{}_\beta {}^{(4)}R^\rho{}_{\sigma\mu\nu} &= (\delta^\mu{}_\rho + n^\mu n_\rho) n^\sigma \gamma^\nu{}_\beta {}^{(4)}R^\rho{}_{\sigma\mu\nu} \\
&= n^\sigma \gamma^\nu{}_\beta {}^{(4)}R_{\sigma\nu} + \gamma^\nu{}_\beta {}^{(4)}R^\rho{}_{\sigma\mu\nu} n_\rho n^\sigma n^\mu \\
&= n^\sigma \gamma^\nu{}_\beta {}^{(4)}R_{\sigma\nu} + \gamma^\nu{}_\beta [{}^{(4)}R^\rho{}_{\sigma\mu\nu}(g_{\rho\alpha} n^\alpha)(g^{\sigma\lambda} n_\lambda)] n^\mu \\
&= n^\sigma \gamma^\nu{}_\beta {}^{(4)}R_{\sigma\nu} + \gamma^\nu{}_\beta [{}^{(4)}R_\alpha{}^\lambda{}_{\mu\nu} n^\sigma n_\lambda] n^\mu \\
&= n^\sigma \gamma^\nu{}_\beta {}^{(4)}R_{\sigma\nu} + \gamma^\nu{}_\beta [{}^{(4)}R_\sigma{}^\rho{}_{\mu\nu} n^\sigma n_\rho] n^\mu \\
&= n^\sigma \gamma^\nu{}_\beta {}^{(4)}R_{\sigma\nu} + \gamma^\nu{}_\beta \underbrace{[-{}^{(4)}R^\rho{}_{\sigma\mu\nu} n^\sigma n_\rho]}_{{}^{(4)}R^\rho{}_{\sigma\mu\nu} = -{}^{(4)}R^\rho{}_{\sigma\mu\nu} = 0} n^\mu \\
&= n^\sigma \gamma^\nu{}_\beta {}^{(4)}R_{\sigma\nu}.
\end{aligned}$$

Therefore, we have the ***contracted Codazzi relation***:

$$n^\sigma \gamma^\nu{}_\beta {}^{(4)}R_{\sigma\nu} = D_\beta K - D_\mu K^\mu{}_\beta.$$

We now project the Einstein equation in its first form once onto Σ_t and once perpendicular to it, and notice the LHS is actually the contracted Codazzi relation, while the RHS is the matter momentum density:

$$\begin{aligned}
&{}^{(4)}R_{\alpha\beta}n^\alpha\gamma^\beta{}_\mu - \frac{1}{2}{}^{(4)}R\underbrace{g_{\alpha\beta}n^\alpha\gamma^\beta{}_\mu}_{=0} = 8\pi T_{\alpha\beta}n^\alpha\gamma^\beta{}_\mu \\
\implies &{}^{(4)}R_{\alpha\beta}n^\alpha\gamma^\beta{}_\mu = 8\pi T_{\alpha\beta}n^\alpha\gamma^\beta{}_\mu \\
\implies &D_\mu K - D_\lambda K^\lambda{}_\mu = -8\pi p_\mu \\
\implies &D_j K^j{}_i - D_i K = 8\pi p_i. \quad \text{(since these tensors are tangent to } \Sigma_t)
\end{aligned} \tag{6.15}$$

This last equation is called the ***momentum constraint***.

6.11. The 3+1 Einstein system

We have now derived all the equations necessary to solve the Einstein system. Indeed, the four constraint equations ((6.14) and (6.15)) are the necessary and sufficient integrability conditions for the embedding of the spacelike hypersurfaces $(\Sigma, \gamma_{\mu\nu}, K_{\mu\nu})$ in the 4D spacetime $(\mathcal{M}, g_{\mu\nu})$, and the evolution equations defined by (6.8) and (6.8) allow us to evolve our unknowns and get the following ***Einstein system***,

$$\begin{cases}
\left(\frac{\partial}{\partial t} - \mathcal{L}_{\boldsymbol{\beta}}\right)\gamma_{ij} = -2NK_{ij} \\
\left(\frac{\partial}{\partial t} - \mathcal{L}_{\boldsymbol{\beta}}\right)K_{ij} = -D_iD_jN + N[R_{ij} + KK_{ij} - 2K_{ik}K^k{}_j + 4\pi((S-E)\gamma_{ij} - 2S_{ij})] \\
R + K^2 - K_{ij}K^{ij} = 16\pi E \\
D_jK^j{}_i - D_iK = 8\pi p_i.
\end{cases}$$

The covariant derivatives can be expressed in terms of partial derivatives and Christoffel symbols in the following way:

$$\begin{aligned}
D_iD_jN &= \frac{\partial^2}{\partial x^i \partial x^j} - \Gamma^k{}_{ij}\frac{\partial N}{\partial x^k}, \\
D_jK^j{}_i &= \frac{\partial K^j{}_i}{\partial x^j} + \Gamma^j{}_{jk}K^k{}_i - \Gamma^k_{ji}K^j{}_k, \\
D_iK &= \frac{\partial K}{\partial x^i}.
\end{aligned}$$

Lie derivatives can also be expressed in terms of partial derivatives:

$$\begin{aligned}
\mathcal{L}_{\boldsymbol{\beta}}\gamma_{ij} &= \frac{\partial \beta_i}{\partial x^j} + \frac{\partial \beta_j}{\partial x^i} - 2\Gamma^k{}_{ij}\beta_k, \\
\mathcal{L}_{\boldsymbol{\beta}}K_{ij} &= \beta^k\frac{\partial K_{ij}}{\partial x^k} + K_{kj}\frac{\partial \beta^k}{\partial x^i} + K_{ik}\frac{\partial \beta^k}{\partial x^j}.
\end{aligned}$$

For completeness, we also mention the expression of the Ricci tensor, scalar curvature and

Christoffel symbols in terms of partial derivatives of the metric:

$$
\begin{aligned}
R_{ij} &= \frac{\partial \Gamma^k{}_{ij}}{\partial x^k} - \frac{\partial \Gamma^k{}_{ik}}{\partial x^j} + \Gamma^k{}_{ij}\Gamma^l{}_{kl} - \Gamma^l{}_{ik}\Gamma^k{}_{lj}, \\
R &= \gamma^{ij} R_{ij}, \\
\Gamma^k{}_{ij} &= \frac{1}{2}\gamma^{kl}\left(\frac{\partial \gamma_{lj}}{\partial x^i} + \frac{\partial \gamma_{il}}{\partial x^j} - \frac{\partial \gamma_{ij}}{\partial x^l}\right).
\end{aligned}
$$

One usually works with systems where matter source terms (E, p_i, S_{ij}) are given, so the Einstein system constitutes a second-order non-linear PDE system of the unknowns $(\gamma_{ij}, K_{ij}, N, \beta^i)$.

Capítulo 7

The Wave Equation on the Schwarzschild Background

7.1. Introduction

We begin with the verification of some equations of motion, with our end goal being usage of finite-difference codes to study the spherically-symmetric dynamics of a massless scalar field on a Schwarzschild background, so the spacetime will be completely known a priori. We focus on the physics describing the absorption and/or scattering of spherically-symmetric pulses of scalar radiation which fall onto a black hole. The domain of our simulations will exclude the interior of a black hole by limiting the spatial domain to the region $r \geq 2M$. Since $r = 2M$ is null, no special boundary conditions are needed for this scalar field; we will apply the equations of motion (the covariant wave equation) up to and including $r = 2M$.

7.2. Wave Equation

Let the general 3+1 form of the Schwartzchild spacetime be:

$$ds^2 = \left(-\alpha^2 + a^2\beta^2\right) dt^2 + 2a^2\beta dt dr + a^2 dr^2 + r^2\left(d\theta^2 + \sin^2\theta d\phi^2\right),$$

where $\alpha \equiv \alpha(r)$ and $a = a(r)$. In 3+1 language, α is known as the lapse function, while β is the radial component of the shift vector, i.e. $\beta^i = (\beta, 0, 0)$, $\beta_i \equiv \gamma_{ij}\beta^j = \left(a^2\beta, 0, 0\right)$, where γ_{ij} is the usual metric of spacelike hypersurfaces defined by $t = \text{const}$, that is to say $\gamma_{ij} \equiv \text{diag}\left(a^2, r^2, r^2\sin^2\theta\right)$.

Let us show the null geodesics of this metric is are given by

$$\left(\frac{dr}{dt}\right)_{\pm} = -\beta \pm \frac{\alpha}{a}.$$

We start noting that we are working with a spherically symmetric metric, so the angular components are not taken into account for the study of $\frac{dr}{dt}$. So we view ds^2 as a tensor field on $\mathbb{R}^2$ that assigns to each point a bilinear function of the following general form:

$$(t, r) \mapsto f(t, r)\, dt \otimes dt + g(t, r)\, dt \otimes dr + h(t, r)\, dr \otimes dt + k(t, r)\, dr \otimes dr.$$

For this problem, we will consider the $\frac{\partial}{\partial t}$ vector in $\mathbb{R}^2$. To see this is in fact a vector in $\mathbb{R}^2$, we note that:

$$\frac{\partial}{\partial t} = \frac{dt}{dt}\frac{\partial}{\partial t} + \frac{dr}{dt}\frac{\partial}{\partial r} = \left(1, \frac{dr}{dt}\right).$$

Therefore, our tensor field can be viewed as a quadratic form that assigns two vectors, $\overrightarrow{a} = (a_1, a_2), \overrightarrow{b} = (b_1, b_2)$, a quadratic form,

$$\begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} -\alpha^2 + a^2\beta^2 & a^2\beta \\ a^2\beta & a^2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Taking our two vectors to be equal to $\frac{\partial}{\partial t}$, we get the following tensor equation:

$$ds^2 = \begin{bmatrix} 1 & \frac{dr}{dt} \end{bmatrix} \begin{bmatrix} -\alpha^2 + a^2\beta^2 & a^2\beta \\ a^2\beta & a^2 \end{bmatrix} \begin{bmatrix} 1 \\ \frac{dr}{dt} \end{bmatrix},$$

which finally solves

$$ds^2_{\left(\frac{\partial}{\partial t}, \frac{\partial}{\partial t}\right)} = \left(-\alpha^2 + a^2\beta^2\right) + 2a^2\beta\frac{dr}{dt} + a^2\left(\frac{dr}{dt}\right)^2.$$

Since we are looking for null geodesics, we impose $ds^2 = 0$, so we are left with:

$$\begin{aligned} 0 &= \left(-\alpha^2 + a^2\beta^2\right) + 2a^2\beta\frac{dr}{dt} + a^2\left(\frac{dr}{dt}\right)^2 \\ &= -\alpha^2 + \left(a\beta + a\frac{dr}{dt}\right)^2. \end{aligned}$$

That finally gives us a solution

$$\left(\frac{dr}{dt}\right)_{\pm} = -\beta \pm \frac{\alpha}{a}.$$

This equations will help us solve for the radial velocities of our wave given some lapse function and shift vector that will determine our hypersurfaces. The two signs correspond to both ingoing and outgoing pulses. When studying these pulses around our black hole, this function will give us a way to solve for the speed of light far away from the event horizon, giving us a boundary condition when simulating in a finite spatial grid.

$$\nabla^a \nabla_a \phi(r, t) = 0, \tag{7.1}$$

can be written as the pair of first-order-in-time equations:

$$\begin{cases} \partial_t \Phi = \partial_r(\beta\Phi + \frac{\alpha}{a}\Pi) \\ \partial_t \Pi = \frac{1}{r^2}\partial_r(r^2(\beta\Pi + \frac{\alpha}{a}\Phi)) \end{cases}$$

where,

$$\begin{cases} \Phi(r, t) \equiv \partial_r \phi \\ \Pi(r, t) \equiv \frac{a}{\alpha}(\partial_t \phi - \beta\partial_r \phi) \end{cases}$$

When solving for the covariant wave equation, one can expand using our connection and partial derivatives. This quickly leads us to its alternate version,

$$\frac{1}{\sqrt{-g}}\frac{\partial}{\partial x^\nu}\left(\sqrt{-g}g^{\nu\mu}\frac{\partial \phi}{\partial x^\mu}\right) = 0,$$

where g is the determinant of the 4-matrix, and where the components $g^{\mu\nu}$ of the inverse 4-metric

are given by:

$$g^{\mu\nu} = \begin{bmatrix} -\alpha^{-2} & \beta\alpha^{-2} & 0 & 0 \\ \beta\alpha^{-2} & a^{-2} - \beta^2\alpha^{-2} & 0 & 0 \\ 0 & 0 & r^{-2} & 0 \\ 0 & 0 & 0 & r^{-2}\sin^{-2}\theta \end{bmatrix}.$$

We can compute the value of $\sqrt{-g}$ by solving the determinant of our metric, and get the value of $\sqrt{-g} = a\alpha r^2 \sin\theta$, and since it is positive, we can cancel it and only study the derivatives.

$$\begin{aligned}
0 &= \partial_t \left(\sqrt{-g}\left(g^{tt}\partial_t\phi + g^{rt}\partial_r\phi\right)\right) + \partial_r\left(\sqrt{-g}\left(g^{tr}\partial_r\phi + g^{rr}\partial_r\phi\right)\right) \\
&= a\alpha r^2\sin\theta\partial_r\left(\frac{-1}{\alpha^2}\partial_t\phi + \left(\frac{1}{a^2} - \frac{\beta^2}{\alpha^2}\right)\partial_r\phi\right) \\
&\quad + \partial_r\left(a\alpha r^2\sin\theta\left(\frac{\beta}{\alpha^2}\partial_t\phi + \left(\frac{1}{a^2} - \frac{\beta^2}{\alpha^2}\right)\partial_r\phi\right)\right) \\
&= \sin\theta\left(ar^2\partial_t\left(\frac{-1}{\alpha}\partial_t\phi\frac{\beta}{\alpha}\partial_r\phi\right) + \partial_r\left(r^2\left[\beta\left(\frac{a}{\alpha}\left(\partial_t\phi - \beta\partial_r\phi\right)\right) + \frac{\alpha}{a}\partial_r\phi\right]\right)\right) \\
&= \sin\theta\left(-r^2\partial_t\left(\frac{a}{\alpha}\left(\partial_t\phi - \beta\partial_r\phi\right)\right) + \partial_r\left(r^2\left(\beta\Pi + \frac{\alpha}{a}\Phi\right)\right)\right) \\
&= \sin\theta\left(-r^2\partial_t\Pi + \partial_r\left(r^2\left(\beta\Pi + \frac{\alpha}{a}\Phi\right)\right)\right).
\end{aligned}$$

Dividing by $\sin\theta$ and rearranging terms we end up with the evolution equation:

$$\partial_t\Pi = \frac{1}{r^2}\partial_r\left(r^2\left(\beta\Pi + \frac{\alpha}{a}\Phi\right)\right).$$

Finally, to describe the evolution of Φ we use the commutation of partial derivatives:

$$\begin{aligned}
\partial_t\Phi &= \partial_t\partial_r\phi \\
&= \partial_r\left(\partial_t\phi\right) \\
&= \partial_r\left(\beta\partial_r\phi + \frac{\alpha}{a}\frac{a}{\alpha}\left(\partial_t\phi - \beta\partial_r\phi\right)\right) \\
&= \partial_r\left(\beta\Phi + \frac{\alpha}{a}\Pi\right),
\end{aligned}$$

leaving us with the desired system of first-order-in-time equations.

7.3. Schwarzchild Solution in Ingoing Eddington-Finkelstein Coordinates

We now consider the Schwarzschild form of the original metric, bearing in mind we will be restricting attention to $r \geq 2M$:

$$ds^2 = -\left(1 - \frac{2M}{r}\right)dt^2 + \left(1 - \frac{2M}{r}\right)^{-1}dr^2 + r^2 d\Omega^2.$$

We now define the "Regge-Wheeler tortoise coordinate", r_*:

$$r_* \equiv r + 2M\ln\left(\frac{r}{2M} - 1\right),$$

then u and v defined by

$$\begin{cases} u \equiv t - r_* \\ v \equiv t + r_*, \end{cases}$$

are outgoing (u) and ingoing (v) null coordinates.

We will show that adopting a timelike coordinate, $\tilde{t}$, based on the ingoing null coordinate, v, defined as

$$\tilde{t} = v - r = t + 2M \ln\left(\frac{r}{2M} - 1\right),$$

then the Schwarzschild metric takes the ingoing Eddington-Finkelstein form

$$ds^2 = -\left(1 - \frac{2M}{r}\right) d\tilde{t}^2 + \frac{4M}{r} d\tilde{t} dr + \left(1 + \frac{2M}{r}\right) dr^2 + r^2 d\Omega^2. \tag{7.2}$$

By definition we have $\tilde{t} = t + 2M \ln\left(\frac{r}{2M} - 1\right)$, then differenciating both sides yields,

$$dt = d\tilde{t} - \left(\frac{r}{2M} - 1\right)^{-1} dr,$$

which is a 1-form on its own. We can 'square both sides' since we are working with tensors and luckily they have the algebra required to do so, so we compute dt^2 as,

$$dt^2 = d\tilde{t}^2 - 2\left(\frac{2}{2M} - 1\right)^{-1} d\tilde{t} dr + \left(\frac{r}{2M} - 1\right)^{-2} dr^2.$$

Replacing these tensors on our original spacetime interval, we get the desired equation,

$$\begin{aligned} ds^2 &= -\left(1 - \frac{2M}{r}\right) dt^2 + \left(1 - \frac{2M}{r}\right)^{-1} dr^2 + r^2 d\Omega^2 \\ &= -\left(1 - \frac{2M}{r}\right) \left[d\tilde{t}^2 - 2\left(\frac{2}{2M} - 1\right)^{-1} d\tilde{t} dr + \left(\frac{r}{2M} - 1\right)^{-2} dr^2 \right] \\ &\quad + \left(1 - \frac{2M}{r}\right)^{-1} dr^2 + r^2 d\Omega^2 \\ &= -\left(1 - \frac{2M}{r}\right) d\tilde{t}^2 + \frac{4M}{r} d\tilde{t} dr + \left(1 + \frac{2M}{r}\right) dr^2 + r^2 d\Omega^2. \end{aligned}$$

We will show that in the limit $r \to \infty$, the wave equation can be written:

$$\partial_{tt}(r\phi) = \partial_{rr}(r\phi).$$

Clearly the limit to the black hole solution when $r \to \infty$ is the Minkowski metric in spherical coordinates, that is,

$$ds^2 = -dt^2 + dr^2 + r^2 d\Omega^2.$$

So we recognize the determinant of the metric being $g = -r^4 \sin^2\theta$, so $\sqrt{-g} = r^2 \sin\theta$. Then,

applying the formula used in problem 2:

$$\begin{aligned}
0 &= \frac{1}{\sqrt{-g}}\frac{\partial}{\partial x^\nu}\left(\sqrt{-g}g^{\nu\mu}\frac{\partial\phi}{\partial x^\mu}\right) \\
&= \frac{1}{\sqrt{-g}}\left(\partial_t\left(-\sqrt{-g}\partial_t\phi\right)+\partial_r\left(\sqrt{-g}\partial_r\phi\right)\right) \\
&= \frac{1}{r^2\sin\theta}\left(\partial_t\left(-r^2\sin\theta\cdot\partial_t\phi\right)+\partial_r\left(r^2\sin\theta\cdot\partial_r\phi\right)\right) \\
&= -\partial_{tt}\phi+\frac{1}{r^2}\partial_r\left(r^2\partial_r\phi\right) \\
&= -\partial_{tt}\phi+\frac{1}{r^2}\left(2r\partial_r\phi+r^2\partial_{rr}\phi\right) \\
&= -\partial_{tt}\phi+\frac{1}{r}\left(\partial_r\phi+\partial_r\phi+r\partial_{rr}\phi\right) \\
&= -\partial_{tt}\phi+\frac{1}{r}\left(\partial_r\phi+\partial_r\left(r\partial_r\phi\right)\right) \\
&= -\partial_{tt}\phi+\frac{1}{r}\left(\partial_r\left(\phi+r\partial_r\phi\right)\right) \\
&= -\partial_{tt}\phi+\frac{1}{r}\left(\partial_r\left(\partial_r\left(r\phi\right)\right)\right) \\
&= -\partial_{tt}\phi+\frac{1}{r}\partial_{rr}\left(r\phi\right).
\end{aligned}$$

Rearranging terms leave us with the final equation

$$\partial_{tt}\left(r\phi\right)=\partial_{rr}\left(r\phi\right).$$

The outgoing solution of this equation can be thought as radial pulses, ie:

$$\left(r\phi\right)\left(r,t\right)=g\left(c_+t-r\right).$$

Now, from our first problem we have that

$$\left(\frac{dr}{dt}\right)_+=c_+=-\beta+\frac{\alpha}{a},$$

so that, asymptotically, we should expect

$$\left(r\phi\right)\left(r,t\right)=g\left(\left(-\beta+\frac{\alpha}{a}\right)t-r\right),$$

which we can also express as

$$\partial_t\left(r\phi\right)+\left(-\beta+\frac{\alpha}{a}\right)\partial_r\left(r\phi\right)=0.$$

So we will solve the wave equation on the finite spatial domain $2M\leq r\leq R$, to finally impose this last equation a a boundary condition at $r=R$.

7.4. Initial Data for the Simulation

To solve the coupled equations, we must supply initial conditions

$$\begin{cases} \Phi\left(r,0\right) = \Phi_0\left(r\right) \\ \Pi\left(r,0\right) = \Pi_0\left(r\right) \end{cases}.$$

We will assume that the initial configuration of the scalar field itself, $\phi(r,0) = \phi_0(r)$ describes some "pulse" shape, such as a Gaussian:

$$\phi_0\left(r; A, r_0, \partial\right) = A\exp\left\{\left(-\left(\frac{r-r_0}{\Delta}\right)^2\right)\right\}, \tag{7.3}$$

for some A, r_0 and Δ adjustable parameters. To get an initial condition for our system of equations, we need a way to get an initial condition for the time derivative of our scalar pulse, that is, $\partial_t\phi(r,0)$, but we cannot derive this from ϕ_0. To solve this, we use the fact that $(r\phi)(r,t)$ initially solves an ingoing wave equation, thus,

$$\partial_t\left(r\phi\right)\left(r,0\right) - \partial_r\left(r\phi\right)\left(r,0\right) = 0.$$

From the above equations, we derive initial conditions $\Phi_0(r)$ and $\Pi_0(r)$, in terms of $\phi_0(r)$ and $\phi_0'(r)$. We can use limits to get the initial condition for Φ, indeed,

$$\begin{aligned} \Phi_0(r) &= \Phi(r,0) \\ &= (\partial_r\phi)(r,0) \\ &= \lim_{h\to 0}\frac{\phi(r+h,t=0)-\phi(r,t=0)}{h} \\ &= \lim_{h\to 0}\frac{\phi_0(r+h)-\phi_0(r)}{h} \\ &= \phi_0'(r), \end{aligned}$$

so all we need is to get an initial condition for $\Pi(r,t)$. Making use of our above approximation, we derive an explicit equation for $\partial_t\phi(r,0)$,

$$\begin{aligned} \partial_t\left(r\phi\right)\left(r,0\right) &= r\partial_t\left(\phi\right)\left(r,0\right) \\ &= \partial_r\left(r\phi\right)\left(r,0\right) \\ &= \phi(r,0) + r\left(\partial_r(\phi)\right)\left(r,0\right) \\ &= \phi_0(r) + r\phi_0'(r). \end{aligned}$$

Therefore, an initial condition for $\Pi(r,t)$ can easily be computed in terms of our initial scalar pulse ϕ_0 as:

$$\begin{aligned} \Pi_0(r) &= \Pi(r,0) \\ &= \frac{a}{\alpha}\left(\partial_t\phi(r,0) - \beta\partial_r\phi(r,0)\right) \\ &= \frac{a}{\alpha}\left(\left[\frac{1}{r}\phi_0(r) + \phi_0'(r)\right] - \beta\phi_0'(r)\right) \\ &= \frac{a}{\alpha}\left(\frac{1}{r}\phi_0(r) + (1-\beta)\phi_0'(r)\right). \end{aligned}$$

7.5. A conserved mass for the model

We recall the stress energy tensor, T_{ab}, for the scalar field satisfying the wave equation is

$$T_{ab} = \nabla_a\phi\nabla_b\phi - \frac{1}{2}g_{ab}\nabla^c\phi\nabla_c\phi.$$

We consider a foliation, Σ_t, of spacetime with an associated unit normal field n^a. Let us assume that the spacetime has a timelike Killing vector field, t^a. Then we can define an energy-momentum 4-vector,

$$J^a \equiv T^{ab}t_b.$$

Which is conserved,

$$\nabla_a J^a = \nabla_a\left(T^{ab}t_b\right) = \left(\nabla_a T^{ab}\right)t_b + T^{ab}\left(\nabla_a t_b\right) = \underbrace{\left(\nabla_a T^{ab}\right)}_{=0}t_b + T^{ab}\underbrace{\nabla_{(a}t_{b)}}_{=0} = 0,$$

by virtue of the conservation of T^{ab} and Killing's equations. If we integrate $\nabla_a J^a$ over a spacetime volume and apply Gauss's theorem:

$$\int_V \nabla_a J^a = \int_{\partial V} J^a n_a = 0 = 0,$$

where ∂V is the (three-dimensional) boundary of the integration region, n^a is the normal vector to ∂V, and both integrals are taken with respect to the natural volume elements on the respective manifolds. If we now take our "Gaussian pillbox" to be the region bounded by any two hypersurfaces, Σ_t, $\Sigma_{t'}$, then assuming $J^a n_a \to 0$ at spatial infinity (the timelike part of the pillbox), we have:

$$\int_{\Sigma_t} J^a n_a - \int_{\Sigma_{t'}} J^a n_a = 0,$$

and since t and t' are chosen to be any value of our time coordinate, we can easily deduce that,

$$m_\infty = \int_{\Sigma_t} J^a n_a = \text{constant},$$

which means that our quantity m_∞, is the mass density integrated at each hypersurface, is constant.

Using our current $3+1$ metric, we have:

$$\begin{aligned} m_\infty &= \int_{\Sigma_t} J^a n_a \\ &= \int T^{\mu\nu}t_\nu n_\mu d\Sigma \\ &= \int T^\mu_\nu t^\nu n_\mu d\Sigma \\ &= \int (-\alpha T^t_t)(ar^2\sin^2\theta)drd\theta d\phi \\ &= 4\pi\int -r^2\alpha a T^t_t dr, \end{aligned}$$

where we have used $t^\mu = (1,0,0,0)$ and $n_\mu = (-\alpha,0,0,0)$. For the purposes of monitoring our calculation, it is convenient to define a space- and time-dependent "mass aspect" function, $m(r,t)$, via

$$m(r,t) = \int_{2M}^r \frac{dm}{d\tilde{r}}(r,t)d\tilde{r},$$

$$\frac{dm}{dr} \equiv -4\pi r^2 \alpha a T_t^t.$$

This function is not time-independent since we are integrating on subsets of our hypersurface, and thus we can calculate the mass thanks for our formula for the energy-momentum tensor T_{ab} of our scalar field satisfying the wave equation. Let us begin getting an explicit value of dm/dr to integrate.

We begin noting that the covariant derivative of any scalar field simplifies to the normal derivative, that is, $\nabla_\mu \phi = \partial_\mu \phi$. Also, since $\phi \equiv \phi(r, t)$, derivatives with respect to angular components will always be zero.

Let us get an expression for $\alpha a T_t^t$:

$$
\begin{aligned}
\alpha a T_t^t &= \alpha a \left(g^{\mu t} T_{\mu t}\right) \\
&= \alpha a \left(g^{tt} T_{tt} + g^{tr} T_{tr}\right) \\
&= \alpha a \left(g^{tt}\left[(\partial_t\phi)^2 - \frac{1}{2} g_{tt}\nabla^c\phi\nabla_c\phi\right] + g^{tr}\left[\partial_t\phi\partial_r\phi - \frac{1}{2} g_{tr}\nabla^c\phi\nabla_c\phi\right]\right) \\
&= \alpha a \left(g^{tt}(\partial_t\phi)^2 + g^{tr}\partial_t\phi\partial_r\phi - \frac{1}{2}\underbrace{\left(g^{tt}g_{tt} + g^{tr}g_{tr}\right)}_{=g^{\mu t}g_{\mu t}=\delta_t^t=1}\left[\nabla^c\phi\nabla_c\phi\right]\right) \\
&= \alpha a \left(g^{tt}(\partial_t\phi)^2 + g^{tr}\partial_t\phi\partial_r\phi - \frac{1}{2}\left[\nabla^c\phi\nabla_c\phi\right]\right) \\
&= \alpha a \left(-\frac{1}{\alpha^2}(\partial_t\phi)^2 + \frac{\beta}{\alpha^2}\partial_t\phi\partial_r\phi - \frac{1}{2}\left[\nabla^c\phi\nabla_c\phi\right]\right) \\
&= -\left(\frac{a}{\alpha}(\partial_t\phi)^2 - \frac{a\beta}{\alpha}\partial_t\phi\partial_r\phi + \frac{\alpha a}{2}\left[\nabla^c\phi\nabla_c\phi\right]\right) \\
&= -\left(\frac{a}{\alpha}(\partial_t\phi)^2 - \frac{a\beta}{\alpha}\partial_t\phi\partial_r\phi + \frac{\alpha a}{2}\left[\nabla^t\phi\nabla_t\phi + \nabla^r\phi\nabla_r\phi\right]\right) \\
&= -\left(\frac{a}{\alpha}(\partial_t\phi)^2 - \frac{a\beta}{\alpha}\partial_t\phi\partial_r\phi + \frac{\alpha a}{2}\left[g^{t\mu}\nabla_\mu\phi\nabla_t\phi + g^{r\nu}\nabla_\nu\phi\nabla_r\phi\right]\right) \\
&= -\left(\frac{a}{\alpha}(\partial_t\phi)^2 - \frac{a\beta}{\alpha}\partial_t\phi\partial_r\phi + \frac{\alpha a}{2}\left[g^{tt}(\nabla_t\phi)^2 + \text{t}2g^{tr}\nabla_t\phi\nabla_r\phi + g^{rr}(\nabla_r\phi)^2\right]\right) \\
&= -\left(\frac{a}{\alpha}(\partial_t\phi)^2 - \frac{a\beta}{\alpha}\partial_t\phi\partial_r\phi + \frac{\alpha a}{2}\left[g^{tt}(\partial_t\phi)^2 + \text{t}2g^{tr}\partial_t\phi\partial_r\phi + g^{rr}(\partial_r\phi)^2\right]\right) \\
&= -\left(\left(\frac{a}{\alpha} + \frac{\alpha a}{2}g^{tt}\right)(\partial_t\phi)^2 + \frac{\alpha a}{2}g^{rr}(\partial_r\phi)^2 + \left(-\frac{a\beta}{\alpha} + \alpha a g^{tr}\right)\partial_t\phi\partial_r\phi\right) \\
&= -\left(\begin{array}{l}\left(\frac{a}{\alpha} + \frac{\alpha a}{2}(-\frac{1}{\alpha^2})\right)(\partial_t\phi)^2 + \left(\frac{\alpha a}{2}\left(\frac{1}{a^2} - \frac{\beta^2}{\alpha^2}\right)\right)(\partial_r\phi)^2 \\ \quad + (-\frac{a\beta}{\alpha} + \alpha a(\frac{\beta}{\alpha^2}))\partial_t\phi\partial_r\phi\end{array}\right) \\
&= -\left(\left(\frac{a}{\alpha} - \frac{a}{2\alpha}\right)(\partial_t\phi)^2 + \left(\frac{\alpha}{2a} - \frac{a\beta^2}{2\alpha}\right)(\partial_r\phi)^2 + (-\frac{a\beta}{\alpha} + \frac{a\beta}{\alpha})\partial_t\phi\partial_r\phi\right) \\
&= -\left(\left(\frac{a}{2\alpha}\right)(\partial_t\phi)^2 + \left(\frac{\alpha}{2a} - \frac{a\beta^2}{2\alpha}\right)(\partial_r\phi)^2 + (-\frac{a}{2\alpha} + \frac{a}{2\alpha})2\beta\partial_t\phi\partial_r\phi\right) \\
&= -\left(\left(\frac{a}{2\alpha}\right)(\partial_t\phi)^2 + \left(\frac{\alpha}{2a} + \frac{a\beta^2}{2\alpha} - \frac{a\beta^2}{\alpha}\right)(\partial_r\phi)^2 + (-\frac{a}{2\alpha} + \frac{a}{2\alpha})2\beta\partial_t\phi\partial_r\phi\right) \\
&= -\left(\begin{array}{l}\left(\frac{a}{2\alpha}\right)\left[(\partial_t\phi)^2 - 2\beta\partial_t\phi\partial_r\phi + (\beta\partial_r\phi)^2\right] + \left(\frac{\alpha}{2a} - \frac{a\beta^2}{\alpha}\right)(\partial_r\phi)^2 \\ \quad + \frac{a}{2\alpha}2\beta\partial_t\phi\partial_r\phi\end{array}\right) \\
&= -\left(\left(\frac{a}{2\alpha}\right)(\partial_t\phi - \beta\partial_r\phi)^2 + \left(\frac{\alpha}{2a} - \frac{a\beta^2}{\alpha}\right)(\partial_r\phi)^2 + \frac{a}{\alpha}2\beta\partial_t\phi\partial_r\phi\right) \\
&= -\left(\left(\frac{\alpha}{2a}\right)\left(\frac{a}{\alpha}\right)^2(\partial_t\phi - \beta\partial_r\phi)^2 + \left(\frac{\alpha}{2a} - \frac{a\beta^2}{\alpha}\right)(\partial_r\phi)^2 + \frac{a}{\alpha}2\beta\partial_t\phi\partial_r\phi\right) \\
&= -\left(\frac{\alpha}{2a}\left[\frac{a}{\alpha}(\partial_t\phi - \beta\partial_r\phi)\right]^2 + \left(\frac{\alpha}{2a}\right)(\partial_r\phi)^2 + \beta(\partial_r\phi)(\partial_t\phi - \beta\partial_r\phi)\right) \\
&= -\left(\frac{\alpha}{2a}\left[\Pi^2 + \Phi^2\right] + \beta\Phi\Pi\right).
\end{aligned}
$$

So that finally, we end up with an expression for our mass radial density,

$$\begin{aligned}\frac{dm}{dr} &= -4\pi r^2 \alpha a T_t^t \\ &= -4\pi r^2 \cdot \left[-\left(\frac{\alpha}{2a} \left[\Pi^2 + \Phi^2 \right] + \beta \Phi \Pi \right) \right] \\ &= 4\pi r^2 \left(\frac{\alpha}{2a} \left[\Pi^2 + \Phi^2 \right] + \beta \Phi \Pi \right).\end{aligned}$$

We notice that to integrate this expression, we must first solve our equation to get our values for Π and Φ, which is what we will do in the next section.

7.6. Solution to the equations of motion

Let us write a program that solves the wave equation on a Schwarzschild background in IEF coordinates. This means that we will solve the system,

$$\begin{cases} \partial_t \Phi = \partial_r(\beta \Phi + \frac{\alpha}{a} \Pi) \\ \partial_t \Pi = \frac{1}{r^2} \partial_r (r^2 (\beta \Pi + \frac{\alpha}{a} \Phi)) \\ \Phi(r,t) \equiv \partial_r \phi \\ \Pi(r,t) \equiv \frac{a}{\alpha}(\partial_t \phi - \beta \partial_r \phi), \end{cases}$$

where $\alpha = \alpha(r) = \left(\frac{r}{r+2M} \right)^{1/2}$, $a = a(r) = \left(\frac{r}{r+2M} \right)^{-1/2}$ and $\beta = \beta(r) = \frac{2M}{r+2M}$.

We will begin changing our notation so our code looks tidier. Our new variables are $X(r,t) \equiv \Phi(r,t)$ and $Y(r,t) \equiv \Pi(r,t)$. This also helps us work on our Python script, for which we work better with one-letter named variables, hence X and Y.

Our new system with its initial conditions are, in terms of X and Y,

$$\begin{cases} X(r,t) \quad , Y(r,t) \in [2M, R] \times [0, T] \\ \partial_t X = \partial_r(\beta X + \frac{\alpha}{a} Y) \\ \partial_t Y = \frac{1}{r^2} \partial_r (r^2 (\beta Y + \ \frac{\alpha}{a} X)) \\ X(r,0) = \phi_0'(r) \\ Y(r,0) = \frac{a}{\alpha} \left(\frac{1}{r} \phi_0(r) + (1-\beta) \phi_0'(r) \right) \end{cases},$$

and finally, we replace for our values of $\alpha(r)$,$a(r)$ and $\beta(r)$,

$$\begin{cases} \partial_t X = \partial_r (\frac{2M}{r+2M} X + \frac{r}{r+2M} Y) \\ \partial_t Y = \frac{1}{r^2} \partial_r (\frac{2Mr^2}{r+2M} Y + \frac{r^3}{r+2M} X)) \\ X(r,0) = \phi_0'(r) \\ Y(r,0) = \frac{r+2M}{r^2} \phi_0(r) + \phi_0'(r) \end{cases}.$$

Now, to solve this system, we use a finite differences scheme on the spatial-temporal grid $[2M, R] \times [0, T]$.

We will assume r lies between p and q, and use this to create our partition and define a function

that will give us our value of r for each i. That is,

$$0 < p \leq r \leq q \quad \implies \quad \pi(i) \equiv p + i\left(\frac{q-p}{N}\right) = p + i\Delta r \quad i \in \{1, ..., N-1\}.$$

To use better notation, we define two functions that represent implicit and explicit discretization.

$$\begin{cases} fx(X,Y)_i^n = \left[\frac{2M}{\pi(i)} X_i^n + \frac{\pi(i)}{\pi(i)+2M} Y_i^n\right] \\ fy(X,Y)_i^n = \left[\frac{2M\pi(i)^2}{\pi(i)+2M} Y_i^n + \frac{\pi(i)^3}{\pi(i)+2M} X_i^n\right] \end{cases}.$$

Our plan is to use the Crank-Nicholson scheme, but we will generalize to a θ-scheme and replace $\theta = 1/2$ when needed. Also, our spatial derivatives will be $\mathcal{O}(h^2)$ centred finite difference approximations in the interior of the domain, with $\mathcal{O}(h^2)$ forwards and backwards approximations for spatial derivatives at the boundaries.

Center terms:

From now on, we will adopt a new variable $\theta' \equiv (1-\theta)$, that will represent the explicit part of our discretization scheme. This helps to make our equations a bit smaller. Our θ-scheme for discretization finally reads,

$$\begin{cases} \frac{X_i^{n+1} - X_i^n}{\Delta t} = \theta\left[\frac{fx(X,Y)_{i+1}^{n+1} - fx(X,Y)_{i-1}^{n+1}}{2\Delta r}\right] + \theta'\left[\frac{fx(X,Y)_{i+1}^{n} - fx(X,Y)_{i-1}^{n}}{2\Delta r}\right] \\ \frac{Y_i^{n+1} - Y_i^n}{\Delta t} = \theta \cdot \frac{1}{\pi(i)^2}\left[\frac{fy(X,Y)_{i+1}^{n+1} - fy(X,Y)_{i-1}^{n+1}}{2\Delta r}\right] + \theta' \cdot \frac{1}{\pi(i)^2}\left[\frac{fy(X,Y)_{i+1}^{n} - fy(X,Y)_{i-1}^{n}}{2\Delta r}\right] \end{cases}.$$

Let us write the equation for the pentadiagonal matrix equation that will be used to calculate the center terms (since boundary points are to be calculated with forward and backwards schemes)

To help with notation, we define our CFL number to be $\varphi = \frac{\Delta t}{2\Delta r}$. Luckily, since we will be using a Crank-Nicolson scheme, our solution will be stable for any value of φ, though we will be fair in our simulations and set it to be small nevertheless. The equations are,

$$\begin{cases} X_i^{n+1} - \theta\varphi\frac{2M}{\pi(i+1)} X_{i+1}^{n+1} + \theta\varphi\frac{2M}{\pi(i-1)} X_{i-1}^{n+1} \\ \quad - \theta\varphi\frac{\pi(i+1)}{\pi(i+1)+2M} Y_{i+1}^{n+1} + \theta\varphi\frac{\pi(i-1)}{\pi(i-1)+2M} Y_{i-1}^{n+1} \\ \qquad = X_i^n + \theta'\varphi\frac{2M}{\pi(i+1)} X_{i+1}^n - \theta'\varphi\frac{2M}{\pi(i-1)} X_{i-1}^n \\ \qquad\quad + \theta'\varphi\frac{\pi(i+1)}{\pi(i+1)+2M} Y_{i+1}^n - \theta'\varphi\frac{\pi(i-1)}{\pi(i-1)+2M} Y_{i-1}^n, \\ \\ Y_i^{n+1} - \frac{\theta\varphi}{\pi(i)^2}\frac{2M\pi(i+1)^2}{\pi(i+1)+2M} Y_{i+1}^{n+1} + \frac{\theta\varphi}{\pi(i)^2}\frac{2M\pi(i-1)^2}{\pi(i-1)+2M} Y_{i-1}^{n+1} \\ \quad - \frac{\theta\varphi}{\pi(i)^2}\frac{\pi(i+1)^3}{\pi(i+1)+2M} X_{i+1}^{n+1} + \frac{\theta\varphi}{\pi(i)^2}\frac{\pi(i-1)^3}{\pi(i-1)+2M} X_{i-1}^{n+1} \\ \qquad = Y_i^n + \frac{\theta'\varphi}{\pi(i)^2}\frac{2M\pi(i+1)^2}{\pi(i+1)+2M} Y_{i+1}^n - \frac{\theta'\varphi}{\pi(i)^2}\frac{2M\pi(i-1)^2}{\pi(i-1)+2M} Y_{i-1}^n \\ \qquad\quad + \frac{\theta'\varphi}{\pi(i)^2}\frac{\pi(i+1)^3}{\pi(i+1)+2M} X_{i+1}^n - \frac{\theta'\varphi}{\pi(i)^2}\frac{\pi(i-1)^3}{\pi(i-1)+2M} X_{i-1}^n. \end{cases}$$

This is equivalent to a matrix equation:

$$AZ^{n+1} = BZ^n \quad \text{where} \quad Z = [X, Y]^T.$$

Boundary terms:
We recall that the forward and backward $\mathcal{O}(h^2)$ approximations are:

$$\begin{cases} f'(x)_{\text{forward}} = \frac{-3f(x)+4f(x+\Delta x)-f(x+2\Delta x)}{2\Delta x}, \\ f'(x)_{\text{backward}} = \frac{3f(x)-4f(x-\Delta x)+f(x-2\Delta x)}{2\Delta x}. \end{cases}$$

This leads us to the final equations for $i = 0$

$$\begin{aligned} & X_0^{n+1} + \theta\varphi\frac{6M}{\pi(0)}X_0^{n+1} - \theta\varphi\frac{8M}{\pi(1)}X_1^{n+1} + \theta\varphi\frac{2M}{\pi(2)}X_2^{n+1} \\ & \quad + \theta\varphi\frac{3\pi(0)}{\pi(0)+2M}Y_0^{n+1} - \theta\varphi\frac{4\pi(1)}{\pi(1)+2M}Y_1^{n+1} + \theta\varphi\frac{\pi(2)}{\pi(2)+2M}Y_2^{n+1} \\ & \qquad = \begin{aligned} & X_0^n - \theta'\varphi\frac{6M}{\pi(0)}X_0^n + \theta'\varphi\frac{8M}{\pi(1)}X_1^n - \theta'\varphi\frac{2M}{\pi(2)}X_2^n \\ & \quad - \theta'\varphi\frac{3\pi(0)}{\pi(0)+2M}Y_0^n + \theta'\varphi\frac{4\pi(1)}{\pi(1)+2M}Y_1^n - \theta'\varphi\frac{\pi(2)}{\pi(2)+2M}Y_2^n. \end{aligned} \end{aligned}$$

For $i = N$, that is, our boundary condition at $r = R$, we will use that in the limit $r \to \infty$ the wave equation becomes,

$$\partial_{tt}(r\phi) = \partial_{rr}(r\phi).$$

The outgoing solution of this equation is:

$$(r\phi)(r, t) = g(t - r),$$

where one considers the characteristic speeds for ingoing and outgoing solutions (which we have been implicitly assuming to be equal to one, that is, $c_+^2 = c_-^2 = c^2 = 1$), and introduce them into our equation,

$$(r\phi)(r, t) = g(c_+t - r),$$

where,

$$c_+ = -\beta + \frac{\alpha}{a} = \frac{r - 2M}{r + 2M}.$$

The above equations can also be expressed as:

$$\partial_t(r\phi) + \left(-\beta + \frac{\alpha}{a}\right)\partial_r(r\phi) = 0.$$

This will be imposed as a boundary condition on $r = R$.

We notice that for $r \to \infty$, the spherical wave equation becomes

$$\begin{cases} \partial_t\Phi(r, t) = \partial_r\Pi(r, t) \\ \partial_t\Pi(r, t) = \partial_r\Phi(r, t) \end{cases},$$

with $\Phi = \partial_r\phi$, and $\Pi = \partial_t\phi$. Again, we use our new variable names $\Phi \equiv X$ and $\Pi \equiv Y$ for the next

equations. Using the above formula for the boundary condition at $r = R$, we get that,

$$r\partial_t\phi + \left(-\beta + \frac{\alpha}{a}\right)[\phi + r\partial_r\phi] = 0.$$

Dividing by r, we get that,

$$\partial_t\phi + \left(-\beta + \frac{\alpha}{a}\right)\left[\frac{1}{r}\phi + \partial_r\phi\right] = 0,$$

which, for big enough r, leaves us with the approximation,

$$\partial_t\phi + \left(-\beta + \frac{\alpha}{a}\right)\partial_r\phi = 0.$$

Considering our change of variables, we represent this as:

$$Y + \left(-\beta + \frac{\alpha}{a}\right)X = 0.$$

Let us begin differentiating with respect to t. Using the fact that $\partial_t X(r,t) = \partial_r Y(r,t)$, we get an equation for Y,

$$\partial_t Y + \left(-\beta + \frac{\alpha}{a}\right)\partial_t X = \partial_t Y + \left(-\beta + \frac{\alpha}{a}\right)\partial_r Y = 0.$$

Now, using the fact that $\partial_t Y(r,t) = \partial_r X(r,t)$, we get an equations for X,

$$\partial_t Y + \left(-\beta + \frac{\alpha}{a}\right)\partial_t X = \partial_r X + \left(-\beta + \frac{\alpha}{a}\right)\partial_t X = 0.$$

Finally, using that $(-\beta + \frac{\alpha}{a}) = \frac{r-2M}{r+2M}$, we get the system of equations at the boundary $r = R$:

$$\begin{cases} \partial_t X = -\frac{r+2M}{r-2M}\partial_r X, \\ \partial_t Y = -\frac{r-2M}{r+2M}\partial_r Y. \end{cases}$$

We will use an $\mathcal{O}(h^2)$ backward approximation at this point, therefore our equation reads.

$$\begin{cases} \frac{X_N^{n+1} - X_N^n}{\Delta t} = CN\left(-\frac{R+2M}{R-2M}\left[\frac{3X_N^n - 4X_{N-1}^n + X_{N-2}^n}{2\Delta r}\right]\right), \\ \frac{Y_N^{n+1} - Y_N^n}{\Delta t} = CN\left(-\frac{R-2M}{R+2M}\left[\frac{3Y_N^n - 4Y_{N-1}^n + Y_{N-2}^n}{2\Delta r}\right]\right). \end{cases}$$

Where our $CN(\cdot)$ function does a Crank Nicholson θ-scheme, and will just serve to make our equation tidier.

Let us save some notation using $c \equiv -\frac{R+2M}{R-2M}$.

Finally, using that $R \equiv q$ in our code (our last value of r), that $\varphi \equiv \frac{\Delta t}{2\Delta x}$ and making the full θ-scheme equation, and simplifying all terms, our equation at the boundary will finally read as:

$$\begin{cases} (1-3c\theta\varphi)X_N^{n+1} + 4c\theta\varphi X_{N-1}^{n+1} - c\theta\varphi X_{N-2}^{n+1} \\ \qquad\qquad = (1+3c\theta'\varphi)X_N^n - 4c\theta'\varphi X_{N-1}^n + c\theta'\varphi X_{N-2}^n, \\ \\ (1-3\dfrac{1}{c}\theta\varphi)Y_N^{n+1} + 4\dfrac{1}{c}\theta\varphi Y_{N-1}^{n+1} - \dfrac{1}{c}\theta\varphi Y_{N-2}^{n+1} \\ \qquad\qquad = (1+3\dfrac{1}{c}\theta'\varphi)Y_N^n - Y\dfrac{1}{c}\theta'\varphi Y_{N-1}^n + \dfrac{1}{c}\theta'\varphi Y_{N-2}^n. \end{cases}$$

We develop our finite difference scheme using all the equations and approximations computed on a Python script, which is added at the 'Codes' section at the end of this thesis.

7.6.1. Analyzing solutions for the equations of motion

We now focus on running our finite differences scheme to solve the previous equations of motion. Firstly though, it is important to study what should happen in our simulations. The wave equation in this form is something we have previously studied, but in a non-flat background it should take a different form due to the curvature of the geodesics governing the scalar pulses. Let us begin noticing a couple things from our equations, which we recall are,

$$\begin{cases} \partial_t \Phi = \partial_r(\beta\Phi + \frac{\alpha}{a}\Pi) \\ \partial_t \Pi = \frac{1}{r^2}\partial_r(r^2(\beta\Pi + \frac{\alpha}{a}\Phi)) \\ \Phi(r,t) \equiv \partial_r \phi \\ \Pi(r,t) \equiv \frac{a}{\alpha}(\partial_t\phi - \beta\partial_r\phi) \\ \Phi(r,0) = \phi_0'(r) \\ \Pi(r,0) = \frac{a}{\alpha}(\frac{1}{r}\phi_0(r) + (1-\beta)\phi_0'(r)), \end{cases}$$

where $\alpha = \alpha(r) = \left(\frac{r}{r+2M}\right)^{1/2}$, $a = a(r) = \left(\frac{r}{r+2M}\right)^{-1/2}$ and $\beta = \beta(r) = \frac{2M}{r+2M}$.

First, we notice as $r \to \infty$, get get that $\frac{\alpha}{a} \sim 1$ and $\beta \sim \frac{1}{r}$. This approximates our equation to the following system:

$$\begin{cases} \partial_t \Phi \sim \partial_r(\frac{1}{r}\Phi + \Pi) \\ \partial_t \Pi \sim \partial_r(\frac{1}{r}\Pi + \Phi)) \\ \Phi(r,t) \sim \partial_r \phi \\ \Pi(r,t) \sim \partial_t\phi - \frac{1}{r}\partial_r\phi \\ \Phi(r,0) \sim \phi_0'(r) \\ \Pi(r,0) \sim \frac{1}{r}\phi_0(r) + \phi_0'(r). \end{cases}$$

So for big r, this system resembles our usual wave equation, but for small r we can see extra terms are taken into account. Also, if we choose an initial condition $\phi_0(r)$ with approximately compact support in big r, then both Π and Φ will start as $\phi_0'(r)$.

Let us generate a Python script that solves our original system, with initial condition depending on

$$\phi_0(r; A, r_0, \Delta) = A\exp\left\{(-(\frac{r-r_0}{\Delta})^2)\right\},$$
$$\phi_0'(r; A, r_0, \Delta) = -2A\frac{r-r_0}{\Delta^2}A\exp\left\{(-(\frac{r-r_0}{\Delta})^2)\right\}.$$

So our initial conditions for Φ and Π are:

$$\Phi(r,0) = \phi_0'(r),$$
$$\Pi(r,0) = \frac{a}{\alpha}(\frac{1}{r}\phi_0(r) + (1-\beta)\phi_0'(r)).$$

We run our Python code, setting $A = 1$, $r_0 = 50$, $\Delta = 1$, and $T = 70$ (T was chosen so that the pulse touches the event horizon eventually), and plot the results for both the Φ (Phi) wave (Fig. 7.1) and the Π (Pi) wave (Fig. 7.2).

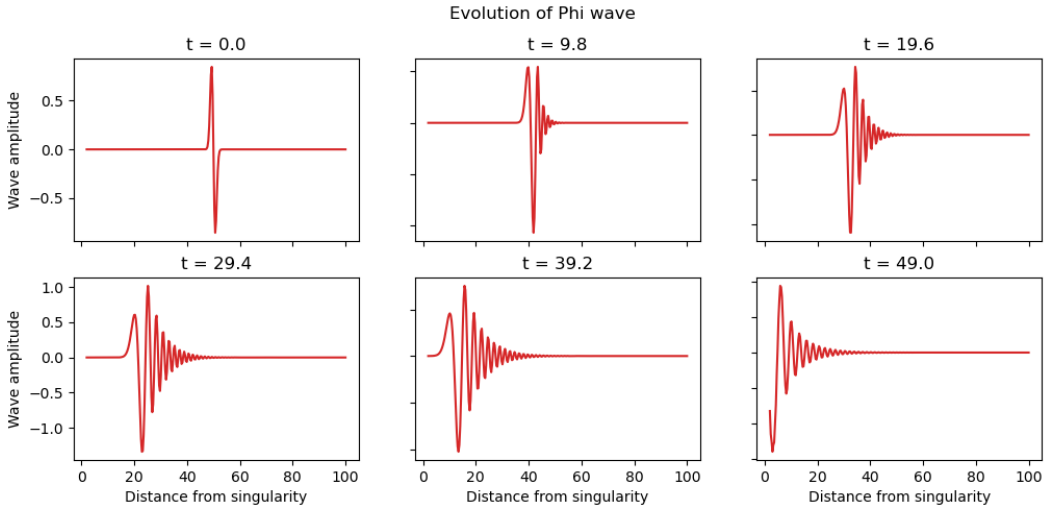

Figura 7.1: Solutions of Φ for different times. We start with our differentiated scalar pulse, and it evolves, moves and grows towards the event horizon, leaving a trail behind.

The metric near the black hole distorts the pulse, and increases the wavelength as it gets closer to the event horizon, thus reducing its frequency due to gravitational redshift.

The pulse does not travel in usual wave motion. Remembering that the initial condition for Φ was the spatial derivative of a Gaussian pulse, the scalar pulse is actually over the r-axis. By noticing what happens as r approaches the event horizon, different frequencies of the r *derivative* start to appear, which would distort our spatial pulse by spreading it apart in a wave-like manner.

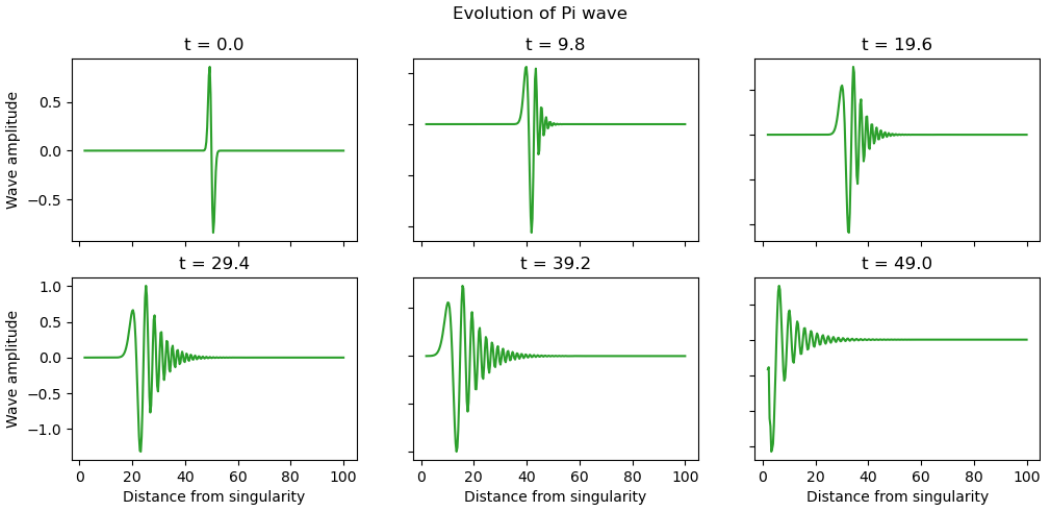

Figura 7.2: Solutions of Π for different times. We start with our differentiated scalar pulse plus a small term, and it evolves, moves and grows towards the event horizon, leaving a trail behind.

The results look pretty similar to one another. This is because both equations are approximately symmetrical for big r (which in our domain $2M < r$ one could say r is still 'big'), and because the initial condition correction term is a Gaussian term divided by r, but since this term has its support for already big r (that is, r around $r_0 = 50$), then it changes little of our equation.

We can check there has been no mistake in plotting both solutions by showing some plots of their absolute difference, and checking if they in fact are the same solutions.

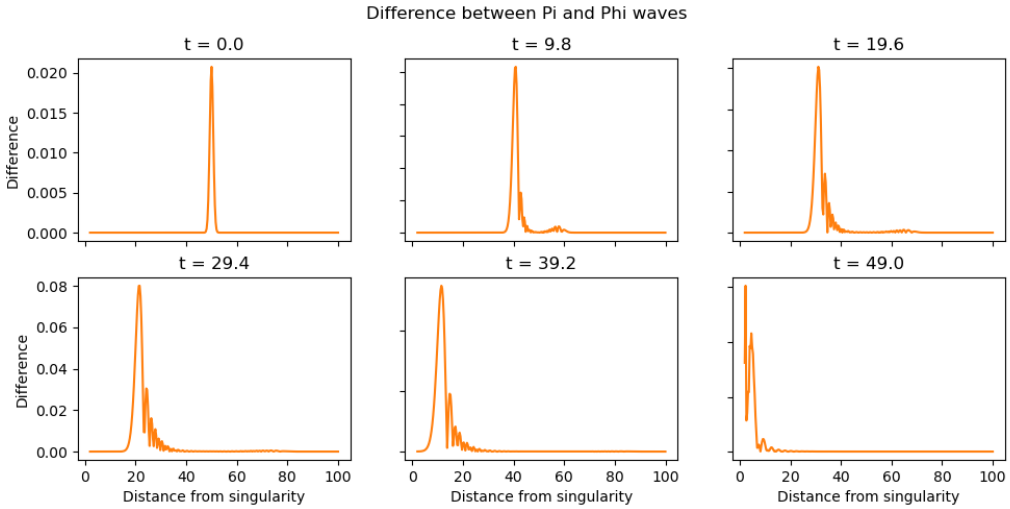

Figura 7.3: Difference between Φ and Π solutions for various time iterations. Since they are not zero everywhere, we can assume they in fact are solutions for our two variables.

The initial difference is of really low order, and thus almost not noticeable by eye in our first plots. As time evolves, the differences grow and we can check that, effectively, the solutions are different.

7.6.2. Analysis of the scalar wave solution

For the analysis of the mass function, we will work with our initial Gaussian wavepacket being really narrow in space, thus having a value of $\Delta = 1$, but let's devote some time into analyzing our original solution $\phi(r, t)$. Let's begin remembering we are studying our covariant wave equation 7.1 on a Schwarzschild metric background written in ingoing Eddington-Finkelstein coordinates (7.2). The regularity of this metric at the event horizon ($r = 2M$) makes numerical simulations more stable near the horizon.

Let's notice what happens for numerical solutions when the wavepacket is very narrow, as in figures 7.4 and 7.5. The pulse begins to 'wiggle' really fast as soon as it starts moving towards the event horizon of our black hole. When this narrow wavepacket propagates through a region of spacetime with varying gravitational potential, different parts of the wavepacket experience vastly different potentials as they traverse different radial positions, making it wiggle due to the interference caused by these rapid changes on potential. We also see how the amplitude of our pulse increases as it gets near the black hole. In the vicinity of the event horizon, an observer would perceive an increase in the apparent amplitude of the wave due to the redshifting. The wave might appear to be growing in amplitude, a behavior sometimes referred as 'superradiance'. As the wave crosses the event horizon (where no boundary conditions were imposed, just the usual wave equation

with boundary derivatives, which we can do by the regularity of our metric at this boundary), it is essentially entering a region from which no signals escape, effectively being absorbed by the black hole and disappearing. Let's also notice no waves are being bounced off our black
hole, a matter that makes sense in the context of the scheme we are studying, in which our coordinates forces our pulse to be as ingoing as possible.

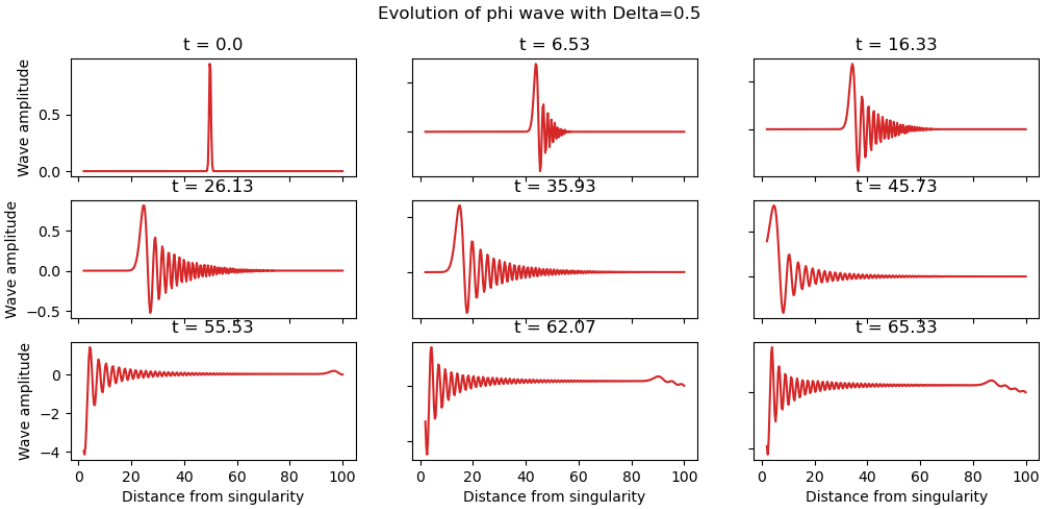

Figura 7.4: Solution of our wave equation, where the initial scalar pulse's Δ has been set to 0.5.

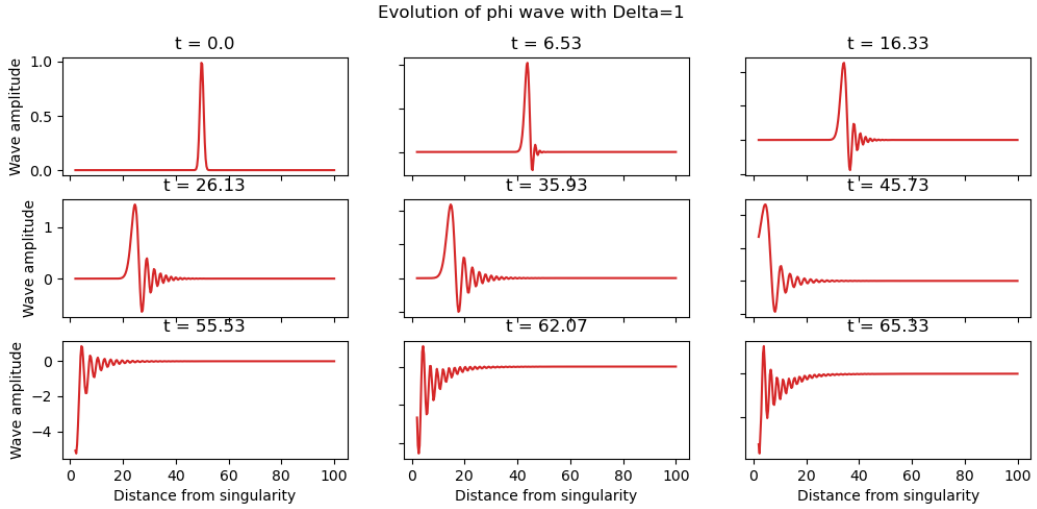

Figura 7.5: Solution of our wave equation, where the initial scalar pulse's Δ has been set to 1.

What happens for higher values of Δ is quite interesting. By having a bigger value of Δ, we notice that our initial conditions for Φ and Π (7.3) begin as a low amplitude wave. Since this works approximately as a normal wave equation, this low amplitude wave moves throughout spacetime, and since $\Phi\ \partial_r\phi$, this means that our wavepacket experiences small change throughout. Let's plot different time steps, from the initial condition up until the wave is almost fully engorged by the black hole, in figures 7.6, 7.7 and 7.8, where our values for Δ are $4, 10$ and 30 respectively.

As seen in these figures, the wavepacket is wider and more spread out in its position space. These

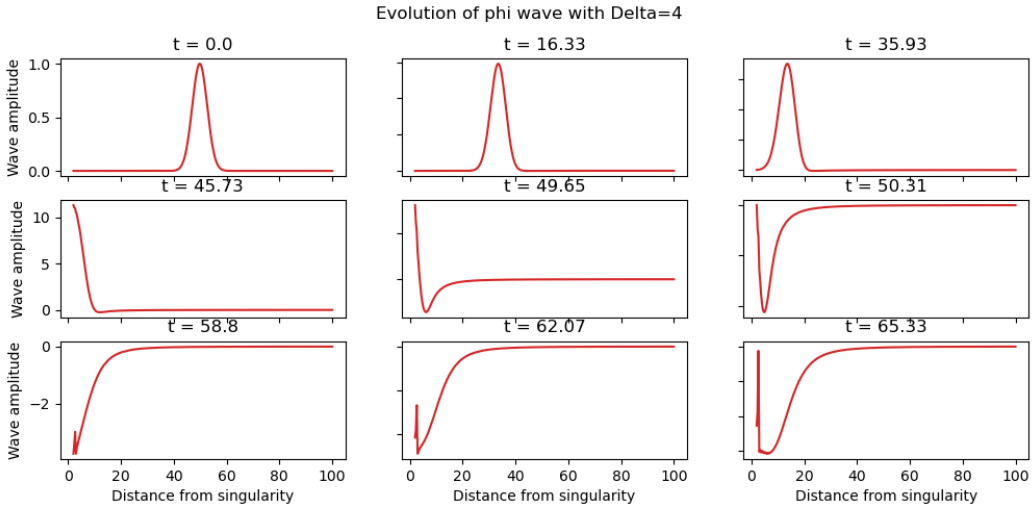

Figura 7.6: Solution of our wave equation, where the initial scalar pulse's Δ has been set to 4.

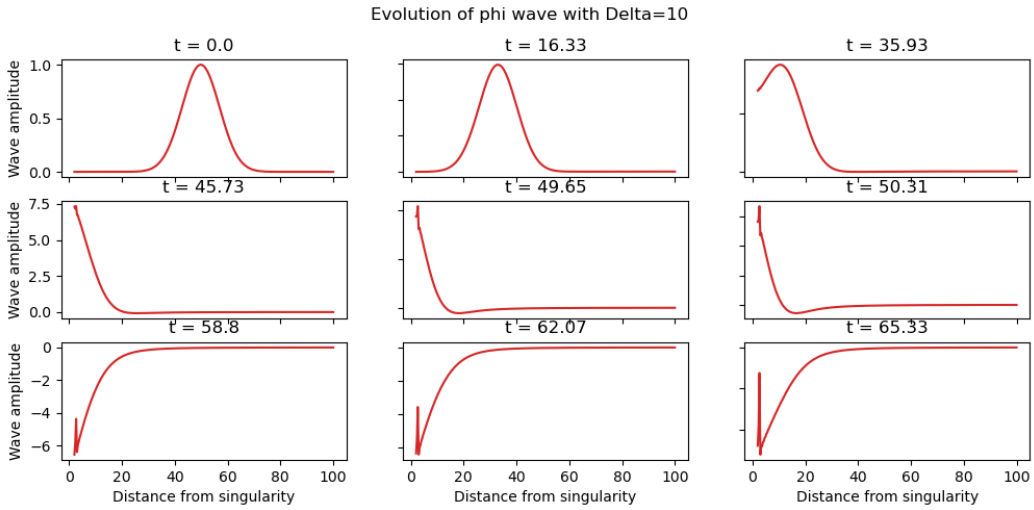

Figura 7.7: Solution of our wave equation, where the initial scalar pulse's Δ has been set to 10.

wavepackets are indeed affected by the gravitational potentials of our black hole, but its width influences its sensitivity to local changes in the potential. The apparent smoothness in the evolution of a wider wavepacket is a result of reduced sensitivity to rapid variations of the gravitational potential. As discussed above, the wave moves towards the black hole almost describing a flat wave solution (due to large values of Δ), reducing the impact of rapid variations of the curvature of spacetime described by the metric behind our equation.

There's another thing that we can notice about these figures. When the wave arrives to the event horizon, it begins to change sign (in the y-axis, which can be understood as the displacement value of the wave). This can be associated with the gravitational redshift near the event horizon. When the wave crosses the event horizon, the redshift can cause a phase shift in the wave, leading to a change in the sign of the amplitude, transitioning its behavior as it enters the strong gravitational field near the black hole. This effect is not seen clearly on our last figure 7.8,

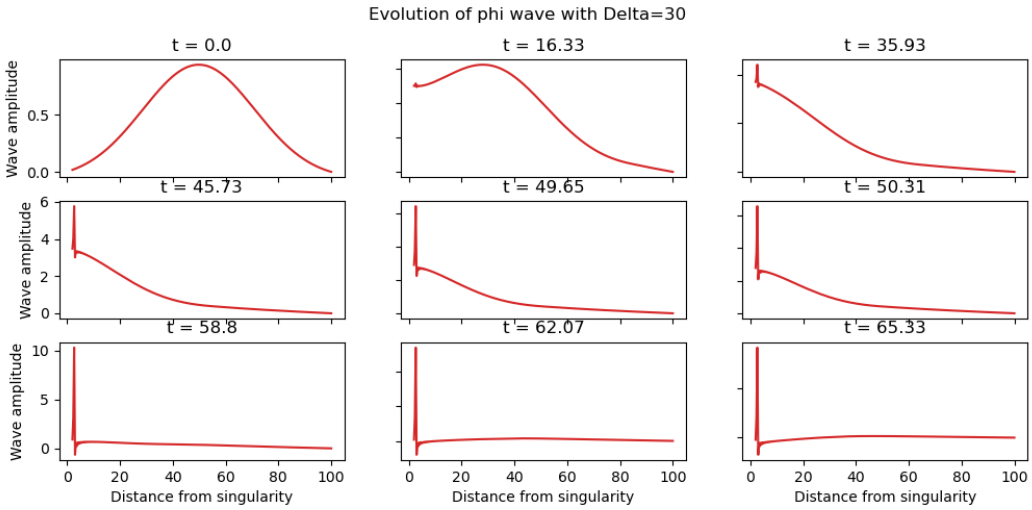

Figura 7.8: Solution of our wave equation, where the initial scalar pulse's Δ has been set to 30.

since the wave is initially so scattered throughout space, that the curvature around the spacetime averages the effect throughout our pulse, making it seem as it just enters the event horizon smoothly.

One last thing to note, is the strange flat vertical line that begins to form while our pulse is entering the black hole. These are numerical 'errors' due to the big spatial discretization used near the black hole for our finite differences scheme. Our original wave $\Phi = \partial_r \phi$ moves toward the black hole growing largely in amplitude, and when we are integrating this big value in our space direction (the radial axis), it translates to extreme variations of the amplitude of ϕ. When considering smaller meshes in our spatial dimension, one could expect smoother growing waves close to the event horizon, which are the remains of our pulse closely entering our black hole.

In conclusion, the metric governing our wave equation forces two things: (1) our pulse be ingoing, and (2) the pulse be regular near and in the event horizon. These are characteristics of our specific wave equation numerical scheme, and its plots can be explained using the background metric's characteristics. We can therefore relate the behavior of our wave simulations, with what the background original metric represents. For narrow wavepackets, the redshift caused by rapid gravitational potential variations (due to our initial conditions for our first-order-in-time wave equation system), cause a 'wiggling' of our wave, changing its sign constantly and finally making its amplitude grow bigger near the event horizon. For more spread out wavepackets, the initial conditions again play a role on explaining the dynamics of our wave, making local changes really small. Physically, the wider wavepackets experience an averaging of the local gravitational potential variations, thus making them move smoothly towards the black hole.

7.7. Solution to the conserved mass model

We recall that we have an expression for $\frac{dm}{dr}$ in terms of Φ and Π, which is

$$\frac{dm}{dr} = 4\pi r^2 \left(\frac{\alpha}{2a} \left(\Phi^2 + \Pi^2 \right) + \beta \Phi \Pi \right).$$

We can integrate our solutions numerically, to get an approximation of what our conserved mass is at each time step, and study it around our black hole. Let us graph our integrated function $m(r,t)$ at different time iterations (Fig. 7.9). The first six iterations correspond to the scalar pulses plotted in the previous section, while the other three correspond to extra time steps that are added to understand the physics behind the problem.

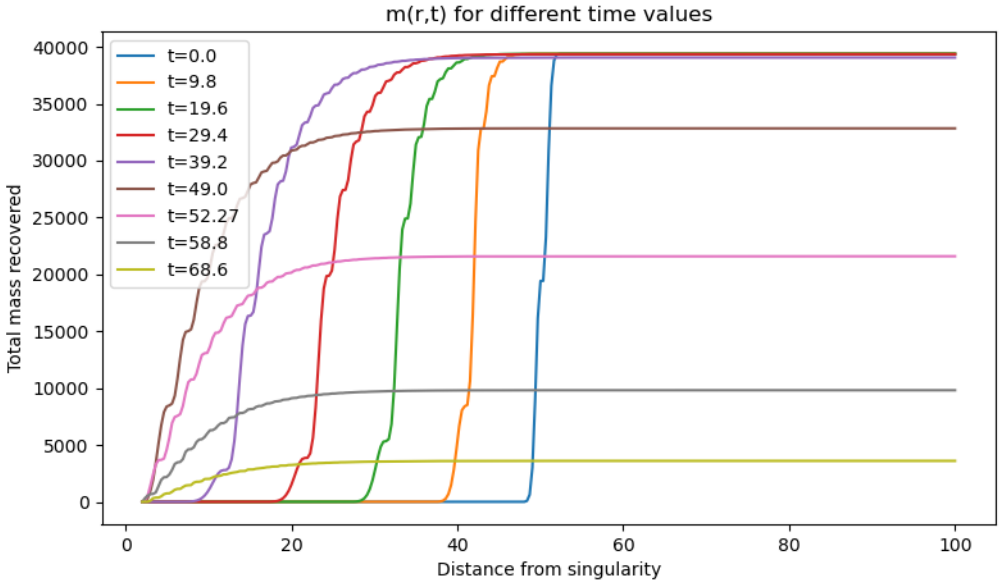

Figura 7.9: The mass around the black hole for different time steps.

Let us begin exploring just what do we mean by 'mass around the black hole'. When the scalar pulse was emitted, we can derive its mass in terms of its momentum, which we know. By studying each graph, for a fixed time, we notice that one can compute how much of the initial mass can still be measured at each distance from the black hole. Whenever Φ and Π are non-zero (that is, starting from our Gaussian pulse and watching it bend toward the black hole, leaving a small trail behind), terms will be contributing for the mass function. As the pulse moves toward the black hole, the mass recovered (or better said, measurable) from the pulse will depend on smaller r each time. For big enough r, the mass function converges towards the total mass of the scalar pulse that we can measure, that is, the mass that hasn't been absorbed by the black hole.

Comparing results for our plots of Φ and Π waves, we notice that for the first 5 time steps, the waves do not yet reach the event horizon. Also, it does not matter that the waves are stretched and leaving lower amplitude waves; we check that the total mass is always the same for those 5 time steps. At the 6th time step however, we check that both Φ and Π waves get to the event horizon and start entering it. Although it reaches the end of our simulation domain, boundary conditions were set to explain the physics of what would happen to a wave entering it. As soon as the waves start to touch the event horizon, we notice that the total mass recovered starts diminishing, that means that it has entered the point of no return. Further time steps are plotted to check that, as time passes, less and less mass is to be recovered, until it must finally reach a value of 0.

Capítulo 8

Using the Einstein Toolkit

8.1. Introduction

The Einstein Toolkit is an open-source community-driven software framework for the numerical simulation of astrophysical systems, particularly in the field of general relativity. It provides a set of tools and libraries that enable researchers to develop and run simulations of phenomena such as black hole collisions, neutron star mergers, and gravitational waves.

This software is based on the Cactus framework, where a collection of scripts called 'thorns' are selected for each algorithm we would desire. The toolkit is designed to be flexible, allowing users to customize and extend its functionality according to their specific research needs.

The primary purpose of the Einstein Toolkit is to provide a comprehensive and flexible environment for conducting research in relativistic astrophysics. By bringing together numerical methods, high-performance computing, and advanced visualization techniques, the toolkit empowers scientists to explore the world of black holes, neutron stars, gravitational waves, and other fundamental aspects of the universe.

8.2. Installation and Setup

The Einstein Toolkit can be either used in the tutorial server, or installed in one's own computer. Considering that we will make much use of this code, we will prefer installing it in our own workstation.

The main requirements our machine needs are:

- **Client tools for Source Code Repositories**: SVN and git (used by the GetComponents utility, which we will talk about later).
- **Compilers**: C, C++ and Fortran 90.
- **MPI Implementation**: The message passing interface is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory. This is needed for the Carpet driver.
- **Standard development tools**: Programs for data management like Python, Perl, etc.
- **Operating System**: Cactus and the Einstein Toolkit is supported on all commonly used variants of Unix (Linux, MacOSX, AIX). On Microsoft Windows machines the Einstein Toolkit can be partially used with the cygwin environment, although this is not regularly tested and we will not be using it.

The installation guide can be easily found the Einstein Toolkit community site, and it can be worked in our laptop's terminal, or in a Jupyter notebook for easier visualization.

8.3. Overview of "Thorns"

One of the Einstein Toolkit is most important aspects involve its thorns. These are special files, self-contained modules that implement specific physics, numerical methods, or analysis tools.

Many thorns are already made by the community and ready to be used, but we are also able to create new thorns suitable to our needs. We start listing the main components of a new Cactus thorn:

- a Cactus thorn must have a name.
- a Cactus thorn must live in an arrangement.
- a Cactus thorn must have four ccl (Cactus Configuration Language) files:
  - **interface.ccl**
  - **schedule.ccl**
  - **param.ccl**
  - **configuration.ccl**
- a Cactus thorn must have a src (source) directory.
- a Cactus thorn must have a file in that source directory.

After all files are set up, we can end making a final file called "make.code.dfn". This file tells Cactus the names of the source code files to compile. In the case of evolution equation thorns, for example, our source code files would be: init.c , evolve.c and boundary.c . These files would contain all the information about initial conditions, evolution equation and boundary conditions, respectively.

8.4. Components and Functionality

The Einstein Toolkit comprises a vast collection of components and modules designed to simulate and analyze relativistic and astrophysical phenomena. These components provide researchers with tools necessary to explore general relativity. We highlight some key components and their associated functionality.

8.4.1. Solver Modules

These modules enable the numerical integration of Einstein's field equations and other relevant equations governing relativistic astrophysics. Examples of solver modules available in the toolkit include:

- **EinsteinEvolve**: This one forms the core of the Einstein Toolkit, providing solvers for evolving Einstein's equation. It employs finite difference or finite volume methods, with adapted mesh refinement and numerical techniques to solve for the dynamics of spacetime.incorporates
- **HydroBase**: This modules solvers for hydrodynamics, who play crucial roles in modelling phenomena involving fluids (accretion disks, supernovae, neutron star mergers, etc.).
- **GWBase**: This module helps work with gravitational wave astronomy. It facilitates the study of gravitational waveforms produced by compact object mergers.

8.4.2. Analysis and Visualization Tools

The Einstein Toolkit provides analysis and visualization tools to help researchers interpret and extract important information from simulation data. Some of these visualization components include:

- **Carpet**: This is a framework for adaptive mesh refinement (AMR), allowing researchers to work with high spatial resolution in regions of interests (like the event horizon in a black hole simulation), while using computational resources efficiently elsewhere. It refines the grid based on different criteria, such as presence of shocks or strength of gravitational fields.
- **VisIt**: This powerful tool offers visualization capabilities for large-scale simulation data. It generates interactive visualizations and custom plots, all needed to successfully understand the data gained in our simulations.

8.4.3. Setting Up a Simulation

We need to configure the Einstein Toolkit according to our specific research requirements. This is done by selecting (if not creating) different files that will be used on the Cactus framework.

Many of the required files have the Cactus Configuration Language (CCL) extension, and are used to define the parameters and setting for our simulations. Below is a list of the most important ".ccl" files:

- **cactus-config.ccl**: This file sets up the overall framework configuration, that is, how the Einstein Toolkit actually runs in our computer.
- **schedule.ccl**: As the name suggests, it specifies the schedule of our simulation, defining the order in which functions are executed during the simulation.
- **param.ccl**: This file contains parameter definitions specific to the simulation we are trying to set up. Here we find information about our numerical parameters, physical constants, initial conditions, boundary conditions, and other settings relevant to our simulation.
- **thornlist.ccl**: Here we find the thorns used in our simulation. This helps to configure which thorns will be active and what are their dependencies.
- **interface.ccl**: Here is where the interface between different thorns is defined. This helps to specify how data is exchanged between modules.

8.5. Parfiles inside the Einstein Toolkit

After all other files and directories are created, we make use of a final file called the '.par' file. These serve as a centralized file where various thorns (modules) within the toolkit read their respective parameters and settings. They will be used to include initial conditions, boundary conditions, numerical parameters, and to specify the behavior of the thorns involved in the simulation.

These files also ensure the integration of multiple thorns, while also allowing them to interact and work together by sharing its parameters among all modules.

Below we show an example of a .par file used to run a Heat Equation code.

```
%%writefile {par_dir}/heat_eqn.par
Cactus::cctk_run_title = "scalar"
Cactus::cctk_full_warnings        = yes
Cactus::highlight_warning_messages = no
Cactus::terminate       = "time"
Cactus::cctk_final_time = 5.0
```

```
7
8  ActiveThorns = "Carpet CarpetLib CarpetInterp CarpetReduce CarpetSlab"
9  Carpet::verbose          = no
10 Carpet::veryverbose      = no
11 Carpet::schedule_barriers = no
12 Carpet::storage_verbose   = no
13 #Carpet::timers_verbose    = no
14 CarpetLib::output_bboxes  = no
15
16 Carpet::domain_from_coordbase = yes
17 Carpet::max_refinement_levels = 1
18
19 driver::ghost_size       = 1
20
21 Carpet::init_fill_timelevels = yes
22
23 ActiveThorns = "NaNChecker"
24 NaNChecker::check_every     = 1 # 512
25 #NaNChecker::verbose         = "all"
26 #NaNChecker::action_if_found = "just warn"
27 NaNChecker::action_if_found = "terminate"
28 NaNChecker::check_vars      = "
29       HeatEqn::U
30 "
31
32 ActiveThorns = "Boundary CartGrid3D CoordBase SymBase"
33 CoordBase::domainsize = "minmax"
34 CoordBase::spacing    = "numcells"
35
36 CoordBase::xmin = -10.0
37 CoordBase::ymin = -10.0
38 CoordBase::zmin = -10.0
39 CoordBase::xmax = +10.0
40 CoordBase::ymax = +10.0
41 CoordBase::zmax = +10.0
42 CoordBase::ncells_x  = 40
43 CoordBase::ncells_y  = 40
44 CoordBase::ncells_z  = 40
45
46 CoordBase::boundary_size_x_lower     = 1
47 CoordBase::boundary_size_y_lower     = 1
48 CoordBase::boundary_size_z_lower     = 1
49 CoordBase::boundary_size_x_upper     = 1
50 CoordBase::boundary_size_y_upper     = 1
51 CoordBase::boundary_size_z_upper     = 1
52
53 CartGrid3D::type = "coordbase"
54
55 ActiveThorns = "Time"
56 Time::dtfac = 0.0125
57
58 ActiveThorns = "InitBase"
```

```
59
60 ActiveThorns = "HeatEqn"
61
62 ActiveThorns = "IOUtil"
63 IO::out_dir = $parfile
64 IO::checkpoint_dir                = $parfile
65 IO::checkpoint_ID                 = no
66 IO::checkpoint_every_walltime_hours = 6.0
67 IO::checkpoint_on_terminate       = yes
68 IO::recover     = "autoprobe"
69 IO::recover_dir = $parfile
70
71 ActiveThorns="CarpetIOHDF5"
72 IOHDF5::out2D_every          = 16
73 IOHDF5::out2D_xz             = no
74 IOHDF5::out2D_yz             = no
75 IOHDF5::output_buffer_points  = yes
76 #IOHDF5::one_file_per_group    = yes
77 IOHDF5::output_symmetry_points = no
78 IOHDF5::compression_level     = 1
79 IOHDF5::use_checksums         = yes
80 IOHDF5::out2D_vars           = "
81       HeatEqn::U
82       Grid::Coordinates{out_every=1000000000} #we only want this to print once
83 "
84 IOHDF5::checkpoint           = no
```

Notice how it includes important parameters of the simulation, while also specifying which thorns will be used to ensure a correct simulation.

Finally, ".par" files can be downloaded from uploaded code in sites such as Github. These would be ready to be ran and worked with in our own terminal. To modify these files, one can open any ".txt" file compiler and change the necessary parameters, and allocating the file accordingly.

8.6. Running an example from the Einstein Toolkit site

Let us begin showing how to run an example from scratch. The Einstein Toolkit community website has all the documentation needed to correctly install the toolkit in your computer, and to understand how everything works. The first thing to do, is of course check if your computer has all the requirements necessary to be able to run code. In our case, we will be working inside **Ubuntu**, a Linux distribution that works best with the Einstein Toolkit. We will be also be working inside a **jupyer notebook**, that will help us better visualize what is happening inside our terminal.

We start going to the Einstein Toolkit community website, https://einsteintoolkit.org/, where we will easily spot two titles: 'Get Started' and 'Documentation' (Fig. 8.1).

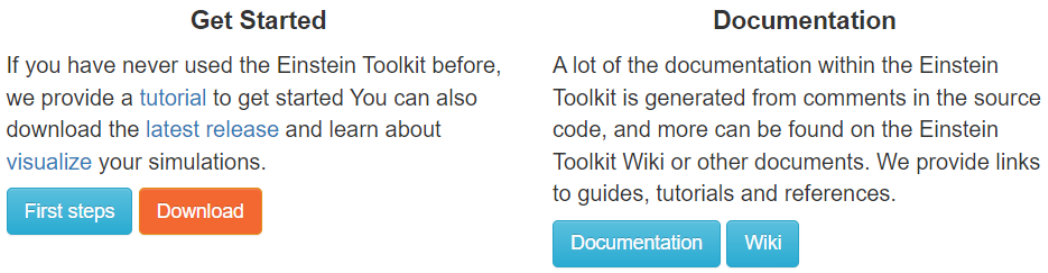

Figura 8.1: Location of tutorials and documentation for the Einstein Toolkit

At **Get Started**, we can find a tutorial, that details how to correctly install the libraries needed for the code. One can either select 'First Steps' or 'Download'. Downloading the notebook in our laptop is easy enough, so when clicking on 'Download' we will be greeted with a read only version of a Jupyter Notebook. This notebook details everything needed to type in different cells of a Jupyter notebook to correctly install the Einstein Toolkit in our machine.

After running every cell with no error, we can assume the Einstein Toolkit works correctly in our computer and therefore we can work with any .par file that is presented to us.

Let us work with a specific example: the **Multipatch Wave Equation**.

We come back to the Einstein Toolkit website to find an image gallery (Fig. 8.2) in the top left corner,

Inside this gallery (Fig. 8.3) we will find various examples. Since some of them take days to run, we will work with a simple example, and click on **Multi Patch Scalar Wave Equation**:

After clicking on this section, we notice a brief description (Fig. 8.4) of the code at hand, which helps set up context of what the parfile is doing internally. Parfiles usually only contain thorns and values of parameters, so it is important to know what is it that we are trying to solve. In this case, the description tells us that the parfile solves the wave equation on a Kerr background. It specifies that it has certain symmetry on its Cartesian coordinates axes, and one can check it does actually use symmetry thorns. Also, one must also know what it is that the simulation shows. In this case, starting from an spherical harmonic with $(l, m) = (2, 2)$ in angular directions, it shows the amplitude of waves outside of the event horizon.

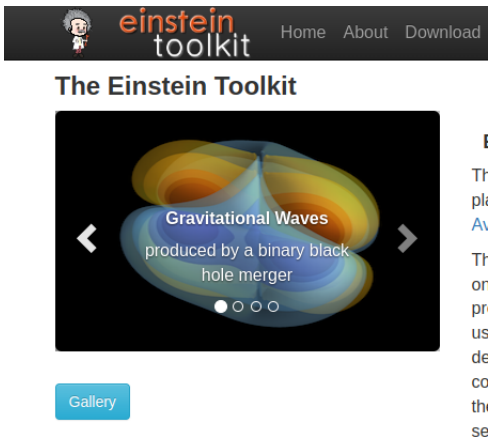

Figura 8.2: Gallery where some examples are presented and ready to be simulated

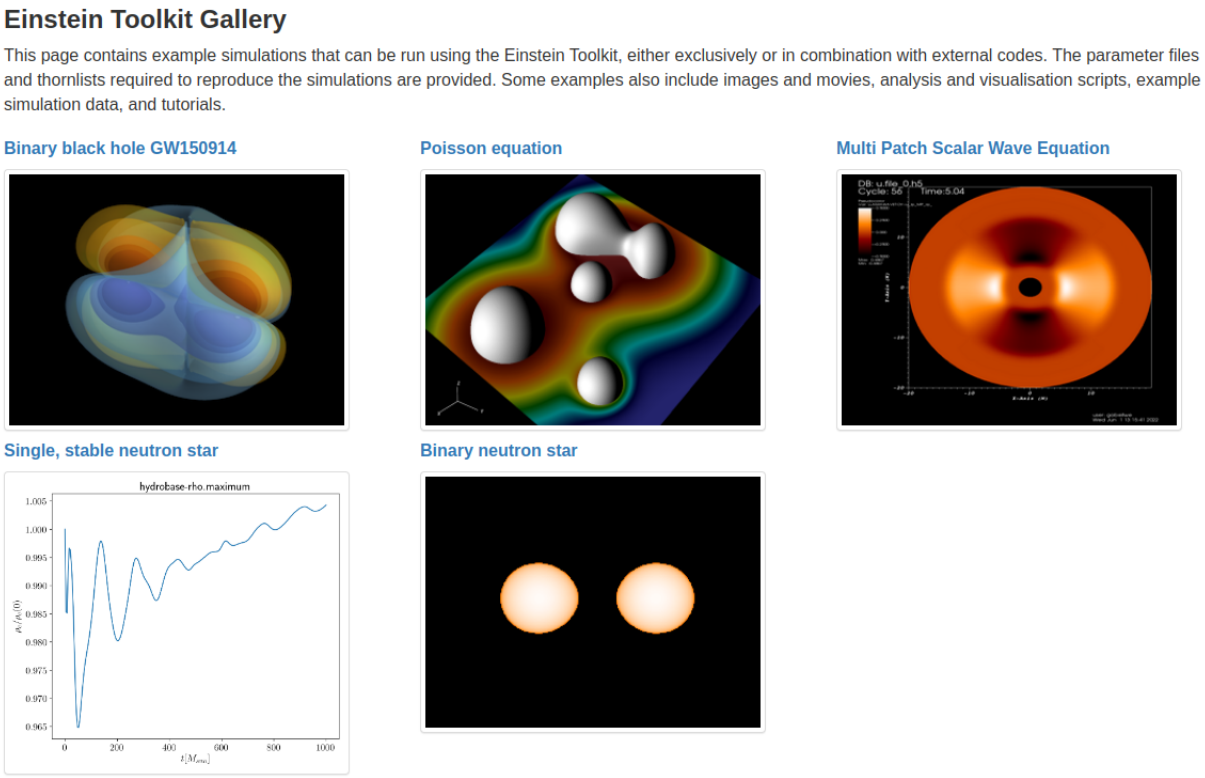

Figura 8.3: 5 examples presented, each with its own tutorial and documentation to understand the physics behind them.

Gallery: Multipatch wave equation

The Llama code is a 3-dimensional multiblock infrastructure for Cactus based on Carpet. It provides different patch systems that cover the simulation domain by a set of overlapping patches. Each of these patches has local cooordinates with a well-defined relation to global Cartesian coordinates. Information between the different patches is communicated via interpolation in the overlap zones.

Here we show an example evolution of a wave equation on a Kerr background using the "Thornburg2004nc" patch-system, consisting of 90 degree by 90 degree wedges centred on each of the +x, -x, +y, -y, +z, -z coordinate axes. This provides a complete covering of a region between rmin and rmax with smooth inner and outer boundaries. The initial data is an l=2, m=2 spherical harmonic in the angular directions, and a Gaussian in the radial direction. An alternative patch system, "Thornburg2004", adds a cubical patch enclosing r < rmin in which standard Carpet box-in-box mesh refinement can be employed. This is used in the GW150914 gravitational wave gallery example.

We ask that if you make use of the Llama code, then please cite Llama, the Einstein Toolkit, the Carpet mesh-refinement driver and Cactus.

Figura 8.4: Brief description of the simulation for the Multipatch wave equation

Just below the description, some important information for our simulation is given, like the one seen in 8.5. The parfile is available for all to download and change as needed. Also, within the Cactus framework one needs to execute a script that calls this parfile, gives the simulation a name, and begins to create all directories necessary for Cactus to run the simulation. This is all done with the 'simfactory/bin/sim' script, which is inside the 'Cactus' folder in the given path. Also, not every simulation can be ran in any cluster, since some need bigger RAM and resources to not run out of memory. In this case, the simulation can run in our laptop, and results can be tested.

Simulation details

Computational details

Parameter File	Kerr-Schild_Multipole.par
Submission command	simfactory/bin/sim create-submit Kerr-Schild_Multipole --parfile arrangements/Llama/LlamaWaveToy/par/Kerr-Schild_Multipole.par
Total memory	800 MB
Run time	A little over 1 minute using 56 MPI tasks and 56 cores on Frontera; or about 10 minutes on two laptop cores
Results (gzip 234MB, uncompressed 468MB)	Kerr-Schild_Multipole-20231116.tar.gz

Figura 8.5: Brief description of the simulation

Finally, results are also given for each simulation example. One can download these, and follow the respective tutorials to plot relevant variables. In this case, one can use the 'VisIt' program to view a 3-D representation of the wave equation. More variables are available for every simulation, all which need some understanding of the parfile to know what is it that is being given by the toolkit.

Capítulo 9

Gravitational Waves

9.1. Introduction

Since the formulation of the field equations of general relativity in 1916, Albert Einstein found that when linearizing the weak-field equations, the solutions had wave behavior. These were transverse waves that travel at the speed of light, and are fundamentally different from other type of waves, for example: while electromagnetic waves propagating in space and time are created by the acceleration of electric charges, gravitational waves are created by the acceleration of mass, and are waves of the spacetime fabric itself.

All objects with mass moving accelerating through spacetime, in theory, generate gravitational waves. However, to truly be able to observe and measure their effects, the sources studied are usually accelerating masses of huge binary stars and black hole mergings. An example of this is depicted in Fig. 9.1.

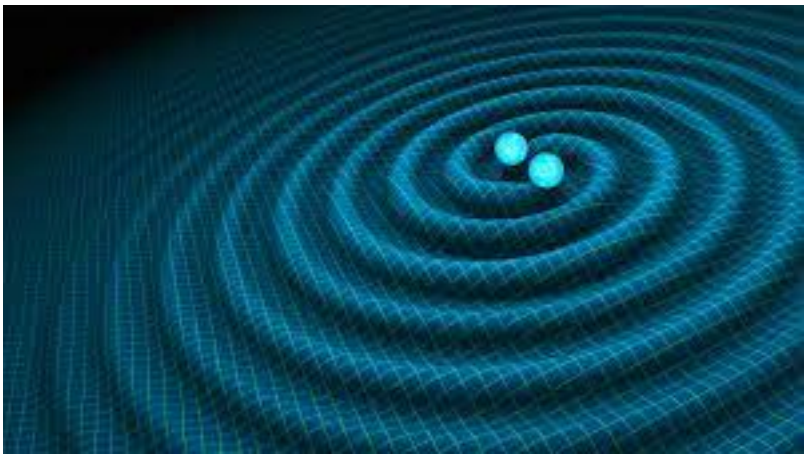

Figura 9.1: Basic display of how the spiraling of two massive objects could cause wave ripples that travel through spacetime.

The significance of gravitational wave detection extends beyond confirming a prediction of General Relativity. Gravitational waves offer a new tool for probing the universe's most energetic events. One notable example is the observation of the binary neutron star merger GW170917. This event, detected in both gravitational waves and gamma-ray bursts, marked the beginning of multi-messenger astronomy. It provided valuable information about the origin of heavy elements in the universe and confirmed that such mergers can be observed through multiple channels.

The most known example of gravitational wave detectors is the first one, detected on September

14, 2015, by the Laser Interferometer Gravitational-Wave Observatory (LIGO).

9.2. Historical context

Recent detection of gravitational waves by the LIGO and Virgo team in 2015 has baffled the physics community as a huge achievement of experimental physics. They not only confirmed black holes existed, but also showed there are binary black hole systems and that they collide to form larger black holes. Though a massive breakthrough for researchers now, the theory behind these waves was not always fully supported by the scientific community. In fact, it was not until the beginning of the 1960s that the experimentalists were given the green light to start designing detectors, for which there was no previous reason to convince anyone that this theory held any weight.

Gravitational waves were solutions to the Einstein field equations when approximating metrics as we, the observers, are at approximately spatial infinity from the black hole mergers. Hence one could approximate solutions to be small perturbations of flat metric and arrive to these wave ripples of spacetime travelling at the speed of light. The fundamental problem in Einstein's time is that not every theory can be proven without experimental teams working on it, and it requires great effort and money to do so. Therefore, the scientific community had to prove, at least, that gravitational waves were something that was worth looking into, and that it had no dead end in sight.

Because of this, Einstein was faced with a number of questions: What is the definition of a plane gravitational wave? Does this plane defined wave exist as a solution of the full Einstein field equations? Do these waves carry energy? What is a definition of a gravitational wave with nonplanar front? What is the energy of such waves? Do there even exist solutions to the full Einstein system satisfying this definition? Does the full theory admit solutions corresponding to the gravitational waves emitted by bounded sources?

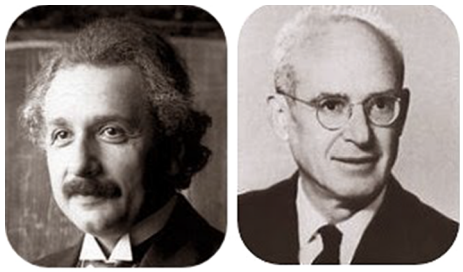

Figura 9.2: Einstein and Rosen, the first scientists to examine exact gravitational wave solutions.

Answering all these questions would, of course, give reason to develop strategies to look further. Nevertheless, since the prediction of Einstein in 1916, the very definition of a gravitational wave in the nonlinear Einstein theory was not provided until Einstein's death. If so, what approach did scientists have to convince the community to begin research? For this, we look into the first question, what's the definition of a plane gravitational wave?

The first approach was simple, a gravitational plane wave is a spacetime metric, which in some coordinates (t, x, y, z), with t of course being timelike, has metric function perturbations depending on $u = t - x$ only. This would then mean that the perturbation satisfies the wave equation in flat metric (by approximating the general metric to first order and eliminating higher order terms) and

could then be called gravitational waves. This approach was tried but failed, as seen in the following example:

Let us consider the metric:

$$\begin{aligned} g_{\mu\nu} &= (\eta_{\mu\nu} + h_{\mu\nu})dx^\mu dx^\nu \\ &= \eta_{\mu\nu}dx^\mu dx^\nu + \cos(t-x)(2+\cos(t-x))dt^2 \\ &\qquad - 2\cos(t-x)(1+\cos(t-x))dtdx + cos^2(t-x)dx^2. \end{aligned}$$

We clearly see that the terms corresponding to the perturbation are oscillatory, and travelling at the speed of light $c = 1$ along the x-axis. Clearly the coefficients $h_{\mu\nu}$ satisfy the wave equations (by depending on the null coordinate $u = t - x$). One can also show that the full metric g had Ricci curvature 0, thus providing a solution of the vacuum Einstein equations $R_{\mu\nu} = 0$ in the fully non-linear theory. The problem is, this was actually *not* a gravitational wave! If we take our Minkowski flat metric $\eta_{\mu\nu} = d\tilde{t}^2 - dx^2 - dy^2 - dz^2$, and changed our time coordinate to $\tilde{t} = t + \sin(t-x)$, we would recover our metric g. Because of this, our original metric $g_{\mu\nu}$ is just the flat metric written in different coordinates, thus not corresponding to a gravitational wave. This posed a problem to physicists: gravitational waves were not solutions of the Einstein field equations that had wave-like behavior, and so a different mathematical precise definitions had to be given.

It was hard for anyone to continue working on this theory, since in 1937, twenty years after the formulation of the concept of a plane wave, Albert Einstein and Nathan Rosen (Fig. 9.2) worked to definitively prove that physically acceptable plane gravitational waves were not admitted by Einstein's theory of General Relativity, in fact, after having found one solution of the vacuum Einstein equation representing a plane wave, they observed that it had singularities and should not be considered physical. Einstein-Rosen's paper was later shown to Howard P. Robertson for an anonymous revision, who recognized said singularities arise from a wrong choice of coordinates, and then suggested they use cylindrical coordinates instead, thus generating cylindrical waves. This story could be further expanded to talk about the gossip it generated amongst the science community, where Robertson's notes angered Einstein and made him switch the publication magazine, but what's really important is that it discouraged physicists to prove the existence of physical gravitational plane waves.

It wasn't until 1957 where Bondi, Felix Pirani, and Robinson wrote papers describing found singularity-free solutions of gravitational plane waves that carried energy. The answers to our first questions were finally here: A gravitational plane wave is a spacetime that satisfies vacuum Einstein's equations $R_{\mu\nu} = 0$ and has a 5-dimensional group of isometries. The motivation came from the isometries of plane electromagnetic waves, and it was later proven that the isometries found on these plane gravitational waves were isomorphic to those of the electromagnetic waves. A sad realization was that these waves had already been discovered in 1925 by a mathematician H. W. Brinkmann who came up with pp-waves (plane-fronted waves) that explained massless radiation, and for which a special case were our gravitational plane waves. However, it was purely mathematical work and it wasn't fully understood in its time.

The final questions were finally answered by Felix Pirani and Andrzej Trautman by the 1960s. Pirani suggested specific conditions that a gravitational wave spacetime must satisfy for gravitational waves to exist, giving rise to a purely geometric definition of this spacetime. He argued gravitational radiation would be detectable when these conditions are met, and hinting how to define what radiation is in general relativity. Trautman made the final move by answering all previous questions in his two papers submitted to Bulletin de l'Academie Polonaise des Sciences. In these papers, he defines boundary conditions to be imposed on gravitational fields due to isolated systems of matter, and consequently proposes a 4-momentum $P^\mu(\sigma)$ of a gravitational field attributed to every

space-like hypersurface σ of a spacetime satisfying the previously mentioned radiative boundary conditions. If solutions satisfied all previous conditions, and also that $p^\mu = P^\mu(\sigma_1) - P^\mu(\sigma_2)$, we could finally know that the waves carried energy. In fact, these solutions were finally found by Robinson and Trautman, and the green light was finally given to begin research on gravitational waves.

9.3. Gravitational Waves in Einstein's linearized theory

We begin trying to understand what 'linearized gravity' is and how it is applied to General Relativity to predict gravitational waves. While trying to study star binaries and black hole collisions, one may try to approximate their effects (on our frame of reference) by thinking that they perturbate our current flat metric $\eta_{\mu\nu}$ by a small amount $h_{\mu\nu}$. Of course the Earth affects the spacetime around us, but at this scale one can say that our metric is basically flat. Using this, instead of working with the whole spacetime metric $g_{\mu\nu}$ to solve Einstein's field equations, one works with $h_{\mu\nu}$ which we already know is small. Let us show that our $h_{\mu\nu}$ tensor admits wave solutions.

Recall that the Einstein field equations describing the geometry of spacetime are:

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} = 8\pi G T_{\mu\nu}.$$

We then assume that the metric that solves the equation takes the form,

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu},$$

where clearly $\partial_\sigma \eta_{\mu\nu} = 0$, and $\|h_{\mu\nu}\| \ll 1$. We restrict ourselves to coordinates in which $\eta_{\mu\nu} = \text{diag}(-1, +1, +1, +1)$. By assuming $h_{\mu\nu}$ is small, we are implicitly setting $h_{\mu\nu} \sim \epsilon h_{\mu\nu}$, so all terms of higher than first order in this quantity are to be ignored. These formulae allow us to simplify the equations by only working on derivatives of $h_{\mu\nu}$ and eliminating all higher order terms. A substitution of the general spacetime metric for this perturbative approximation in the formula for the Ricci tensor yields:

$$R_{\mu\nu} = \frac{1}{2}(\partial_\sigma \partial_\mu h^\sigma{}_\nu + \partial_\sigma \partial_\nu h^\sigma{}_\mu - \partial_\mu \partial_\nu h - \Box h_{\mu\nu}),$$

where $h = \eta^{\mu\nu} h_{\mu\nu}$ is the trace of the perturbation, and $\Box = \eta^{\mu\nu}\partial_\mu \partial_\nu$ is the D'Alambert operator in flat space.

We notice we have raised indices on our perturbation. The formula for the inverse metric can be easily derived as follows.

First, we assume the metric $g^{\mu\nu}$ will take the same form of flat space $\eta^{\mu\nu}$ plus a small perturbation $h'^{\mu\nu}$. We derive the form of $h'^{\mu\nu}$ in the following way:

$$
\begin{aligned}
g_{\sigma\mu}g^{\mu\nu} = \delta^{\nu}{}_{\sigma} &\implies (\eta_{\sigma\mu} + h_{\sigma\mu})(\eta^{\mu\nu} + h'^{\mu\nu}) = \delta^{\nu}{}_{\sigma} \\
&\implies \eta_{\sigma\mu}\eta^{\mu\nu} + \eta_{\sigma\mu}h'^{\mu\nu} + h_{\sigma\mu}\eta^{\mu\nu} + \underbrace{h_{\sigma\mu}h'^{\mu\nu}}_{\approx 0} = \delta^{\nu}{}_{\sigma} \\
&\implies \eta^{\nu}{}_{\sigma} + h'^{\nu}{}_{\sigma} + h^{\nu}{}_{\sigma} = \delta^{\nu}{}_{\sigma} \\
&\implies \delta^{\nu}{}_{\sigma} + h'^{\nu}{}_{\sigma} + h^{\nu}{}_{\sigma} = \delta^{\nu}{}_{\sigma} \\
&\implies h'^{\nu}{}_{\sigma} + h^{\nu}{}_{\sigma} = 0 \quad / \cdot \eta^{\sigma\mu} \\
&\implies h'^{\mu\nu} = -h^{\mu\nu}.
\end{aligned}
$$

Therefore, our equation for the linearized inverse metric is:

$$g^{\mu\nu} = \eta^{\mu\nu} - h^{\mu\nu}.$$

Combining this with the equation for the Ricci scalar,

$$R = \eta_{\mu\nu}R^{\mu\nu} = \partial_\mu\partial_\nu h^{\mu\nu} - \Box h,$$

allows us to reduce the left side of the field equation to

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} = \frac{1}{2}(\partial_\sigma\partial_\mu h^{\sigma}{}_{\nu} + \partial_\sigma\partial_\nu h^{\sigma}{}_{\mu} - \partial_\mu\partial_\nu h - \Box h_{\mu\nu} - \eta_{\mu\nu}\partial_\rho\partial_\lambda h^{\rho\lambda} + \eta_{\mu\nu}\Box h).$$

We have obtained the linearized form of the Einstein field equations, expressing the perturbations to the spacetime metric caused by the presence of matter and energy. Now, let's delve into the physical interpretation of these equations in the context of gravitational waves.

Gravitational waves are ripples in spacetime, propagating outward at the speed of light. In our linearized theory, the metric perturbation $h_{\mu\nu}$ encapsulates the effect of these waves on our flat background spacetime, represented by $\eta_{\mu\nu}$. The key components of our linearized field equations are associated with the dynamics of these perturbations.

The term $\partial_\sigma\partial_\mu h^{\sigma}{}_{\nu} + \partial_\sigma\partial_\nu h^{\sigma}{}_{\mu}$ captures the spatial derivatives of the metric perturbation, signifying the stretching and squeezing of space as gravitational waves pass through. The trace $h = \eta^{\mu\nu}h_{\mu\nu}$ represents the overall amplitude of the wave, providing insight into the strength of the gravitational disturbance. Additionally, the appearance of the D'Alambert operator $\Box$ in the equations reflects the wave-like nature of gravitational perturbations. It describes how these waves propagate through spacetime, akin to the behavior of electromagnetic waves.

Since we are still in the grounds of General Relativity, these equations are valid in any choice of coordinates. This will give us some freedom in the selection of our system from which we will study gravitational waves.

9.3.1. Gauge freedom

In classical Newtonian mechanics, the motion due to a force field is governed by $\boldsymbol{f} = m\boldsymbol{a}$. Whenever we have $\boldsymbol{f} = -\nabla\Phi$, the particle's equation is,

$$m\frac{d^2x^i}{dt^2} = -\partial_i\Phi.$$

This is a second-order differential equation for $x^i(t)$, which we can rewrite as a system of first-order equations by introducing the momentum $\boldsymbol{p}$:

$$\frac{dp^i}{dt} = -\partial_i \Phi$$
$$\frac{dx^i}{dt} = \frac{1}{m} p^i.$$

This helps us set an initial-value problem, specifying a state (x^i, p^i), which serves as a boundary condition with which the above system is uniquely solved. As seen in previous methods, a first-order in time system of equations allows us to specify coordinates and momenta at some time t, and evolve it forward to an infinitesimal amount $t+\delta t$, thus obtaining the entire solution by iteration.

We we would like is to formulate the analogous problem in general relativity. Recall Einstein's equations are $G_{\mu\nu} = 8\pi G T_{\mu\nu}$, where both sides are covariant; there's no preferred notion of 'time'. One can nevertheless pick a spacelike hypersurface Σ, specify initial data on that hypersurface, and see how it would evolve to the next hypersurface (as seen in the chapter on Numerical Relativity).

Since our unknown in General Relativity is the metric $g_{\mu\nu}$, we guess that we should consider the values $g_{\mu\nu}|_\Sigma$ of the metric on the hypersurface to be the 'coordinates', and time derivatives $\partial_t g_{\mu\nu}|_\Sigma$ (after specifying the time coordinate) to be the 'momenta'. Recall the equations $G_{\mu\nu} = 8\pi G T_{\mu\nu}$ involve second derivatives of the metric. However, the Bianchi identity tells us that the contravariant Einstein tensor is conserved, that is, $\nabla_\mu G^{\mu\nu} = 0$. This equation can be rewritten as

$$\partial_0 G^{0\nu} = -\partial_i G^{i\nu} - \Gamma^\mu{}_{\mu\lambda} G^{\lambda\nu} - \Gamma^\nu{}_{\mu\lambda} G^{\mu\lambda}.$$

We notice that the right hand side has no third-order time derivatives, meaning that the components $G^{0\nu}$ cannot have second-order time derivatives. This means that the four equations

$$G^{0\nu} = 8\pi G T^{0\nu},$$

cannot be used to evolve initial data $(g_{\mu\nu}, \partial_t g_{\mu\nu})|_\Sigma$. They are thus **constraints** on this initial data, setting conditions on combination of the metric and its time derivative on the hypersurface. The remaining equations,

$$G^{ij} = 8\pi G T^{ij},$$

are the dynamical evolution equations for the metric. These are six equations for the ten unknown functions $g_{\mu\nu}(x^\sigma)$, so the solution involves a fourfold ambiguity. This is the freedom to choose four coordinate functions throughout spacetime.

All up to this point tells us we must determine the evolution of the metric, up to an unavoidable ambiguity in fixing the remaining components $g_{0\nu}$. This is analogous to electromagnetism, where one cannot determine the evolution uniquely since there's always freedom to perform a gauge transformation $A_\mu \to A_\mu + \partial_\mu \lambda$. In General Relativity, coordinate transformations will play a role reminiscent of gauge transformations in electromagnetism, introducing ambiguity into the time evolution. This we will call our **gauge freedom**.

Physically, gauge freedom implies that different observers might use different coordinates to describe the same gravitational wave, yet they will all agree on the observable effects of the wave. This freedom to choose coordinates reflects the inherent flexibility in our mathematical description of the gravitational field.

To cope with this problem we can simply "choose a gauge". A popular choice in the **harmonic**

gauge, in which

$$\Box x^\mu = 0.$$

Remembering that x^μ are just functions, not components of a vector, thus covariant derivatives reduce to partial derivatives. So our equation can be expanded as follows:

$$\begin{aligned}
0 &= \Box x^\mu \\
&= \nabla^\sigma \nabla_\sigma x^\mu \\
&= \nabla^\sigma (\partial_\sigma x^\mu) \\
&= g^{\rho\sigma} \nabla_\rho (\partial_\sigma x^\mu) \\
&= g^{\rho\sigma} (\partial_\rho \partial_\sigma x^\mu - \Gamma^\lambda{}_{\rho\sigma} \partial_\lambda x^\mu) \\
&= g^{\rho\sigma} \partial_\rho \partial_\sigma x^\mu - g^{\rho\sigma} \Gamma^\lambda{}_{\rho\sigma} \partial_\lambda x^\mu \\
&= g^{\rho\sigma} \partial_\rho \delta^\mu{}_\sigma - g^{\rho\sigma} \Gamma^\lambda{}_{\rho\sigma} \delta^\mu{}_\lambda \\
&= -g^{\rho\sigma} \Gamma^\mu{}_{\rho\sigma}.
\end{aligned}$$

By expanding the Christoffel symbol, this condition can be translated into a second-order differential equation for the previously unconstrained metric components $g^{0\nu}$. This finally fixes our gauge freedom, in that we can now solve for the evolution of the entire metric in harmonic coordinates.

Let us use this gauge in our weak field approximation. We have already shown that the gauge specified by $\Box x^\mu = 0$, was equivalent to,

$$g^{\rho\sigma} \Gamma^\mu{}_{\rho\sigma} = 0.$$

In the weak field limit this becomes,

$$\frac{1}{2} \eta^{\rho\sigma} \eta^{\mu\lambda} (\partial_\rho h_{\sigma\lambda} + \partial_\sigma h_{\lambda\rho} - \partial_\lambda h_{\rho\sigma}) = 0,$$

which reduces to,

$$\partial_\mu h^\mu{}_\lambda - \frac{1}{2} \partial_\lambda h = 0.$$

In this gauge, the linearized Einstein equations $G_{\mu\nu} = 8\pi G T_{\mu\nu}$ simplify to,

$$\Box h_{\mu\nu} - \frac{1}{2} \eta_{\mu\nu} \Box h = -16\pi G T_{\mu\nu},$$

while the vacuum equations $R_{\mu\nu}$ take the form,

$$\Box h_{\mu\nu} = 0,$$

which is the conventional relativistic wave equation. These equations determine the evolution of a disturbance in the gravitational field in vacuum in the harmonic gauge.

One often works with a different description of the metric perturbation. We define the "trace-reversed" perturbation $\bar{h}_{\mu\nu}$ by,

$$\bar{h}_{\mu\nu} = h_{\mu\nu} - \frac{1}{2} \eta_{\mu\nu} h.$$

The name comes from that fact that $\bar{h}^\mu{}_\mu = -h^\mu{}_\mu$. In terms of $\bar{h}_{\mu\nu}$, the harmonic gauge condition becomes,

$$\partial_\mu \bar{h}^\mu{}_\lambda = 0,$$

and the full field equations become,

$$\Box\bar{h}_{\mu\nu} = -16\pi G T_{\mu\nu}.$$

9.3.2. Application to gravitational radiation

Let us apply the weak-field limit with our harmonic gauge to study gravitational radiation. We begin considering the linearized equations in vacuum,

$$\bar{h}_{\mu\nu} = 0.$$

The D'Alembertian in flat space has the form $\Box = -\partial_t^2 + \nabla^2$, so the field equation above is in the worm of a wave equations for $\bar{h}_{\mu\nu}$. To get solutions to this equations, we propose an ansatz of complex valued wave solutions, given by

$$\bar{h}_{\mu\nu} = C_{\mu\nu} e^{ik_\sigma x^\sigma},$$

where $C_{\mu\nu}$ is a constant, symmetric, (0,2) tensor, and k^σ is a constant vector known as the **wave vector**. To check conditions it be a solution, we plug in:

$$\begin{aligned} 0 &= \Box\bar{h}_{\mu\nu} \\ &= \eta^{\rho\sigma}\partial_\rho\partial_\sigma\bar{h}_{\mu\nu} \\ &= \eta^{\rho\sigma}\partial_\rho(ik_\sigma\bar{h}_{\mu\nu}) \\ &= -\eta^{\rho\sigma}k_\rho k_\sigma\bar{h}_{\mu\nu} \\ &= -k_\sigma k^\sigma\bar{h}_{\mu\nu}. \end{aligned}$$

This forces a condition on our wave vector, since we are focusing on solutions where $\bar{h}_{\mu\nu}$ is not zero everywhere, we must have,

$$k_\sigma k^\sigma = 0.$$

This is loosely translated into the statement that *gravitational waves travel at the speed of light.* The timelike component of the wave vector is often referred to as the *frequency* of the wave, so we write $k^\sigma = (\omega, k^1, k^2, k^3)$. The condition that the wave vector be null then becomes,

$$\omega^2 = \delta_{ij}k^i k^j.$$

The free parameters that specify the wave are then: ten numbers for the $C_{\mu\nu}$ coefficients and three for the null wave vector k^σ. We begin imposing the harmonic gauge condition:

$$\begin{aligned} 0 &= \partial_\mu\bar{h}^{\mu\nu} \\ &= \partial_\mu(C^{\mu\nu}e^{ik_\sigma x^\sigma}) \\ &= iC^{\mu\nu}k_\mu e^{ik_\sigma x^\sigma}, \end{aligned}$$

which is only true if,

$$k_\mu C^{\mu\nu} = 0.$$

This means that the harmonic gauge conditions forces the wave vector be orthogonal to $C^{\mu\nu}$. These are four equations, which reduce our ten independent components of $C_{\mu\nu}$ to six.

Even setting the harmonic gauge conditions, we can still make a coordinate transformation by setting $x^\mu \to x^\mu + \zeta^\mu$, which will still satisfy the harmonic gauge condition as long as $\Box\zeta^\mu = 0$.

We then have another wave equation for ζ^μ, so once we choose a solution, we will have used up

all of our gauge freedom. We choose the solution,

$$\zeta_\mu = B_\mu e^{ik_\sigma x^\sigma},$$

where k_σ is the previous wave vector and B_μ are constant coefficients.

We claim that this remaining freedom allows us to convert from whatever coefficients $C^{(\text{old})}_{\mu\nu}$ that characterize our gravitational wave to a new set of coefficients $C^{(\text{new})}_{\mu\nu}$, such that

$$C^{(\text{new})\mu}{}_\mu = 0 \quad \text{and} \quad C^{(\text{new})}{}_{0\nu} = 0.$$

Let us see how this is possible by solving for our B_μ coefficients. Under our $x^\mu \to x^\mu + \zeta^\mu$ transformation, the change on our perturbation can be written

$$\begin{aligned} h^{(\text{new})}_{\mu\nu} &= h^{(\text{old})}_{\mu\nu} - \mathcal{L}_\zeta h_{\mu\nu} \\ &= h^{(\text{old})}_{\mu\nu} - \mathcal{L}_\zeta g_{\mu\nu} \\ &= h^{(\text{old})}_{\mu\nu} - \partial_\mu \zeta_\nu - \partial_\nu \zeta_\mu, \end{aligned}$$

which induces a change in the trace-reversed perturbation,

$$\begin{aligned} \bar{h}^{(\text{new})}_{\mu\nu} &= h^{(\text{new})}_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu} h^{(\text{new})} \\ &= h^{(\text{old})}_{\mu\nu} - \partial_\mu \zeta_\nu - \partial_\nu \zeta_\mu - \frac{1}{2}\eta_{\mu\nu}(h^{(\text{old})} - 2\partial_\lambda \zeta^\lambda) \\ &= \bar{h}^{(\text{old})}_{\mu\nu} - \partial_\mu \zeta_\nu - \partial_\nu \zeta_\mu + \eta_{\mu\nu}\partial_\lambda \zeta^\lambda. \end{aligned}$$

Using the specific form of our perturbation $\bar{h}_{\mu\nu} = C_{\mu\nu} e^{ik_\sigma x^\sigma}$ and for our translation vector $\zeta_\mu = B_\mu e^{ik_\sigma x^\sigma}$, we obtain that:

$$C^{(\text{new})}_{\mu\nu} = C^{(\text{old})}_{\mu\nu} - ik_\mu B_\nu - ik_\nu B_\mu + in_{\mu\nu} k_\lambda B^\lambda.$$

Imposing our condition $C^{(\text{new})\mu}{}_\mu = 0$, we get that

$$0 = C^{(\text{old})\mu}{}_\mu + 2ik_\lambda B^\lambda \quad \implies k_\lambda B^\lambda = \frac{i}{2} C^{(\text{old})\mu}{}_\mu.$$

Imposing our other condition $C^{(\text{new})}{}_{0\nu} = 0$, we start setting $\nu = 0$:

$$\begin{aligned} 0 &= C^{(\text{old})}_{00} - 2ik_0 B_0 - ik_\lambda B^\lambda \\ &= C^{(\text{old})}_{00} - 2ik_0 B_0 + \frac{1}{2} C^{(\text{old})\mu}{}_\mu, \end{aligned}$$

or,

$$B_0 = -\frac{1}{2k_0}(C^{(\text{old})}_{00} + \frac{1}{2} C^{(\text{old})\mu}{}_\mu).$$

Next, imposing for $\nu = j$, we get:

$$\begin{aligned} 0 &= C^{(\text{old})}_{0j} - ik_0 B_j - ik_j B_0 \\ &= C^{(\text{old})}_{0j} - ik_0 B_j - ik_j[-\frac{i}{2k_0}(C^{(\text{old})}_{00} + \frac{1}{2} C^{(\text{old})\mu}{}_\mu)], \end{aligned}$$

or,

$$B_j = \frac{i}{2(k_0)^2}[-2k_0 C_{0j}^{(\text{old})} + k_j(C_{00}^{(\text{old})} + \frac{1}{2}C^{(\text{old})\mu}{}_{\mu})].$$

Assuming we have performed this transformation, we will refer the new components $C_{\mu\nu}^{(\text{new})}$ simply as $C_{\mu\nu}$.

Let us summarize what we've done. We began with ten independent components in the symmetric matrix $C_{\mu\nu}$. Choosing the harmonic gauge, brought the number of independent components down to six. Using the remaining gauge freedom led to (1) $C^{(\text{new})\mu}{}_{\mu} = 0$ and the four conditions (2) $C^{(\text{new})}{}_{0\nu} = 0$. (So 6 minus 1 minus 4 = 1). But when $\nu = 0$, we get that (2) implies the one of original four constraints $k_\mu C^{\mu\nu} = 0$, adding one additional constraint. So then we have a total of two independent components. This means we have used all our possible freedom, and these two numbers can represent the physical information characterizing our plane wave in this gauge. We can see this explicitly by choosing our spatial coordinates such that the wave is travelling in the x^3 direction, that is:

$$k^\mu = (\omega, 0, 0, k^3) = (\omega, 0, 0, \omega),$$

where $k^3 = \omega$ since the wave vector is null. In this case, $k^\mu C_{\mu\nu} = 0$ and $C_{0\nu} = 0$ together imply that,

$$C_{3\nu} = 0.$$

Therefore the only nonzero components of $C_{\mu\nu}$ are therefore C_{11}, C_{12}, C_{21} and C_{22}. Since $C_{\mu\nu}$ is traceless and symmetric, we can actually write the matrix as:

$$C_{\mu\nu} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & C_{11} & C_{12} & 0 \\ 0 & C_{12} & -C_{11} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

This means that for a plane wave in this gauge travelling in the x^3 direction, the two components C_{11} and C_{12} (along with the frequency ω) completely characterize the wave.

We are finally close to exploring the true nature of these gravitational waves. For this, let us consider the motion of test particles in presence of a wave. To obtain coordinate-independent measures of the wave's effect, we will consider the relative motion of nearby particles, as described by the *geodesic deviation equation*. Let us consider nearby particles with four-velocities described by a single vector field $U^\mu(x)$ and a separation vector S^μ. The geodesic deviation equation then becomes:

$$\frac{D^2}{d\tau^2}S^\mu = R^\mu{}_{\nu\rho\sigma}U^\nu U^\rho S^\sigma.$$

We want to compute the left-hand side to first order in $h_{\mu\nu}$. If we take our test particles to be moving slowly (since we are studying the effects of the gravitational waves in their weak field limit), then we can express the four-velocity as an unit vector in the time direction plus correction of order $h_{\mu\nu}$ and higher, but the Riemann tensor is already first order, so the correction to U^ν can be ignored, and so,

$$U^\nu = (1, 0, 0, 0).$$

Therefore we only need to compute $R^\mu{}_{00\sigma}$, or equivalently $R_{\mu 00\sigma}$. From the equation of the weak field approximation of the Riemann tensor, we have

$$R_{\mu 00\sigma} = \frac{1}{2}(\partial_0\partial_0 h_{\mu\sigma} + \partial_\sigma\partial_\mu h_{00} - \partial_\sigma\partial_0 h_{\mu 0} - \partial_\mu\partial_0 h_{\sigma 0}).$$

But we have already computed values for $h_{\mu 0}$,

$$\begin{aligned} h_{\mu 0} &= C_{\mu 0} e^{ik_\sigma x^\sigma} \\ &= 0, \end{aligned}$$

since $C_{\mu 0} = 0$. The equation for the Riemann tensor then becomes

$$R_{\mu 00\sigma} = \frac{1}{2}\partial_0\partial_0 h_{\mu\sigma}.$$

Meanwhile, for slow moving partials we have $\tau = x^0 = t$ to lowest order, so the geodesic deviation equation becomes:

$$\frac{\partial^2}{\partial t^2} = \frac{1}{2} S^\sigma \frac{\partial^2}{\partial t^2} h^\mu{}_\sigma,$$

where we can compute that $h^\mu{}_\sigma = \eta^{\mu\lambda} h_{\lambda\sigma} = h_{\mu\sigma}$, whenever $\mu, \sigma \in \{1, 2\}$. Since we know the wave is travelling in the x^3 direction, we will have $h^\mu{}_\sigma$ will only be nonzero when $\mu, \sigma \in \{1, 2\}$. Therefore only S^1 and S^2 will be affected.

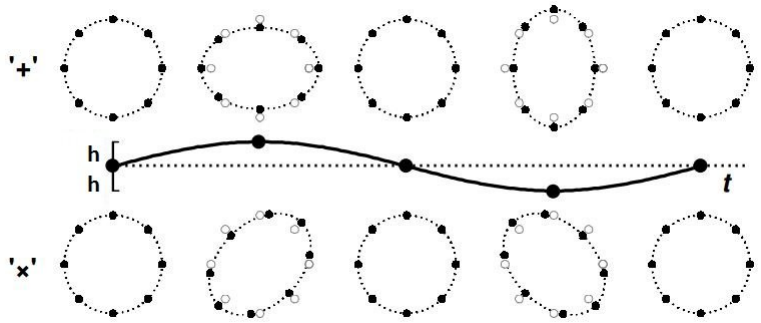

Figura 9.3: Both $'+'$ and $'\times'$ polarizations. White dots represent the positions of our test particles without the presence of gravitational waves, while black dots show their position as the wave passes through.

Since our wave is characterized by the two numbers C_{11} and C_{12}, we will from now on rename them as $C_+ = C_{11}$ and $C_\times = C_{12}$. Let us consider their effects separately.

When $C_\times = 0$, we have the equations

$$\frac{\partial^2}{\partial t^2} S^1 = \frac{1}{2} S^1 \frac{\partial^2}{\partial t^2}(C_+ e^{ik_\sigma x^\sigma}) \quad \text{and} \quad \frac{\partial^2}{\partial t^2} S^2 = -\frac{1}{2} S^2 \frac{\partial^2}{\partial t^2}(C_+ e^{ik_\sigma x^\sigma}).$$

These can be solved to yield, to lowest order,

$$S^1 = (1 + \frac{1}{2} C_+ e^{ik_\sigma x^\sigma}) S^1(0) \quad \text{and} \quad S^2 = (1 - \frac{1}{2} C_+ e^{ik_\sigma x^\sigma}) S^2(0).$$

This means that particles initially separated in the x^1 direction oscillate back and forth in the x^1 direction, and likewise for those with x^2 separation. If we start with a ring of particles in the $x - y$ plane, they bounce back and forth in the shape of a " + ", as seen in Fig. 9.3.

The equivalent analysis for the case when $C_+ = 0$ and $C_\times \neq 0$ would yield the solution,

$$S^1 = S^1(0) + \frac{1}{2} C_\times e^{ik_\sigma x^\sigma} S^2(0) \quad \text{and} \quad S^2 = S^2(0) + \frac{1}{2} C_\times e^{ik_\sigma x^\sigma} S^1(0).$$

In this case, the ring of particles bounce back and forth in the shape of an " $\times$ ", thus explaining why we have labelled these constants this way. This whole analysis therefore explains how a gravitational wave affects the spacetime it travels through in their weak field limit.

Now, how to these theoretical considerations connect to actual observations? In experiments such as LIGO, which is designed to detect gravitational waves, the impact of these waves on test masses in crucial. Gravitational waves passing through the Earth cause tiny, periodict stretching and squeezing effects on spacetime. The geodesic deviation equation, as we've derived, describes the behavior of these particles induced by gravitational waves.

9.4. Observation of Gravitational Waves

Not long after having confirmed the theory behind gravitational waves, experiments began with Weber and his resonant mass detectors in the 1960s. Interferometric detectors were also suggested in the early 1960s and further developed on the 1970s. These showed a lot of promise by being studied for their noise and performance, which led to proposals for long-baseline broadband laser interferometers with the potential for increased sensitivity. It wasn't until the early 2000s that the first detectors were completed, including TAMA 300 in Japan, GEO 600 y Germany, the Laser Interferometer Gravitational-Wave Observatory (LIGO) in the United States, and Virgo in Italy. These detectors worked together to make joint observations from 2002 through 2011. In 2015, Advanced LIGO became the first of a significantly more sensitive network of advanced detectors to begin observations of a higher variety of gravitational-wave sources.

9.4.1. The LIGO detectors

The LIGO sites, situated in Washington (LIGO Hanford) and Louisiana (LIGO Livingston), are separated by a distance of 3.002km. They each operate a single Advanced LIGO detector, which is a modified Michelson interferometer that measures gravitational-wave strain as a difference in length of its orthogonal arms. The idea of how these work is pretty simple and illustrated in figure 9.4; each arm is formed by two mirrors acting as test masses, and are separated by $L_x = L_y = L = 4km$. A passing gravitational wave would then alter the length of these arms, where the difference is measured as $\Delta L(t) = \delta L_x - \delta L_y = h(t)L$, where h is the gravitational-wave strain amplitude projected onto the detectors. By measuring the length variation, one can notice a difference on the phase of the two light fields returning to the beam splitter, transmitting measurements that can then be used to effectively make assumptions on how and where the gravitational wave was produced.

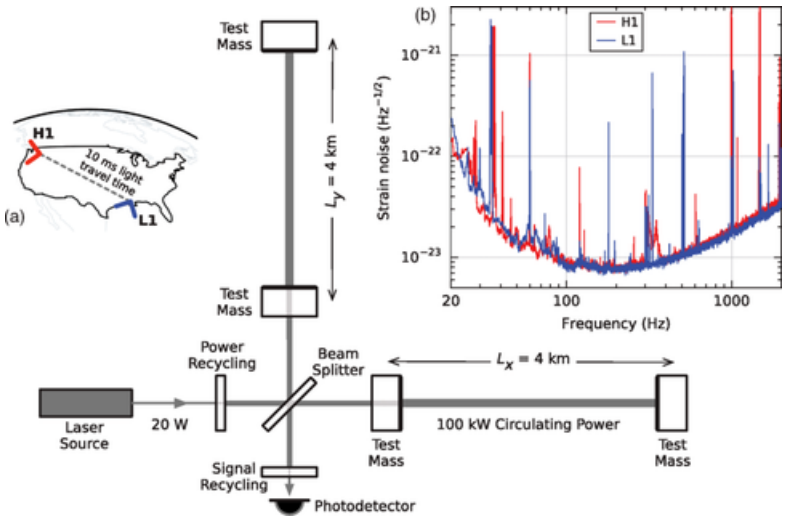

Figura 9.4: Simplified diagram of an Advanced LIGO detector.

The strain of a gravitational wave is the amount by which distances are stretched and compressed by a passing gravitational wave, relative to the original length; thus, it is a dimensionless number. The LIGO detectors, being receptors of such sensitive data as are gravitational waves, whose strain is of the order of 10^{-21}, must achieve sufficient sensitivity to (1) be able to measure gravitational waves and (2) be able to differenciate noise from relevant data, as is noise, light refraction between the mirrors, ground vibrations, etc. The detectors included several enhancements to the basic Michelson interferometers. These include adding mirrors to amplify the effects of a gravitational

wave, operating in a vacuum to reduce light refraction, and optimizing the gravitational-wave signal extraction by broadening the bandwidth of the arm cavities. To reduce seismic noise, test masses (ones emitting the light beams) are suspended on quadruple-pendulum systems, minimizing low frequencies. To minimize additional noise sources, all components, except the laser source, are mounted on vibration isolation stages in ultrahigh vacuum.

9.4.2. The GW150914 signal

On September 14, 2015 at 09:50:45 UTC, the LIGO Hanford, WA, and Livingston, LA, observatories detected the coincident signal of GW150914 (Fig. 9.5). Subsequent to the initial detection, the data were successfully acquired and reported. Matched-filter analyses that use relativistic models of compact binary waveforms cataloged GW150914 as the most significant event from each detector for the observations reported.

One might wonder why there is no mention of observations from the Virgo and GEO 600 detectors for this event. Unfortunately, Virgo was being upgraded at that moment, and GEO 600 was not sufficiently sensitive to detect this event. Luckily, two detectors operating was enough to locate the source position with 90% credibility.

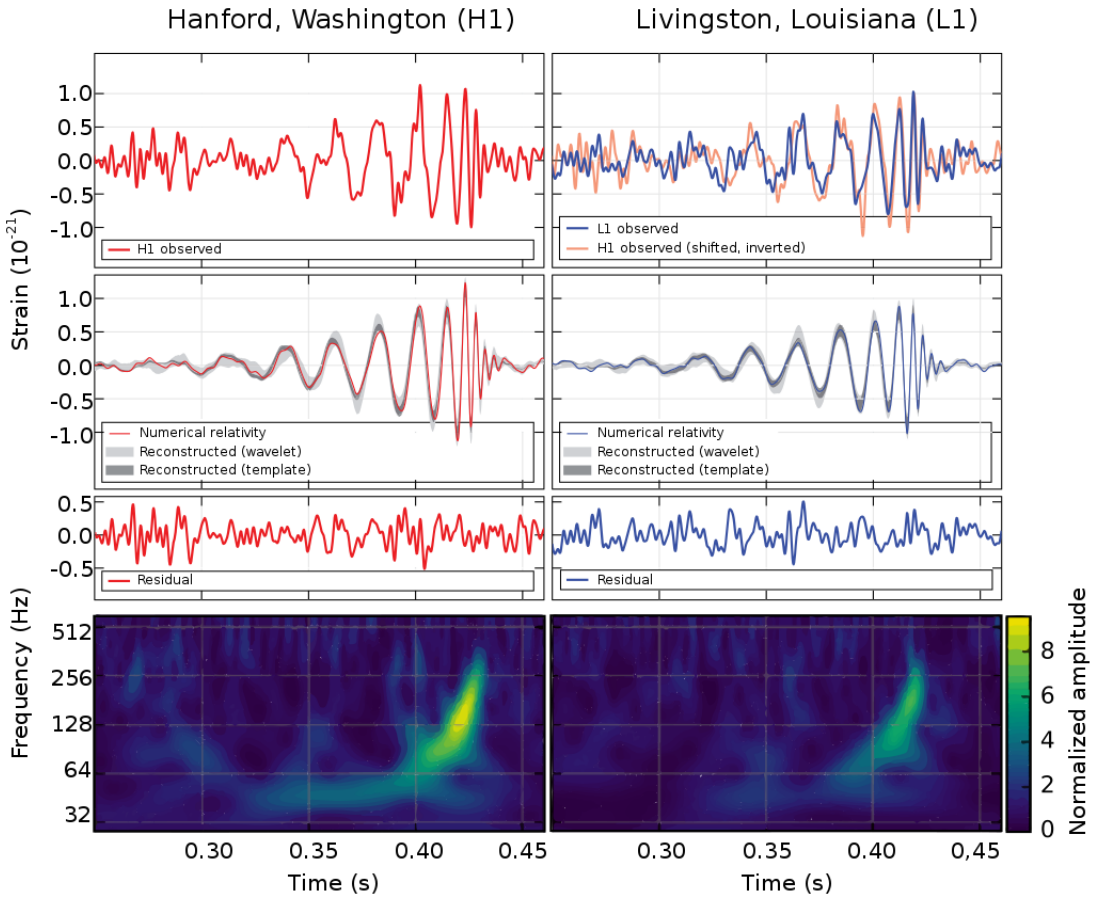

Figura 9.5: LIGO measurement of gravitational waves.

Basic features of GW150914 point to it being produced by the spiraling and merging of two black holes. Over 0.2s, the signal undergoes a remarkable transformation, increasing in frequency

and amplitude across approximately 8 cycles, ranging from 35 to 150Hz. This progression occurs at velocities extremely close to the speed of light, where the amplitude of the wave reaches a maximum and then decays in the black hole ringdown, as visually depicted in Fig. 9.6. This evolution is explained by a two body problem in general relativity; the inspiral of two orbiting masses m_1 and m_2, due to gravitational-wave emission. What explains the motion at lower frequencies is the chirp mass:

$$\mathcal{M} = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}} = \frac{c^3}{G}\left[\frac{5}{96}\pi^{-8/3} f^{-11/3}\dot{f}\right]^{3/5},$$

where f and $\dot{f}$ are the observed frequency and its time derivative.

The chirp mass of a compact binary system is a scalar that determines the leading-order orbital evolution of the system as a result of energy loss from emitting gravitational waves. It also determines the frequency evolution of the gravitational wave signal emitted during a binary's inspiral phase. When analyzing gravitational wave data, it is easier to measure the chirp mass than the two masses individually. By estimating f and $\dot{f}$ from the matched-filtering analysis of the LIGO detection, a chirp mass of $\mathcal{M} \approx 30\mathrm{M}_\odot$ is obtained, suggesting that the total mass $M = m_1 + m_2$ is approximately $70 \geq \mathrm{M}_\odot$. This bounds the sum of the Schwarzschild radii of the binary components to approximately $2GM/c^2 \geq 210$km. Since the gravitational wave frequency was 150Hz at its peak, we would need an orbital frequency of 75Hz. To reach this, objects must've been very close and very compact. Equal Newtonian point masses orbiting at this frequency (and having said chirp mass) would only be approximately 350km apart. While a pair of neutron stars would have been compact enough, they would not have had the required mass. On the other hand, a black hole neutron star binary would have a very large total mass, and thus, it would merge at a much lower frequency. This leaves black holes as the only objects compact enough to reach orbital frequency of 75Hz without contact. Furthermore, the decay of the waveform on the merging ringdown is consistent with damped oscillations of a black hole relaxing to a stationary Kerr configuration, leaving this as the only candidate to have formed the famous GW150914 gravitational wave detection.

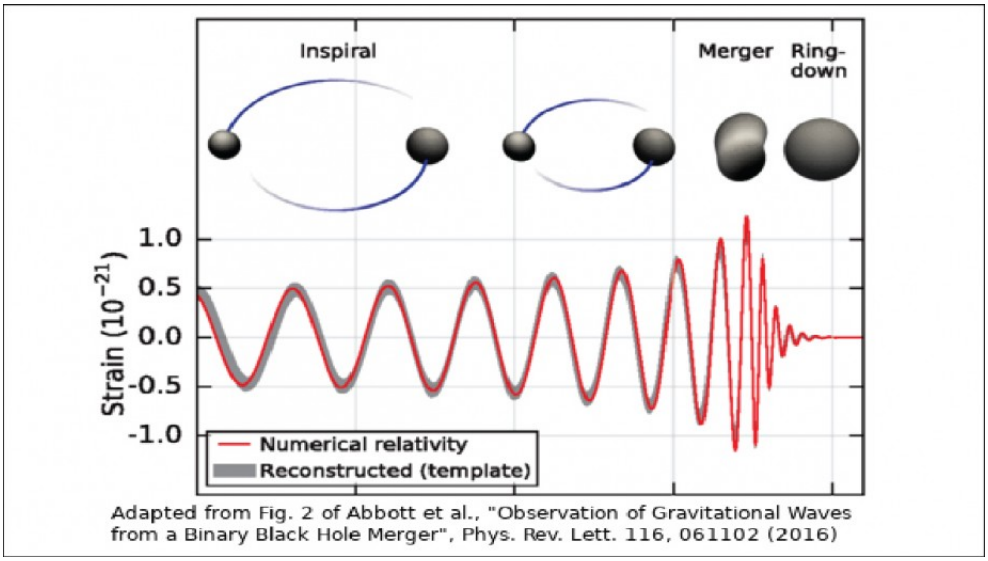

Figura 9.6: Diagram of a binary black hole merger , starting from a low frequency inicial inspiral, followed by the merging just after maximum frequencies and ending with a decay in the ringdown phase.

9.5. Using the NLHPC

For all simulations we are in need of high power computing, and for this we were given an account in the National Laboratory for High Performance Computing (NLHPC) cluster, who helped us finish this thesis successfully.

9.5.1. Introduction to the NLHPC

The NLHPC is a Chilean initiative that aims to provide high-performance computing (HPC) resources and support various scientific research and projects. It focuses on advancing computational research across a great variety of disciplines. It is currently located at the Center for Mathematical Modeling (CMM) in the University of Chile, being to date Chile's most powerful supercomputer available to researchers throughout the country.

We translate and quote NLHPC's mission:
'The consolidation of a national HPC facility by offering high-quality services and advanced training to respond to the high demand for scientific computing, developing connections between research groups, industry, and the public sector.'

The NLHPC system uses two different different clusters, Guacolda and Leftraru. Leftraru was the first cluster to begin operating in 2014, Guacolda coming in second in 2019 to expand the center's processing capacity by five times, reaching 266 teraflops and 5236 cores, with a hard drive of 274 terabytes and a RAM of 23 terabytes.

To begin using the NLHPC, one must enter its website http://www.nlhpc.cl and request an account. After being accepted, one is given a specific username. For these simulations, the account used is 'eiubini@leftraru.nlhpc.cl'.

Accessing the NLHPC clusters requires the use of a terminal, preferably an Ubuntu bash terminal. One uses the *SSH* protocol to log in remotely. With our specific account, the bash command needed would be:

```
$ ssh eiubini@leftraru.nlhpc.cl
```

After typing our password, we are finally inside the cluster, and free to use its features.

9.5.2. The SLURM system

Since there's a lot of researchers working on the NLHPC's clusters, they need a system that manages resources given to each user. It is central to the cluster's accessibility and one must learn how to use it to follow ones simulations.

SLURM (Simple Linux Utility for Resource Management) is an open-source job scheduling and cluster management system, fault-tolerant, and highly scalable for both large and small Linux clusters. As a cluster workload manager, Slurm servers three key functions: First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for a specified period, allowing them to perform their work. Second, it provides a framework for initiating, executing, and monitoring jobs (typically parallel jobs) on the allocated set of notes. Finally, it arbitrates resource contention by managing a queue of pending jobs. If you wand to launch tasks within Leftraru, you must do so through Slurm.

SLURM manages user jobs with the following key characteristics:

- Requested resource set:
  - Number of coputing resources: nodes (including all their CPU's and cores) or CPPU's (including all their cores) or only cores.
  - Amount of memory: per node or per CPU.
  - Time needed for user tasks to complete their work.
- Requested node partition (job queue).
- Requested Quality of Service (QoS) granting specific access to users.
- Requested account with limited resources.

The SLURM system manages different partitions in the NLHPC cluster, each one requested and used according to specific needs of our simulation. NLHPC expects responsibility in the use of these clusters, requesting small clusters for low-cost simulations and allowing others to request more powerful clusters if they need it. The partitions SLURM manages (up to date) are as following:

- general: 48 nodes, 44 CPUs each, 187GB RAM
- largemem: 9 nodes, 44 CPUs each, 765GB
- gpus: 2 nodes, 44 CPUs each, 187GB RAM, with 4 GPUs Nvidia Tesla V100
- slims: 132 nodes, 20 CPUs each, 46GB RAM
- debug: 4 nodes, 20 CPUs each, 59GB RAM, destined for jobs lasting no more than 30 minutes.

By default, users submit jobs to a specific partition called *debug* (designated for all users) and under a specific account (predefined by the user). When submitting a job to the SLURM system, it is assignated a specific Job ID, a number that grants access to the full time status of our simulation.

Also, SLURM has 5 basic commands used to perform most of the cluster's fundamental operations:

- srun: This command executes a specific command or script in designated nodes.
- sbatch: This command sends scripts to the squeue, allowing the user to monitor the script even when exiting the shell.
- squeue: This command displays our job's status, letting us algo know which nodes are assigned, Job ID, and current time running.
- scancel: This command erases a job running in the system.
- sinfo: This command displays the status on all the cluster's nodes, showing which nodes are available or in use.

9.5.3. The Eistein Toolkit inside the NLHPC

Since the Einstein Toolkit is an open-source framework, one must first install it inside the NLHPC. This must be done by someone familiar with the cluster, since they can be a little tricky to work with.

Fortunately, the ET was successfully installed previous to our simulations, but our best bet to download it in any cluster is to go to the Einstein Toolkit download tutorial, and follow the installation guide. In this case, the code needed to be executed inside our cluster is:

```
curl -kLO https://raw.githubusercontent.com/gridaphobe/CRL/master/GetComponents
chmod a+x GetComponents
./GetComponents --parallel --shallow https://bitbucket.org/einsteintoolkit/manifest/raw/
    ↪ master/einsteintoolkit.th
```

If this does not work, further assistance should be requested by the NLHPC support team.

9.6. Reproducing a quasi-circular binary merger

The Einstein Toolkit possesses all the tools to simulate the dynamics of black hole binaries: their inspiral, merger, the resulting gravitational wave emission, and even the quasi-normal modes in their ringdown phase. By working with a specific parfile, corresponding to the evolution of a system of two black holes, one can adjust parameters to explore different possible mergings.

Several parameters are available for adjustment:

- **Initial conditions:** Specify the masses of each black hole, their initial separation, their initial momenta (which sets the motion of the system), and also their initial spin.
- **The eccentricity of the binary orbit**
- **Grid resolution:** Allows for more detailed simulations. Users can specify higher resolutions near the black holes, with easy-to-set parameters for precision.
- **Simulation Time Steps**
- **Physical parameters:** Include the equation of state, aiding simulations involving matter. This is particularly relevant for modeling systems beyond black hole binaries, such as neutron star binaries (NS-NS or NS-BH systems).
- **Simulation Domain Size Parameters:** Important to prevent radiation in the spatial boundaries from reflecting and affecting the central system's motion.
- **Numerical Techniques Parameters:** Related to the numerical methods used to solve the Einstein field equations.
- **Output Intervals:** Specify the quantity and type of data to be displayed.

We will start with an initial simulation of a quasi-circular binary, understanding the parfile used and the parameters set to analyze relevant output. Following this, we will conduct a simulation of the first gravitational wave detection, GW150914, on a supercomputer cluster. This process involves analyzing the parameters set and comparing the simulated results with the actual detection.

9.6.1. Analyzing the parfile

The following parfile is an example of a black hole binary that takes about 20 hours using 40 CPUs in a cluster to compute. To know what our simulation is doing, we analyze our Einstein Toolkit parfile. All code inside this parfile is important on its own, but there are some we will focus on to discuss how it changes our simulation.

Each parfile found or made can be different on its own, since they're coded by programmers. While every line in the parfile is essential, we specifically focus on those not prefixed with a '#' sign. Cactus interprets lines with '#' as comments, excluding them from the software's reading.

```
1 #=============================================================================
    ↪
2 # BBH: TwoPunctures - Mclachlan
3 #=============================================================================
    ↪
4 # Changed output options wrt to example file
5 #-----------------------------------------------------------------------------
6 # Cactus parameters:
7 #-----------------------------------------------------------------------------
8 #Cactus::cctk_run_title = "BBH"
9 #Cactus::cctk_full_warnings        = "yes"
10 #Cactus::terminate="runtime"
11 #Cactus::max_runtime= 5120
```

```
12
13 TerminationTrigger::max_walltime = 12.0
14 TerminationTrigger::on_remaining_walltime       = 30 # minutes
15 TerminationTrigger::output_remtime_every_minutes = 60
16 Cactus::terminate                              = time
17 Cactus::cctk_final_time                        = 200
```

The initial segment of the parfile is dedicated to configuring the essential parameters that govern the simulation. One crucial aspect established in the parfile is the final time of the simulation. This parameter, denoted as "cctk_final_time" determines the duration of the simulation run. Care must be taken to set an appropriate final time, as an excessively large value may lead to undesired effects, such as the gravitational waves generated 'bouncing' off the simulation boundaries.

To mitigate this issue, one common strategy is to adjust the spatial domain of the simulation. By ensuring that the spatial domain is sufficiently big, the gravitational waves can propagate without encountering the boundaries too soon. In the provided example, the final time is set to 200, representing the time duration of the simulation. It is worth noting that this value is carefully chosen to balance between the capturing the relevant dynamics of the system, and preventing unwanted reflections.

Additionally, within the parfile, there is typically a comment indicating the maximum permissible value for the final time. In this specific case, the comment specifies that our simulation's maximum allowable final time is set at 5120. .

```
1  #------------------------------------------------------------------------------
2  # Activate all necessary thorns:
3  #------------------------------------------------------------------------------
4
5  ActiveThorns = "Boundary CartGrid3D CoordBase Fortran InitBase IOUtil LocalReduce
       ↪ SymBase Time"
6  ActiveThorns = "AEILocalInterp LocalInterp"
7  #ActiveThorns = "MoL ReflectionSymmetry RotatingSymmetry90 RotatingSymmetry180
       ↪ Slab SpaceMask SphericalSurface"
8  ActiveThorns = "MoL ReflectionSymmetry RotatingSymmetry180 Slab SpaceMask
       ↪ SphericalSurface"
9  #ActiveThorns = "MoL ReflectionSymmetry Slab SpaceMask SphericalSurface"
10 ActiveThorns = "Carpet CarpetInterp CarpetIOASCII CarpetIOHDF5 CarpetIOScalar
       ↪ CarpetLib CarpetIOBasic CarpetReduce CarpetRegrid2 CarpetSlab CarpetTracker
       ↪ CarpetMask LoopControl SystemTopology"
11 #ActiveThorns = "Formaline NaNChecker TerminationTrigger TimerReport"
12 ActiveThorns = "NaNChecker TerminationTrigger TimerReport"
13 ActiveThorns = "ADMbase ADMcoupling ADMmacros CoordGauge StaticConformal"
14 ActiveThorns = "PunctureTracker"
15 ActiveThorns = "TmunuBase"
16 ActiveThorns = "QuasiLocalMeasures"
17 #ActiveThorns = "ADMConstraints"
18 #ActiveThorns = "BLAS LAPACK GSL HDF5"
19 ActiveThorns = "TwoPunctures"
20 ActiveThorns = "SummationByParts"
21 ActiveThorns = "GenericFD NewRad"
22 ActiveThorns = "ML_BSSN ML_BSSN_Helper ML_ADMConstraints"
23 ActiveThorns = "Dissipation"
24 ActiveThorns = "AHFinderDirect"
25 ActiveThorns = "WeylScal4 Multipole"
```

```
ActiveThorns = "SystemStatistics"
```

Next, we tell Cactus which thorns it will use on our simulations. We will give a brief explanation of some relevant thorns, though it is important to mention that each thorn plays a part on our coding.

- **MoL thorns:** These thorns correspond to Method of Lines thorns, which is a numerical scheme to solve evolution equations. There are several MoL schemes, so one must later specify which one to use.
- **Symmetry thorns:** When working with symmetrical problems (like our spherically symmetric black hole, for instance), these thorns are in charge of reducing our computational work when applicable.
- **NaNChecker:** This thorn makes sure our simulation ends or warns us when there are NaN (Not a Number) characters found, for example, when gradients explode or divisions by 0 are being made.
- **PunctureTracker:** This one is part of the thorns we will use most. This thorn keeps track of our black hole punctures (singularities) position and momentum in our grid, and its used to evolve to the next time step in our simulation.
- **QuasiLocalMeasures:** This thorn holds a data array of 111 columns. Some important columns are: coordinate spins, areas of the black holes, radii of the black holes, energy of each black hole, etc. It implements the calculation of mass and spin multipoles from the isolated and dynamical horizon formalism. It takes as input a horizon surface, or any other surface that the user specifies, and calculates useful quantities such as the Weyl or Ricci scalar, in addition to physical observables such as mass and momenta.
- **WeylScal4:** This thorn contains the Weyl scalars, which is a set of five complex scalars $\Psi_0, ..., \Psi_4$ which encode the independent components of the Weyl tensor on a 3+1 spacetime. We will discuss how these are related to our gravitational wave radiation later.

```
#-----------------------------------------------------------------------------
# Run parameters:
#-----------------------------------------------------------------------------

#------
# Grid:
#------

Time::dtfac = 0.25
MoL::ODE_Method             = "rk4"
MoL::MoL_Intermediate_Steps = 4
MoL::MoL_Num_Scratch_Levels = 1

CartGrid3D::type          = "coordbase"
CartGrid3D::domain          = "full"
CartGrid3D::avoid_origin    = "no"

CoordBase::domainsize       = "minmax"
CoordBase::spacing          = "gridspacing"  # "gridspacing" or "numcells"

# On a workstation (Total required memory: ~1.9GB):
#CoordBase::xmin =  0.00
```

```
#CoordBase::ymin =-12.00
#CoordBase::zmin =  0.00
#CoordBase::xmax = 12.00
#CoordBase::ymax = 12.00
#CoordBase::zmax = 12.00
#CoordBase::dx   =  0.40  # dx or ncells_x
#CoordBase::dy   =  0.40  # dy or ncells_y
#CoordBase::dz   =  0.40  # dz or ncells_z

# On a HPC (Total required memory: ~6.5GB):
CoordBase::xmin =   0.00
CoordBase::ymin =-120.00
CoordBase::zmin =   0.00
CoordBase::xmax = 120.00
CoordBase::ymax = 120.00
CoordBase::zmax = 120.00
CoordBase::dx   =   1.50  # dx or ncells_x
CoordBase::dy   =   1.50  # dy or ncells_y
CoordBase::dz   =   1.50  # dz or ncells_z
```

Within this section, we instruct our MoL thorn on the specific numerical evolution method it should employ. For the current simulation, we opt for the widely-used Runge-Kutta 4 (rk4) method. It's noteworthy that many parfiles provide guidance on configuring simulations to match the computational capabilities of the hosting machine. In our case, these simulations are executed on the NLHPC, and adhering to the advice in the parfile becomes very important. Consequently, we uncomment the relevant lines tailored to the NLHPC environment.

Furthermore, the parfile configures the spatial grid for our simulations. In this instance, we set up a cubic grid with dimensions of 120 units on each side. The initial spatial step size is established at 1.5, setting up parameters for the discretization of the spatial domain. This step size is chosen to balance computational efficiency and precision.

```
CarpetRegrid2::regrid_every = 32
CarpetRegrid2::num_centres  = 2

# On a workstation (Total required memory: ~1.9GB):
#CarpetRegrid2::num_levels_1        =  6
#CarpetRegrid2::position_x_1        = +3.0
#CarpetRegrid2::radius_1[ 1]        =  2.8
#CarpetRegrid2::radius_1[ 2]        =  2.0
#CarpetRegrid2::radius_1[ 3]        =  1.2
#CarpetRegrid2::radius_1[ 4]        =  0.6
#CarpetRegrid2::radius_1[ 5]        =  0.3
#CarpetRegrid2::movement_threshold_1 =   0.16

#CarpetRegrid2::num_levels_2        =  6
#CarpetRegrid2::position_x_2        = -3.0
#CarpetRegrid2::radius_2[ 1]        =  2.8
#CarpetRegrid2::radius_2[ 2]        =  2.0
#CarpetRegrid2::radius_2[ 3]        =  1.2
#CarpetRegrid2::radius_2[ 4]        =  0.6
#CarpetRegrid2::radius_2[ 5]        =  0.3
#CarpetRegrid2::movement_threshold_2 =   0.16
```

```
22
23 # On a HPC (Total required memory: ~6.5GB):
24 CarpetRegrid2::num_levels_1        =  7
25 CarpetRegrid2::position_x_1        = +3.0
26 CarpetRegrid2::radius_1[ 1]        =  64.0
27 CarpetRegrid2::radius_1[ 2]        =  16.0
28 CarpetRegrid2::radius_1[ 3]        =   8.0
29 CarpetRegrid2::radius_1[ 4]        =   4.0
30 CarpetRegrid2::radius_1[ 5]        =   2.0
31 CarpetRegrid2::radius_1[ 6]        =   1.0
32 CarpetRegrid2::movement_threshold_1 =   0.16
33
34 CarpetRegrid2::num_levels_2        =  7
35 CarpetRegrid2::position_x_2        = -3.0
36 CarpetRegrid2::radius_2[ 1]        =  64.0
37 CarpetRegrid2::radius_2[ 2]        =  16.0
38 CarpetRegrid2::radius_2[ 3]        =   8.0
39 CarpetRegrid2::radius_2[ 4]        =   4.0
40 CarpetRegrid2::radius_2[ 5]        =   2.0
41 CarpetRegrid2::radius_2[ 6]        =   1.0
42 CarpetRegrid2::movement_threshold_2 =   0.16
```

This section of our parfile holds significant importance as it contains very relevant details for our simulation. Initially, it specifies the number of black holes in our system, determining the quantity of centers to be studied. In this particular case, we consider two black holes.

The specialized thorn, CarpetRegrid2, allows us to change our grid depending on where the black holes are located. The line *CarpetRegrid2::num_levels_1 = 7* indicates that it will make 7 different grid resolutions (and could be different for each black hole, as seen in the two different blocks of code). Starting from the box containing the black hole, it will make our spatial grid 64x thinner, then 16x thinner, and so on until it reaches most outside levels.

An important line is *CarpetRegrid2::position_x_1 = +3.0*, which specifies the starting position of the black hole. In this parfile we work with a quasi-circular binary, in which both masses will be equal, and momenta will just differ on initial direction, one going strictly upwards on our plane and the other backwards. Therefore, we start with a symmetric starting positions of $+3$ and -3. Other black hole configurations might need different starting positions so as to not lose our black holes from our simulation domain.

```
1 #------------------------------------------------------------------------------
2 # Analysis:
3 #------------------------------------------------------------------------------
4 AHFinderDirect::find_every = 32
5
6 #AHFinderDirect::run_at_CCTK_ANALYSIS = "yes"
7 #AHFinderDirect::run_at_CCTK_POSTSTEP = "no"
8 AHFinderDirect::run_at_CCTK_POST_RECOVER_VARIABLES = "no"
9
10 AHFinderDirect::move_origins           = "yes"
11 AHFinderDirect::reshape_while_moving    = "yes"
12 AHFinderDirect::predict_origin_movement = "yes"
13
```

```
# Hermite to order 3 to avoid discontinuities in the metric spatial derivatives:
#AHFinderDirect::geometry_interpolator_name = "Hermite polynomial interpolation"
#AHFinderDirect::geometry_interpolator_pars = "order=3"
#AHFinderDirect::surface_interpolator_name  = "Hermite polynomial interpolation"
#AHFinderDirect::surface_interpolator_pars  = "order=3"
AHFinderDirect::geometry_interpolator_name = "Lagrange polynomial interpolation"
AHFinderDirect::geometry_interpolator_pars = "order=4"
AHFinderDirect::surface_interpolator_name  = "Lagrange polynomial interpolation"
AHFinderDirect::surface_interpolator_pars  = "order=4"

AHFinderDirect::output_h_every = 0

AHFinderDirect::N_horizons = 3

AHFinderDirect::origin_x                            [1] = +3.0
AHFinderDirect::initial_guess___coord_sphere___x_center[1] = +3.0
AHFinderDirect::initial_guess___coord_sphere___radius  [1] =  0.25
AHFinderDirect::which_surface_to_store_info         [1] = 0
AHFinderDirect::set_mask_for_individual_horizon     [1] = no
AHFinderDirect::reset_horizon_after_not_finding     [1] = no
AHFinderDirect::track_origin_from_grid_scalar       [1] = yes
AHFinderDirect::track_origin_source_x               [1] = "PunctureTracker::pt_loc_x[0]"
AHFinderDirect::track_origin_source_y               [1] = "PunctureTracker::pt_loc_y[0]"
AHFinderDirect::track_origin_source_z               [1] = "PunctureTracker::pt_loc_z[0]"
AHFinderDirect::max_allowable_horizon_radius        [1] = 3

AHFinderDirect::origin_x                            [2] = -3.0
AHFinderDirect::initial_guess___coord_sphere___x_center[2] = -3.0
AHFinderDirect::initial_guess___coord_sphere___radius  [2] =  0.25
AHFinderDirect::which_surface_to_store_info         [2] = 1
AHFinderDirect::set_mask_for_individual_horizon     [2] = no
AHFinderDirect::reset_horizon_after_not_finding     [2] = no
AHFinderDirect::track_origin_from_grid_scalar       [2] = yes
AHFinderDirect::track_origin_source_x               [2] = "PunctureTracker::pt_loc_x[1]"
AHFinderDirect::track_origin_source_y               [2] = "PunctureTracker::pt_loc_y[1]"
AHFinderDirect::track_origin_source_z               [2] = "PunctureTracker::pt_loc_z[1]"
AHFinderDirect::max_allowable_horizon_radius        [2] = 3

AHFinderDirect::origin_x                            [3] = 0
AHFinderDirect::find_after_individual_time          [3] = 100.0
AHFinderDirect::initial_guess___coord_sphere___x_center[3] = 0
AHFinderDirect::initial_guess___coord_sphere___radius  [3] = 1.0
AHFinderDirect::which_surface_to_store_info         [3] = 2
AHFinderDirect::reset_horizon_after_not_finding     [3] = no
AHFinderDirect::max_allowable_horizon_radius        [3] = 6
```

Now focusing on the analysis segment of our parfile, we look into the functionality of the AHFinderDirect thorn. This thorn employs numerical methods to computationally determine the locations of black hole event horizons. The term 'AH' in AHFinderDirect stands for Apparent Horizon, representing the boundary beyond which even light cannot escape the gravitational pull of a black hole, as calculated by computer formulae.

One noticeable aspect of this analysis is the inclusion of lines containing the term 'guess.' These lines reveal a dynamic process wherein the thorn strategically refines its search for the apparent horizons (with the help of the PunctureTracker thorn). The PunctureTracker thorn keeps track of the black hole centers, enabling AHFinderDirect to make informed 'guesses' about the probable locations of event horizons.

```
PunctureTracker::track                     [0] = "yes"
PunctureTracker::initial_x                 [0] = 3.0
PunctureTracker::which_surface_to_store_info[0] = 0
PunctureTracker::track                     [1] = "yes"
PunctureTracker::initial_x                 [1] = -3.0
PunctureTracker::which_surface_to_store_info[1] = 1

QuasiLocalMeasures::num_surfaces   = 3
QuasiLocalMeasures::spatial_order  = 4
QuasiLocalMeasures::interpolator = "Lagrange polynomial interpolation"
QuasiLocalMeasures::interpolator_options = "order=4"
QuasiLocalMeasures::surface_index  [0] = 0
QuasiLocalMeasures::surface_index  [1] = 1
QuasiLocalMeasures::surface_index  [2] = 2

Multipole::nradii       = 4
Multipole::radius[0]    = 30
Multipole::radius[1]    = 40
Multipole::radius[2]    = 50
Multipole::radius[3]    = 60
Multipole::ntheta       = 120
Multipole::nphi         = 240
Multipole::variables    = "WeylScal4::Psi4r{sw=-2 cmplx='WeylScal4::Psi4i' name='psi4'}"
Multipole::out_every    = 4
Multipole::l_max        = 4

WeylScal4::fd_order = "4th"
```

The next section of the analysis output focuses on various thorns:

We set the PunctureTracker to effectively track the black hole positions throughout the simulation, and give this information to other thorns.

For Multipole, we specify the number of radii in which we will calculate the Weyl scalar, specifying each of their sizes. This helps to better analyze how the gravitational waves (as studied with the Weyl scalar Psi4) travels outside the black hole.

We note that the Weyl scalar is calculated up to 4th order in *WeylScal4::fd_order = "4th"*. The Einstein Toolkit currently calculates the Weyl scalar up to 8th order, but the calculations get pretty messy, and it is not really necessary to look much further to higher orders when working on binary systems.

```
#------------------------------------------------------------------------
# Checkpoint/Recovery:
#------------------------------------------------------------------------
IOHDF5::checkpoint                 = "yes"
```

```
IO::checkpoint_ID                 = "yes"
IO::recover                       = "autoprobe"
IO::checkpoint_every_walltime_hours = 6.0
# IO::out_proc_every              = 2
IO::checkpoint_keep               = 3
IO::checkpoint_on_terminate       = "yes"
IO::checkpoint_dir                = "../checkpoints"
IO::recover_dir                   = "../checkpoints"
IO::abort_on_io_errors                  = yes
CarpetIOHDF5::open_one_input_file_at_a_time = yes
```

The final part of the parfile just specifies certain checkpoint settings. Since simulations take so much time, one usually allows the computer to store checkpoints. If anything went wrong with our simulation in say, two days, we could resume the simulation starting from the last checkpoint, and not lose relevant data.

9.6.2. Plot and analysis of the quasi-circular merger

After running the script, we have retrieved from the simulation the positions of our binary black hole system in the $x-y$ plane, and subsequently plotted them in Fig. 9.7. Positions of both black holes in Cartesian coordinates are all stored in an ASC file called *'puncturetracker-ptloc..asc'*, along with several other parameters. The trajectories are both retrieved using the following Python function, which specifies the columns of each parameter plotted, namely; time and (x, y, z) positions for the two black holes,

```
def get_trajectories(path):

    ascii_grid = np.loadtxt(f'{path}puncturetracker-pt_loc..asc')

    idx = [8, 22, 23, 32, 33, 42, 43]
    cols = ['t', 'x1', 'x2', 'y1','y2','z1','z2']
    id_dict = dict(zip(cols, idx))

    t = ascii_grid[:,id_dict['t']]
    trajectory1 = ascii_grid[:,id_dict['x1']], ascii_grid[:,id_dict['y1']]
    trajectory2 = ascii_grid[:,id_dict['x2']], ascii_grid[:,id_dict['y2']]

    return t, trajectory1, trajectory2
```

Since files contain such variety of parameters, it is important to check which variables are being plotted. Most of this information can be checked going to the respective output file and checking the column names, where some previous conventions for names must be known, such as p for momenta, or m for masses.

Since no momentum outside this plane was initially given, and neither was spin for our black holes, the it makes sense to only study the system solely on this plane. Units for the distance are given in terms of the solar mass, and taking remaining constants to be equal to one, that is, $G = c = 1$.

As specified in the parfile, starting positions for our black holes are set on $+3$ and -3 according to our labelling. Initial momentums for both black holes specified before are given by $P_+ = +0.13808$ and $P_- = -.13.808$ respectively. These values contribute to the symmetry of our system, which makes our black holes orbit in a circular manner until their collision and merging.

Having shown the trajectories of our binary black hole system, we now turn our attention to the analysis of gravitational waves. The positions of the black holes in the $x-y$ plane have been plotted, providing information of their motion over time (Fig. 9.7). To get further understanding of the astrophysical phenomena, we examine the gravitational waves emitted during the quasi-circular merger.

These plots do not represent in the slightest how fast they would be orbiting each other, neither the gravitational waves produces by this spiraling. For this, we look at files called *mp_psi4_l{l}_m{m}_r{r}*, where parameters inside brackets are replaces by their namesake value. In this case, (l, m) represent the respective coefficient in the spherical harmonics decomposition of the Ψ_4 scalar, and r represents the radius at which this scalar is computed. which in the asymptotic limit completely describes the outgoing gravitational radiation field. This means that far from a source, a gravitational wave is locally plane, and Ψ_4 is directly related to the metric perturbation in a 'traceless-transverse' gauge of our perturbation tensor $h_{\mu\nu}$ as,

$$\Psi_4 = \partial_t^2 (h_+ - i h_\times).$$

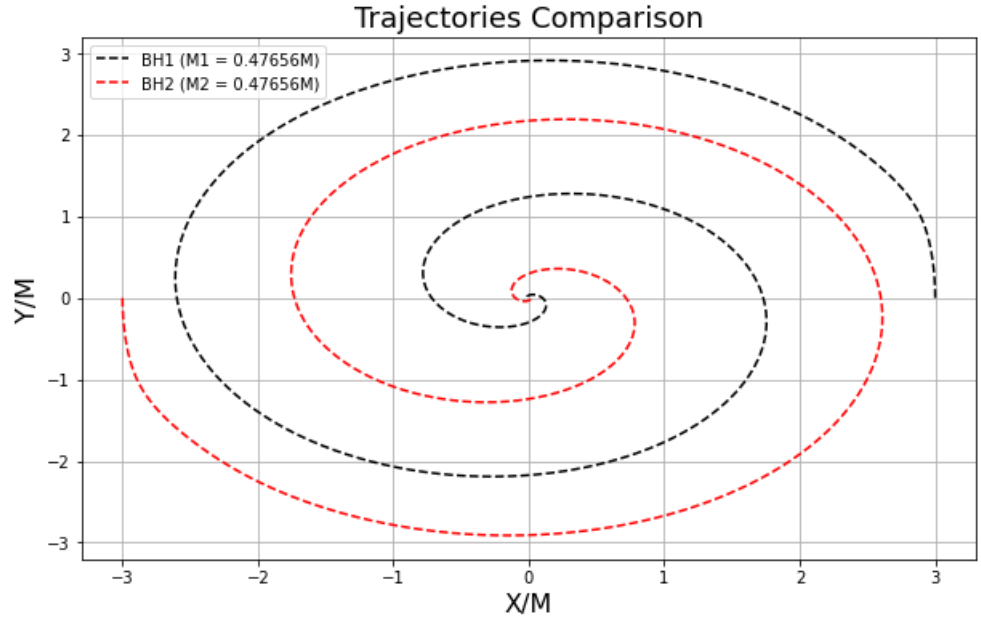

Figura 9.7: The trajectories over time of our quasi-circular binary black hole system. Starting positions are -3 and 3, beginning their spiraling at increasing frequencies and ending in their merging in the center.

In an asymptotically flat spacetime using appropriate coordinates, we can prove that Ψ_4 falls off as r^{-1}, thus gravitational waves are often described not by Ψ_4 but by $r\Psi_4$, which should be evaluated for big enough r. This notation is introduced on our code to read and define the Ψ_4 scalar,

```
def read_psi4(path, r, lm = (2,2)):

    l,m = lm
    gw_data = np.loadtxt(f'{path}mp_psi4_l{l}_m{m}_r{r}.asc')

    t = gw_data[:,0]
    re_psi4 = gw_data[:,1]
    im_psi4 = gw_data[:,2]

    psi4 = float(r)*np.array(re_psi4 + 1j*im_psi4)

    return t, psi4
```

The Ψ_4 scalar is defined on our whole spacetime, but using our methods we can derive an equation to get its expression into a form involving hypersurface quantities only.

After this, Ψ_4 is decomposed into spin-weighted spherical harmonics,

$$\Psi_4 = \sum_{l\geq 2, |m|\leq l} \Psi_4^{lm} {}_{-2}Y^{lm}.$$

Normally, the dominant part of the gravitational wave signal is in the lowest modes, with $l = 2$. The other modes are also important to gravitational-wave data analysis, recoil calculations, etc, but

as we will notice, they are of far lower order than $l = 2$.

Let us begging plotting (Fig. 9.8) the strain of the gravitational waves at a radius of $r = 50$. The spherical harmonics nodes which are dominant for a binary are $(l, m) = (2, 2)$, for which we plot both the real part and the imaginary part; the real part corresponding to the amplitude and the imaginary to a shift.

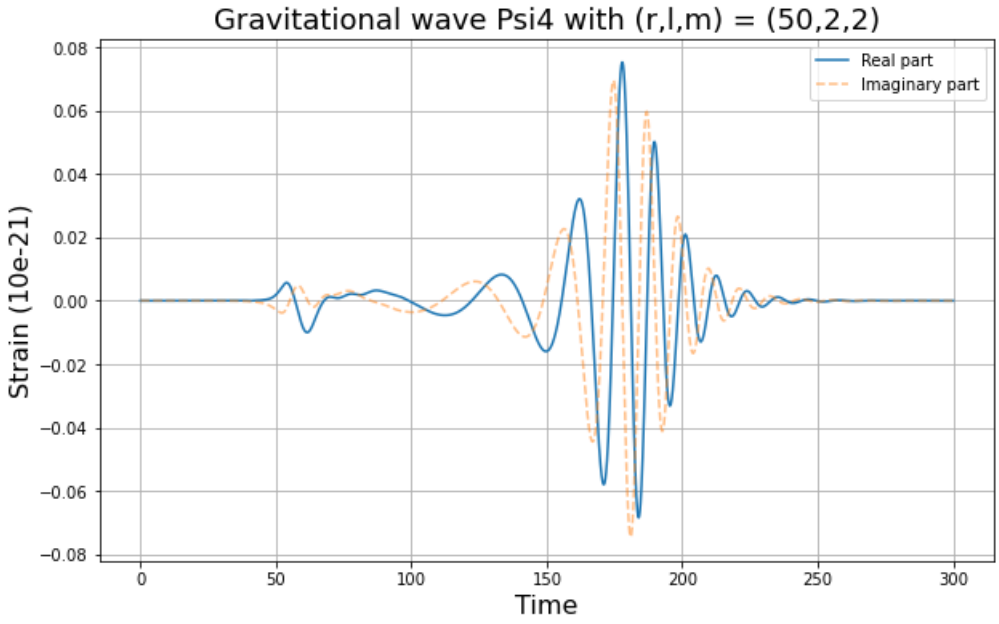

Figura 9.8: The real part and imaginary part of the gravitational wave strain for our quasi-circular binary for $(r, l, m) = (50, 2, 2)$.

It is important to know that the initial wiggle on our simulation is not physical; numerical errors arise when applying an initial momentum to our system, which rapidly stabilizes. We notice that at the 100 time mark, low frequency waves begin to appear, which grow quickly to high frequency and high amplitude waves. These correspond to the faster spiraling of our system until the black holes are so close together that they begin to lose energy, and so the amplitude starts decreasing, all this while the black holes continue to spiral around each other at high speeds. After the merging, although the resulting merged black hole could be moving in spacetime, there are no gravitational waves being formed by a binary system, so we consider the ringdown phase as the part in the graph where there the amplitud goes back to zero.

Let us show how other values for (l, m) contribute to the Ψ_4 scalar. Let us consider a higher order node, in this case, $(l, m) = (4, 4)$ at the same radius $r = 50$, plotted in Fig. 9.9. We notice these nodes contribute to high frequencies, but at a much lower amplitude that the dominant node $(2, 2)$. By being even numbered nodes and considering the highest value of m for this particular l, the strain although small, still contributes to the overall gravitational wave.

These are of the same order of the dominant node, but are not completely necessary to the study of gravitational waves unless we make a more detailed visualization of gravitational waves. One can also plot all possible values of (l, m) up to $l = 8$. Higher values of l are usually not relevant to any kind of study on gravitational waves, since they are of such low order that they are insignificant. Other value of pairs can algo be neglected, as seen in the following plot (Fig. 9.10) where we choose the values $(l, m) = (3, 1)$,

A natural question might be how well our simulated results align with theoretical expectations

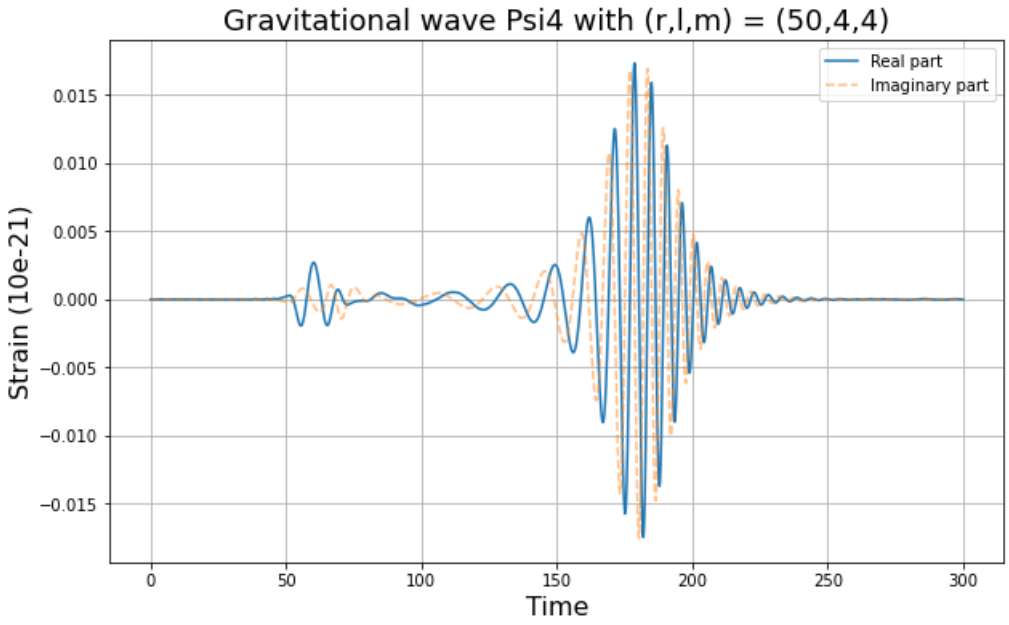

Figura 9.9: The real part and imaginary part of the gravitational wave strain for our quasi-circular binary for $(r, l, m) = (50, 4, 4)$.

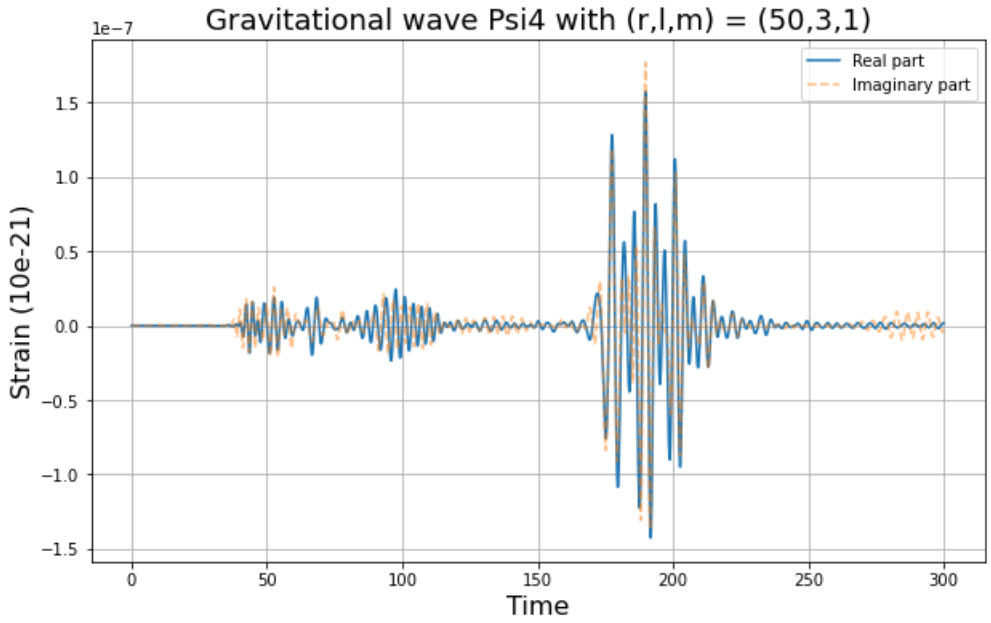

Figura 9.10: The real part and imaginary part of the gravitational wave strain for our quasi-circular binary for (r,l,m)=(50,3,1).

or observational data from actual binary black hole mergers detected by gravitational wave observatories. While our simulated system provides a controlled environment for studying gravitational waves, comparisons with observed events can validate the accuracy of our model and contribute to a more thorough understanding of these phenomena.

9.7. Reproducing the GW150914 merger

Finally we have all the tools necessary to study the merging of the first-ever detected binary black hole system, the GW150915 signal. This simulation's parfile is already available in the Einstein's Toolkit website, with tutorials on how to plot relevant data, as well as multiple programs to visualize data as one would desire. For the purposes of this thesis, we focus on the black hole trajectories and their gravitational waves' strain.

This simulation has a cost of 8700 core hours, which amounts to approximately 2.8 days on 128 cores, according to the Einstein Toolkit computational details of this simulation. Also, a total memory of 98 GB is needed, thus we must have a high performance computing at hand so as to not run out of memory and finish our simulations at reasonable time. This merging uses the same parfile we have used before, but with data that has already been recovered from the LIGO measurements. This means that we will use parameters which were already observed from the GW150914 merger, and try to reproduce this signal using our computational code.

Throughout the simulation, we use normalized units where $G = c = 1$. This choice simplifies the equations and results in dimensionless quantities. For clarity, all physical parameters are expressed in terms of these dimensionless units, unless specified otherwise.

We notice some important parameters for this simulation, all of which are specified on the parfile. These are,

- Physical parameters :
  - An initial separation D of $10M$,
  - A mass ratio $q = m_1/m_2 = 36/29$,
  - A spin of $\chi_1 = a_1/m_1 = 0.31$ for our first black hole,
  - A spin of $\chi_2 = a_2/m_2 = -0.46$ for our second black hole.
- Physical properties:
  - A number of orbits of 6,
  - A time from the beginning of the simulation until the merging of $899M$ (in normalized units $G = c = 1$),
  - The mass of our final black hole $0.95M$,
  - The spin of our final black hole 0.69.

The choice of parameters reflects the observed characteristics of GW150914. For instance, the specified mass ratio and spins align with LIGO's findings, and the time parameters indicate the duration of the simulated merger, resulting in a final black hole with specific mass and spin properties.

As seen by the physical parameters, the LIGO analysis indeed found that the merger consisted of a 36+29 solar mass binary black hole system, with a remnant of a 62 solar mass black hole, while the remaining 3 solar masses were radiated as gravitational waves. This simulation evolves the last 6 orbits and merger of a binary black hole merger with parameters to match the GW150914 event. These properties allow us to begin our simulation up until the merging occurs.

9.7.1. Plot and analysis of the GW150914 merger

These simulations were ran on a high performance cluster (NLHPC) using 100 cores. We plot the initial trajectories of the two black holes in Fig. 9.11, starting with an initial separation of $10M$. The black dotted line represents the heavier black hole, and the red dotted line represents the lighter.

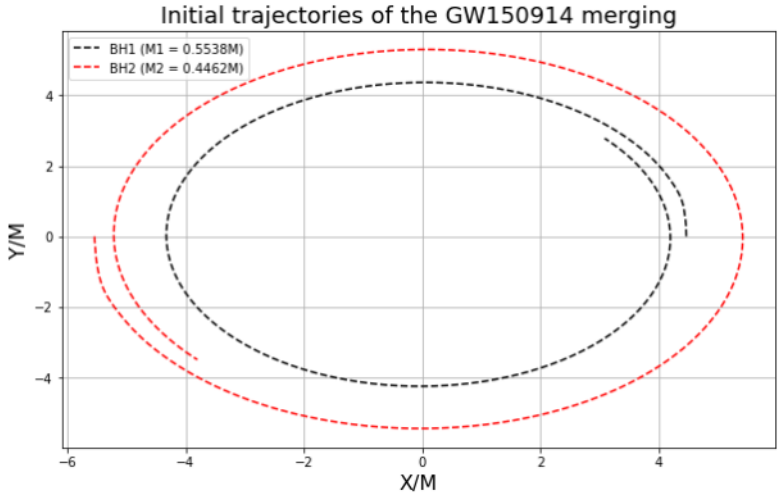

Figura 9.11: Initial trajectories of the GW150914 merger in the $x-y$ plane. The black dotted line represents the heavier black hole, and the red dotted line represents the lighter. This corresponds to about one full orbit of the two black holes, equivalent to approximately two gravitational waves.

It is important to notice we are plotting results in the $x-y$ plane. Due to spin and momentum initial conditions, the system should be moving through all space axes, and it is possible to graph them all in whichever program is able to read the corresponding files. For the sake of simplicity, we focus on $x-y$ movements of both black holes. After 6 orbits around each other, our simulations show how the binary collides and merges, as seen in the following plot (Fig. 9.12) depicting the final trajectories just before the merging.

The final black hole has three solar masses less than what the system had initially. All the remnant energy would come out in the form on gravitational waves. In the following plot, we plot the Ψ_4 scalar as seen by a radius of $r = 500$, far away from the collision. We again plot the dominant spherical harmonics node in the black hole binary, which is $(l, m) = (2, 2)$, which better describes the gravitational waves formed by the system. This famous signal's simulation is plotted below in Fig. 9.13.

These waveforms are usually compared to the ones observed by the LIGO detectors, and results have shown that numerical relativity is very accurate when predicting gravitational wave behavior for binary systems. A visual representation of this comparison is usually seen in any article about the famous LIGO GW150914 detection (Fig. 9.5).

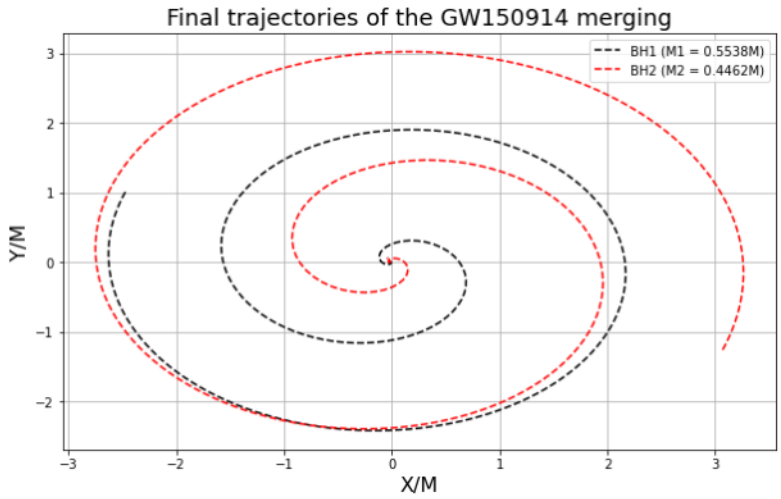

Figura 9.12: Final trajectories of the GW150914 merger in the $x - y$ plane. The higher frequency in this phase results in more visible orbits of the black holes, indicating a faster spiral compared to the initial phase.

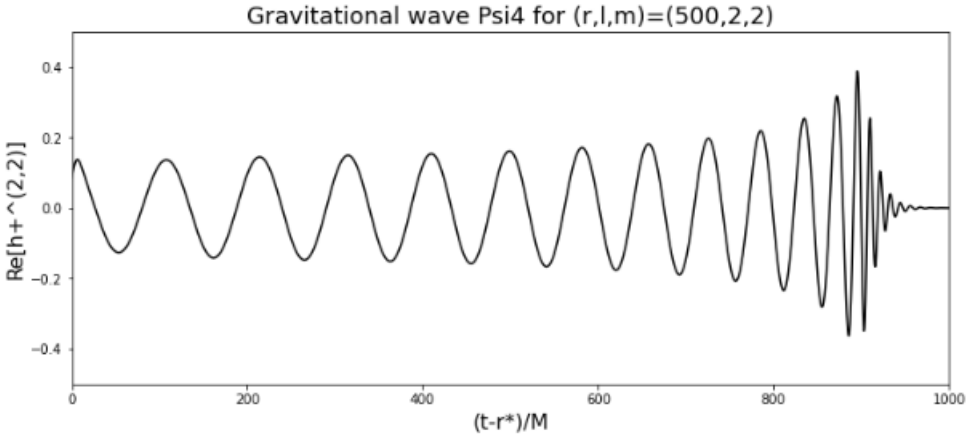

Figura 9.13: Real part of the Ψ_4 scalar associated with the dominant spherical harmonics node $(l, m) = (2, 2)$, simulated using parameters obtained from the GW150914 LIGO detection.

Capítulo 10

Conclusion

The research done in this project aimed to work our way up from basic numerical schemes to solve partial differential equations, to the formulation of the equations describing binary black hole collisions, and consequently investigating the numerical schemes behind the numerical approximations done to solve these systems of equations. With all this knowledge, we managed to solve for the evolution of the black hole binary system, and plot both their trajectories and the Ψ_4 scalar describing the gravitational waves generated.

To solve the complex systems of equations that govern the movement of two black hole spiraling towards one another, we made use of an open-source programming tool called the Einstein Toolkit. The methodology for the use of this program was deeply studied to understand all the way from basic examples to the most important one, which was the simulation of the famous GW150914 gravitational wave signal.

The physics behind this project was all General Relativity, describing both the metric and curvature of spacetime due to the presence of a black hole, and the way that we receive the signal due to gravitational waves emitted by a black hole binary. To solve these equations, we also explored Numerical relativity; numerical schemes necessary to study curved spacetime where the flow of time is not unique to all points in our domain space. These schemes are used by the Einstein Toolkit internally to solve all equations derived in this project.

In the need of more computational resources to solve our equations, we were provided with an account on the National Laboratory for High Performance Computing (NLHPC) in Chile, and submitted all simulations to be solved in this cluster. All results were later plotted and studied on a Jupyter Notebook using Python.

In conclusion, we were able to study the way different parameters affect the spiraling and merging of a black hole binary, as well as the way gravitational waves are emitted through it and later detected at distances really far away, where all waves are merely seen as perturbations of a flat metric. This opens field for new studies, such as higher order wave perturbations, spinning and/or charged black hole systems, quasinormal modes of the ringdown of our black hole merging due to the damping by the emission of gravitational radiation, and further different ways to detect the huge number of astrophysical phenomena that may be happening in our Universe.

Capítulo 11

Codes

We present this section so anyone may make use of the scripts generated for our simulations. These codes are all to be used with Python, and libraries added when necessary.

0. A code that creates matrices to solve for our first-order-in-time wave equation, with a value for the wave speed (which can be set to 1 to get a symmetrical problem). It later solves for the wave equation variables $v = u_t$ and $w = u_x$. To obtain u one may need to integrate numerically, although for most energy problems this may not be necessary.

```
def pi(i,N,xmin,xmax):
    #function that partitions xmin and xmax in N+1 parts
    return xmin+i*((xmax-xmin)/N)
```

```
def A_wave(N,theta,varphi,c_speed):
    # Create A 7-diagonal sparse matrix with size 2N

    #we create 4 submatrices and stack them using scipy.sparse.v(h)stack
    #we will create the four matrices separately and stack them accordingly
    #we wil name them by quadrant

    ones = np.ones(N)
    zeros = np.zeros(N)
    offset = [-1,0,1]
    zeros_m = np.zeros(N-1)

    ###### quadrant I
    f_I_plus = np.ones(N-1)*-theta*(c_speed**2)*varphi
    f_I_minus = np.ones(N-1)*theta*(c_speed**2)*varphi

    k1 =  np.array([f_I_minus,zeros,f_I_plus], dtype= object)
    A1 = sp.sparse.diags(k1,offset).toarray()
          #boundary derivatives
    A1[0][0] = 0
    A1[0][1] = 0
    A1[0][2] = 0
    A1[-1][-1] = 0
    A1[-1][-2] = 0
```

```
A1[-1][-3] = 0


#quadrant II

k2 =  np.array([zeros_m,ones,zeros_m], dtype= object)
A2 = sp.sparse.diags(k2,offset).toarray()

A2[0][0] = 1+ 3*theta*varphi
A2[0][1] = -4*theta*varphi
A2[0][2] = theta*varphi
A2[-1][-1] = 1+3*theta*varphi
A2[-1][-2] = -4*theta*varphi
A2[-1][-3] = theta*varphi


#quadrant III
f_III_plus = np.ones(N-1)*-theta*varphi
f_III_minus = np.ones(N-1)*theta*varphi

k3 =  np.array([f_III_minus,zeros,f_III_plus], dtype= object)
A3 = sp.sparse.diags(k3,offset).toarray()

A3[0][0] = 0
A3[0][1] = 0
A3[0][2] = 0
A3[-1][-1] = 0
A3[-1][-2] = 0
A3[-1][-3] = 0


#quadrant IV



k4 =  np.array([zeros_m,ones,zeros_m], dtype= object)
A4 = sp.sparse.diags(k4,offset).toarray()

A4[0][0] = 1+ 3*theta*varphi
A4[0][1] = -4*theta*varphi
A4[0][2] = theta*varphi
A4[-1][-1] = 1+ 3*theta*varphi
A4[-1][-2] = -4*theta*varphi
A4[-1][-3] = theta*varphi



#stack matrices by quadrant
horiz1 = np.hstack((A2,A1))
horiz2 = np.hstack((A3,A4))
vert = np.vstack((horiz1,horiz2))

```

```
    A = csr_matrix(vert)
    return A


def B_wave(N,theta,varphi,c_speed):
    # Create A 7-diagonal sparse matrix with size 2N

    #we create 4 submatrices and stack them using scipy.sparse.v(h)stack
    #we will create the four matrices separately and stack them accordingly
    #we wil name them by quadrant

    ones = np.ones(N)
    zeros = np.zeros(N)
    offset = [-1,0,1]
    zeros_m = np.zeros(N-1)

    ###### quadrant I
    f_I_plus = np.ones(N-1)*(1-theta)*(c_speed**2)*varphi
    f_I_minus = np.ones(N-1)*-(1-theta)*(c_speed**2)*varphi

    k1 =  np.array([f_I_minus,zeros,f_I_plus], dtype= object)
    B1 = sp.sparse.diags(k1,offset).toarray()
    B1[0][0] = 0
    B1[0][1] = 0
    B1[0][2] = 0
    B1[-1][-1] = 0
    B1[-1][-2] = 0
    B1[-1][-3] = 0


    ###### quadrant II
    k2 =  np.array([zeros_m,ones,zeros_m], dtype= object)
    B2 = sp.sparse.diags(k2,offset).toarray()

    B2[0][0] = 1- 3*(1-theta)*varphi
    B2[0][1] = 4*(1-theta)*varphi
    B2[0][2] = -(1-theta)*varphi
    B2[-1][-1] = 1- 3*(1-theta)*varphi
    B2[-1][-2] = 4*(1-theta)*varphi
    B2[-1][-3] = -(1-theta)*varphi


    ###### quadrant III
    f_III_plus = np.ones(N-1)*(1-theta)*varphi
    f_III_minus = np.ones(N-1)*-(1-theta)*varphi

    k3 =  np.array([f_III_minus,zeros,f_III_plus], dtype= object)
    B3 = sp.sparse.diags(k3,offset).toarray()

    B3[0][0] = 0
    B3[0][1] = 0
    B3[0][2] = 0
```

```
    B3[-1][-1] = 0
    B3[-1][-2] = 0
    B3[-1][-3] = 0

    ###### quadrant IV
    k4 =  np.array([zeros_m,ones,zeros_m], dtype= object)
    B4 = sp.sparse.diags(k4,offset).toarray()

    B4[0][0] = 1- 3*(1-theta)*varphi
    B4[0][1] = 4*(1-theta)*varphi
    B4[0][2] = -(1-theta)*varphi
    B4[-1][-1] = 1- 3*(1-theta)*varphi
    B4[-1][-2] = 4*(1-theta)*varphi
    B4[-1][-3] = -(1-theta)*varphi

    #stack matrices by quadrant
    horiz1 = np.hstack((B2,B1))
    horiz2 = np.hstack((B3,B4))
    vert = np.vstack((horiz1,horiz2))

    B = csr_matrix(vert)
    return B

amp= 1
xc = 0.5
xwid = 0.1
idsignum = 1


def solve_wave(N,theta,c_speed,varphi,xmin,xmax,tmin,tmax,V_0,W_0):
    #solve an AZ^{n+1}=ZU^{n} system, with initial condition
    #Z[0] = [X[0],Y[0]], CFL number s, with a theta-scheme finite
    #difference
    #and diffusion coefficient a

    #spatial step size
    dx = (xmax-xmin)/N

    #temporal step size
    dt = 2*varphi*dx

    #number of spatial and temporal points in our grid
    totalx = int(N+1)
    totalt = int(np.floor((tmax-tmin)/dt) + 1)

    #division of our spatial and temporal grids
    x=np.linspace(xmin, xmax, totalx)
    t=np.linspace(tmin, tmax, totalt)

    #initialize final approximation
    z=np.zeros((totalt,2*totalx))

```

```
    #impose initial condition
    #current x is pi(j,N,xmin,xmax)
    for j in range(totalx):
        z[0][j] = V_0(pi(j,N,xmin,xmax))
        z[0][j+totalx] = W_0(pi(j,N,xmin,xmax))

    #compute matrices
    A_m = A_wave(N+1,theta,varphi,c_speed)
    B_m = B_wave(N+1,theta,varphi,c_speed)
    step = inv(A_m)*B_m.toarray()
    step[np.abs(step)<0.00001] = 0
    #compute next time step from previous one
    for i in range(totalt-1):
        Unext = step.dot(z[i].transpose())
        Unext = np.array(Unext)
        for j in range(2*totalx):
            z[i+1][j]= Unext[j]


    return x,t,z
```

1. Create matrices to solve our Φ and Π wave equations

```
import numpy as np
from scipy.sparse import diags, kron, csr_matrix, csc_matrix
from scipy.sparse.linalg import spsolve, norm, inv, eigsh
import matplotlib.pyplot as plt
from matplotlib import rc
rc('text', usetex=False) # para usar latex en matplotlib
import scipy as sp
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd

def pi(i,N,p,q):
    #function that partitions p and q in N+1 parts
    return p+i*((q-p)/N)

def A_project(N,M,theta,varphi,p,q):
    # Create A 7-diagonal sparse matrix with size 2N

    #we create 4 submatrices and stack them using scipy.sparse.v(h)stack
    #we will name them by quadrant, and stack them acoordingly

    ones = np.ones(N)
    zeros = np.zeros(N)
    offset = [-1,0,1]
    c = -(q+2*M)/(q-2*M)

    ###### quadrant I
    f_I_plus = np.ones(N-1)
    f_I_minus = np.ones(N-1)
    for k in range(N-1):
```

```
    f_I_plus[k] = -theta*varphi*(pi(k+1,N,p,q)/(pi(k+1,N,p,q)+2*M))
    f_I_minus[k] = theta*varphi*(pi(k-1,N,p,q)/(pi(k-1,N,p,q)+2*M))

k1 =  np.array([f_I_minus,zeros,f_I_plus], dtype= object)
A1 = sp.sparse.diags(k1,offset).toarray()
      #boundary derivatives
A1[0][0] = theta*varphi*3*pi(0,N,p,q)/(pi(0,N,p,q)+2*M)
A1[0][1] = -theta*varphi*4*pi(1,N,p,q)/(pi(1,N,p,q)+2*M)
A1[0][2] = theta*varphi*pi(2,N,p,q)/(pi(2,N,p,q)+2*M)
A1[-1][-1] = 0
A1[-1][-2] = 0
A1[-1][-3] = 0



#quadrant II
f_II_plus = np.ones(N-1)
f_II_minus = np.ones(N-1)

for k in range(N-1):
    f_II_plus[k] = -theta*varphi*(2*M/pi(k+1,N,p,q))
    f_II_minus[k] = theta*varphi*(2*M/pi(k-1,N,p,q))


k2 =  np.array([f_II_minus,ones,f_II_plus], dtype= object)
A2 = sp.sparse.diags(k2,offset).toarray()
A2[0][0] = 1+(theta*varphi*6*M)/(pi(0,N,p,q))
A2[0][1] = -(theta*varphi*8*M)/pi(1,N,p,q)
A2[0][2] = (theta*varphi*2*M)/pi(2,N,p,q)
A2[-1][-1] = 1-3*c*theta*varphi
A2[-1][-2] = 4*c*theta*varphi
A2[-1][-3] = -c*theta*varphi

#quadrant III
f_III_plus = np.ones(N-1)
f_III_minus = np.ones(N-1)

for k in range(N-1):
    f_III_plus[k] = -(theta*varphi/pi(k,N,p,q)**2)*((pi(k+1,N,p,q)**3)/
                                        (pi(k+1,N,p,q)+2*M))
    f_III_minus[k] = (theta*varphi/pi(k,N,p,q)**2)*((pi(k-1,N,p,q)**3)/
                                        (pi(k-1,N,p,q)+2*M))

k3 =  np.array([f_III_minus,zeros,f_III_plus], dtype= object)
A3 = sp.sparse.diags(k3,offset).toarray()
A3[0][0] = (theta*varphi*3*pi(0,N,p,q)**3)/(pi(0,N,p,q)+2*M)
A3[0][1] = -(theta*varphi*4*pi(1,N,p,q)**3)/(pi(1,N,p,q)+2*M)
A3[0][2] = (theta*varphi*pi(2,N,p,q)**3)/(pi(2,N,p,q)+2*M)
A3[-1][-1] = 0
A3[-1][-2] = 0
A3[-1][-3] = 0

```

```
    #quadrant IV
    f_IV_plus = np.ones(N-1)
    f_IV_minus = np.ones(N-1)

    for k in range(N-1):
        f_IV_plus[k] = (-theta*varphi/pi(k,N,p,q)**2)*((2*M*pi(k+1,N,p,q)**2)/
                                                (pi(k+1,N,p,q)+2*M))
        f_IV_minus[k] = (theta*varphi/pi(k,N,p,q)**2)*((2*M*pi(k-1,N,p,q)**2)/
                                                (pi(k-1,N,p,q)+2*M))


    k4 =  np.array([f_IV_minus,ones,f_IV_plus], dtype= object)
    A4 = sp.sparse.diags(k4,offset).toarray()
    A4[0][0] = 1+(theta*varphi*6*M*pi(0,N,p,q)**2)/(pi(0,N,p,q)+2*M)
    A4[0][1] = -(theta*varphi*8*M*pi(1,N,p,q)**2)/(pi(1,N,p,q)+2*M)
    A4[0][2] =(theta*varphi*2*M*pi(2,N,p,q)**2)/(pi(2,N,p,q)+2*M)
    A4[-1][-1] = 1-3*(1/c)*theta*varphi
    A4[-1][-2] = 4*(1/c)*theta*varphi
    A4[-1][-3] = -(1/c)*theta*varphi



    #stack matrices by quadrant
    horiz1 = np.hstack((A2,A1))
    horiz2 = np.hstack((A3,A4))
    vert = np.vstack((horiz1,horiz2))

    A = csr_matrix(vert)
    return A




def B_project(N,M,theta,varphi,p,q):
    # Create A 7-diagonal sparse matrix with size 2N

    #we create 4 submatrices and stack them using scipy.sparse.v(h)stack
    #we will name them by quadrant, and stack them acoordingly

    ones = np.ones(N)
    zeros = np.zeros(N)
    offset = [-1,0,1]
    c = -(q+2*M)/(q-2*M)

    ####### quadrant I
    f_I_plus = np.ones(N-1)
    f_I_minus = np.ones(N-1)
    for k in range(N-1):
        f_I_plus[k] = (1-theta)*varphi*(pi(k+1,N,p,q)/(pi(k+1,N,p,q)+2*M))
        f_I_minus[k] = (-(1-theta)*varphi*(pi(k-1,N,p,q)/(pi(k-1,N,p,q)+2*M)))

```

```
k1 =  np.array([f_I_minus,zeros,f_I_plus], dtype= object)
B1 = sp.sparse.diags(k1,offset).toarray()
      #boundary derivatives
B1[0][0] = -(1-theta)*varphi*3*pi(0,N,p,q)/(pi(0,N,p,q)+2*M)
B1[0][1] = +(1-theta)*varphi*4*pi(1,N,p,q)/(pi(1,N,p,q)+2*M)
B1[0][2] = -(1-theta)*varphi**pi(2,N,p,q)/(pi(2,N,p,q)+2*M)
B1[-1][-1] = 0
B1[-1][-2] = 0
B1[-1][-3] = 0



#quadrant II
f_II_plus = np.ones(N-1)
f_II_minus = np.ones(N-1)
for k in range(N-1):
   f_II_plus[k] = ((1-theta)*varphi*(2*M/pi(k+1,N,p,q)))
   f_II_minus[k] = (-(1-theta)*varphi*(2*M/pi(k-1,N,p,q)))

k2 =  np.array([f_II_minus,ones,f_II_plus], dtype= object)
B2 = sp.sparse.diags(k2,offset).toarray()
B2[0][0] = 1-((1-theta)*varphi*6*M)/(pi(0,N,p,q))
B2[0][1] = +((1-theta)*varphi*8*M)/pi(1,N,p,q)
B2[0][2] = -((1-theta)*varphi*2*M)/pi(2,N,p,q)
B2[-1][-1] = 1+3*c*(1-theta)*varphi
B2[-1][-2] = -4*c*(1-theta)*varphi
B2[-1][-3] = c*(1-theta)*varphi

#quadrant III
f_III_plus = np.ones(N-1)
f_III_minus = np.ones(N-1)
for k in range(N-1):
   f_III_plus[k] = (+(1-theta)*varphi/pi(k,N,p,q)**2)*((pi(k+1,N,p,q)**3)/
                                          (pi(k+1,N,p,q)+2*M))
   f_III_minus[k] = (-(1-theta)*varphi/pi(k,N,p,q)**2)*((pi(k-1,N,p,q)**3)/
                                          (pi(k-1,N,p,q)+2*M))


k3 =  np.array([f_III_minus,zeros,f_III_plus], dtype= object)
B3 = sp.sparse.diags(k3,offset).toarray()
B3[0][0] = -((1-theta)*varphi*3*pi(0,N,p,q)**3)/(pi(0,N,p,q)+2*M)
B3[0][1] = +((1-theta)*varphi*4*pi(1,N,p,q)**3)/(pi(1,N,p,q)+2*M)
B3[0][2] = -((1-theta)*varphi*pi(2,N,p,q)**3)/(pi(2,N,p,q)+2*M)
B3[-1][-1] = 0
B3[-1][-2] = 0
B3[-1][-3] = 0

#quadrant IV
f_IV_plus = np.ones(N-1)
f_IV_minus = np.ones(N-1)


```

```
    for k in range(N-1):
        f_IV_plus[k] = (+(1-theta)*varphi/pi(k,N,p,q)**2)*((2*M*pi(k+1,N,p,q)**2)/
                                                (pi(k+1,N,p,q)+2*M))
        f_IV_minus[k] = (-(1-theta)*varphi/pi(k,N,p,q)**2)*((2*M*pi(k-1,N,p,q)**2)/
                                                (pi(k-1,N,p,q)+2*M))

    k4 =  np.array([f_IV_minus,ones,f_IV_plus], dtype= object)
    B4 = sp.sparse.diags(k4,offset).toarray()
    B4[0][0] = 1-((1-theta)*varphi*6*M*pi(0,N,p,q)**2)/(pi(0,N,p,q)+2*M)
    B4[0][1] = +((1-theta)*varphi*8*M*pi(1,N,p,q)**2)/(pi(1,N,p,q)+2*M)
    B4[0][2] =-((1-theta)*varphi*2*M*pi(2,N,p,q)**2)/(pi(2,N,p,q)+2*M)
    B4[-1][-1] = 1+3*(1/c)*(1-theta)*varphi
    B4[-1][-2] = -4*(1/c)*(1-theta)*varphi
    B4[-1][-3] = (1/c)*(1-theta)*varphi

    #stack matrices by quadrant
    horiz1 = np.hstack((B2,B1))
    horiz2 = np.hstack((B3,B4))
    vert = np.vstack((horiz1,horiz2))

    B = csr_matrix(vert)


    return B
```

2. Function that takes relevant parameters, takes our initial condition of:

$$\phi_0(r; A, r_0, \Delta) = A \exp\left\{(-(\frac{r - r_0}{\Delta})^2)\right\}$$
$$\phi_0'(r; A, r_0, \Delta) = -2A\frac{r - r_0}{\Delta^2} A \exp\left\{(-(\frac{r - r_0}{\Delta})^2)\right\},$$

and solves the equation for a θ-scheme, with CFL condition $\frac{\Delta t}{2\Delta x} = \varphi$, in a domain $[p, q]$ discretized in N steps, in a Schwarzchild metric where $r_S = 2M$, from time $tmin$ to time $tmax$.

```
#adjustable parameters
A = 1
r_0 = 50
Delta = 1

#phi_0's
phi0 = lambda r: A*np.exp( -( (r-r_0)/Delta )**2 )
drphi0 = lambda r: -2*(r-r_0)/(Delta**2) * A*np.exp( -( (r-r_0)/Delta )**2 )

#initial conditions
X_0 = lambda r: drphi0(r)
Y_0 = lambda r: ((r+2*M)/r**2) * phi0(r) +  drphi0(r)


def solve_phi(N,M,theta,varphi,p,q,tmin,tmax):
    #solve an AZ^{n+1}=ZU^{n} system, with initial condition
    #Z[0] = [X[0],Y[0]], CFL number s, with a theta-scheme finite
```

```
    #difference
    #and diffusion coefficient a

    #spatial step size
    dr = (q-p)/N

    #temporal step size
    dt = 2*varphi*dr

    #number of spatial and temporal points in our grid
    totalr = int(N+1)
    totalt = int(np.floor((tmax-tmin)/dt) + 1)

    #division of our spatial and temporal grids
    r=np.linspace(p, q, totalr)
    t=np.linspace(0, 1, totalt)

    #initialize final approximation
    z=np.zeros((totalt,2*totalr))

    #impose initial condition
    #current r is pi(N,i,p,q)
    for j in range(totalr):
        z[0][j] = X_0(pi(j,N,p,q))
        z[0][j+totalr] = Y_0(pi(j,N,p,q))

    #compute matrices
    A_m = A_project(N+1,M,theta,varphi,p,q)
    B_m = B_project(N+1,M,theta,varphi,p,q)
    step = inv(A_m)*B_m.toarray()
    #compute next time step from previous one
    for i in range(totalt-1):
        Unext = step.dot(z[i].transpose())
        Unext = np.array(Unext)
        for j in range(2*totalr):
            z[i+1][j]= Unext[j]


    return r,t,z
```

3. The following cell solves the wave equation for appropiate parameters, and later animates both Φ and Π wave functions for several time steps. The command *frames* specifies how many time iteration our animations must have (in this case, for $tmax = 70$, we had 10500 time iterations).

```

N=300
M=1
theta=0.5
varphi=0.01
p=2*M
q=100
tmin= 0
```

```
tmax= 70

z = solve_phi(N,M,theta,varphi,p,q,tmin,tmax)

#remember z[0] is r, z[1] is t, and z[2] is the matrix solution
#z[2][0] is the solution X,Y, so X is z[2][0][0:N+1]


# Function animation
%matplotlib notebook
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
fig, ax = plt.subplots()

line, = ax.plot([])
ax.set_xlim([0,100])
ax.set_ylim([-1.5,1.5])
ax.set_xlabel('r')

wave = 'Phi'  #change between Phi and Pi
ax.set_title('Animation for '+str(wave))


def animate(i):

    if wave=='Phi':
        y = z[2][i][0:N+1] # Animate Phi
    elif wave=='Pi':
        y = z[2][i][N+1:] # Animate Pi

    line.set_data((z[0],y))
    return line

anim = FuncAnimation(fig, animate, frames=10500, interval=1)
plt.show()


def solve_mass(N,M,theta,varphi,p,q,tmin,tmax,Delta):
    A = 1
    r_0 = 50

    #phi_0's
    phi0 = lambda r: A*np.exp( -( (r-r_0)/Delta )**2 )
    drphi0 = lambda r: -2*(r-r_0)/(Delta**2) * A*np.exp( -( (r-r_0)/Delta )**2 )

    #initial conditions
    X_0 = lambda r: phi0(r)
    Y_0 = lambda r: ((r+2*M)/r**2) * phi0(r) +  drphi0(r)

    z = solve_phi(N,M,theta,varphi,p,q,tmin,tmax)

    r = z[0]
```

```
    t = z[1]

    Phi = z[2][0:,0:501]
    Pi = z[2][0:,501:]

    totalr = len(r)
    totalt = len(t)

    N = totalr-1
    dr = (q-p)/N

    m = np.zeros((totalt,totalr))

    for i in range(totalt):
        m[i,0] = 0


    for j in range(totalr-1):
        actual_r = pi(j,N,p,q)
        alpha_over_2a = (actual_r/(2*actual_r + 4*M))
        beta = (2*M)/(actual_r + 2*M)
        for i in range(totalt):
            m[i,j+1] = m[i,j]+ 4*np.pi*(actual_r**2)*(alpha_over_2a*(Phi[i,j]**2 + Pi[i,j]**2)
     ↪ +beta*Phi[i,j]*Pi[i,j])*dr

    return m
```

4. We add a code to solve the mass function, given all previous parameters. This gives us a 2D array, to be studied however one find useful.

```
    def solve_mass(N,M,theta,varphi,p,q,tmin,tmax,Delta):
    A = 1
    r_0 = 50

    #phi_0's
    phi0 = lambda r: A*np.exp( -( (r-r_0)/Delta )**2 )
    drphi0 = lambda r: -2*(r-r_0)/(Delta**2) * A*np.exp( -( (r-r_0)/Delta )**2 )

    #initial conditions
    X_0 = lambda r: drphi0(r)
    Y_0 = lambda r: ((r+2*M)/r**2) * phi0(r) +  drphi0(r)

    z = solve_phi(N,M,theta,varphi,p,q,tmin,tmax)

    r = z[0]
    t = z[1]

    Phi = z[2][0:,0:N+1]
    Pi = z[2][0:,N+1:]

    totalr = len(r)
    totalt = len(t)
```

```
23
24      N = totalr-1
25      dr = (q-p)/N
26
27      m = np.zeros((totalt,totalr))
28
29      for i in range(totalt):
30          m[i,0] = 0
31
32
33      for j in range(totalr-1):
34          actual_r = pi(j,N,p,q)
35          alpha_over_2a = (actual_r/(2*actual_r + 4*M))
36          beta = (2*M)/(actual_r + 2*M)
37          for i in range(totalt):
38              m[i,j+1] = m[i,j]+ 4*np.pi*(actual_r**2)*(alpha_over_2a*(Phi[i,j]**2 + Pi[i,j]**2)
    ↪ +beta*Phi[i,j]*Pi[i,j])*dr
39
40      return m
```

5. This cell takes an i_grid, which is an array of time iteration, and plots the mass function for each time iteration. One may change the value of these time iterations as necessary.

```
1  i_grid = np.array([0,1500,3000,4500,6000,7500,8000,9000,10500])
2
3  dr = (q-p)/N
4  dt = 2*varphi*dr
5  current_t = lambda x: tmin + x*dt
6
7  fig = plt.figure(figsize=(9, 5))
8
9  for i in i_grid:
10     plt.plot(pi(j_grid,N,p,q),m[i,j_grid],label='t='+str(round(current_t(i),2)))
11
12 plt.legend()
13 plt.xlabel('Distance from singularity')
14 plt.ylabel('Total mass recovered')
15 plt.title('m(r,t) for different time values')
```

Bibliografía

[1] Carrol, Sean M. *Spacetime and geometry: An introduction to general relativity*, Harlow: Pearson, 2013.

[2] Bishop, Nigel T.; Rezzolla, Luciano. *Extraction of Gravitational Waves in Numerical Relativity*, arXiv:1606.02532, 2016.

[3] Choquet-Bruhat, Yvonne. *General Relativity and the Einstein Equations*, Oxford: Oxford University Press, 2009.

[4] Guzmán, F. S. *Solución de la ecuación de onda como un problema de valores iniciales usando diferencias finitas*, Michoacán: Revista mexicana de física, 2010.

[5] Ansorg, Marcus and Brügmann, Bernd and Tichy, Wolfgang. *A single-domain spectral method for black hole puncture data*, Phys. Rev. D, 2004.

[6] Baiotti, Luca; Hawke, Ian; Montero, Pedro J.; Löffler, Frank; Rezzolla, Luciano; Stergioulas, Nikolaos; Font, Jose A.; Seidel, Ed. *Three-dimensional relativistic simulations of rotating neutron star collapse to a Kerr black hole*, Phys, Rev. D, 2005.

[7] Choustikov, Nicholas. *The Einstein Toolkit: A student's guide*, arXiv:2011.13314, 2020.

[8] Gourgoulhon, Éric. *3+1 Formalism and Bases of Numerical Relativity*, arXiv:0703035v1, Laboratoire Univers et Théories, 2007.