Universidad Nacional Autónoma de México

Facultad de Ciencias

Resolvedor elíptico para relatividad numérica y su aplicación en ondas de Brill

T E S I S

que para obtener el grado de:
Físico

presenta:
Santiago Ontañón Sánchez

director de tesis:
Dr. Miguel Alcubierre Moya

Ciudad Universitaria, CDMX Abril, 2018

Resolvedor elíptico para relatividad numérica y su aplicación en ondas de Brill

por

Santiago Ontañón Sánchez

Tesis presentada para obtener el grado de

Físico

en la

Facultad de Ciencias

Universidad Nacional Autónoma de México

Ciudad Universitaria, CDMX. Abril, 2018

Para S., S., M. y G.
. . . y para Mariana.

Agradecimientos

Agradezco de la manera más sincera a todas las personas que me han ayudado, motivado y tenido paciencia. No sólo en el desarrollo de este trabajo y de esta carrera, sino en el transcurso de mi vida. Este trabajo es especialmente para ustedes.

No es en absoluto sorprendente que el mayor de mis agradecimientos sea a mis padres, Silvia y Santiago, a quienes les debo una vida entera. Les agradezco enormemente la confianza y fe que han puesto en su hijo. Pero sobre todo les agradezco el lujo que me han dado en la vida para poder elegir este y cualquier camino. También agradezco especialmente a mis hermanos, 'Marix' y Gonzalo. Les deseo el mayor de los éxitos ahora que cada uno de ustedes está definiendo su camino por la vida. Espero que aprendan de mí en los errores y aciertos.

Doy gracias a mi familia y amigos cuyo apoyo y presencia no puede ser descrita. En particular, quiero agradecer a mis abuelos Silvia y José Luis, a mi tía madrina Margarita y a mis amigos Alberto, Andrés, Antonio, Bastián, César, Emiliano, Esteban, Nicolás, Rodrigo, Sebastián y Víctor. Pienso que en la vida se requiere de motivación para ser la mejor versión de uno mismo. Mariana, tú eres esa motivación para mí. Gracias, te amo.

En un ámbito más profesional (aunque no por ello menos sincero), agradezco enormemente al tutor de este trabajo, Miguel Alcubierre. Aún si no considerara su enorme apoyo, fructífera discusión y paciencia, una rápida consulta a la bibliografía de este trabajo concluye que usted es sin duda el mayor contribuidor. Gracias por abrirme a mí y a tantos estudiantes las puertas hacia la relatividad numérica.

Como últimas gracias, agradezco al jurado presente en este examen profesional, al Departamento de Física de la Facultad de Ciencias, a la misma Facultad de Ciencias y a la Universidad Nacional Autónoma de México. Al final de este viaje comprendo mejor el significado de que "Por mi raza hablará el espíritu".

Resumen

Este trabajo estudia la evolución de ondas de Brill tipo Holz en un espacio axisimétrico usando relatividad numérica en el formalismo $3+1$. Específicamente, se utiliza una formulación tipo BSSN con coordenadas curvilíneas siguiendo el trabajo de Alcubierre y Méndez presentado en [3]. Dicha implementación resulta en el código *OllinBrill* disponible a la comunidad de manera pública en [44]. El código realiza la evolución de ondas de Brill con un lapso tipo maximal para el cual es necesario resolver una ecuación elíptica lineal en cada paso de integración. Como resolvedor elíptico se usa el resolvedor directo lineal *PARDISO* (véase [46] y [47]) que se ha optimizado cuidadosamente para este problema particular ya que el grueso del costo computacional consiste en estar resolviendo sistemas con estructura matricial idéntica en cada paso de integración. Adicionalmente, siguiendo el trabajo de Alcubierre en [1], se escribe un encontrador de horizontes aparentes dentro de *OllinBrill*.

Los resultados de la evolución demuestran la convergencia esperada de cuarto orden para *OllinBrill* y permiten estudiar distintos fenómenos en las ondas de Brill como son su posible colapso a un horizonte aparente para fuertes amplitudes, o la resultante evolución al espacio plano de Minkowski si la amplitud es débil.

Abstract

The following works studies Holz type Brill wave evolution in axisymmetric space using $3+1$ numerical relativity. Specifically, this work utilizes Alcubierre and Méndez's BSSN formulation developed for curvilinear coordinates presented in [3]. The resulting evolution code is named *OllinBrill* and is publicly available for download in [44]. For this particular problem, maximal slicing has been chosen, which requires that the code solve a linear elliptic equation at every integration step. *PARDISO* (see [46] and [47]) has been chosen as a direct linear solver and has been carefully optimized for this particular problem, since the main computational cost comes from solving linear systems with identical matrix structure at every integration step. Additionally, following Alubierre's work in [1], an apparent horizon finder has also been written inside of *OllinBrill*.

The evolution results prove that *OllinBrill* converges to the expected fourth order and thus allows the study of several phenomena in Brill waves. Examples are the possible collapse into an apparent horizon for waves of strong amplitude, or the evolution into Minkowski flat space if the amplitude is weak.

Índice general

Prefacio **1**

1. Evolución numérica en el formalismo 3 + 1 **3**
1.1. Las ecuaciones ADM . . . 5
1.2. Formulación BSSN . . . 5
1.3. Formulación BSSN en coordenadas curvilíneas . . . 7
1.4. Condiciones de norma . . . 9
1.4.1. Lapso maximal y $1 + \log$ . . . 10
1.4.2. Vector de corrimiento hiperbólico . . . 11
1.5. Aplicación a coordenadas cilíndricas . . . 12
1.5.1. Restricciones sobre la métrica . . . 12
1.5.2. Regularización . . . 12

2. Datos iniciales de ondas de Brill **17**
2.1. Descomposición conforme . . . 17
2.2. Ecuación elíptica . . . 18
2.2.1. Condiciones de frontera . . . 19
2.3. Introducción compatible al formalismo BSSN . . . 19
2.4. Resultados previos . . . 20

3. Resolvedor elíptico **22**
3.1. Discretización en diferencias finitas . . . 22
3.1.1. Malla computacional . . . 22
3.1.2. Ghost zones . . . 23
3.1.3. Sistema lineal $Au = f$ . . . 24
3.2. Análisis del error para una frontera de Robin . . . 28
3.3. Matrices Sparse . . . 31
3.3.1. Formato CSR . . . 33
3.3.2. Cálculo de elementos distintos de cero . . . 34
3.3.3. Algoritmo de llenado . . . 35
3.4. Resolvedor directo LU . . . 39
3.5. El resolvedor directo PARDISO . . . 41
3.5.1. Estructuras y parámetros de control . . . 42

4. Encontrador de horizontes aparentes **44**
4.1. Teoría en simetría axial . . . 44
4.2. Algoritmo . . . 45
4.2.1. Resultados con datos iniciales de Schwarschild . . . 48

4.2.2. Resultados con datos iniciales de Brill 50

5. El código OllinBrill **53**
5.1. Subrutinas de geometría 53
5.2. Condiciones de frontera 57
5.3. Análisis CFL 60
5.4. Archivo de parámetros 61

6. Resultados **66**
6.1. Pruebas de convergencia con Minkowski y Schwarschild 67
6.1.1. Minkowski con lapso armónico 67
6.1.2. Schwarschild con lapso $1 + \log$ 71
6.1.3. Schwarschild con lapso maximal 75
6.2. Ondas de Brill 79
6.2.1. Onda débil $a = 1$ 79
6.2.2. Onda fuerte $a = 11$ 86
6.2.3. Onda cercana a la amplitud crítica $a = 4.5$ 99

7. Conclusión **102**

A. Uso de memoria, escalamiento y perfilado **104**
A.1. Memoria 104
A.2. Escalabilidad 105
A.3. Perfilado 106

Prefacio

La presente tésis consiste en un trabajo de relatividad numérica, un área relativamente joven de la física, ya que los primeros trabajos no anteceden los años 60's (específicamente se tiene el primer trabajo de Hahn y Lindquist [31]). La relatividad numérica nace de resolver computacionalmente las ecuaciones de la relatividad general de Einstein quien presentó en [26] una serie de ecuaciones de campo conectando la curvatura del espacio a la materia y energía contenida en el mismo. En forma explícita

$$G_{\mu\nu} = 8\,\pi\,T_{\mu\nu}\,, \tag{1}$$

donde $G_{\mu\nu}$ es el tensor de Einstein y $T_{\mu\nu}$ el tensor de energía-momentum. Dentro de la sencilla expresión (1) existen diez ecuaciones diferenciales parciales, no-lineales, acopladas y en cuatro dimensiones. En general, las soluciones cerradas de dichas soluciones sólo se pueden obtener al tener un problema con alta simetría. El hecho de tener un formalismo en cuatro dimensiones donde está incluido el tiempo permite que las simetrías sean variadas: es decir, existen las típicas simetrías espaciales como son la esférica y axial pero también se debe considerar en el tiempo que la solución pueda ser estática y estacionaria. Sin embargo, la clave central es que para simular eventos astrofísicos realistas, donde existe muy poca o ninguna simetría, es simplemente imposible resolver las ecuaciones de Einstein de forma exacta.

Gracias a una comunidad activa y el increíble avance en el equipo y capacidades computacionales de hoy en día, la relatividad numérica ya existe como un área madura y productiva de la física. En particular, existe una diversidad grande de grupos en el planeta que se dedican a estudiarla. Por lo tanto (y sin duda, por fortuna), hoy existen diversos códigos computacionales. Entre ellos destaca el trabajo *Einstein Toolkit* (véase [41]), un trabajo extensivo y comunitario que resuelve las ecuaciones de Einstein usando el formalismo BSSN que será detallado en el Capítulo 1. Dicho trabajo puede simular colisiones de agujeros negros, estrellas de neutrones y muchas otras geometrías de interés actual. Por lo tanto, una pregunta válida sería, ¿por qué vale la pena desarrollar códigos adicionales cuando ya existen códigos profesionales con capacidades de simular todo tipo de geometrías y sin simetría alguna?

Para responder hay que destacar que la infraestructura de la malla del trabajo anterior es totalmente cartesiana en tres dimensiones. Esta malla es ideal para una situación donde no hay simetría alguna pero no tanto para un problema axisimétrico que se reduce a ser un problema bidimensional. Por tanto, una simulación axisimétrica en *Einstein Toolkit* no está del todo optimizada a ser un problema bidimensional[1].

Más allá del problema de optimización, hay otra diferencia central. Cuando se plantea un problema con simetría axial, las coordenadas que vienen en mente son las cilíndricas $\{\rho, z, \varphi\}$. Sin embargo, la mayoría de los códigos manejan coordenadas cartesianas ya que son ideales para la

[1] Sin embargo, véase [8] para entender cómo trabaja dicho código la simetría axial. Básicamente se rota una malla bidimensional para generar la malla tridimensional entera.

formulación BSSN que pide, por ejemplo, que el determinante de la métrica conforme sea unitario. Además, muchas cantidades en BSSN no son tensores o vectores, es decir, no son cantidades covariantes. Esto ocasiona que se origen muchos inconvenientes a la hora de usar las coordenadas cilíndricas dentro del BSSN usual.

El problema de escribir un replantamiento del BSSN en coordenadas curvilíneas es totalmente equivalente al de reescribirlo en cantidades totalmente covariantes y ha sido desarrollado cabalmente por Alcubierre y Méndez en [3]. Con dicha referencia, es posible escribir códigos de relatividad numérica en cualquier sistema de coordenadas curvilíneas, en particular el cilíndrico de interés.

Juntando la discusión anterior, se puede afirmar que vale la pena realizar un código de evolución que trabaje la simetría axial como un problema de realmente dos dimensiones con una formulación covariante. Es decir, el enfoque es proporcionar a la comunidad un código optimizado para realizar simulaciones de simetría axial y adicionalmente, se quiere demostrar que es posible realizar las simulaciones con coordenadas cilíndricas via la formulación BSSN covariante.

Sin embargo, dicho objetivo ya lo realizó Torres en [61] con su código titulado *OllinAxis*. La diferencia central con este trabajo es el objetivo de implementar un resolvedor elíptico para resolver datos iniciales de ondas de Brill así como para calcular el lapso maximal en cada paso de integración. Si bien este código fue guiado invaluablemente por *OllinAxis*, consiste en un trabajo totalmente independiente por el simple hecho de que la infraestructura requerida por el resolvedor elíptico es incompatible con la estructura de mallas trabajada en *OllinAxis*.

El trabajo resultante se ha bautizado como *OllinBrill* y existe como un código disponible a la comunidad en un repositorio público [44]. Se ha tenido especial atención en que el uso y compilación del código sea lo más sencillo posible. Además, una especial ventaja de que la simetría axial implica un problema bidimensional es que el menor costo computacional (comparado con un problema completo de tres dimensiones) permite que este código corra en máquinas de uso personal, como puedes ser las de un estudiante interesado y recién introducido al área.

En resumen, este trabajo presenta en el Capítulo 1 la formulación BSSN en coordenadas curvilíneas que son la base del código de evolución. El Capítulo 2 presenta las ondas de Brill que serán la geometría principal de estudio. Para poder generar datos iniciales de dichas ondas y evolucionar con lapso maximal, se presenta en el Capítulo 3 el resolvedor elíptico. Las simulaciones previas de las ondas de Brill, en particular el trabajo de Alcubierre *et al.* en [5] reportan que las ondas de Brill pueden colapsar a un agujero negro. Por lo tanto, en el Capítulo 4 se presenta un encontador de horizontes aparentes. El Capítulo 5 presenta el uso general del código *OllinBrill* y finalmente, el Capítulo 6 discute los resultados y convergencia del mismo.

Capítulo 1

Evolución numérica en el formalismo 3 + 1

Existen diversos formalismos para hacer simulaciones numéricas de relatividad general. Entre ellos, uno de los más robustos y usados en la comunidad es el formalismo 3 + 1. En dicho, el espaciotiempo se folía en hípersuperficies tridimensionales con caracter espacial cada una. El propósito de este trabajo no es dar una introducción a la teoría detrás de este formalismo, pues se pueden consultar en trabajos con gran completez como son [1] y [53]. Sin embargo, aún con asumir que el lector tiene familiaridad en el tema, es relevante esclarecer las convenciones utilizadas en este trabajo.

Como es usual en trabajos de relatividad general, se usa la convención de Einstein de sumar sobre índices repetidos, así como unidades geométricas donde la velocidad de la luz c y la constante de gravitación universal G son tomadas como uno. En cuanto a los símbolos para las cantidades geométricas, se presentan en el siguiente Cuadro 1.1.

$g_{\mu\nu}$	La métrica completa en el espacio tiempo. Como convención usual en relatividad general, los índices griegos denotan los cuatro índices $\{0, 1, 2, 3\}$, con el 0 el tiempo.
$T_{\mu\nu}$	El tensor de energía momento en cuatro dimensiones que se une a la métrica $g_{\mu\nu}$ via las ecuaciones de Einstein.
n^μ	El vector normal a la hipersuperfice con el cual se obtiene la proyección γ_{ij}.
P^μ_ν	El operador de proyección dado por $\delta^\mu_\nu + n^\mu n_\nu$.
γ_{ij}	La métrica espacial obtenida como proyección de la métrica completa $g_{\mu\nu}$. En este caso, la presencia de índices latinos indica que sólo abarcan las tres dimensiones espaciales $\{1, 2, 3\}$.
γ	El determinante o elemento de volumen de la métrica espacial γ_{ij}.
$\Gamma^i{}_{jk}$	Los símbolos de Christoffel asociados a la métrica espacial γ_{ij}.
∇_i	La derivada covariante con la conexión anterior $\Gamma^i{}_{jk}$.
R_{ij}	El tensor de Ricci asociado a la métrica espacial γ_{ij}.
R	El escalar de Ricci correspondiente a R_{ij}.
K_{ij}	La curvatura extrínseca de la métrica espacial γ_{ij}.
K	La traza de dicha curvatura extrínseca.
α	La función de lapso.
β^i	El vector tridimensional de corrimiento o *shift* en inglés.
$\pounds_{\mathbf{v}}$	La derivada de Lie en la dirección del vector tridimensional v^i. Las reglas usuales de la derivada de Lie aplican para cantidades escalares, vectoriales, tensoriales, etc.
$\{r, \theta, \varphi\}$	Las coordenadas esféricas correspondientes al radio, ángulo azimutal y polar, respectivamente.
$\{\rho, z, \varphi\}$	Las coordenadas cilíndricas donde $\rho = r \sin\theta$, $z = r\cos\theta$ y φ continua siendo el ángulo polar.

Tabla 1.1: Convenciones utilizadas.

Es importante hacer énfasis que en este trabajo los cálculos son siempre en la proyección espacial salvo que se haga notar lo contrario. Es decir, todas las trazas y movimientos de índices se hacen con la métrica espacial γ_{ij} y su inversa γ^{ij}. Del mismo modo, las derivadas covariantes y cantidades geométricas siempre son asociadas a la métrica espacial. En algunos trabajos se suele utilizar D_i para la derivada covariante en tres dimensiones pero como en este trabajo no se utiliza nunca la derivada covariante en cuatro dimensiones, se prefiere poner su símbolo usual ∇_i.

1.1. Las ecuaciones ADM

En función de las convenciones definidas arriba, las ecuaciones ADM (Arnowitt-Deser-Misner) introducidas en [12] y reescritas en su forma convencional para 3 + 1 por York en [63] toman la forma

$$\partial_t \gamma_{ij} - \pounds_\beta \gamma_{ij} = -2\alpha K_{ij} \tag{1.1}$$

$$\begin{aligned} \partial_t K_{ij} - \pounds_\beta K_{ij} = &- \nabla_i \nabla_j \alpha + \alpha \left(R_{ij} + K K_{ij} - 2K_{ik} K^k{}_j \right) \\ &+ 4\pi\alpha \left(\gamma_{ij} (S - \rho) - 2S_{ij} \right) , \end{aligned} \tag{1.2}$$

donde las cantidades relacionadas al tensor de energía momento son

$$\rho \equiv n^\mu n^\nu T_{\mu\nu} , \tag{1.3}$$

$$j^i \equiv - P^{i\mu} n^\nu T_{\mu\nu} , \tag{1.4}$$

$$S_{ij} \equiv P^\mu{}_i P^\nu{}_j T_{\mu\nu} . \tag{1.5}$$

Unidas a estas ecuaciones de evolución (pues involucran derivadas temporales), tenemos las constricciones que deben satisfacer las soluciones a cualquier tiempo y en toda la hipersuperfice espacial.

$$H \equiv \frac{1}{2} \left(R + K^2 - K_{ij} K^{ij} \right) - 8\pi\rho = 0 , \tag{1.6}$$

$$M^i \equiv \nabla_j \left(K^{ij} - \gamma^{ij} K \right) - 8\pi j^i = 0 . \tag{1.7}$$

Dichas constricciones se conocen como la constricción hamiltoniana y la constricción de momento, respectivamente.

1.2. Formulación BSSN

Si bien las ecuaciones ADM (junto con las constricciones) son un replantamiento entero de las ecuaciones de Einstein, no son ecuaciones *bien-planteadas* en el sentido hiperbólico de un sistema de ecuaciones diferenciales parciales. En otras palabras, esto significa que las soluciones no dependen continuamente de los datos iniciales, o que pequeñas perturbaciones en los datos iniciales pueden llevar a grandes cambios en las soluciones. Afortunadamente, el hecho de que les podemos sumar múltiplos de las constricciones sin cambiar el aspecto físico de las ecuaciones nos permite formar una infinidad de sistemas equivalentes. *Matemáticamente*, dichos sistemas pueden tener una estructura totalmente distinta.

La discusión y teoría de hiperbolicidad está fuera del enfoque de este trabajo pero se puede consultar con amplio detalle en [1] para comprender por qué ciertos múltiplos de las constricciones llevan a una formulación estable y además bien-planteada. Por ende, dentro del formalismo 3 + 1 existen diversas formulaciones que se han probado hiperbólicas y robustas tanto en un aspecto teórico como empírico. La más usada por la comunidad actualmente es la formulación BSSN de Baumgarte, Shapiro, Shibata y Nakamura presentada originalmente en [15] y [54]. La siguiente discusión sobre las formulación BSSN toma de manera cercana la forma presentada en [1] y [3].

La formulación BSSN empieza con una descomposición conforme de la métrica de la forma:

$$\hat{\gamma_{ij}} = e^{-4\phi} \gamma_{ij} . \tag{1.8}$$

De ahora en adelante, se adhiere a la convención adicional de que las cantidades denotadas con un circunflejo $\hat{}$ son cantidades conformes relacionadas por el factor conforme $e^{-4\phi}$. Por convención adicional, *los índices de las cantidades conformes se manipulan con la métrica conforme*. El nombre de factor conforme normalmente se reserva para la cantidad ϕ y justamente en la formulación BSSN se empieza por pedir que el determinante de la métrica conforme $\hat{\gamma}$ sea uno. Esto impone como restricción para ϕ la siguiente relación:

$$\phi \equiv \frac{1}{12} \log \gamma \,. \tag{1.9}$$

Esta relación no se toma como constricción, sino que ϕ se promueve a una variable dinámica y ahora se le necesita una ecuación de evolución obtenida a partir de (1.1):

$$\partial_t \phi - \pounds_\beta \phi = -\frac{1}{6} \alpha K \,, \tag{1.10}$$

donde se ha utilizado el hecho de que la derivada de Lie está dada por

$$\pounds_\beta \phi = \beta^m \partial_m \phi + \frac{1}{6} \partial_m \beta^m \,, \tag{1.11}$$

pues ϕ no es un escalar sino una densidad tensorial de peso $1/6$, como se puede comprobar en (1.9).

En adición al factor conforme, se introduce una separación entre la curvatura extrínseca y su traza:

$$A_{ij} = K_{ij} - \frac{1}{3} \gamma_{ij} K \,. \tag{1.12}$$

Siguiendo la convención ya estipulada, tendremos

$$\hat{A}_{ij} \equiv e^{-4\phi} A_{ij} = \hat{K}_{ij} - \frac{1}{3} \hat{\gamma}_{ij} K \,. \tag{1.13}$$

Después se añaden tres variables auxiliares llamadas las funciones de conexión conformes:

$$\hat{\Gamma}^i \equiv \hat{\gamma}^{jk} \hat{\Gamma}^i{}_{jk} = -\partial_j \hat{\gamma}^{ij} \,, \tag{1.14}$$

donde se ha utilizado que $\hat{\gamma} = 1$ para hacer cero un logaritmo y llegar a la segunda igualdad.

El sistema BSSN toma como variables dinámicas $\{\hat{\gamma}_{ij}, \phi, \hat{A}_{ij}, K, \hat{\Gamma}^i\}$, lo cual requiere una reescritura de las ecuaciones ADM:

$$\partial_t \hat{\gamma}_{ij} - \pounds_\beta \hat{\gamma}_{ij} = -2\alpha \hat{A}_{ij} \,, \tag{1.15}$$

$$\partial_t \phi - \pounds_\beta \phi = -\frac{1}{6} \alpha K \,, \tag{1.16}$$

$$\partial_t \hat{A}_{ij} - \pounds_\beta \hat{A}_{ij} = e^{-4\phi} \left(-\nabla_i \nabla_j \alpha + \alpha R_{ij} + 4\pi\alpha \left(\gamma_{ij} (S - \rho) - 2 S_{ij}\right)\right)^{\mathrm{TF}} \,, \tag{1.17}$$

$$\partial_t K - \pounds_\beta K = -\nabla^2 \alpha + \alpha \left(\hat{A}_{ij} \hat{A}^{ij} + \frac{1}{3} K^2\right) + 4\pi\alpha (\rho + S) \,. \tag{1.18}$$

En el conjunto anterior de ecuaciones hay que tener especial cuidado para saber qué cantidades están ligadas a cuáles. Siguiendo las convenciones, ∇_i es la derivada covariante de la métrica física γ_{ij}, junto con R_{ij} y R que también son de la métrica física. Del mismo modo, el laplaciano $\nabla^2 = \nabla^m \nabla_m$ es la traza del operador físico $\nabla_i \nabla_j$, i.e. $\nabla^2 = \gamma^{ij} \nabla_i \nabla_j$. También es importante recalcar que la constricción hamiltoniana (1.6) se ha usado para eliminar el escalar de Ricci en (1.18).

En lugar de tener que formar la métrica física y luego calcular el tensor de Ricci, es posible formar una relación entre el tensor de Ricci asociado a la métrica física y el asociado a la métrica conforme via

$$R_{ij} = \hat{R}_{ij} + R^{\phi}_{ij}\,, \tag{1.19}$$

donde $\hat{R}_{ij}$ es el tensor de Ricci de la métrica conforme calculado con una expresión usual como es:

$$\hat{R}_{ij} = -\frac{1}{2}\hat{\gamma}^{kl}\,\partial_k\,\partial_l\,\hat{\gamma}_{ij} + \hat{\gamma}_{k(i}\,\partial_{j)}\,\hat{\Gamma}^k + \hat{\Gamma}^k\,\hat{\Gamma}_{(ij)k} + 2\hat{\Gamma}^{kl}{}_{(i}\hat{\Gamma}_{j)kl} + \hat{\Gamma}_{kli}\,\hat{\Gamma}^{kl}{}_j\,. \tag{1.20}$$

El término R^{ϕ}_{ij} denota los términos adicionales que vienen del factor conforme ϕ y están dados por

$$R^{\phi}_{ij} = -2\hat{\nabla}_i\,\hat{\nabla}_j\,\phi - 2\hat{\gamma}_{ij}\,\hat{\nabla}^k\,\hat{\nabla}_k\,\phi + 4\hat{\nabla}_i\,\phi\,\hat{\nabla}_j\,\phi - 4\hat{\gamma}_{ij}\,\hat{\nabla}^k\,\phi\,\hat{\nabla}_k\,\phi\,. \tag{1.21}$$

Nótese que todas las derivadas y métricas en la expresión anterior son conformes. Debido a esto, otra expresión de particular uso es la que relaciona las conexiones:

$$\Gamma^k{}_{ij} = \hat{\Gamma}^k{}_{ij} + 2\left(\delta^k_i\,\partial_j\,\phi + \delta^j_i\,\partial_i\,\phi - \hat{\gamma}_{ij}\,\hat{\gamma}^{kl}\,\partial_l\,\phi\right)\,. \tag{1.22}$$

Es importante resaltar que las expresiones (1.19), (1.21) y (1.22) *son válidas en todas las circunstancias*. Es decir, sin importar la forma o restricciones del factor conforme ϕ.

Regresando a las ecuaciones del sistema BSSN, hay que hacer la aclaración de que las derivadas de Lie se toman para $\hat{\gamma}_{ij}$ y $\hat{A}_{ij}$ considerando que son densidades tensoriales de peso $-2/3$. K es un escalar puro y ϕ ya se discutió como densidad de peso $1/6$. Finalmente, la ecuación para $\hat{\Gamma}^i$ resulta ser mucho más detallada porque justamente es esta la que considera múltiplos de las constricciones. Específicamente se añaden múltiplos de la constricción de momento, quedando:

$$\begin{aligned}\partial_t\,\hat{\Gamma}^i - \pounds_\beta\,\Gamma^i =& \hat{\gamma}^{jk}\,\partial_j\,\partial_k\,\beta^i + \frac{1}{3}\,\hat{\gamma}^{ij}\,\partial_j\,\partial_k\,\beta^k - 2\hat{A}^{ij}\,\partial_j\,\alpha - \alpha\,(2-\xi)\,\partial_j\,\hat{A}^{ij} \\ &+ \alpha\,\xi\,\left(\hat{\Gamma}^i{}_{jk}\,\hat{A}^{jk} + 6\hat{A}^{ij}\,\partial_j\,\phi - \frac{2}{3}\,\hat{\gamma}^{ij}\,\partial_j\,K - 8\pi\hat{j}^i\right)\,.\end{aligned} \tag{1.23}$$

El parámetro que indica el múltiplo de la constricción de momento es ξ. El análisis de hiperbolicidad como se hace en [1] determina que para $\xi > 1/2$ el sistema es fuertemente hiperbólico. Por ejemplo, la formulación estándar de BSSN toma $\xi = 2$.

1.3. Formulación BSSN en coordenadas curvilíneas

La formulación BSSN detallada arriba resulta muy útil y natural para un sistema cartesiano pero al extenderlo a coordenadas curvilíneas uno encuentra que hay asunciones poco naturales. Primero existe el punto de partida de pedir que $\hat{\gamma} = 1$. Si elegimos una métrica conforme plana en coordenadas esféricas, i.e. $\hat{\gamma}_{ij} = \mathrm{diag}(1, r^2, r^2\,\sin^2\theta)$, se sigue que $\hat{\gamma} = r^4\,\sin^2\theta$. Sería mucho más conveniente pedir que inicialmente $\hat{\gamma}$ se reduzca al determinante plano. Sin embargo, existe un aspecto mucho más fundamental y es que las cantidades introducidas en el BSSN no son covariantes: el factor conforme ϕ no es un escalar, las cantidades auxiliares $\hat{\Gamma}^i$ no son vectores, etc.

Haciendo una formulación estrictamente covariante se puede llegar a un BSSN modificado y perfectamente adecuado para coordenadas curvilíneas que además se reduce al BSSN usual en

coordenadas cartesianas. Dicha extensión covariante ha sido realizada por Brown en [19] y por Alcubierre y Méndez en [3].

Resumiendo en particular el último trabajo, se parte del conocido hecho de que la conexión no es un tensor pero que la diferencia de dos conexiones cancela los términos que no transforman como tensores y así se forma un tensor adecuado. Se define entonces

$$\hat{\Delta}^i{}_{jk} \equiv \hat{\Gamma}^i{}_{jk} - \mathring{\Gamma}^i{}_{jk}\,, \tag{1.24}$$

donde $\mathring{\Gamma}^i{}_{jk}$ denota la conexión de la métrica plana y, como seguro ya era de imaginarse, el círculo $^\circ$ denotará las cantidades planas como la métrica plana $\mathring{\gamma}_{ij}$ y el determinante plano $\mathring{\gamma}$.

Después de introducir esta cantidad hay que reescribir el tensor de Ricci asociado a la métrica conforme $\hat{R}_{ij}$ en términos de la $\Delta^i{}_{jk}$. El análisis cuidadoso se puede consultar en [3] donde se llega a la expresión:

$$\hat{R}_{ij} = -\frac{1}{2}\,\hat{\gamma}^{kl}\mathring{\nabla}_k\,\mathring{\nabla}_l\,\hat{\gamma}_{ij} + \hat{\gamma}_{(ki}\mathring{\nabla}_{j)}\hat{\Delta}^k + \hat{\Delta}^k\,\hat{\Delta}_{(ij)k} + 2\hat{\Delta}^{kl}{}_{(i}\hat{\Delta}_{j)kl} + \hat{\Delta}^{kl}{}_i\,\hat{\Delta}_{klj}\,. \tag{1.25}$$

En la expresión anterior $\mathring{\nabla}$ denota la derivada covariante respecto a la métrica plana, los índices de $\hat{\Delta}^i{}_{jk}$ han sido manipulados con la métrica conforme y además se ha definido

$$\hat{\Delta}^i = \hat{\gamma}^{jk}\,\hat{\Delta}^i{}_{jk} = \hat{\Gamma}^i - \hat{\gamma}^{jk}\,\mathring{\Gamma}^i{}_{jk}\,. \tag{1.26}$$

Obsérvese que $\hat{\Delta}^i$ es un vector, a diferencia de $\hat{\Gamma}^i$. Adicionalmente, si la métrica plana es cartesiana, $\mathring{\Gamma}^i{}_{jk} = 0$ y así la expresión para el Ricci (1.25) se reduce a la conocida (1.20).

En el sistema covariante queremos ampliar la restricción del BSSN original de tener como determinante conforme igual a uno. En vez, a tiempo cero se establece que

$$\hat{\gamma}(t=0) = \mathring{\gamma}\,. \tag{1.27}$$

Sin embargo, hay que determinar cómo va a evolucionar esta cantidad. Es posible que $\partial_t\,\hat{\gamma} = 0$ pero también podemos pedir que $\partial_t\,\hat{\gamma} - \pounds_\beta\,\hat{\gamma} = 0$. Estas dos condiciones coinciden cuando no hay vector de corrimiento pero en general son distintas y siguiendo las convenciones de dinámica de fluidos, corresponden a la condición Lagrangiana y Euleriana, respectivamente. Es decir, en la condición Lagrangiana, el determinante conforme es constante en las líneas de tiempo, mientras que en la Euleriana es constante a través de las líneas normales (que son las trayectorias de los observadores Eulerianos). Ambas condiciones se puede escribir como

$$\partial_t\,\hat{\gamma} = s\,\left(2\hat{\gamma}\,\hat{\nabla}_m\,\beta^m\right)\,, \tag{1.28}$$

donde

$$s = \begin{cases} 0 & \text{Lagrangiana} \\ 1 & \text{Euleriana} \end{cases}\,. \tag{1.29}$$

Así como para BSSN usual, $\hat{\gamma} = 1$ estableció la relación (1.9) para ϕ, ahora se deduce que

$$\phi = \frac{1}{12}\,\log\frac{\gamma}{\hat{\gamma}}\,. \tag{1.30}$$

De esta manera, la ecuación de evolución del factor conforme queda como

$$\partial_t \phi - \pounds_\beta \phi = -\frac{1}{6}\alpha K + \frac{1}{6}\sigma \hat{\nabla}_m \beta^m \,, \tag{1.31}$$

donde se define la cantidad

$$\sigma \equiv 1 - s \,. \tag{1.32}$$

En esta formulación, ϕ es un escalar puro, como se puede comprobar de que es la razón entre dos elementos de volumen. Esto implica que no hay un peso en la derivada de Lie y así

$$\pounds_\beta \phi = \beta^m \partial_m \phi \,. \tag{1.33}$$

En analogía con el BSSN usual ahora las variables dinámicas son $\{\hat{\gamma}_{ij}, \phi, \hat{A}_{ij}, K, \hat{\Delta}^i\}$. Las ecuaciones son altamente similares pero ahora se trata de cantidades totalmente covariantes, lo cual implica que las derivadas de Lie ya no tienen pesos involucrados. Sin embargo, la posibilidad de evolución Lagrangiana/Euleriana debe ser considerada con cuidado via el parámetro σ. El sistema resulta ser:

$$\partial_t \hat{\gamma}_{ij} - \pounds_\beta \hat{\gamma}_{ij} = -2\alpha \hat{A}_{ij} - \frac{2}{3}\sigma \hat{\gamma}_{ij} \hat{\nabla}_m \beta^m \,, \tag{1.34}$$

$$\partial_t \phi - \pounds_\beta \phi = -\frac{1}{6}\alpha K + \frac{1}{6}\sigma \hat{\nabla}_m \beta^m \,, \tag{1.35}$$

$$\begin{aligned} \partial_t \hat{A}_{ij} - \pounds_\beta \hat{A}_{ij} = {} & e^{-4\phi}\left(-\nabla_i \nabla_j \alpha + \alpha R_{ij} + 4\pi\alpha\left(\gamma_{ij}(S-\rho) - 2S_{ij}\right)\right)^{\mathrm{TF}} \\ & -\frac{2}{3}\sigma \hat{A}_{ij} \hat{\nabla}_m \beta^m \,, \end{aligned} \tag{1.36}$$

$$\partial_t K - \pounds_\beta K = -\nabla^2 \alpha + \alpha\left(\hat{A}_{ij}\hat{A}^{ij} + \frac{1}{3}K^2\right) + 4\pi\alpha(\rho + S) \,, \tag{1.37}$$

$$\begin{aligned} \partial_t \hat{\Delta}^i - \pounds_\beta \hat{\Delta}^i = {} & \hat{\gamma}^{jk} \mathring{\nabla}_j \mathring{\nabla}_k \beta^i - 2\hat{A}^{ij}\partial_j \alpha - \alpha(2-\xi)\hat{\nabla}_j \hat{A}^{ij} + 2\alpha \hat{A}^{jk} \hat{\Delta}^i{}_{jk} \\ & +\alpha\xi\left(6\hat{A}^{ij}\partial_j \phi - \frac{2}{3}\hat{\gamma}^{ij}\partial_j K - 8\pi \hat{j}^i\right) \\ & +\frac{\sigma}{3}\left(\hat{\nabla}^i\left(\hat{\nabla}_m \beta^m\right) + 2\hat{\Delta}^i \hat{\nabla}_m \beta^m\right) \,. \end{aligned} \tag{1.38}$$

Hay que recordar que el tensor y escalar de Ricci ahora se calculan con las expresiones (1.20), (1.21) y (1.25).

En la ecuación (1.38) aparecen nuevamente los múltiplos de la constricción de momento. Para esta formulación se siguen los mismos resultados de fuerte hiperbolicidad con $\xi > 1/2$.

Para recuperar el sistema BSSN usual, basta poner $\sigma = 1$ en coordenadas cartesianas. Esto parece por un momento incorrecto pero hay que recordar que en el BSSN usual las derivadas de Lie llevaban pesos y el término proporcional a la σ es el que los proveé.

1.4. Condiciones de norma

En las secciones anteriores no se ha hablado en absoluto de las cantidades α, β^i. Estas son las variables de norma y están asociadas a la libertad de norma en las ecuaciones de Einstein de poder escoger un sistema coordenado. Dicha libertad puede ser un problema en que no hay una opción obvia de qué tipo de norma utilizar. La experiencia empírica y algunos resultados geométricos son los que han guiado a la comunidad a elegir ciertas normas en determinados problemas. Para una

discusión de las diversas normas, se puede consultar [1]. El presente trabajo va reducir su enfoque a sólo dos tipos de lapsos y un solo tipo de corrimiento.

1.4.1. Lapso maximal y $1 + \log$

La primera pregunta que uno puede hacer es ¿por qué no hacer el lapso uno en toda la hípersuperfice a todo tiempo? Este es el lapso geodésico basado en que hace que el tiempo t coincida con el tiempo propio de los observadores Eulerianos en caída libre. Esto es una mala idea en una simulación numérica por el efecto de que si existe un campo gravitacional no uniforme, los observadores pueden colisionar entre ellos. Esto genera una cáustica que numéricamente implica que las coordenadas dejan de ser uno a uno. Por lo tanto, usar lapso geodésico en la mayoría de los casos va a parar la simulación en poco tiempo gracias a este problema.

Para evitar dicho problema se propone que los elementos de volumen asociados a los observadores Eulerianos permanezcan constantes. Puesto que

$$K_{\mu\nu} = -P^{\alpha}_{\mu} \nabla_{\alpha} n_{\nu} \,, \tag{1.39}$$

se sigue que

$$\nabla_{\mu} n^{\mu} = -K \,, \tag{1.40}$$

así que pedir que $K = \partial_t K = 0$ garantiza que dichos elementos de volumen permanezcan constantes. De la ecuación (1.37) esto implica la condición de lapso maximal primero sugerida por Lichnerowicz [39]

$$\nabla^2 \alpha = \alpha \left(\hat{A}_{ij} \hat{A}^{ij} + \frac{1}{3} K^2 + 4\pi(\rho + S) \right) . \tag{1.41}$$

El nombre de maximal proviene del hecho de que al tomar $K = 0$ el volumen de la hípersuperfice espacial es máximo con respecto a pequeñas variaciones de la hípersuperficie. Al utilizar lapso maximal no sólo se evita enfocar los observadores Eulerianos sino que además se tiene el fenómeno de *evitación de singularidad*. Es decir, las hípersuperficies espaciales no pueden tocar la singularidad en la variedad. Este fenómeno se debe a que el lapso maximal α se colapsa en la vecindad de la singularidad, lo que efectivamente detiene el avance del tiempo hacia su interior.

La teoría detrás de este colapso esta fuera del enfoque de este trabajo pero se pueden enlistar los resultados principales. Tomando (1.41) para el caso de vacío y utilizando la constricción hamiltoniana para regresar al escalar de Ricci se encuentra que

$$\nabla^2 \alpha = \alpha \, R \,. \tag{1.42}$$

Smarr y York construyen un modelo analítico [55] donde $R = R_0$ dentro de una esfera y afuera $R = 0$. Dicho modelo encuentra que el lapso colapsa exponencialmente como e^{-R_0} al centro de la esfera. Del mismo modo, para un hoyo negro de Schwarschild se pueden construir las rebanadas maximales de manera analítica (aunque implican integrales que deben ser evaluadas numéricamente).

El colapso del lapso maximal resulta ser tan característico que empíricamente se suele tomar como un indicador de que se ha formado un hoyo negro. Para el caso de Schwarschild [1] se conoce bien que a tiempo tardíos $\alpha \sim 0.3248$ en el horizonte. Este valor se suele tomar como base para empezar a buscar horizontes.

Habiendo detallado las ventajas claras del lapso maximal, hay que considerar las desventajas. La más clara se esconde detrás de la ecuación (1.41) y es que se trata de una ecuación elíptica y resolverla numéricamente a cada paso de integración resulta muy demandante en el sentido de

tiempo de simulación. Por ser solución a una ecuación elíptica, también hay que imponer condiciones de frontera de manera cuidadosa. Para un espacio-tiempo asintóticamente plano lo que se suele hacer es tomar una condición de Robin en la frontera externa

$$\partial_r \, \alpha + \frac{(\alpha - 1)}{r} \sim 0 \, , \tag{1.43}$$

conforme $r \to \infty$. Por otro lado, en la singularidad se pide que el lapso tenga gradiente cero, i.e. que sea una función par. Las condiciones de frontera son más detalladas en el capítulo del resolvedor elíptico.

La desventaja particular de tiempo de simulación fue circumventada introduciendo otro tipo de lapsos tanto de manera teórica pero sobre todo de manera empírica. Entre estos lapsos, el más exitoso es la familia de lapsos hiperbólicos de Bona-Massó [17]. En dicha familia, el lapso es una variable dinámica que sigue la ecuación de evolución

$$\frac{\mathrm{d}\alpha}{\mathrm{d}t} = -\alpha^2 \, f(\alpha) \, K \, , \tag{1.44}$$

donde f es una función positiva de α. El caso de $f = 2/\alpha$ ha sido probado muy robusto empíricamente y se conoce como lapso $1+\log$ y es utilizado de manera casi universal en los códigos modernos de la comunidad. El nombre no es accidental ya que equivale a tener

$$\alpha = 1 + \log \, \gamma \, , \tag{1.45}$$

ya que la ecuación de evolución para el determinante γ via (1.1)

$$\partial_t \, \gamma - \pounds_\beta \, \gamma = -2\gamma \, (\alpha \, K - \partial_i \, \beta^i) \, , \tag{1.46}$$

implica que

$$\partial_t \, \alpha - \pounds_\beta \, \alpha = -2\alpha \, K \, . \tag{1.47}$$

La razón por la que este lapso se denomina hiperbólico es porque se puede demostrar que sigue una especie de ecuación de onda (véase [1]). Por lo mismo, son condiciones mucho más económicas y eficientes para resolver numéricamente.

El lapso $1 + \log$ también satisface propiedades de evitación de singularidad, donde ahora el resultado central es que el lapso se vuelve cero en la singularidad o se vuelve cero antes de llegar a dicha singularidad.

1.4.2. Vector de corrimiento hiperbólico

Para el caso del vector de corrimiento, también existen condiciones fundamentadas en un aspecto geométrico. Sin embargo, se han tomado resultados mucho más pragmáticos basados en ecuaciones hiperbólicas. Partiendo de un corrimiento fundamentado en un resultado geométrico, *Gamma-freezing*, donde $\partial_t \, \hat{\Gamma}^i = 0$, se llega a una ecuación similar con forma hiperbólica propuesta por Alcubierre *et al.* en [4] denominada *Gamma-driver*:

$$\partial_t^2 \, \beta^i = c_2 \, \partial_t \, \hat{\Gamma}^i - \eta \, \partial_t \, \beta^i \, , \tag{1.48}$$

donde c_2 es un parámetro tomado usualmente como $c_2 = 3/4$ para que la velocidad longitudinal del corrimiento sea igual a la de la luz asintóticamente (véase [1]) y η añade un término de amortiguamiento que suele tomarse como $2/M_{\mathrm{ADM}}$ (M_{ADM} es la masa ADM del espacio tiempo). Siguiendo

la idea de escribir las ecuaciones de manera covariante, es claro que llegamos a la condición de corrimiento *Delta-driver*

$$\partial_t^2 \, \beta^i = c_2 \, \partial_t \, \hat{\Delta}^i - \eta \, \partial_t \, \beta^i \, . \tag{1.49}$$

1.5. Aplicación a coordenadas cilíndricas

Este trabajo tiene como desarrollo principal una evolución en un espacio axisimétrico que es descrito por un sistema coordenado cilíndrico $\{\rho, z, \varphi\}$. Dicho espacio se define como uno donde existe un vector de Killing ξ^i para el cual sus curvas son cerradas y además $\xi^i = \delta^i_\varphi$. En otras palabras, un espacio axisimétrico es uno en el que las cantidades son independientes del ángulo φ.

1.5.1. Restricciones sobre la métrica

La primera pregunta que se hace es ¿cuál es la forma más general para un métrica en un espacio axisimétrico? No resulta ser tan trivial esta pregunta porque a diferencia del caso esférico, no se puede asumir que la métrica puede escribirse de forma diagonal, sino hay que considerar los seis componentes independientes.

Puesto que la métrica física está relacionada con la conforme via la simple relación (1.8), la estructura de las dos métricas es la misma y cualquier discusión que aplica para una aplica para la otra. Como en la formulación BSSN la variable dinámica es la métrica conforme, vamos a centrarnos en esta.

La primera simplificación es considerar el caso relevante a este trabajo que es que no haya momento angular. Eso claramente implica que $\hat{\gamma}_{\rho\varphi} = \hat{\gamma}_{z\varphi} = 0$, lo cual reduce los componentes independientes a cuatro. Esta reducción tiene como consecuencia que no habrán modos impares de radiación gravitacional, pero de todos modos hay diversos fenómenos de interés que se pueden estudiar.

De esta manera podemos proponer un ansatz para la métrica de la forma

$$\hat{\mathrm{d}l}^2 = a \, \mathrm{d}\rho^2 + b \, \mathrm{d}z^2 + \rho^2 h \, \mathrm{d}\varphi^2 + 2\rho c \, \mathrm{d}\rho \, \mathrm{d}z \, . \tag{1.50}$$

Es decir,

$$a \equiv \hat{\gamma}_{\rho\rho} \, , \tag{1.51}$$

$$b \equiv \hat{\gamma}_{zz} \, , \tag{1.52}$$

$$h \equiv \frac{\hat{\gamma}_{\varphi\varphi}}{\rho^2} \, , \tag{1.53}$$

$$c \equiv \frac{\hat{\gamma}_{\rho z}}{\rho} \, . \tag{1.54}$$

La razón por la que se ponen factores de ρ en h, c es simplemente para facilitar un esquema de regularización que se detalla abajo.

1.5.2. Regularización

Hay algunos detalles respecto a las funciones a, b, h, c. El primero viene de pedir simetría axial $\rho \to -\rho$. Esto significa de inmediato que como está escrita la métrica en (1.50), a, b, h, c son funciones

pares y suaves de ρ. La segunda condición, que además justifica la factorización de ρ en h, c viene de pedir que la métrica sea localmente plana en el eje $\rho = 0$.

Una manera muy completa de llevar a cabo este análisis es como se hace en [49]: resolviendo la ecuación de Killing y llegando a una forma muy general para la métrica. En este caso se sigue un argumento sacado de [50] para llevar a cabo la regularización.

Básicamente, primero hay que regresar a coordenadas cartesianas para las cuales tener simetría axial implica que podemos intercambiar x por y y además podemos reflejar libremente por dichos ejes. Entonces, para un z fijo podemos concluir que tener una métrica localmente plana requiere que

$$\hat{\gamma}_{xx} \sim k_\rho + \mathcal{O}(\rho^2)\,, \tag{1.55}$$
$$\hat{\gamma}_{yy} \sim k_\rho + \mathcal{O}(\rho^2)\,, \tag{1.56}$$
$$\hat{\gamma}_{zz} \sim k_z + \mathcal{O}(\rho^2)\,, \tag{1.57}$$
$$\hat{\gamma}_{xy} \sim \mathcal{O}(xy) \sim \mathcal{O}(\rho^2)\,, \tag{1.58}$$
$$\hat{\gamma}_{xz} \sim \mathcal{O}(x) \sim \mathcal{O}(\rho)\,, \tag{1.59}$$
$$\hat{\gamma}_{yz} \sim \mathcal{O}(y) \sim \mathcal{O}(\rho)\,, \tag{1.60}$$

donde k_ρ, k_z son constantes. Si se hace un cambio de coordenadas a cilíndricas se sigue después de algo de álgebra que

$$\hat{\gamma}_{\rho\rho} \sim k_\rho + \mathcal{O}(\rho^2)\,, \tag{1.61}$$
$$\hat{\gamma}_{zz} \sim k_z + \mathcal{O}(\rho^2)\,, \tag{1.62}$$
$$\hat{\gamma}_{\varphi\varphi} \sim \rho^2 \left(k_\rho + \mathcal{O}(\rho^2)\right)\,, \tag{1.63}$$
$$\hat{\gamma}_{\rho z} \sim \mathcal{O}(\rho)\,, \tag{1.64}$$
$$\hat{\gamma}_{\rho\varphi} \sim \mathcal{O}(\rho^3)\,, \tag{1.65}$$
$$\hat{\gamma}_{z\varphi} \sim \mathcal{O}(\rho^2)\,. \tag{1.66}$$

Por ende, se confirma que a, b son funciones pares en el eje; que al factorizar ρ^2 de $\hat{\gamma}_{\varphi\varphi}$ se tiene una función par; y que también se puede factorizar un factor ρ de $\hat{\gamma}_{\rho z}$. Sin embargo, esto no es todo, puesto que las ecuaciones anteriores implican que $\hat{\gamma}_{\rho\rho} - \hat{\gamma}_{\varphi\varphi}/\rho^2 = a - h \sim \mathcal{O}(\rho^2)$, gracias a la constante k_ρ. Esto permite definir la variable

$$\lambda \equiv \frac{a-h}{\rho^2}\,, \tag{1.67}$$

que debe ser regular y par en el eje $\rho = 0$. Este análisis de regularización se ha llevado a cabo en formas similares en trabajos como [2], [50], [1] hasta aterrizarlo en la forma actual como está en [61] y [62].

El punto principal de la regularización es que nos impone condiciones de simetría en el eje. Al llevarse a cabo la simulación, algunos términos que van como $1/\rho$ introducen problemas numéricos en el código. Dichas divergencias no existen a un nivel analítico como consecuencia de la axisimetría y la condición de métrica localmente plana. Sin embargo, para mantener este comportamiento de un modo numérico lo que se necesita hacer es imponer la paridad correspondiente como una condición de frontera.

En la práctica no se tienen puntos de la malla sobre el eje, sino que se tiene una malla recorrida por medio paso, i.e. puntos en $\rho = +\Delta\rho/2$ y una zona llamada *ghost zone* en $\rho = -\Delta\rho/2$. De esta

manera las condiciones de paridad/simetría se imponen copiando el valor de la función al ghost zone con un signo positivo para funciones pares y con uno negativo para una impar.

Aún así, imponer que a, h sean pares no es suficiente: λ debe de serlo también. Pero esto causa un problema de sobredeterminación gracias a que se tienen tres condiciones y sólo dos variables dinámicas. La solución es promover a λ a una variable dinámica y revisar su comportamiento por medio de una nueva constricción

$$C_\lambda \equiv \rho^2 \lambda - (a - h) = 0 \,. \tag{1.68}$$

La descripción anterior no sólo aplica para la métrica sino también para cualquier tensor regular, en particular la curvatura extrínseca sin traza $\hat{A}_{ij}$ para la que también hay que introducir una cantidad

$$\hat{A}_\lambda \equiv \frac{\hat{A}_a - \hat{A}_h}{\rho^2} \,. \tag{1.69}$$

Por tanto, se necesitan dos ecuaciones de evolución adicionales que deben ser sacadas del sistema original. Dichas ecuaciones van a ser analíticamente regulares pero numéricamente hay que tener un gran cuidado en cómo escribirlas. Dichas ecuaciones están presentadas en [62] y se repiten por completez a continuación:

$$\begin{aligned} \partial_t \lambda = {} & \beta^m \partial_m \lambda + 2\lambda \left(\frac{\beta^\rho}{\rho} \right) + 2\lambda \partial_\rho \beta^\rho + 2h \left(\partial_\rho \left(\frac{\beta^\rho}{\rho} \right) / \rho \right) \\ & + 2c \left(\frac{\partial_\rho \beta^z}{\rho} \right) - 2\alpha \hat{A}_\lambda - \frac{2}{3} \sigma \lambda \hat{\nabla}_m \beta^m \,. \end{aligned} \tag{1.70}$$

$$\begin{aligned} \partial_t \hat{A}_\lambda = {} & \beta^m \partial_m \hat{A}_\lambda + 2\hat{A}_\lambda \left(\frac{\beta^\rho}{\rho} \right) + 2\hat{A}_\lambda \partial_\rho \beta^\rho + 2\hat{A}_h \left(\partial_\rho \left(\frac{\beta^\rho}{\rho} \right) / \rho \right) \\ & + 2\hat{A}_c \left(\frac{\partial_\rho \beta^z}{\rho} \right) - \frac{2}{3} \sigma \hat{A}_\lambda \hat{\nabla}_m \beta^m \\ & + e^{-4\phi} \left((\nabla_i \nabla_j)_\lambda \alpha + \alpha R_\lambda - 8\pi \alpha S_\lambda \right) \\ & - \frac{\lambda}{3} \left(-\nabla^2 \alpha + \alpha R - 8\pi \alpha S \right) \\ & + \alpha \left(K \hat{A}_\lambda - 2 \left(\hat{A}_{ik} \hat{A}^k{}_j \right)_\lambda \right) \,. \end{aligned} \tag{1.71}$$

Nótese el énfasis en como se escriben los términos $1/\rho$: siempre se tiene un término manifestamente impar en el numerador para reducir el riesgo de ruido numérico en la simulación. Por ejemplo, en (1.70) se tiene $\left(\partial_\rho \left(\frac{\beta^\rho}{\rho} \right) / \rho \right)$. Primero se hace la división β^ρ / ρ que resulta en una cantidad par. Después se deriva este resultado, produciendo una cantidad impar que finalmente se divide por el término ρ y resulta en un término globalmente par.

En realidad, no se ha resuelto del todo el problema en la ecuación (1.71), pues sólo se han introducido nomenclaturas para $R_\lambda, (\nabla_i \nabla_j)_\lambda \alpha, \left(\hat{A}_{ik} \hat{A}^k{}_j \right)_\lambda$. Estos términos se deben de escribir de manera regular también.

Primero hay que introducir la métrica inversa $\hat{\gamma}^{ij}$ que tiene una forma parecida a la métrica usual:

$$g_a \equiv \hat{\gamma}^{\rho\rho} = \frac{b}{ab - \rho^2 c^2}\,, \tag{1.72}$$

$$g_b \equiv \hat{\gamma}^{zz} = \frac{a}{ab - \rho^2 c^2}\,, \tag{1.73}$$

$$g_h \equiv \rho^2\, \hat{\gamma}^{\varphi\varphi} = \frac{1}{h}\,, \tag{1.74}$$

$$g_c \equiv \frac{\hat{\gamma}^{\rho z}}{\rho} = -\frac{c}{ab - \rho^2 c^2}\,. \tag{1.75}$$

Nótese que ahora g_h involucra una multiplicación de ρ^2, a diferencia de h que involucra una división. Para la métrica inversa también definimos

$$g_\lambda = \frac{g_a - g_h}{\rho^2} = \frac{c^2 - b\lambda}{ab - \rho^2 c^2}\,, \tag{1.76}$$

misma que se comprueba como totalmente regular.

El algoritmo básico para la regularización es primero hacer a un lado los términos que son explícitamente regulares que no se deben modificar en absoluto. Después se toman los términos que parecen ser irregular pero en realidad son regulares via las condiciones de paridad. Por ejemplo β^ρ/ρ, $\partial_\rho a/\rho$, etc. Finalmente quedan los términos que sólo se pueden regularizar con las λ's. El mejor ejemplo para verla en acción es R_λ. Para este, los términos explícitamente regulares son

$$\begin{aligned}\hat{R}^r_\lambda =& -\lambda^2 g_a g_h - 2c\lambda g_c g_h + 2c g_c\, \left(g_c \partial_z\, a + g_b \partial_z\, c\right) \\ &+ \frac{1}{2}\, g_c^2\, \left(-(\partial_z\, a)^2 + \partial_\rho\, a\, \left(2\rho \partial_z\, c + \partial_\rho\, b\right)\right) \\ &+ 2 g_c\, \left(\rho g_c \partial_z\, a + \rho g_b \partial_z\, c + g_a \partial_\rho\, a\right)\, \partial_\rho\, c + g_a g_b (\partial_\rho\, c)^2\,.\end{aligned} \tag{1.77}$$

Los términos que se pueden regularizar usando paridad son

$$\begin{aligned}\hat{R}^n_\lambda =& \frac{1}{2}\, \left(\frac{\hat{\Delta}^\rho}{\rho}\right)\, \left(\frac{\partial_\rho\, a}{\rho}\right) + 2c g_a g_c\, \left(\frac{\partial_\rho\, a}{\rho}\right) + \lambda g_h^2\, \left(\frac{\partial_\rho\, h}{\rho}\right) \\ &- g_c g_h \partial_z\, h\, \left(\frac{\partial_\rho\, h}{\rho}\right) - \frac{g_a g_h}{2}\, \left(\frac{\partial_\rho\, h}{\rho}\right)^2 - \frac{g_h^2}{4}\, \left(\frac{\partial_\rho\, h}{\rho}\right)^2 \\ &- g_c\, \left(\frac{\partial^2_{\rho z}\, a}{\rho}\right) + g_c\, \left(\frac{\partial^2_{\rho z}\, h}{\rho}\right) + c\, \left(\frac{\partial_\rho \hat{\Delta}^z}{\rho}\right) - \frac{1}{2}\, \left(\frac{\hat{\Delta}^\rho}{\rho}\right)\, \left(\frac{\partial_\rho\, h}{\rho}\right) \\ &+ g_a g_c \partial_z\, a\, \left(\frac{\partial_\rho\, a}{\rho}\right) + \frac{3 g_a^2}{4}\, \left(\frac{\partial_\rho\, a}{\rho}\right)^2 + g_b g_c \partial_z\, a\, \left(\frac{\partial_\rho b}{\rho}\right) \\ &+ g_b^2 \partial_z\, c\, \left(\frac{\partial_\rho\, b}{\rho}\right) - \frac{g_b^2}{4}\, \left(\frac{\partial_\rho\, b}{\rho}\right)^2 + 2c g_a g_b\, \left(\frac{\partial_\rho\, c}{\rho}\right)\,.\end{aligned} \tag{1.78}$$

Después vienen los términos para los que hay que hacer manipulaciones reales.

- Hay términos triviales donde sólo hay que formar λ's:

$$\frac{c^2 g_a g_b}{\rho^2} - \frac{c^2 g_b g_h}{\rho^2} = c^2 g_b g_\lambda\,. \tag{1.79}$$

$$\frac{\hat{\Delta}^z\,\partial_z\,a}{2\rho^2}-\frac{\hat{\Delta}^z\,\partial_z\,h}{2\rho^2}=\frac{\hat{\Delta}^z}{2}\,\partial_z\,\lambda\,. \tag{1.80}$$

$$-\frac{g_b\,\partial^2_{zz}\,a}{2\rho^2}+\frac{g_b\,\partial^2_{zz}\,h}{2\rho^2}=-\frac{g_b}{2}\,\partial^2_{zz}\,\lambda\,. \tag{1.81}$$

- Existen otros donde hay que factorizar una a o una h y luego hacer otra reducción. Esto puede ocasionar que haya dos maneras de hacer la regularización como en los siguientes ejemplos.

$$\begin{aligned}-\frac{\hat{\Delta}^\rho h}{\rho^3}+\frac{a\partial_\rho\,\hat{\Delta}^\rho}{\rho^2}=&\lambda\left(\frac{\hat{\Delta}^\rho}{\rho}\right)+a\left(\frac{\partial_\rho\left(\hat{\Delta}^\rho/\rho\right)}{\rho}\right),\\ =&\lambda\,\partial_\rho\,\hat{\Delta}^\rho+h\left(\frac{\partial_\rho\left(\hat{\Delta}^\rho/\rho\right)}{\rho}\right).\end{aligned} \tag{1.82}$$

$$\begin{aligned}\frac{g_a g_b(\partial_z\,a)^2}{2\rho^2}-\frac{g_b g_h(\partial_z\,h)^2}{2\rho^2}=&\frac{g_b g_\lambda}{2}\,(\partial_z\,a)^2+\frac{g_b g_h}{2}\,\partial_z\,\lambda\,(\partial_z\,a+\partial_z\,h)\,,\\ =&\frac{g_b g_\lambda}{2}\,(\partial_z\,h)^2+\frac{g_a g_b}{2}\,\partial_z\,\lambda\,(\partial_z\,a+\partial_z\,h)\end{aligned} \tag{1.83}$$

- El último tipo de términos son los que involucran derivadas de λ con respecto a ρ como es en este caso:

$$\begin{aligned}-\frac{g_a\partial^2_{\rho\rho}\,a}{2\rho^2}+\frac{g_a\partial^2_{\rho\rho}\,h}{2\rho^2}-\frac{g_h\partial_\rho\,a}{2\rho^3}+\frac{g_h\partial_\rho\,h}{2\rho^3}+\frac{2g_h\lambda}{\rho^2}&\\ =\frac{g_\lambda}{2}\,\left(\partial^2_{\rho\rho}\,h-\partial^2_{\rho\rho}\,a\right)-\frac{g_h}{2}\,\left(\partial^2_{\rho\rho}\,\lambda+5\left(\frac{\partial_\rho\,\lambda}{\rho}\right)\right),&\\ =\frac{g_\lambda}{2}\,\left(\rho\,\partial_\rho\,\lambda-2\lambda\right)-\frac{g_a}{2}\,\left(\partial^2_{\rho\rho}\,\lambda+5\left(\frac{\partial_\rho\,\lambda}{\rho}\right)\right).&\end{aligned} \tag{1.84}$$

Como se vio en este esquema, puede haber cierta ambigüedad en la regularización. ¿Qué expresión se debe tomar? En la práctica, lo que se hace en este trabajo es algo inspirado en el código numérico de [61]: abrir la posibilidad de tomar múltiplos complementarios de las dos formas. En teoría las dos formas son equivalentes pero a nivel numérico es sumamente complejo decir qué va a ocurrir si se toma una forma por encima de la otra.

Capítulo 2

Datos iniciales de ondas de Brill

Las ondas de Brill son el objeto de estudio principal de este trabajo, pues serán los datos iniciales para las simulaciones numéricas y el cálculo de dichos datos iniciales requiere el resolvedor elíptico detallado en el Capítulo 4. Por lo tanto, antes de entrar al aspecto numérico, se requiere un conocimiento general de ellas.

2.1. Descomposición conforme

Las ondas de Brill son las soluciones no triviales más sencillas en el vacío (para fines modernos, la solución de Schwarschild ya resulta demasiado trivial). Pese a la aparente sencillez en el planteamiento de la métrica y en algunas ecuaciones presentadas a continuación, el fenómeno resultante tiene suficiente complejidad y similitudes importantes a otros fenómenos más complicados en relatividad numérica. Por lo tanto, estudiar ondas de Brill puede dar y ha dado nuevas guías para implementar en sistemas complejos como es la simulación de un sistema binario de hoyos negros y la extracción de ondas gravitacionales del mismo (véase [5]).

La construcción original debida a Dieter Brill en [18] empieza por considerar un espacio-tiempo inicialmente en axisimetría y con una métrica conforme de la forma:

$$\tilde{\mathrm{d}l}^2 = e^{2q}\left(\mathrm{d}\rho^2 + \mathrm{d}z^2\right) + \rho^2\,\mathrm{d}\varphi^2\,, \tag{2.1}$$

donde q es una función de ρ, z general con única restricción de que ella y sus derivadas valgan cero en el eje z y que decaiga más rápido que r^{-1}. Nótese que se usa una tilde para denotar esta métrica en vez del circunflejo usual. Esto se explicará más adelante pero tiene que ver con que *a priori* no es una métrica compatible con el formalismo introducido en el capítulo anterior. Regresando a q, sus restricciones se vuelven

$$\begin{aligned} q\big|_{\rho=0} &= 0\,, \\ \partial_\rho^n\, q\big|_{\rho=0} &= 0 \quad \text{para } n \text{ impar}\,, \\ q\big|_{r\to\infty} &= \mathcal{O}(r^{-2})\,. \end{aligned} \tag{2.2}$$

La razón para estas restricciones es explicada tanto en [18] como en [43], pero en resumen, lo único que hacen es garantizar que la integral del escalar de Ricci sea cero, i.e. $\int R\,\mathrm{d}v = 0$ y que la masa ADM salga siempre positiva.

Ahora que se tiene la forma general de la métrica conforme, hay que resolver las constricciones hamiltoniana y de momento para que realmente sea solución de las ecuaciones de Einstein y pueda

ser usada como datos iniciales. La primera simplificación considerada es que el espacio empieza en un estado de simetría temporal. Esto implica gracias a las ecuación ADM para la métrica, (1.1), que $K_{ij} = 0$ y por tanto, la constricción de momento (1.7) se cumple trivialmente. Para resolver la constricción hamiltoniana seguimos una descomposición conforme de York-Lichnerowicz como se detalla en [1]. Para ello, se introduce el otro factor conforme $\tilde{\psi}$, de tal manera que la transformación conforme está dada por

$$\tilde{\gamma}_{ij} = \tilde{\psi}^{-4}\, \gamma_{ij}\,, \tag{2.3}$$

lo que implica, vía la comparación con (1.8) que

$$\tilde{\psi} = e^{\tilde{\phi}}\,. \tag{2.4}$$

De esta manera, la constricción hamiltoniana determina la siguiente ecuación para $\tilde{\psi}$:

$$8\,\tilde{\nabla}^2\,\tilde{\psi} - \tilde{R}\,\tilde{\psi} + \tilde{\psi}^5\,\left(K_{ij}\,K^{ij} - K^2\right) + 16\pi\tilde{\psi}^5\,\rho = 0\,. \tag{2.5}$$

Gracias a que estamos en el vacío donde $\rho = 0$, entonces, unido a la simetría temporal ya mencionada se simplifica la ecuación anterior a

$$\tilde{\nabla}^2\,\tilde{\psi} - \frac{1}{8}\,\tilde{R}\,\tilde{\psi} = 0\,. \tag{2.6}$$

Ahora hay que determinar la forma de $\tilde{\nabla}^2$ y $\tilde{R}$ a partir de la métrica conforme (2.1). Después de unos cálculos sencillos se calcula que

$$\tilde{R} = -2e^{-2q}\,\left(\partial^2_{\rho\rho}\,q + \partial^2_{zz}\,q\right)\,, \tag{2.7}$$

mientras que

$$\tilde{\nabla}^2 = e^{-2q}\,\mathring{\nabla}^2\,, \tag{2.8}$$

donde, por las convenciones que se han estado utilizando, $\mathring{\nabla}^2$ es el laplaciano plano. Consecuentemente, la ecuación final es

$$\mathring{\nabla}^2\,\tilde{\psi} + \frac{1}{4}\,\left(\partial^2_{\rho\rho}\,q + \partial^2_{zz}\,q\right)\,\tilde{\psi} = 0\,. \tag{2.9}$$

2.2. Ecuación elíptica

Dada la forma expandida de la ecuación (2.9),

$$\frac{\partial^2\tilde{\psi}}{\partial\rho^2} + \frac{\partial^2\tilde{\psi}}{\partial z^2} + \frac{1}{\rho}\,\frac{\partial\tilde{\psi}}{\partial\rho} + \frac{1}{4}\,\left(\partial^2_{\rho\rho}\,q + \partial^2_{zz}\,q\right)\,\tilde{\psi} = 0\,, \tag{2.10}$$

podemos clasificarla como se suele hacer para una ecuación de derivadas parciales. De hecho, es sencillo ver que el discriminante de esta ecuación es la constante negativa $\Delta = b^2 - 4ac = -4$, lo cual implica que se trata de una ecuación elíptica.

2.2.1. Condiciones de frontera

La ecuación elíptica anterior en cualquiera de sus formas (2.9) ó (2.10) requiere de una condición de frontera que lleve a la solución del factor $\tilde{\psi}$ adecuado. Para determinar una condición apropiada, podemos guiarnos por un aspecto físico y de soluciones asintóticas.

Claramente, el decaimiento de q implica que el espacio es asintóticamente plano, i.e. que mientras r crece, la solución se aproxima a un espacio de Schwarschild con $\tilde{\psi} = 1 + M/2r$. Esto implica a su vez una conclusión que era originalmente más sencilla de visualizar: que $\tilde{\psi} \to 1$ en infinito. Sin embargo, la expresión con la masa de Schwarschild es mucho más fuerte y valiosa para implementar numéricamente. Esto es porque en la práctica no se suele usar una malla numérica que contenga al infinito (salvo en excepciones que compactifican el espacio, pero esto no se explora en este trabajo). Por lo tanto, no se puede poner directamente $\tilde{\psi} = 1$ en infinito como una condición de Dirichlet. En vez, se usa una condición de Robin en el borde de la malla como ya se anticipó en la sección de lapso maximal del capítulo anterior.

La condición de Robin es una mezcla de las condiciones de Dirichlet y Neumann. Es decir, es una relación entre las derivadas y el valor de una función con una sutil diferencia: consiste en una aproximación en el sentido de que es una relación que se cumple mientras $r \to \infty$. Sin embargo, sí existen bases teóricas para saber qué tan lejos se debe poner la frontera para llegar a un error numérico debajo de una tolerancia estipulada. Esto es explicado a mayor detalle en el Capítulo 4 del resolvedor elíptico. Por ahora, es suficiente decir que en la frontera externa $r \gg 1$ se va a implementar una condición de Robin,

$$\frac{\partial \tilde{\psi}}{\partial r} + \frac{\tilde{\psi} - 1}{r} \approx 0\,, \tag{2.11}$$

en una frontera suficientemente lejos pero claramente no infinita.

2.3. Introducción compatible al formalismo BSSN

Uno podría pensar que una vez que se obtiene el factor conforme $\tilde{\psi}$ todo está listo para introducirse en el formalismo del capítulo anterior. Esto no es cierto debido a un detalle sutil que sale del hecho de que la métrica conforme (2.1) no tiene un determinante igual al determinante plano $\mathring{\gamma} = \rho^2$. De hecho, con esta métrica inicial, se tiene

$$\hat{\gamma} = \rho^2\, e^{4q}\,. \tag{2.12}$$

Para poder ser compatible con el formalismo, hay que absorber e^{4q} dentro del factor conforme. Un cálculo rápido define el verdadero factor conforme

$$\psi = e^{q/3}\,\tilde{\psi}\,, \tag{2.13}$$

como el nuevo factor conforme. De esta manera, la nueva métrica conforme,

$$\hat{\mathrm{d}l}^2 = e^{2q/3}\,\left(\mathrm{d}\rho^2 + \mathrm{d}z^2\right) + \rho^2\, e^{-4q/3}\,\mathrm{d}\varphi^2\,, \tag{2.14}$$

tiene determinante $\hat{\gamma} = \rho^2 = \mathring{\gamma}$ y además, junto con el nuevo factor conforme describe a la misma métrica física, pues:

$$\psi^4\,\hat{\mathrm{d}l}^2 = \tilde{\psi}^4\,\left(e^{2q}\,\left(\mathrm{d}\rho^2 + \mathrm{d}z^2\right) + \rho^2\,\mathrm{d}\varphi^2\right) = \tilde{\psi}^4\,\tilde{\mathrm{d}l}^2 = \mathrm{d}l^2\,. \tag{2.15}$$

Por lo tanto, se usará (2.14) para inicializar la métrica conforme en el código de evolución y hay que recordar que el factor $\tilde{\psi}$ será reescalado por $e^{q/3}$ después de obtener su primera versión vía el resolvedor numérico.

Esto se detalla con énfasis no sólo porque se trate de un tecnicismo para poder ser compatible con el formalismo BSSN en coordenadas curvilíneas. Más bien se trata de una advertencia, pues el factor conforme que aparece en otros trabajos de la comunidad y que juega un papel importante a la hora de dar ciertos resultados no es el mismo que se tiene en este trabajo. Sin embargo, es claro cómo convertir entre ellos por si fuera relevante.

2.4. Resultados previos

Antes de terminar este capítulo se van a enlistar algunos resultados previos que se consideran relevantes sobre simulaciones de ondas de Brill con el fin de que se puedan comparar algunos cuando se presenten los resultados de este trabajo.

Previo a realizar evoluciones de ondas de Brill, la comunidad las consideró como buenas candidatas para fuentes de ondas gravitacionales. Eppley [27] las estudió con el fin de calcularles su masa ADM con distintas expresiones e integrales. Después calculó el flujo de energía radiada por ondas gravitacionales para estimar una pérdida de masa. Eppley utilizó una forma concreta para la q:

$$q = a\,\frac{\rho^2}{1+(r/\lambda)^n}\,, \tag{2.16}$$

donde recordando el Cuadro 1.1 de convenciones $r = \sqrt{\rho^2+z^2}$ y a, λ, n son parámetros para los datos iniciales para los cuales Eppley estudió $\lambda = 1$, $n = 5$ y varió la amplitud a.

Después, Holz *et al.* [34] usaron ondas de Brill como un modelo que simula la producción de hoyos negros a partir del colapso de ondas gravitacionales de fuerte amplitud. En dicho trabajo, se buscaron superficies atrapadas con el fin de determinar en qué punto existe el colapso a un hoyo negro. Holz *et al.* utilizaron la q que se ha vuelto más estándar y que es en realidad la que se utiliza en este trabajo:

$$q = a\,\rho^2\,e^{-r^2}\,. \tag{2.17}$$

Estos datos iniciales están únicamente caracterizados por la amplitud a para los que Holz *et al.* encontraron una superficie atrapada a partir del valor crítico de $a_* = 7.5$ con una masa ADM de $m_* = 1.546$.

La búsqueda de horizontes aparentes fue continuada cuando Alcubierre *et al.* [7] presentaron un trabajo cabal con el fin de poner a prueba algunos encontradores de horizontes aparentes (AHF's por sus siglas en inglés) en distintas geometrías. Una de las geometrías utilizadas fueron tanto las ondas de Brill con la forma de Eppley, así como la forma de Holz *et al.*. En el análisis realizado, se hicieron cálculos tanto en 2D como en 3D con el fin de poder dar mayor cimiento y credibilidad a los resultados cuando los dos métodos coinciden. Alcubierre *et al.* concuerdan totalmente en el cálculo de las masas ADM con Holz pero difieren en la amplitud crítica, pues reportan que sus AHF's en 2D y 3D encontraron el primer horizontes en $a_* \in [11.81, 11.82]$ con una masa ADM $m_* \approx 4.5$. Como último detalle, se presenta una tabla con las masas para dar una idea de cómo se comportan las masas en función de la amplitud.

a	M_{ADM}
1	0.0338 ± 0.0006
2	0.126 ± 0.001
3	0.270 ± 0.002
4	0.459 ± 0.003
5	0.698 ± 0.004
6	0.991 ± 0.005
10	2.91 ± 0.01
12	4.67 ± 0.02

Tabla 2.1: Masas ADM para ondas de Brill con q tipo Holz para distintas amplitudes. Masas tomadas con permiso de [1].

Los trabajos anteriores se han basado en las ondas de Brill como datos iniciales. Es decir, la amplitud critica a_* encontrada por Alcubierre *et al.* genera un horizonte aparente desde el tiempo cero. La evolución de las ondas de Brill también ha sido estudiada, con el ejemplo notable de Alcubierre *et al.* [5].

En dicho trabajo, uno de los enfoques es estudiar la evolución y hacer comparaciones relevantes, como son las ondas gravitacionales radiadas con las de un colisión de dos hoyos negros. Otro enfoque más relevante para este trabajo es que Alcubierre *et al.* encuentran que si bien no se genera un hoyo negro en los datos iniciales para amplitudes menores a 11.8, sí puede haber un colapso en el curso de la evolución. La amplitud crítica para la cual la evolución lleva a un colapso se reporta como $a_* = 4.85 \pm 0.15$. En el caso particular de $a = 6$, se encuentra que el horizonte aparece a $t \approx 7.7$ (usando lapso maximal). Además, el AHF puede seguir el horizonte durante la evolución, lo que hace posible hacer un cálculo de su área y masa en función del tiempo.

Un objetivo principal de este trabajo es reproducir de forma confiable los resultados ya mencionados, en particular los siguientes dos:

1. Encontrar un horizonte aparente en los datos iniciales con una amplitud crítica que coincida con [7]. Esto no sólo servirá para verificar la generación de los datos iniciales, sino que también pondrá a prueba el AHF escrito para este trabajo (el siguiente Capítulo 3 detalla la teoría detrás de este AHF).

2. Realizar una evolución de las ondas de Brill donde haya un colapso subsecuente y que el comportamiento del lapso, horizonte, etc. sean consistentes con [5].

Sin embargo, otra intención es poder realizar estas simulaciones con mejoras como pueden ser mayor resolución, mayor tiempo de evolución y mayor accesibilidad con respecto al equipo de cómputo en donde corre dicha simulación.

En los siguientes capítulos se detallan las herramientas y bloques que forman el programa escrito para poder cumplir estos objetivos.

Capítulo 3

Resolvedor elíptico

3.1. Discretización en diferencias finitas

La idea central del resolvedor elíptico de este trabajo consiste en discretizar la ecuación de derivadas parciales en diferencias finitas para formar un sistema de ecuaciones lineales. Sin embargo, antes de plantear dichas ecuaciones, hay que detallar la geometría de la malla sobre la que se va a trabajar.

3.1.1. Malla computacional

Existen una gran clase de mallas cuando se resuelve un problema numéricamente. Hay las llamadas AMR's (de adaptive mesh refinement en inglés) que aumentan su resolución en áreas de interés y la disminuyen en otras para poder cubrir un mayor dominio computacional. Otras mallas compactifican el espacio para poder insertar al infinito en un punto de la memoria. Sin embargo, para este trabajo se usa la clase más sencilla de mallas que son las cartesianas de paso fijo. Puesto que se tiene un problema bidimensional en las coordenadas ρ, z, la malla consiste en un arreglo cartesiano en donde cada punto está separado por un paso espacial h en cada dirección[1]. Específicamente las coordenadas de un punto están dada por dos enteros i, j tal que

$$(\rho_i, z_j) \equiv (\rho(i), z(j))\,. \tag{3.1}$$

Para que el paso entre los puntos sea fijo, ρ_i, z_j deben ser funciones lineales de i, j respectivamente y basta con determinar un punto por el que pasan. Es posible que la primera idea que surja es hacer que las funciones pasen por el cero y se tenga algo de la forma $\rho_i = ih$. Sin embargo, esto lleva a problemas numéricos cuando se divide entre las coordenadas, porque causa una división entre cero que si bien puede tener un límite definido analíticamente, numéricamente llevará a comportamiento indeterminado. Por lo tanto, se usa una malla *alternada* donde se pasa por los puntos $\pm h/2$, evitando al cero efectivamente. En conclusión, las funciones coordenadas toman la forma

$$\rho(i) = ih - h/2\,, \quad z(j) = jh - h/2\,, \tag{3.2}$$

con i, j enteros.

[1]El paso puede ser variable en cada dirección, i.e. $\Delta\rho$ y Δz. Sin embargo, para simplificar la discusión y no llenar de símbolos para cada caso, se asume que el paso es fijo en las dos direcciones. También hay que tener cuidado de no confundir h con la métrica $\hat{\gamma}_{\phi\phi}/\rho^2$ pero esto no debe ser difícil puesto en contexto.

3.1.2. Ghost zones

La existencia de posibles puntos negativos $\rho = -h/2, -3h/2, \ldots$ al tomar enteros como $i = 0, -1, -2$ no sólo sirve para evitar problemas numéricos sino que también se utilizan para fijar condiciones de simetría. Como se mencionó en el Capítulo 1, para el caso de estudio de simetría axial, toda función $u(\rho, z)$ debe tener una paridad establecida en el eje $\rho = 0$. Los puntos negativos no tienen en realidad sentido pues el rango físico es $\rho \in [0, \infty)$. La gracia que tienen es que sirven como condiciones de frontera o constricciones sobre la función u. En primera instancia no se ve de inmediato como esto limita a u, ya que uno puede pensar que sólo se utilizan sus valores en los puntos positivos a la hora de hacer cálculos numéricos. Sin embargo, cuando se calculan derivadas con diferencias finitas tenemos que garantizar que si la función es par $\frac{\partial u}{\partial \rho}(\rho = 0) = 0$ y en contraparte, que una función impar cumpla $u(\rho = 0) = 0$ y $\frac{\partial^2 u}{\partial \rho^2}(\rho = 0) = 0$. Si se utiliza ingenuamente una aproximación de diferencias finitas ladeada esto puede no cumplirse, lo cual empezará a acumular errores en pocos pasos de simulación. La existencia de los puntos negativos parece casi redundante (y por eso se llaman *ghost zones* en inglés) pero sirven para restringir las funciones a su forma física correcta.

Si tenemos una aproximación de diferencias finitas a segundo orden, la primera y segunda derivada centrada utilizan un punto a la izquierda y derecha del punto en dónde se está evaluando como se puede ver en sus fórmulas explícitas

$$\begin{aligned} \frac{\partial u}{\partial \rho}(\rho_i, z_j) &= \frac{u(\rho_{i+1}, z_j) - u(\rho_{i-1}, z_j)}{2h} + \mathcal{O}(h^2)\,, \\ \frac{\partial^2 u}{\partial \rho^2}(\rho_i, z_j) &= \frac{u(\rho_{i+1}, z_j) - 2u(\rho_i, z_j) + u(\rho_{i-1}, z_j)}{h^2} + \mathcal{O}(h^2)\,. \end{aligned} \tag{3.3}$$

(*Nota*: De ahora en adelante se usará la notación reducida $u_{i,j}$ para representar a $u(\rho_i, z_j)$.) Por lo tanto, si queremos la primera y segunda derivada de una función en el punto $\rho = h/2$, necesitamos un ghost zone correspondiente a $\rho = -h/2$. De manera similar, para una aproximación a cuarto orden se requieren ahora dos ghost zones en $\rho = -h/2, -3h/2$ para poder calcular las derivadas considerando la simetría axial.

Para el problema de este trabajo no sólo se tiene simetría axial sino también simetría ecuatorial, lo cual quiere decir que las funciones también tienen un comportamiento definido bajo la reflexión por el ecuador $z \to -z$. Entonces, también se pueden implementar ghost zones en z, lo cual sirve para reducir efectivamente el dominio computacional a la mitad, lo que se traduce a un menor costo computacional.

Denotando por g el número de ghost zones apropiado para el orden de aproximación en diferencias finitas, la forma de las coordenadas se vuelve

$$\rho(i) = (i - g)h - h/2\,, \quad z(j) = (j - g)h - h/2\,, \tag{3.4}$$

donde se ha instaurado la convención de que $\rho(1)$ es el primer punto de la malla con $\rho(1) = -(2g - 1)h/2$ (*i.e. los índices empiezan en 1*). Además se sigue que $\rho(g + 1) = h/2$ es el primer punto interno de la malla (y vice versa para z).

Como es natural para una malla computacional, los índices i, j deben tener un límite finito. Se suele caracterizar una malla por el número de puntos internos, denotados por N_ρ para ρ y N_z para z. En adición a estos puntos internos, se pone un último punto en la frontera externa donde se trabajarán las condiciones de frontera externa. Por lo tanto, la malla de trabajo tiene $g + N_\rho + 1$ puntos en ρ y $g + N_z + 1$ puntos en z y queda dibujada en la siguiente Figura 3.1.

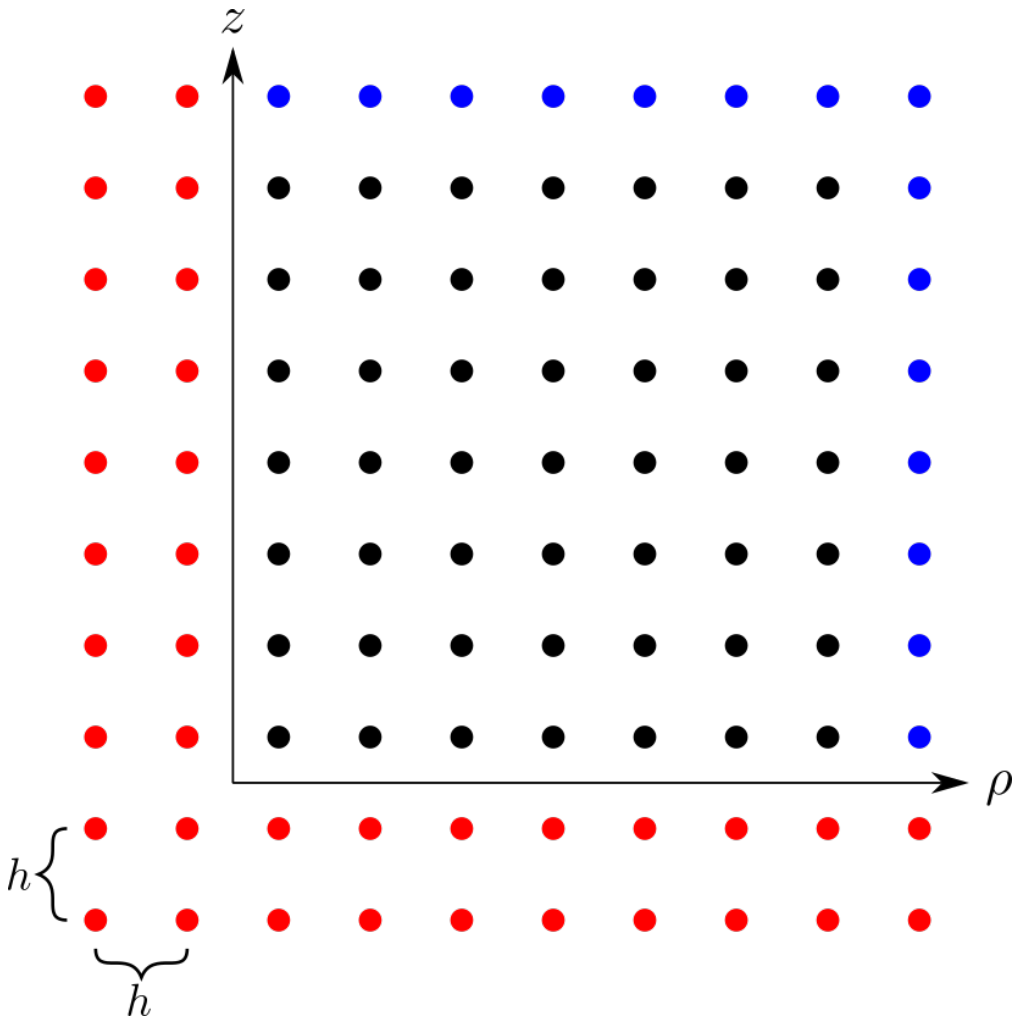

Figura 3.1: Estructura de la malla computacional usada. El espaciamiento es de h uniformemente en las dos direcciones. En cada dimensión hay $g = 2$ ghost zones que se dibujan en rojo (correspondientes a trabajar a cuarto orden). Los puntos asociados a la frontera externa son azules y para este ejemplo, hay $N_\rho = N_z = 7$ puntos internos negros. Por lo tanto, la malla es de $2 + 7 + 1 = 10$ puntos en cada dirección.

3.1.3. Sistema lineal $Au = f$

Ahora que se ha explicado la forma de la malla se puede discretizar la ecuación de derivadas parciales en diferencias finitas. Para formar un sistema lineal, la ecuación obviamente debe ser lineal desde su planteamiento. Esta limitación no es inconveniente para el problema de estudio de este trabajo, ya que las dos ecuaciones elípticas a resolver: de lapso maximal (1.41) y de datos iniciales de ondas de Brill (2.9) son perfectamente lineales. De todos modos, es relevante hacer la observación de que este resolvedor no funcionará con ecuaciones no-lineales.

Para formar un sistema completo de ecuaciones necesitaremos una ecuación por cada punto de la malla, es decir $(N_\rho + g + 1)(N_z + g + 1)$ ecuaciones. Para las ghost zones la condición de paridad en el eje ρ se establece como

$$u_{g-i,j} - p_\rho \, u_{g+1+i,j} = 0 \,, \tag{3.5}$$

donde $i = 1, 2, \ldots, g$ y p_ρ es una constante que denota la paridad en ρ, i.e. $p_\rho = 1$ para una función par y $p_\rho = -1$ para una impar. Del mismo modo, las ghost zones debajo del ecuador obedecen una ecuación análoga

$$u_{i,g-j} - p_z\, u_{i,g+1+j} = 0\,. \tag{3.6}$$

Finalmente, para la diagonal se utiliza una reflexión por los dos ejes

$$u_{g-i,g-j} - p_\rho\, p_z\, u_{g+1+i,g+1+j} = 0\,. \tag{3.7}$$

Con este conjunto de ecuaciones manifiestamente lineales se establece la paridad de la solución. En los puntos internos $i \in [g+1, g+N_\rho]$, $j \in [g+1, g+N_z]$ se utiliza la discretización de la ecuación elíptica. Aquí va a depender de a qué orden se están trabajando las diferencias finitas. Por ejemplo, para segundo orden, el laplaciano plano junto con una fuente lineal s se discretizan como

$$\begin{aligned}\mathring{\nabla}^2\, u + s\, u \to \left(-\frac{4}{h^2} + s_{i,j}\right)\, u_{i,j} + \frac{1}{h^2}\, (u_{i,j+1} + u_{i,j-1}) \\ + \left(\frac{1}{h^2} - \frac{1}{2h\rho_{i,j}}\right)\, u_{i-1,j} + \left(\frac{1}{h^2} + \frac{1}{2h\rho_{i,j}}\right)\, u_{i+1,j}\,.\end{aligned} \tag{3.8}$$

En la práctica, se suele multiplicar todo por un factor de h^2 para trabajar con cantidades adimensionales. La llamada molécula o plantilla de este esquema queda dibujada en la Figura 3.2.

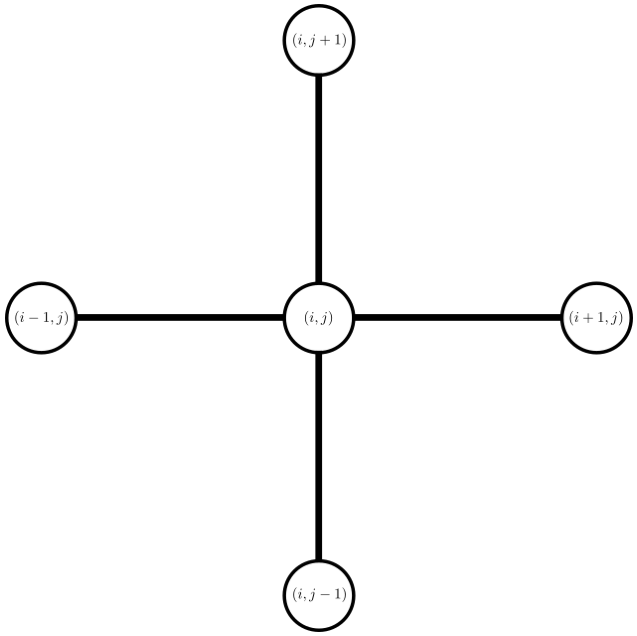

Figura 3.2: Plantilla de la discretización a segundo orden de $\mathring{\nabla}^2\, u + s\, u$.

Es fácil convencerse de que esta plantilla puede generar una discretización en todos los puntos internos de la malla: en los lados izquierdo e inferior se completa con el ghost zone de simetría y en los lados derecho y superior se completa con el punto adicional de frontera. Es decir, a segundo orden se puede utilizar una sola plantilla, cosa que no es verdad para cuarto orden. Esto se debe a que en cuarto orden la plantilla ahora se extiende cuatro puntos más que la anterior (uno en cada dirección).

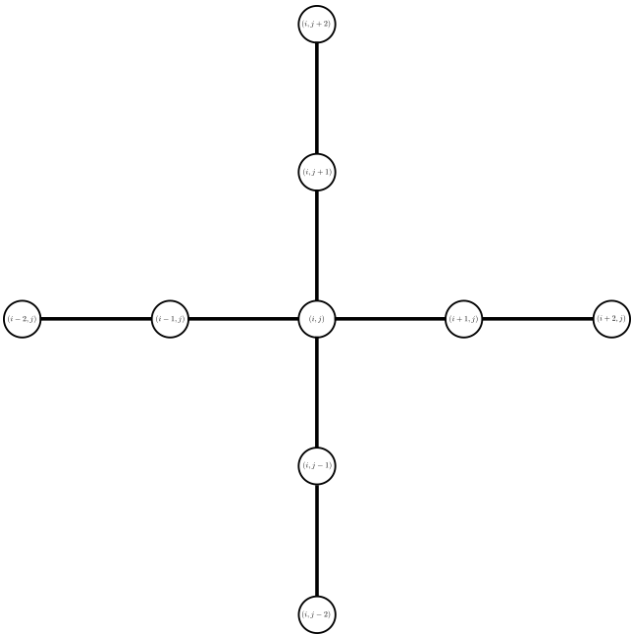

Figura 3.3: Plantilla de la discretización a cuarto orden de $\mathring{\nabla}^2 u + s\, u$.

Por lo tanto, esta plantilla cabe en los lados izquierdos e inferior gracias a las dos ghost zones, pero no cabe en los lados superior y derecho a falta de un punto adicional de frontera externa. Esto no es un problema catastrófico pues sólo indica que en estos puntos es necesario utilizar una aproximación semi-ladeada para las diferencias finitas para poder insertar la plantilla.

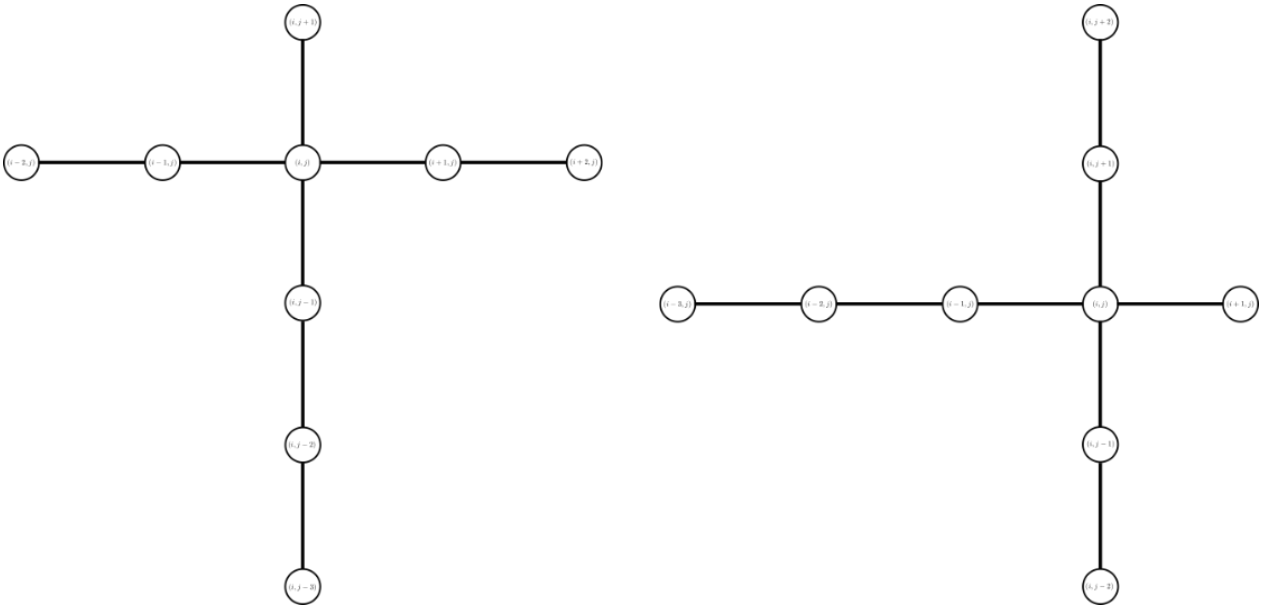

Figura 3.4: Plantillas de cuarto orden semi-ladeadas.

Finalmente, se debe poner una ecuación para la frontera externa. Aquí el resolvedor tiene tres distintas opciones:

1. **Dirichlet:** La frontera externa es un arreglo de valores fijos proporcionados por el usuario

$u^*_{i,j}$ y la ecuación lineal en los puntos de frontera se reduce a

$$u_{i,j} = u^*_{i,j}\,, \tag{3.9}$$

con al menos uno de los índices cumple $i = g + N_\rho + 1$ ó $j = g + N_z + 1$ para estar en la frontera.

2. **Neumann:** Para este caso se da el valor de la derivada normal a la frontera. Lo cual quiere decir que el usuario debe establecer un arreglo de valores $u^{\rho*}_j$ para $\frac{\partial u}{\partial \rho}$ en la frontera derecha y otro arreglo de valores u^{z*}_i para $\frac{\partial u}{\partial z}$ en la frontera superior. Por ejemplo, para segundo orden se tiene en la frontera derecha

$$\frac{3}{2h}\,u_{g+N_\rho+1,j} - \frac{2}{h}\,u_{g+N_\rho,j} + \frac{1}{2h}\,u_{g+N_\rho-1,j} = u^{\rho*}_j\,, \tag{3.10}$$

donde se ha utilizado una derivada ladeada a segundo orden para $\frac{\partial u}{\partial \rho}$. El caso de la frontera superior es análogo al discretizar la derivada $\frac{\partial u}{\partial z}$. Sin embargo, para la esquina lo que se hace es tomar la derivada en la dirección diagonal, i.e.

$$\frac{3}{2h'}\,u_{g+N_\rho+1,g+N_z+1} - \frac{2}{h'}\,u_{g+N_\rho,g+N_z} + \frac{1}{2h'}\,u_{g+N_\rho-1,g+N_z-1} = u'\,, \tag{3.11}$$

donde $h' = \sqrt{2}\,h$ (ó $\sqrt{(\Delta\,\rho)^2 + (\Delta\,z)^2}$ si la malla es de paso variable) y $u' = u^{\rho*}_{g+N_z+1} = u^{z*}_{g+N_\rho+1}$.

3. **Robin:** La última condición es la más importante para este problema y viene de pedir que la solución u pueda ser expandida como

$$u = u_\infty + \frac{v_1}{r} + \mathcal{O}(r^{-2})\,, \tag{3.12}$$

donde u_∞ es el valor constante en infinito de la solución y v_1 es otra constante que no juega importancia en el análisis por ahora. A partir de este comportamiento asintótico, se deduce que

$$\frac{\partial u}{\partial r} + \frac{u - u_\infty}{r} = \mathcal{O}(r^{-3})\,. \tag{3.13}$$

En la práctica, esta última expresión se toma como igual a cero en la frontera externa y el usuario sólo necesita decir cuál es la constante u_∞. El único problema es que se trata de una derivada radial y la malla no está equipada para calcularlas directamente. Consecuentemente, hay que hacer otra aproximación que es asumir que la solución no tiene dependencia angular en la frontera, i.e. $\frac{\partial u}{\partial \theta} \approx 0$. Entonces, de la regla para la cadena

$$\frac{\partial u}{\partial \rho} = \frac{\partial r}{\partial \rho}\,\frac{\partial u}{\partial r} + \frac{\partial \theta}{\partial \rho}\,\frac{\partial u}{\partial \theta} \approx \frac{\partial r}{\partial \rho}\,\frac{\partial u}{\partial r} = \frac{\rho}{r}\,\frac{\partial u}{\partial r}\,, \tag{3.14}$$

se concluye que

$$\frac{\partial u}{\partial r} \approx \frac{r}{\rho}\,\frac{\partial u}{\partial \rho}\,, \tag{3.15}$$

y análogamente para la otra frontera

$$\frac{\partial u}{\partial r} \approx \frac{r}{z}\,\frac{\partial u}{\partial z}\,. \tag{3.16}$$

Así la ecuación en las fronteras se vuelve

$$\frac{r^2}{x}\,\frac{\partial u}{\partial x} + u = u_\infty\,, \tag{3.17}$$

con $x = \{\rho, z\}$ según se trate de la frontera derecha o superior, respectivamente. Para el ejemplo de segundo orden, la frontera derecha tendrá la discretización

$$\left(1 + \frac{3r^2}{2h\rho}\right) u_{g+N_\rho+1,j} - \frac{2^2}{h\rho}\,u_{g+N_\rho,j} + \frac{r}{2h\rho}\,u_{g+N_\rho-1,j} = u_\infty\,, \tag{3.18}$$

donde se han abreviado $r^2 = r^2_{g+N_\rho+1,j}$, $\rho = \rho_{g+N_\rho+1,j}$. Al igual que en la condición de Neumann, la condición de la frontera superior es análoga y la de la esquina sigue discretizando sobre la diagonal con un paso $h' = \sqrt{2}\,h$.

Antes de continuar se hace la observación de que las condiciones de frontera anteriores introducen un término inhomogéneo al sistema lineal $Au = f$ que está dentro del lado derecho f. Si existía un valor previo para f en estos puntos, es importante que sea reemplazado por el nuevo valor derecho correspondiente a la condición de frontera apropiada.

3.2. Análisis del error para una frontera de Robin

La condición de frontera de Robin resulta ser absolutamente fundamental para los dos problemas que se resuelven en este trabajo, en particular vale la pena ahondar en el posible error que introduce en la solución numérica.

El error en cuestión no viene del típico error de diferencias finitas sino del hecho mucho más fundamental que es que la condición de Robin es una aproximación asintótica. Visto de otro modo, la pregunta esencial es ¿qué tan lejos se debe poner la frontera externa para que la aproximación de Robin sea suficientemente buena como para introducir un error menor a una tolerancia determinada?

Dicha pregunta no es tan sencilla de responder aunque uno podría guiarse primero por la ecuación (3.13) donde se ve que el lado derecho de la condición de Robin es $\mathcal{O}(r^{-3})$, así que si llamamos r_∞ al menor radio de la frontera externa de la malla, es posible asumir que el error introducido sería $\mathcal{O}(r_\infty^{-3})$. Sin embargo, esto no es cierto ya que el carácter de este error es local y cuando se calculan errores, los errores locales pueden acumularse a otro orden. Pero al menos podemos ver que la cota mínima para el error será $\mathcal{O}(r^{-3})$.

El problema de resolver ecuaciones elípticas usando condiciones de frontera asintóticas no es nada nuevo y de hecho sí existen resultados que pueden guiarnos. En particular, el estudio de estabilidad y convergencia para condiciones de Robin en dos dimensiones utilizando un método de elementos finitos (FEM) ha sido estudiado previamente en [42] y [32]. Dichos resultados nos proporcionan una base para poder comprobar que el problema está bien planteado: como es de esperar intuitivamente, ya que se puede probar para el problema de Poisson (véase por ejemplo [11], [36] y [51]) que las condiciones de frontera de Dirichlet y Neumann son estables y bien-planteadas, y Robin se puede ver como una mezcla de ambas. Sin embargo, no se pueden usar todos los resultados de estos trabajos, principalmente debido a que el método de este resolvedor es de diferencias finitas y no FEM's.

Para un resultado que aplique directamente a diferencias finitas, se sigue el trabajo pionero de Bayliss *et al.* que introduce en [16] un método para no sólo calcular el orden del error sino también presentan indicaciones de cómo construir condiciones de frontera que tengan un orden de error variable. Es decir, se puede reescribir la condición de Robin (3.13) usando operadores diferenciales de mayor orden, teniendo como consecuencia indirecta que ya no es necesario tener la frontera externa tan alejada.

Dicho análisis está limitado a las ecuaciones de Poisson y Helmholtz pero también se puede extender a problemas más complicados para los cuales el comportamiento cerca de las fronteras se vuelve esencialmente Poisson o Helmholtz. Por ejemplo, para los datos iniciales de ondas de Brill, sabemos que el operador diferencial (el laplaciano plano $\mathring{\nabla}^2$) se comporta de manera idéntica al operador diferencial de Poisson. Para saber qué tipo de ecuación se tiene en las fronteras, hay que estudiar el término lineal que en el caso de usar una q tipo Holz (2.17) tiene la forma analítica

$$s = \frac{1}{4}\left(\frac{\partial^2 q}{\partial \rho^2} + \frac{\partial^2 q}{\partial z^2}\right) = a e^{-r^2}\left(\frac{1}{2} + \rho^2\left(r^2 - 3\right)\right). \tag{3.19}$$

Es claro que este término lineal decae rápidamente a cero gracias a e^{-r^2}. De hecho, como regla empírica: usando amplitudes $a \in [0, 12]$, el valor absoluto de s es menor a 10^{-15} para $r > 8$ y para estos radios se tiene esencialmente un problema de Laplace debido a que el lado derecho es homogéneo. La ecuación de Laplace $\mathring{\nabla}^2 u = 0$ es un caso especial de Poisson, así que el análisis puede proceder.

Para la ecuación de lapso maximal (1.41), el termino lineal puede reducirse vía la constricción hamiltoniana al escalar de Ricci R. Si tenemos un espacio asintóticamente plano y en vacío, también será cierto que lejos la ecuación también se vuelve Laplace, aunque en este caso no es posible deducir un valor empírico para el cual empieza a suceder este comportamiento. Pese a ésto, se puede asumir que debe ser un radio similar al radio de decaimiento de s ya que para un radio mayor el comportamiento dominante será tipo Schwarschild con una métrica física de la forma

$$\mathrm{d}l^2 = \left(1 + \frac{M}{2r}\right)^4 \left(\mathrm{d}r^2 + r^2\,\mathrm{d}\Omega^2\right) = e^{4\phi}\left(\mathrm{d}r^2 + r^2\,\mathrm{d}\Omega^2\right), \tag{3.20}$$

donde se ha definido $\phi = \log\left(1 + 2M/r\right)$. Entonces, el laplaciano físico puede ser calculado como

$$\nabla^2 = e^{4\phi}\left(\mathring{\nabla}^2 - 2\mathring{\gamma}^{kj}\left(\partial_j\,\phi\right)\partial_k\right). \tag{3.21}$$

con esta expresión, no es difícil convencerse de que asíntoticamente sí se tiene $\mathring{\nabla}^2\,\alpha = 0$ y también puede aplicar el análisis subsecuente.

Habiendo justificado que los dos problemas son asintóticamente Laplace, se puede proceder a detallar el análisis de Bayliss *et al.*, para el caso particular de Poisson/Laplace. El primer paso es trabajar con $w \equiv u - u_\infty$ y expandir en multipolos

$$w = \frac{1}{r}\sum_{j=0}^{\infty}\frac{F_j(\theta,\varphi)}{r^j}, \tag{3.22}$$

donde $F_j(\theta,\varphi)$ son combinaciones de armónicos esféricos (véase [11], por ejemplo). Como es bien sabido, esta es una solución a la ecuación de Laplace y de hecho se puede definir el operador de Robin como

$$B_1 \equiv \frac{\partial}{\partial r} + \frac{1}{r}, \tag{3.23}$$

de tal manera que la condición de Robin (3.13) se vuelve

$$B_1\, w\Big|_{r=r_\infty} = \mathcal{O}(r_\infty^{-3})\,. \tag{3.24}$$

(Recuérdese que r_∞ es el valor del radio más pequeño de la malla). Esto quiere decir que el operador B_1 aniquila el primer término en la expansión (3.22). La idea central de Bayliss *et al.* es pedir que se no sólo se aniquile el primer término sino poder pedir que se aniquilen m términos en la frontera externa r_∞. Esto se hace definiendo los operadores

$$B_m = \prod_{j=1}^{m} \left(\frac{\partial}{\partial r} + \frac{2j-1}{r} \right)\,. \tag{3.25}$$

Por ejemplo,

$$B_2 = \frac{\partial^2}{\partial r^2} + \frac{4}{r}\,\frac{\partial}{\partial r} + \frac{2}{r^2}\,. \tag{3.26}$$

Como B_m aniquila los primeros m términos se sigue que

$$B_m\, w\Big|_{r=r_\infty} = \mathcal{O}(r_\infty^{-(2m+1)})\,. \tag{3.27}$$

Todo esto es relevante debido a que el resultado final de Bayliss *et al.* es que si se tiene el problema de Poisson con una frontera externa en r_∞ para el que se ha planteado una condición de frontera $B_m w = 0$, el error ϵ producido por esta aproximación satisface

$$\|\epsilon\| = \mathcal{O}(r_\infty^{-(m+1)})\,, \tag{3.28}$$

para alguna norma definida. Por lo tanto, en la condición de Robin original, el error es $\mathcal{O}(r_\infty^{-2})$ y no $\mathcal{O}(r_\infty^{-3})$ como se pude haber pensado primero. El resultado nos permite entonces determinar dónde se debe poner la frontera r_∞ para el resolvedor elíptico. Supongamos que se está trabajando a segundo orden con una resolución h. Entonces, si se desea que el error de Robin sea del mismo orden h^2 se determina que

$$r_\infty = h^{-\frac{2}{m+1}}\,, \tag{3.29}$$

dará un error de orden h^2. Usando el valor $m = 1$ para la condición típica de Robin junto con una resolución de $h = 0.02$, sale que $r_\infty = 50$. El verdadero problema no es este valor de r_∞ sino el número de puntos interiores que implica: una malla de aproximadamente 2500×2500 puntos son necesarios para cumplir esta condición. Este sistema lineal es demasiado costoso computacionalmente, y se está resolviendo en regiones de muy poco interés: sobre todo porque gracias al análisis empírico, uno quisiera no extenderse mucho más allá de $r_\infty = 8$. Esto sólo empeora si se considera cuarto orden ya que ahora

$$r_\infty = h^{-\frac{4}{m+1}}\,, \tag{3.30}$$

implica que para la misma resolución de $h = 0.02$ se necesitan 125000×125000 puntos. Por lo tanto, si se quiere trabajar con altas resoluciones como es $h = 0.04, 0.02$, va a ser necesario implementar una condición de Robin con $m > 1$.

Analíticamente queda claro cómo hacer esto. Se pueden calcular en principio los operadores a cualquier m y discretizar en diferencias finitas. Sin embargo, numéricamente no es posible llegar a

cualquier m debido a que al operador B_m le corresponde una derivada $m-$ésima. La derivada dará una diferencia dividida entre h^m lo cual puede generar errores de truncamiento cuando h es muy pequeña o m demasiado grande.

En la práctica se encuentra que para segundo orden en las resoluciones altas, se puede implementar el operador B_3 con bastante éxito lo cual deja

$$r_\infty = h^{-1/2}\,, \tag{3.31}$$

que para $h = 0.02$ implica que $r_\infty = 7.1$ y una malla de aproximadamente 350×350 puntos. Entonces ahora la limitante ya no es la r_∞ calculada arriba sino la r_∞ donde se puede asumir que el comportamiento es tipo Laplace que se determinó empíricamente como $r_\infty = 8$ con una malla razonable de 400×400 puntos.

A pesar del éxito a segundo orden, los errores de truncamiento hacen imposible implementar el operador B_7 que llevaría a usar la misma r_∞ a cuarto orden que B_3 a segundo orden. De hecho, se encuentra que lo mejor que se puede hacer es implementar la misma B_3 que implica que

$$r_\infty = h^{-1}\,. \tag{3.32}$$

Por ejemplo, en el caso de cuartocon una resolución de $h = 0.03$ se sigue que $r_\infty = 33.3$ y que la malla es de 1100×1100 puntos.

3.3. Matrices Sparse

El sistema lineal $Au = f$ generado por la discretización de la ecuación elíptica puede ser un sistema enorme en término de la dimensión de la matriz. Esto es porque para la malla de trabajo u se ve como un *vector* con $n \equiv (g + N_\rho + 1)(g + N_z + 1)$ puntos, lo que quiere decir que A es una matriz con n^2 elementos. Para mallas de 400×400 puntos esto implica que A tiene 2.5×10^{10} elementos. Por lo tanto, el sistema lineal no puede ser resuelto con modos convencionales. De hecho, ni siquiera puede ser almacenado de modo convencional porque para el ejemplo anterior en doble precisión (8 bytes por número) implica alrededor de 200 Gb de almacenamiento.

La solución a estos problemas resulta del hecho de que gran mayoría de los elementos de la matriz A son ceros. Por ejemplo, a segundo orden cada ecuación se traduce en una fila de la matriz donde sólo cinco elementos son distintos de cero. Este patrón de ceros puede ser graficado, como sucede para la siguiente Figura 3.5.

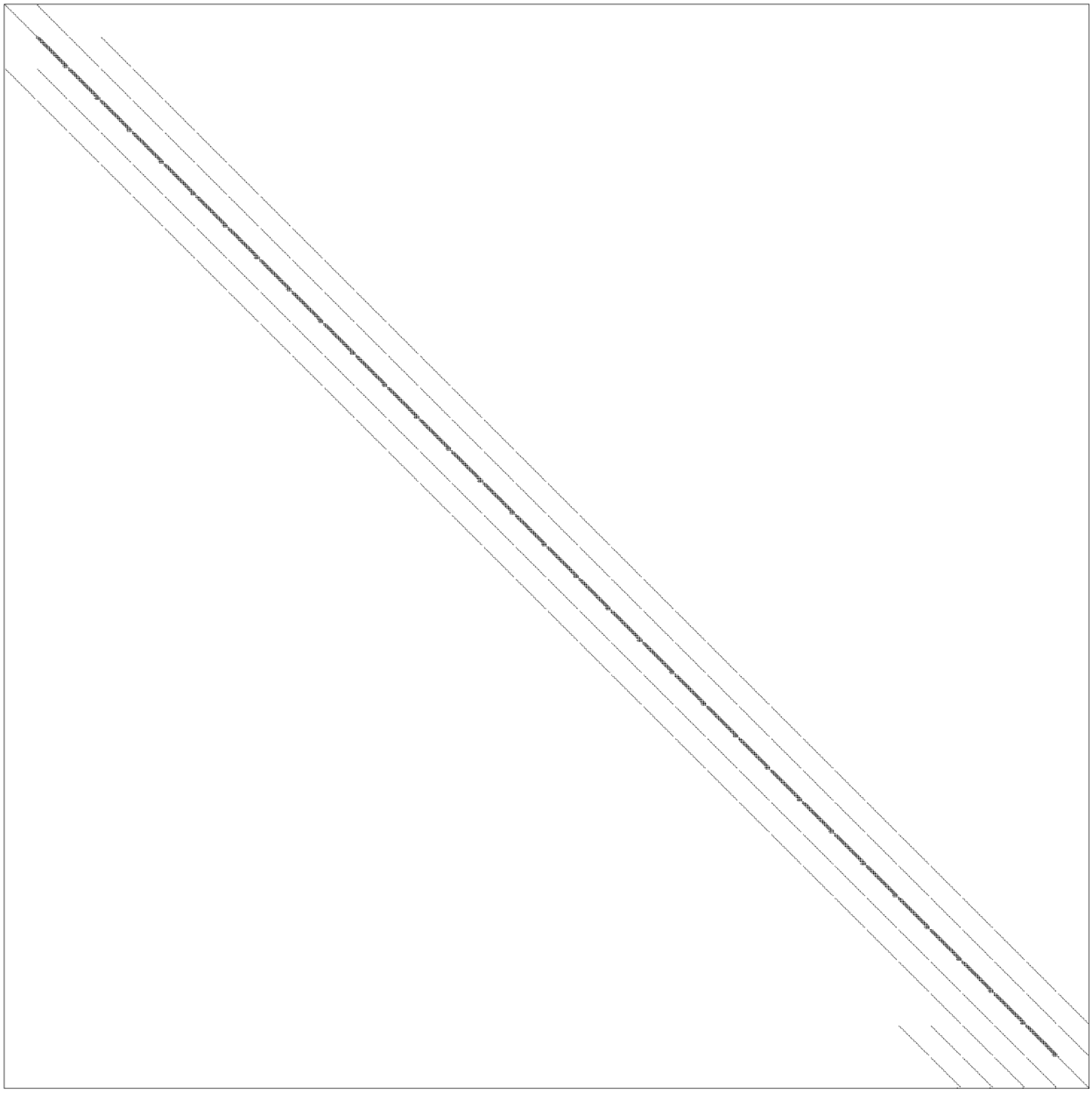

Figura 3.5: Gráfica de la matriz A resultante de discretizar el sistema de datos iniciales de ondas de Brill a cuarto orden y con condición de Robin B_1. Los elementos distintos de cero son presentados como pixeles negros y los ceros son el espacio en blanco. El sistema considera una malla muy pequeña de 32×32 que no se usaría en la práctica pero aquí sí porque para mallas mayores la visualización es totalmente impráctica.

Los sistemas que tienen este comportamiento son denominados *sparse* en inglés[2] y debido a que la mayoría de las discretizaciones de diferencias finitas producen un sistema sparse, la solución de dichos problemas es de crucial importancia. En las siguientes secciones se presenta el formato de almacenamiento usado para este resolvedor, así como el algoritmo de cómo calcular el número de

[2]Una traducción válida al español puede ser 'poco denso' o 'esparcido'. Se espera que no sea molesto para el lector que se use dicho anglicismo con el fin de mantener la discusión fluida.

elementos distintos de cero. Después se procede a detallar el método de solución relevante a este trabajo. La teoría detrás de estos métodos, así como de las matrices sparse en general es bastante extensiva y puede ser consultado a mayor detalle en [51], [38], [57], entre otros.

3.3.1. Formato CSR

El primer problema cuando se lidia con sistemas lineales con tantas ecuaciones es cómo almacenarlo en memoria. Dicho método debe estar optimizado para no tener que guardar los ceros irrelevantes en memoria. Es fácil ver que la mínima cantidad de información requerida para construir el sistema debe ser un arreglo que guarde los elementos distintos de cero y además se complemente con una manera de construir los índices de dicho elemento. Por ejemplo, se podría trabajar con un arreglo de los valores y dos arreglos de enteros: uno con el índice de columna y otro con el índice de fila. Este formato se denomina COO (de *coordinate* en inglés).

Sin embargo, el formato COO no está realmente optimizado para grandes matrices (véase [51]), por lo que, en la práctica se utiliza el formato CSR (de *compressed sparse row*). Primero se asume que se tiene un matriz A de m filas por n columnas. Además, la matriz tiene n_{nz} número de elementos distintos de cero (las siglas nnz abrevian *number of nonzeros*, i.e. número de no-ceros). Entonces, el formato CSR consiste en tres arreglos:

- El arreglo $A.a$ de longitud n_{nz} que almacena todos los elementos de A distintos de cero leídos de izquierda a derecha recorriendo las filas hacia abajo (esto se suele indicar como que la matriz está ordenada por filas primero). Es decir, la matriz

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \tag{3.33}$$

  se lee como $A.a = (1,\ 2,\ 3,\ 4,\ 5,\ 6)$.

- También se requiere el arreglo $A.j$ de longitud n_{nz} igualmente. Dicho arreglo contiene el índice correspondiente a la columna de cada elemento de $A.a$.

- Finalmente, el arreglo $A.ia$ de longitud $m+1$. Por tanto, hay un elemento por cada fila de A y uno adicional. La última entrada de $A.ia$ es igual a $n_{nz}+1$. El resto de entradas de $A.i$ denotan qué número de elemento de $A.a$ empieza la fila m-ésima de A.

La experiencia dice que la mejor manera de entender cómo funciona el formato CSR es con un ejemplo pequeño. Pero antes se advierte que *los índices de los arreglos empiezan en 1.* Entonces, sea la matriz

$$A = \begin{pmatrix} 0 & 9 & 3 & 0 \\ 4 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 2 & 0 & 0 & 8 \\ 0 & 8 & 0 & 7 \end{pmatrix}. \tag{3.34}$$

Es claro que esta matriz tiene $n_{nz} = 9$ elementos distintos de cero y que tiene $m = 5$ filas y $n = 4$ columnas. Lo primero que se puede hacer es escribir el arreglo de tamaño n_{nz} correspondiente a los no-ceros:

$$A.a = \begin{pmatrix} 9, & 3, & 4, & 1, & 5, & 2, & 8, & 8, & 7 \end{pmatrix}. \tag{3.35}$$

Después se determina la columna de cada elemento de $A.a$, por ejemplo, $A.a(1) = 9$ tiene la columna 2, $A.a(7) = 8$ la columna 4, etc. Entonces:

$$A.ja = \begin{pmatrix} 2, & 3, & 1, & 2, & 3, & 1, & 4, & 2, & 4 \end{pmatrix}. \tag{3.36}$$

Por último hay que ver dónde empiezan las filas de A. La primera fila tiene como primer elemento a 9 que es el elemento 1. Entonces $A.ia(1) = 1$. Luego, la segunda fila empieza en el 4 que es el elemento número 3, así que $A.ia(2) = 3$. Análogamente se concluye

$$A.ia = \begin{pmatrix} 1, & 3, & 5, & 6, & 8, & 10 \end{pmatrix}. \tag{3.37}$$

Esto termina por explicar cómo funciona el formato CSR y se infiere que un problema central de este trabajo será escribir los sistemas matriciales de manera correcta y optimizada, cosa que se detalla en las siguientes secciones.

3.3.2. Cálculo de elementos distintos de cero

Con el conocimiento del formato CSR, es evidente que se requiere saber cuántos elementos no-ceros habrá en la discretización del sistema lineal. Esto va a depender de qué tipo de problema se está resolviendo y a qué orden se está resolviendo. En su forma final, el resolvedor puede tratar a segundo y cuarto orden un laplaciano plano

$$\left(\frac{\partial^2}{\partial \rho^2} + \frac{\partial^2}{\partial z^2} + \frac{1}{\rho} \frac{\partial}{\partial \rho} + s(\rho, z) \right) u(\rho, z) = f(\rho, z)\,, \tag{3.38}$$

así como una ecuación elíptica general

$$\left(a(\rho, z) \frac{\partial^2}{\partial \rho^2} + b(\rho, z) \frac{\partial^2}{\partial \rho \partial z} + c(\rho, z) \frac{\partial^2}{\partial z^2} \right.$$
$$\left. + d(\rho, z) \frac{\partial}{\partial \rho} + d(\rho, z) \frac{\partial}{\partial z} + s(\rho, z) \right) u(\rho, z) = f(\rho, z)\,. \tag{3.39}$$

Para este trabajo, se presenta el análisis del laplaciano plano a segundo orden porque es el más corto de todos y muestra todos los elementos necesarios para entender los otros casos más generales.

Usando la plantilla/molécula de la Figura 3.2, junto con la discretización de (3.8), se deduce que cada punto interno de la malla va a generar cinco elementos en el sistema lineal. Después, las condiciones de simetría (3.5), (3.6) y (3.7) dicen que las ghost zones ocupan dos elementos. Finalmente, al discretar Robin B_1 como está en (3.18), se sigue que cada punto en la frontera externa tendrá tres elementos asociados en la matriz (incluyendo a la esquina donde la discretización es diagonal). Si se usa el operador B_2 serán cuatro elementos por punto y similarmente B_3 tiene cinco elementos. Entonces, si n_{R} denota el número de operador de Robin a utilizar ($n_{\mathrm{R}} = 1, 2, 3$), se puede escribir la forma de n_{nz} para este problema:

$$n_{nz} = 5\, N_\rho\, N_z + (4 + n_{\mathrm{R}})\ (N_\rho + N_z) + (8 + n_{\mathrm{R}})\,. \tag{3.40}$$

Esta ecuación justifica por completo que la matriz sea sparse, por ejemplo, si $N_\rho = N_z = 1000$, los elementos no-cero del sistema lineal resultante son sólo $0.0005\,\%$ del número total de elementos en el sistema denso.

3.3.3. Algoritmo de llenado

A continuación hay que detallar cómo se llenan los tres arreglos asociados al formato CSR $A.a$, $A.ia$, $A.ja$. El cuidado especial en llenar los arreglos viene de seguir una serie de convenciones y de ordenamientos. Primero, aunque tenemos una malla u bidimensional, sabemos que a la hora de introducirla al sistema lineal $Au = f$, la malla se toma como un vector unidimiensional. Por lo tanto, la primera convención que se estipula es que el arreglo en memoria $u_{i,j}$ (donde i es el índice asociado a ρ y j el asociado a z) está ordenado de tal manera que j es el índice rápido. En otras palabras, $u_{i,j}$ está ordenado primeramente por ρ y así en memoria lineal puede ser visto como

$$u = \left(u_{1,1},\, u_{1,2},\, u_{1,3}, \ldots u_{1,g+N_z+1},\, u_{2,1},\, u_{2,2},\, \ldots u_{g+N_\rho+1,g+N_z+1}\right). \tag{3.41}$$

La manera de convertir un índice bidimensional a uno lineal es bien conocida y para este caso se puede nombrar dicho índice k:

$$k(i,j) = (i-1)\,(g+N_z+1)+j \quad \Rightarrow \quad i = 1+\left\lfloor \frac{k}{g+N_z+1} \right\rfloor, \quad j = k \mod (g+N_z+1). \tag{3.42}$$

Con base en el índice k, cada punto i, j determina una ecuación lineal según sea un ghost zone, un punto interno, o un punto de frontera externa. Por lo tanto, al llenar los arreglos CSR empezamos con el punto $1, 1$ que es un ghost zone. Este punto genera dos elementos vía (3.7): $A.a(1) = 1$ y $A.a(2) = -p_\rho\, p_z$ que están asociados a los índices $k(1,1) = 1$ y $k(g+1, g+1) = g(g+N_z+1)+g+1$, respectivamente, de donde se deduce que $A.ja(1) = 1$ y $A.ja(2) = g(g+N_z+1)+g+1$. Puesto que esta es la primera fila del sistema lineal y empieza en el primer elemento de $A.a$, $A.ia(1) = 1$. El algoritmo de llenado se esquematiza en forma resumida/seleccionada en el siguiente listado. El código entero puede consultarse en [44]. Nótese especialmente que el algoritmo está paralelizado con *OpenMP* (véase [23]), para tener un llenado más rápido.

```
// Incluye parámetros globales y declaraciones.
// ...
void csr_gen_flat_laplacian_robin(csr_matrix A, // Matriz sparse.
  const int NrInterior, // Número de puntos internos en r.
  const int NzInterior, // Número de puntos internos en z.
  const int order,    // Orden a trabajar: 2 o 4.
  const double dr,    // Paso espacial en r.
  const double dz,    // Paso espacial en z.
  const double *s,    // Termino lineal.
  double *f,        // Lado derecho.
  const double uInf,    // Valor en infinito para Robin.
  const int robin,    // Tipo de operador de Robin: 1, 2, 3.
  const int r_sym,    // Simetría en r: -1, 1.
  const int z_sym)    // Simetría en z: -1, 1.
{
  // Variables auxiliares.
  // ...
  // Calcula nnz.
  int nnz = flat_laplacian_nnz(NrInterior, NzInterior, order);
  // Cuantos elementos se han llenado. Cero al inicio.
  int offset = 0;

  // Matriz a segundo orden.
  if (order == 2) {
    // Simetría diagonal.
    // Fila IDX(1,1) = 1 empieza en elemento 1.
    A.ia[IDX(1, 1)] = 1 + offset;
    // Llena dos elementos.
    A.a[offset] = 1.0;
    A.a[offset + 1] = -(double)(r_sym * z_sym);
    // Indice de columnas.
    A.ja[offset] = IDX(1, 1);
    A.ja[offset + 1] = IDX(2, 2);
    // Lado derecho homogéneo.
    f[IDX(0, 0)] = 0.0;
    // Se han llenado dos elementos ahora.
    offset += 2;

    // Guarda variable offset temporal.
    int t_offset = offset;
    // Llena lado izquierda con simetría axial.
    // Paraleliza llenado con variable privada offset.
    #pragma omp parallel shared(A, f) private(offset) {
      #pragma omp for schedule(guided)
      // Recorre indice z con j = 2,...,NzInterior + 1
      for (j = 2; j < NzInterior + 2; j++) {
        // Cada iteración llena dos elementos.
        offset = t_offset + 2 * (j - 2);
        // Ecuación correspondiente a fila IDX(1,j).
        A.ia[IDX(1, j)] = 1 + offset;
        A.a[offset] = 1.0;
        A.a[offset + 1] = -(double)r_sym;
        A.ja[offset] = IDX(1, j);
        A.ja[offset + 1] = IDX(2, j);
        // Lado derecho homogéneo.
        f[IDX(1, j)] = 0.0;
      }
    }
    // Ahora se han llenado...
```

```
60      offset = 2 + 2 * NzInterior;
61
62      // Esquina superior izquierda: simetría axial.
63      A.ia[IDX(1, NzInterior + 2)] = 1 + offset;
64      A.a[offset] = 1.0;
65      A.a[offset + 1] = -(double)r_sym;
66      A.ja[offset] = IDX(1, NzInterior + 2);
67      A.ja[offset + 1] = IDX(2, NzInterior + 2);
68      f[IDX(1, NzInterior + 2)] = 0.0;
69      offset += 2;
70
71      // Variable temporal.
72      t_offset = offset;
73      // Ahora se llenan los puntos internos y las fronteras superiores e inferiores
        con simetría ecuatorial y Robin, respectivamente.
74      #pragma omp parallel shared(A, f) private(offset, j, r, z, ir, rr2, robin1,
        robin2, robin3) {
75        // Recorre el eje r.
76        #pragma omp for schedule(guided)
77        for (i = 2; i < NrInterior + 2; i++) {
78          // Cada iteración llena 5 * NzInterior + (2 + n_robin) valores.
79          offset = t_offset + (i - 2) * (5 * NzInterior + 2 + n_robin);
80          // Coordenada r.
81          r = (double)i - 1.5;
82          // Inverso.
83          ir = 1.0 / r;
84
85          // Punto inferior con simetría ecuatorial.
86          A.ia[IDX(i, 1)] = offset;
87          A.a[offset] = 1.0;
88          A.a[offset + 1] = -(double)z_sym;
89          A.ja[offset] = IDX(i, 1);
90          A.ja[offset + 1] = IDX(i, 2);
91          f[IDX(i, 1)] = 0.0;
92          offset += 2;
93
94          // Ahora se recorren los puntos internos.
95          for (j = 2; j < NzInterior + 2; j++) {
96            // Fila.
97            A.ia[IDX(i, j)] = offset;
98            // Valores.
99            A.a[offset] = 1.0 - 0.5 * ir;
100           A.a[offset + 1] = 1.0;
101           A.a[offset + 2] = dr * dz * s[IDX(i, j)] - 4.0;
102           A.a[offset + 3] = 1.0;
103           A.a[offset + 4] = 1.0 + 0.5 * ir;
104           // Columnas.
105           A.ja[offset] = IDX(i - 1, j);
106           A.ja[offset + 1] = IDX(i, j - 1);
107           A.ja[offset + 2] = IDX(i, j);
108           A.ja[offset + 3] = IDX(i, j + 1);
109           A.ja[offset + 4] = IDX(i + 1, j);
110           // Multiplica f por dr*dz para ser adimensional.
111           f[IDX(i, j)] *= dr * dz;
112           // Se llenan 5 elementos.
113           offset += 5;
114         }
115
116         // Luego viene la frontera con Robin.
```

```
117         j = NzInterior + 2;
118         // Coordenada z.
119         z = (double)j - 1.5;
120         // Coordenada radial.
121         rr2 = r * r + z * z;
122         // Fila.
123         A.ia[IDX(i, j)] = BASE + offset;
124         // Escoge operador de Robin.
125         switch (robin) {
126         case 1:
127           A.a[offset] = 0.5 * rr2 / z;
128           A.a[offset + 1] = -2.0 * rr2 / z;
129           A.a[offset + 2] = 1.0 + 1.5 * rr2 / z;
130           A.ja[offset] = IDX(i, NzInterior);
131           A.ja[offset + 1] = IDX(i, NzInterior + 1);
132           A.ja[offset + 2] = IDX(i, NzInterior + 2);
133           break;
134         //...
135         }
136         // Lado derecho se sustituye.
137         f[IDX(i, j)] = uInf;
138       }
139     }
140     // En este punto se han llenado...
141     offset = 4 + 2 * NzInterior + 5 * NrInterior * NzInterior + (2 + n_robin) *
      NrInterior;
142     // Esquina inferior derecha con simetría ecuatorial.
143     // ...
144     offset += 2;
145     // ...
146     // Frontera derecha con Robin.
147     // ...
148     // En este punto se han llenado...
149     offset = 6 + (2 + n_robin) * (NzInterior + NrInterior) + 5 * NrInterior *
      NzInterior;
150     // Esquina superior derecha con Robin.
151     // ...
152     offset += n_robin;
153     // Ultimo elemento es el numero de no-ceros.
154     A.ia[IDX(NrInterior + 2, NzInterior + 2) + 1] = 1 + nnz;
155   }
156 }
```

Ahora que se ha visto cómo se llena el sistema lineal, se puede presentar la Figura 3.6 como complemento a la Figura 3.5. Aquí se ve la parte central del sistema que corresponde a las líneas 74-139 del listado de arriba.

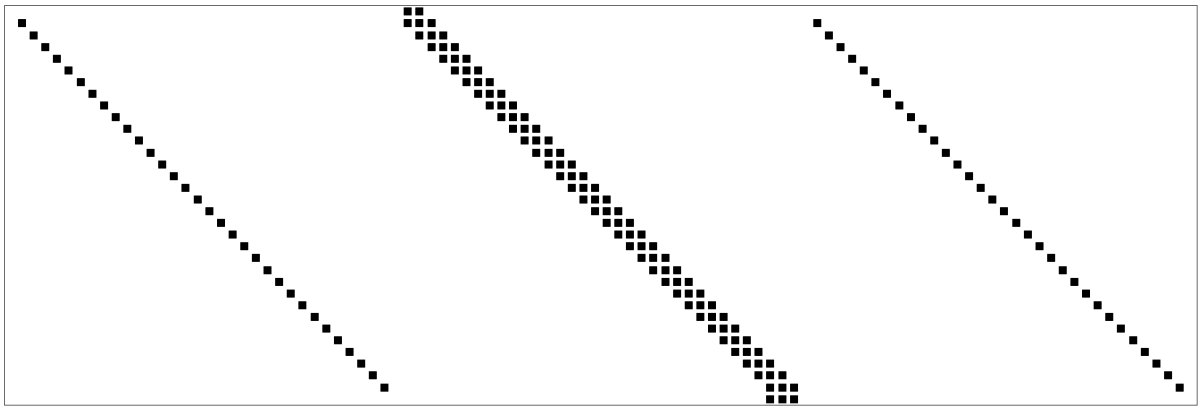

Figura 3.6: Gráfica de la matriz A resultante de discretizar el sistema de datos iniciales de ondas de Brill a segundo orden y con condición de Robin B_1. Se hace un acercamiento a una parte que escribe los puntos internos del laplaciano.

Nótese en particular los primeros dos elementos superiores. Estos son la condición de paridad (líneas 85-92). Luego viene la parte de matriz bandeada que son los puntos internos (líneas 95-113). Finalmente se ven tres elementos de la condición de Robin B_1 (líneas 116-137).

La metodología para generar las matrices para los otros casos de cuarto orden y del resolvedor general (3.39) es análoga aunque no trivial porque hay que tener cuidado en poner diferencias finitas ladeadas en el caso de cuarto orden. Nuevamente se señala que el código entero puede consultarse en el repositorio de [44].

3.4. Resolvedor directo LU

Al momento de resolver sistemas lineales existen docenas de métodos. Desde la básica eliminación gaussiana hasta métodos iterativos con precondicionadores. La intención de esta sección no es dar un resumen de todos los distintos métodos ya que se han escrito referencias enteras con ese propósito como son [51], [38], [25].

El método que se elige para resolver los sistemas matriciales cuya escritura se detalló arriba es el de un resolvedor directo LU. La justificación recae en que al escribir el sistema en forma explícita CSR, se puede aprovechar de poderosos resolvedores que están optimizados para aprovechar la estructura sparse y que pueden generar información adicional para que las futuras factorizaciones sean más rápidas. Después de explicar a continuación en qué consiste una factorización LU se retomará esta discusión de por qué se elije este tipo de resolvedor a favor de, por ejemplo, un método iterativo *GMRES*, *SOR*, o multigrid.

Dicho lo anterior, se debe definir en qué consiste una factorización LU. Si se tiene una matriz A, la factorización o descomposición LU consiste en factorizar la matriz A en dos matrices L y U tal que

$$A = LU\,, \tag{3.43}$$

donde L es una matriz triangular inferior y U una matriz triangular superior. Por ejemplo, si A es una matriz de 3×3:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = LU = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}. \tag{3.44}$$

La existencia de las matrices L y U se sigue de un teorema (véase [51]) que declara que existe dicha factorización si y sólo si la matriz es invertible y todos sus menores principales (i.e. los menores M_{ij} donde $i = j$) son distintos de cero. Por ejemplo, para la matriz anterior, la condición de invertibilidad pide obviamente que

$$\det A \neq 0\,, \tag{3.45}$$

mientras que la condición sobre los menores principales dice que

$$a_{11},\, a_{22},\, a_{33} \neq 0\,, \quad \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \det \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix}, \det \begin{pmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{pmatrix} \neq 0\,. \tag{3.46}$$

Un corolario adicional al teorema es que dicha factorización es única si se pide que la diagonal de L o de U sea unitaria de manera excluyente (generalmente esto se pide sobre L).

Sin analizar cómo se realiza la factorización, la pregunta es de qué manera se puede resolver el sistema lineal $Ax = f$ con LU. Para ello, supóngase que ya se tiene dicha factorización. Entonces, el sistema lineal se traduce a

$$LU\,x = f = L(U\,x) = f\,. \tag{3.47}$$

Obsérvese cómo se ha hecho énfasis en separar $U\,x$. Esto es porque si ahora se define

$$y = U\,x\,, \tag{3.48}$$

el sistema lineal puede escribirse como

$$L\,y = f\,. \tag{3.49}$$

Esto es relevante porque el sistema anterior (3.49) es muy sencillo de resolver gracias a la forma triangular de L. Dicho tipo de solución se conoce como sustitución delantera (*forward substiution*). Brevemente se puede ver que (3.49) declara inmediatamente que $y_1 = f_1/l_{11}$, de donde ahora se puede insertar el valor de y_1 en la siguiente ecuación, i.e. $y_2 = (f_2 - l_{12}\,y_1)/l_{22}$. De esta manera se generan los valores sucesivos de y en tiempo $\mathcal{O}(n)$ si se hace de manera paralela [30]. Recuérdese que n es el número de columnas de A o el número de entradas de x, y, f y en el problema principal de este trabajo $n \approx N_\rho\, N_z$.

Al tener el vector y a partir del algoritmo anterior, se puede resolver (3.48) para x. El método es muy análogo pero ahora se llama sustitución trasera (*backward substitution*) porque empieza desde la última ecuación del sistema $x_n = y_n/u_{nn}$ y regresa hasta generar todas las entradas de x. Debe ser evidente que el tiempo de este algoritmo es el mismo que el de sustitución delantera.

Se puede concluir que si se factoriza la matriz en LU, la solución del vector x se sigue fácilmente. Sobre todo para algoritmos paralelizados que pueden utilizar la forma de L y U para hacer sustitución por bloques. Es decir, si la forma de L y U es también sparse, cada ecuación del sistema $L\,y = f$ no requiere muchas entradas de y y así se puede subdividir el sistema en bloques independientes. Sin embargo, aunque se puede creer inicialmente que si A es sparse, sus factores L y U también lo son, esto no es cierto automáticamente.

El hecho de que L y U también puedan ser sparse resulta ser un problema fundamental desde el aspecto de memoria para almacenar cada una de estas matrices, así como el hecho básico de que si resultan tener estructura sparse no se necesita tanto tiempo para llenar y calcular sus elementos. Este tiempo se conoce en la literatura como *fill-in time*. Para reducir dicho llenado se puede permutar la matriz via $A' = P^T\, A\, P$. La diferencia puede ser dramática como se puede apreciar en el siguiente ejemplo tomado de [24]. Ahí el sistema no permutado y su factor L son

$$A = \begin{pmatrix} 9 & 3/2 & 6 & 3/4 & 3 \\ 3/2 & 1/2 & 0 & 0 & 0 \\ 6 & 0 & 12 & 0 & 0 \\ 3/4 & 0 & 05/8 & 0 & \\ 3 & 0 & 0 & 0 & 16 \end{pmatrix} \quad \Rightarrow \quad L = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 2 & -2 & 2 & 0 & 0 \\ 1/4 & -1/4 & -1/2 & 1/2 & 0 \\ 1 & -1 & -2 & -3 & 1 \end{pmatrix}, \tag{3.50}$$

mientras que el sistema permutado y su factor L son

$$A' = \begin{pmatrix} 16 & 0 & 0 & 0 & 3 \\ 0 & 1/2 & 0 & 0 & 3/2 \\ 0 & 0 & 12 & 0 & 6 \\ 0 & 0 & 0 & 5/8 & 3/4 \\ 3 & 3/2 & 6 & 3/4 & 9 \end{pmatrix} \quad \Rightarrow \quad L = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 0 & 0 \\ 0 & 0 & 2\sqrt{3} & 0 & 0 \\ 0 & 0 & 0 & \sqrt{10}/4 & 0 \\ 3/4 & 3/\sqrt{2} & \sqrt{3} & 3/\sqrt{10} & \sqrt{3/5}/4 \end{pmatrix}. \tag{3.51}$$

Ergo, se puede inferir con buena confianza que si el resolvedor puede permutar el sistema lineal para generar factores L y U sparse, el tiempo de ejecución será mucho menor. El hecho de que permutar los elementos no requiere conocer los valores numéricos de la matriz, sino sólo su estructura, hace que este proceso se llame *factorización simbólica*.

Con esto en mente, se puede regresar a la discusión de por qué se elige un resolvedor LU para este trabajo. La ventaja cae particularmente en la permutación discutida pues el problema principal de este trabajo será resolver el lapso maximal (1.41) a cada paso de integración. En cada instancia, el sistema lineal generado será distinto pero estructuralmente equivalente a los resueltos en pasos anteriores. Por lo tanto, si se calcula la permutación via la factorización simbólica en el paso inicial, se tiene para todo futuro paso. Adicionalmente, con la justificación de que las entradas de la matriz varían lentamente entre cada entrada, los factores L, U calculados previamente pueden utilizarse como precondicionares para este nuevo paso. Entonces, aunque no se trata de un método de relajación donde se puede utilizar la solución obtenida en el paso anterior, el aspecto iterativo se conserva y puede acelerar la solución de los sistemas a estudiar en este trabajo.

3.5. El resolvedor directo PARDISO

Pese a que pudiera parecer inicialmente que los aspectos detallados arriba son muy específicos, hay una gran variedad de resolvedores a elegir. Entre las opciones más usadas en la comunidad están MUMPS(*MUltifrontal Massively Parallel sparse direct Solver*: [9] y [10]), PETSc/Tao (*Portable, Extensible Toolkit for Scientific Computation* [13]), y PARDISO (*Parallel Sparse Direct and Multi-Recursive Iterative Linear Solver*: [46] y [47]). Como el objetivo de este trabajo es escribir un código que pueda correr en una máquina de uso personal pero sí pueda hacer uso de subrutinas paralelas, el que resulta más ideal es PARDISO. Tanto MUMPS como PETSc tienen un enfoque orientado más bien a *clusters* con un gran número de procesadores. PARDISO tiene una funcionalidad OOC (*out-of core*) para correr en *clusters*, así que si resulta necesario adaptar este trabajo para un *cluster* no debe resultar muy difícil.

Como último comentario sobre la comparación entre los resolvedores, se cita un resultado que los pone directamente a comparación [29]. Ahí se encuentra que aunque PARDISO suele tomar más tiempo en la etapa de análisis y factorización simbólica, en la factorización numérica y las sustituciones delantera/trasera tiene ventajas importantes sobre los otros resolvedores. Resulta muy conveniente que el área débil del resolvedor sea la que se puede optimizar usando el hecho de que las matrices que se están resolviendo tienen la misma estructura y sólo es necesario hacer el análisis y factorización simbólica una vez.

3.5.1. Estructuras y parámetros de control

PARDISO tiene como ventaja que puede resolver muchos tipos de problemas: simétricos, no-simétricos, reales, complejos, transpuestos, múltiples lados derechos, etc. La siguiente discusión no tiene como propósito explicar el uso del resolvedor de manera exhaustiva ya que la mejor manera de entender su uso es leer las referencias [52], [20] junto con algunos ejemplos que ahí se presentan. Lo que se pretende es resaltar algunos parámetros especialmente relevantes para el problema principal de este trabajo. Dicho lo anterior, la estructura central que pasa los parámetros se llama *iparm* y consiste en un arreglo de 64 enteros.

Primero que nada, todos los valores de *iparm* se inicializan a su valor default para matrices no-simétricas como están dados en [20]. A continuación se listan en el Cuadro 3.1 los parámetros que no se ponen en su valor predeterminado o que son relevantes para la discusión posterior.

Componente	Valor	Comentario
$iparm(2)$	3	Este es el algoritmo que reordena la matriz. Con la opción 3 se usa una versión paralelizada en *OpenMP*.
$iparm(4)$	0 ó 61	Indica si se usa CGS (*conjugate gradient squared* [56]) como precondicionador. Es necesario haber calculado la factorización LU antes. Si se trata del primer paso donde no se ha calculado nada debe ser 0. En cambio, si se desea prender en pasos futuros puede valer 61: el 6 indica una tolerancia en el precondicionador de 10^{-6} y el 1 establece que sí se debe usar CGS.
$iparm(5)$	0,1 ó 2	Este parámetro controla si se usa una permutación dada por el usuario. Si es 0 no hay permutación dada; con 1 se usa como permutación un arreglo dado por el usuario; con 2 se calcula el arreglo de permutación. Por lo mismo, puede ser 2 en el primer paso y luego 1 en los demás si se utiliza el arreglo devuelto en el primer paso.
$iparm(8)$	10	Número máximo de pasos que refinan iterativamente la solución. Véase [52].
$iparm(10)$	13	Cómo se manejan los pivotes pequeños para matrices no-simétricas. Véase [52].
$iparm(11)$	1	Reescala la matriz para que la diagonal se unitaria.
$iparm(24)$	10	*Específico a Intel MKL PARDISO*: utiliza un algoritmo paralelo de dos niveles para factorizar la matriz.
$iparm(25)$	1	Utiliza un algoritmo paralelo en las sustituciones delanteras y traseras.

Tabla 3.1: Tabla con parámetros de control para el resolvedor PARDISO.

Los parámetros más importantes son $iparm(4)$ e $iparm(5)$ porque son los que permiten acelerar la subrutina al precondicionar con una factorización previa y/o usar una permutación calculada en la primera solución. En la práctica, para una malla común de 500×500 puntos internos, se encuentra que al usar ambas opciones la solución se acelera en más de 1.25 veces comparada con estar siempre calculando la permutación y sin precondicionar. El decremento de tiempo es principalmente en la factorización simbólica donde se puede reducir más de 2.5 veces. Sin embargo, el tiempo principal de uso es la factorización numérica donde se usan los algoritmos de pivoteo. Aunque no parezca mucho, 1.25 es muy relevante ya que en una simulación típica puede haber miles de llamadas al resolvedor elíptico y reducir el tiempo de ejecución en un cuarto puede ser la diferencia entre dejar corriendo el código tres o cuatro días como en la simulación más larga que se detalla en el Capítulo 6 de resultados.

Capítulo 4

Encontrador de horizontes aparentes

4.1. Teoría en simetría axial

La teoría del los horizontes aparentes es detallada con cuidado en [1], [7] y [59]. Tomando como base dichas fuentes, podemos resumir que la condición de que una superficie bidimensional S, dentro de la hipersuperficie espacial Σ, sea un horizonte aparente es que la expansión H de las geodésicas nulas salientes sea cero. Es decir, si s^μ es el vector espacial normal a S, este induce una métrica $h_{\mu\nu}$ sobre S dada por

$$h_{\mu\nu} = \gamma_{\mu\nu} - s_\mu \, s_\nu \,, \tag{4.1}$$

donde $\gamma_{\mu\nu}$ es la métrica espacial usual inducida en Σ. Además, el vector nulo saliente de S deberá ser una combinación de s^μ y la normal temporal a Σ n^μ. Si llamamos a este vector l^μ, entonces, hasta un factor de normalización

$$l^\mu = n^\mu + s^\mu \,. \tag{4.2}$$

Es con esta expresión para el vector nulo que podemos calcular la expansión de sus geodésicas sobre S con la derivada de Lie, i.e. la cantidad H:

$$H = -\frac{1}{2} \, h^{\mu\nu} \, \pounds_l \, h_{\mu\nu} \,. \tag{4.3}$$

Esto se expande con algo de álgebra e identidades relacionadas a la curvatura extrínseca de S y de Σ. Eventualmente se puede llegar a que

$$H = \nabla_i \, s^i + K_{ij} \, s^i \, s^j - K \,, \tag{4.4}$$

y con un poco más de álgebra que

$$H = \left(\gamma^{ij} - s^i \, s^j\right) \, \left(\nabla_i \, s_j - K_{ij}\right) \,. \tag{4.5}$$

Como ya se había afirmado, la condición de horizonte aparente es que $H = 0$. En resumen, el horizonte aparente (que comúnmente se abreviara AH por sus siglas en inglés) es la superficie más externa tal que $H = 0$ en toda la superficie S.

Sin pérdida de generalidad, podemos asumir que la superficie se escribe como el nivel cero de una función, i.e.

$$F(x^i) = 0 \,. \tag{4.6}$$

Esto nos permite reescribir H en términos de F y sus derivadas, ya que el vector normal es sencillamente

$$s^i = \frac{\nabla^i F}{|\nabla F|}\,. \tag{4.7}$$

Nótese que s^i es unitario (esto es necesario porque para deducir (4.5) se usó que $s^k\,\nabla_i\, s_k = 0$). Entonces, al insertar en (4.5) se tiene que

$$H = \left(\gamma^{ij} - \frac{(\nabla^i F)\,(\nabla^j F)}{|\nabla F|^2}\right)\,\left(\frac{\nabla_i\,\nabla_j\, F}{|\nabla F|} - K_{ij}\right) = 0\,. \tag{4.8}$$

Hasta ahora, este tratamiento ha sido completamente general, sin ninguna restricción a alguna geometría. Se puede esperar que para situaciones simétricas esféricas o axiales se pueda simplificar esta última ecuación. El caso de simetría esférica está tratado en [1] donde resulta una ecuación muy sencilla para un valor de r que cumple la condición de AH.

El caso de simetría axial (también tratado en [1] y [60]) es más complejo debido al hecho que ahora se trata la superficie como un cuerpo de revolución alrededor del eje z. La parametrización usada es que

$$F(r,\theta) = r - R(\theta)\,, \tag{4.9}$$

que se denomina *strahlkörper* ó cuerpo de rayos, debido a que se trata de una superficie trazada por rayos salientes por el origen. Al insertar esta función en (4.8) se obtiene una ecuación diferencial de segundo orden ordinaria no lineal para R (véase [1]):

$$\frac{\partial^2 R}{\partial\theta^2} = -\frac{1}{\gamma^{rr}\,\gamma^{\theta\theta} - (\gamma^{r\theta})^2}\,\left(u^2\,\gamma^{ij} - (\partial^i F)\,(\partial^j F)\right)\,\left(\Gamma^k{}_{ij}\,\partial_k F + u\,K_{ij}\right)\,, \tag{4.10}$$

donde

$$u = \sqrt{(\partial_i F)(\partial^i F)}\,. \tag{4.11}$$

Para resolver esta ecuación diferencial necesitaremos dos condiciones de frontera. La axisimetría pide que R sea suave en el eje z, i.e. $\theta = 0$. Naturalmente, esto implica que $\partial_\theta R(0) = 0$ como primera condición de frontera. Como segunda condición vemos que al tener simetría ecuatorial en $\theta = \pi/2$, también $\partial_\theta R(\pi/2) = 0$.

Como se trata de una ecuación no lineal y con condiciones de Neumann de frontera, el método más común para resolverla es el método de *disparo* donde se va recorriendo el eje z para un valor $R(0)$. Después, partiendo de que $\partial_\theta\, R(0) = 0$, se integra hasta llegar a $\pi/2$. En la práctica es casi imposible que en la primera adivinanza se obtenga $\partial_\theta R(\pi/2) = 0$ para esta función integrada. Más bien, conforme recorramos el eje de afuera hacia adentro nos fijaremos en qué momento cambia de signo la derivada en el ecuador y luego se hacen refinaciones con un método de bisección hasta obtener un valor de $\partial_\theta\, R(\theta = \pi/2)$ tan pequeño como una tolerancia estipulada.

4.2. Algoritmo

La parte del método de disparo ya se ha detallado brevemente en el párrafo anterior. Sin embargo, es importante ahondar en ciertos detalles intrínsecos a la implementación usada en este trabajo. Primero que nada, el problema central es que las cantidades usadas en (4.10) son r, θ. Esto contrasta fuertemente con como se ha diseñado la infraestructura axisimétrica en términos de mallas en ρ, z que

además están escalonadas con el propósito de no tocar los ejes y tener problemas de regularización. Para integrar, se van a requerir cantidades en los ejes, ya que es imposible resolver este problema sin partir de $\theta = 0$ y $\theta = \pi/2$ que es donde tenemos las condiciones de frontera para $R(\theta)$.

Con esto en mente, el primer paso es escribir las cantidades esféricas como γ^{rr}, $\gamma^{\theta\theta}$, etc. en términos de las cantidades cilíndricas. Esto no es difícil al conocer la transformación de coordenadas. Para simplificar y verificar el trabajo, estas expresiones se han reducido en *Mathematica* (véase [35]) y no se presentan en seguida debido a su tamaño y por no contribuir realmente a la discusión.

Pese a esto, no se ha solucionado la cuestión de que las cantidades viven en mallas ρ, z. En vez de hacer nuevas mallas para r, θ, basta con poder tener una subrutina que pueda interpolar las cantidades al pedirlas en un punto (r, θ). Para ello se implementa un algoritmo de interpolación bicúbica (véase por ejemplo [48]) usando multiplicación rápida de matrices. Así se obtienen todas las cantidades necesarias para calcular el lado derecho de (4.10).

El algoritmo empieza con una adivinanza inicial $R(0) = r_{max}$ que puede ser el valor máximo de r soportado por las mallas ρ, z o un valor estipulado por el usuario. De esta manera, el encontrador de horizontes intentará encontrar un horizonte en el rango $r \in [0, r_{max}]$. Partiendo de la condición de frontera $\partial_\theta R(0) = 0$ y usando la interpolación bicúbica para calcular las fuentes, se integra usando Runge-Kutta de cuarto orden hasta llegar al ecuador. En este punto se almacena el valor de la derivada en una variable auxiliar que servirá para saber si hay un cambio de signo en una futura integración. La subrutina determina que ya se encontró el horizonte si ya se llegó a $|\partial_\theta R(\pi/2)| < t$, donde t es una tolerancia arbitraria normalmente igual a h^2 o h^4 (h es la separación en las mallas ρ, z) según el código esté trabajando a segundo o cuarto orden, respectivamente. Si no se ha llegado a dicha tolerancia, se toma una nueva adivinanza $R(0)$ menor a la anterior y se procede iterativamente hasta encontrar un cambio de signo en $\partial_\theta R(\pi/2)$. En este punto se refina con un número finito de bisecciones hasta llegar a la condición de salida $|\partial_\theta R(\pi/2)| < t$. Si no se encuentra un cambio de signo en todo el rango de búsqueda, se determina que no existe el AH.

En la práctica el usuario puede no sólo proporcionar un valor de r_{max} sino también un valor de r_{min} si ya tiene una idea de dónde puede estar el horizonte. Si se trata de una evolución donde ya se había encontrado un horizonte en un tiempo previo, estos valores se pueden restringir con base en la adivinanza correcta para el horizonte encontrado en el tiempo anterior. Del mismo modo, con el fin de minimizar el tiempo tomado por esta subrutina, el usuario también puede determinar a partir de qué tiempo de integración se debe prender el encontrador de horizontes.

El pseudocódigo de la subrutina detallada se encuentra resumido en el siguiente listado.

```
// AHF: Encontrador de horizontes. Se llama en la etapa de análisis geométrico para
    encontrar un AH en el espacio.

// Incluye parámetros globales y declaraciones.
// ...
void ah(void) {
  // Calcula geometría adicional relacionada al AHF.
  ah_geometry();
  // Itera sobre un rango de radios.
  for (R = ah_start + 20 * drr; R > ah_min_rr; R -= drr) {
    // Integra usando la adivinanza.
    ah_integrator(R, &flag);
    // Revisa que la integración sea exitosa.
    // ...
    // Guarda dR(pi/2) en la variable temporal t_dR1.
    t_dR1 = array_dR[NThetaTotal - 1];
    // Revisa si se puede comparar la ultima integración
    // y compara el cambio de signo.
    if (last_int && (t_dR1 * t_dR2 <= 0.0)) {
      // AH encontrado.
      ah_found = 1;
      // Bisección con máximo ah_bis_iter iteraciones.
      for (l = 0; l < ah_bis_iter; l++) {
        // Bisecta en el punto medio entre Ra, Rb e integra.
        // ...
        // Revisa convergencia de salida.
        if (ABS(Rc - Ra) < tol) || ABS(dRc) < tol) {
          // Escribe a archivo, nuevo ah_start.
          // ...
          // Calcula máscara.
          ah_mask();
          return;
        }
      }
    }
    // Continua si no hay cambio de signo.
    else {
      // Transfiere las variables.
      t_dR2 = t_dR1;
    }
  }
  // Si se llega a este punto, no se encontró AH.
  return;
}
```

4.2.1. Resultados con datos iniciales de Schwarschild

El primer caso que se puede utilizar para comprobar el buen funcionamiento del AHF es poner datos iniciales de Schwarschild en coordenadas isotrópicas. En este caso la métrica es conformamente plana y el factor conforme es

$$\psi = 1 + 2M/r, , \tag{4.12}$$

con M la masa ADM del agujero negro. Como es bien sabido, el horizonte de eventos está en $r = M/2$ y al ser una situación estática debe coincidir con el horizonte aparente (véase [33]). Específicamente, en la siguientes simulaciones, si $M = 1$, el horizonte es la esfera $r = 1/2$. Esto lo podemos ver en las corridas a continuación donde las resoluciones han sido $h = \{1/16, 1/32, 1/64, 1/128\}$ y abarcan una malla cuadrada con $N = \{128, 256, 512, 1024\}$ puntos interiores en cada dimensión, respectivos a cada resolución. Por ahora, se trata de sólo datos iniciales sin evolucionar, lo que significa que sólo se está buscando el horizonte a $t = 0$.

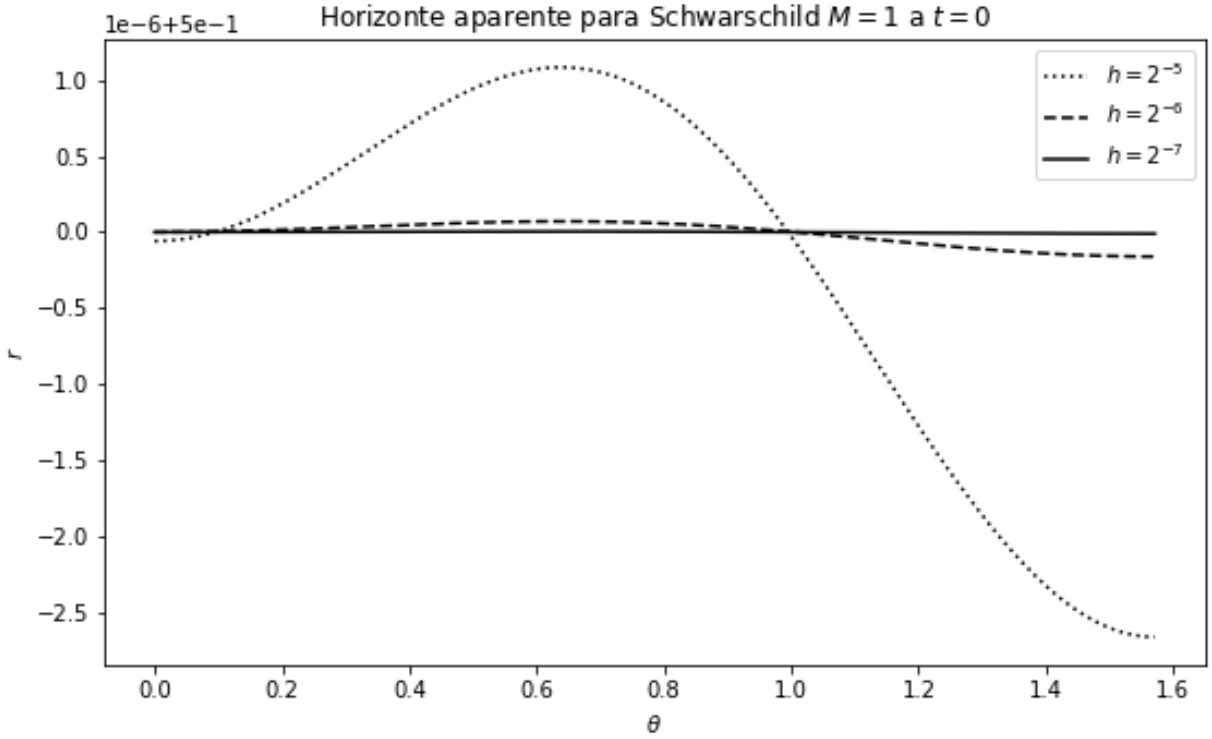

Figura 4.1: Horizonte aparente para Schwarschild con $M = 1$. Obsérvese que se ha restado $1/2$ en el eje vertical y que el error es de orden 10^{-6} incluso para la menor resolución.

Como se puede ver, las cuatro resoluciones parecen estar muy cercanas entre sí en el sentido de que incluso la menor de las resoluciones difiere en sólo en el orden de 10^{-6} del valor analítico. De todos modos, sí se alcanzan a ver distinciones entre las resoluciones. Sobre todo se aprecia como al incrementar la resolución la curva se vuelve visiblemente más esférica y se acercan al radio de $r = 1/2$.

Estos argumentos se pueden fundamentar con mayor solidez si se pone una medida mucho más cuantitativa que el acercamiento visual de curvas. Para ello se pretende ahora calcular el área del horizonte usando el hecho de que

$$A = \oint_S \sqrt{h}\, \mathrm{d}s = 4\,\pi \int_0^{\pi/2} \sqrt{\gamma_{\phi\phi}(R(\theta))\, \gamma_{\theta\theta}(R(\theta))}\, \mathrm{d}\theta \ , \tag{4.13}$$

donde h es el determinante de la métrica inducida sobre el horizonte, mismo que se calcula fácilmente del lado derecho. Para dicho cálculo se han hecho cero los términos provenientes de $\gamma_{\theta\phi}$ para un espacio axisimétrico donde además no hay momento angular. El factor de 4π viene de la simetría ecuatorial y de la integral sobre ϕ. Se sigue también que en la integral de línea que usa la métrica física evaluada en el horizonte $R(\theta)$. Esta integral se puede realizar numéricamente con un método de Simpson a cuarto orden [48]. Al tener un área es sencillo calcular la masa vía

$$M = \sqrt{\frac{A}{16\pi}}\,. \tag{4.14}$$

Es así como podemos obtener masas para cada resolución y para cada una podemos ver su diferencia con el valor analítico de $M = 1$.

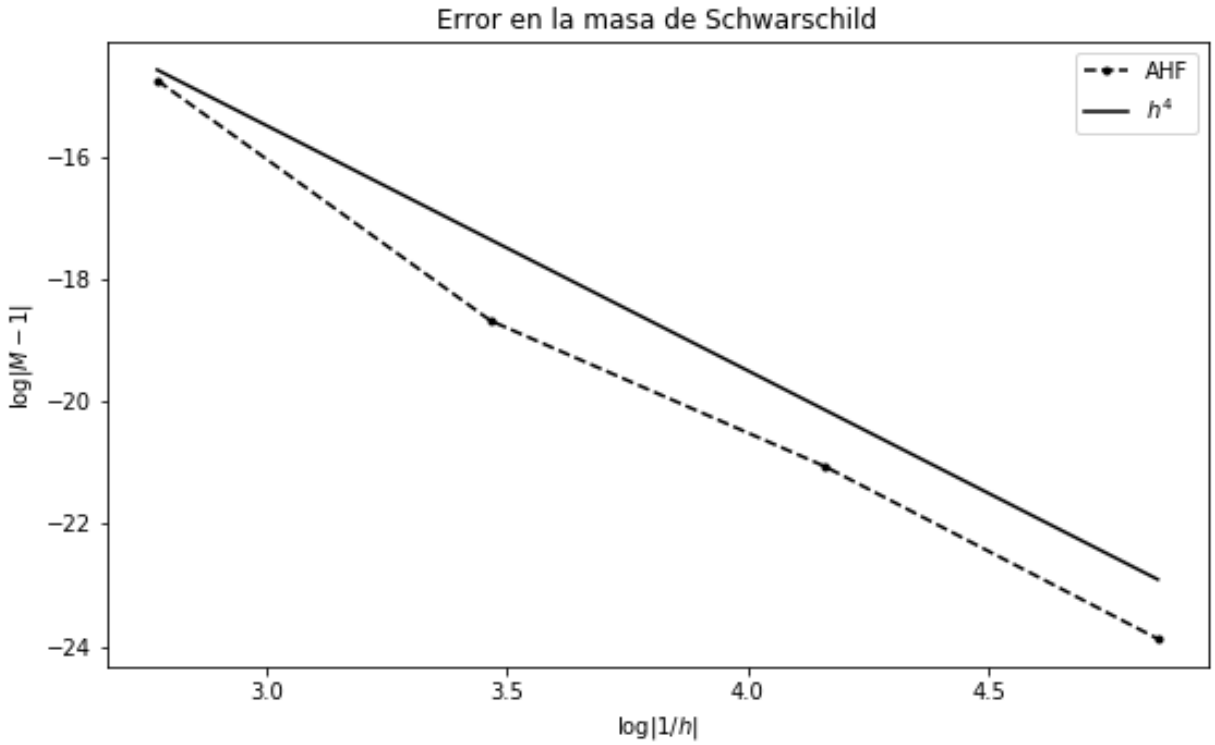

Figura 4.2: Gráfica log − log del error en la masa de los horizontes contra el inverso de la resolución $1/h$. También se presenta una recta $\mathcal{O}(h^4)$.

Podemos verificar en la gráfica anterior que tanto el AHF como el integrador que determina el área y masa están funcionando como se espera y además convergen a cuarto orden sin duda gracias a múltiples factores como son la interpolación bicúbica, las derivadas a cuarto orden y la integración por Simpson a cuarto orden.

Como último aditivo a la discusión de Schwarschild está la importancia de cómo ayuda el AH a quitar el interior del horizonte de eventos donde existe la singularidad. Justamente para Schwarschild esto está muy pronunciado por el polo $1/r$ en el factor conforme. A continuación se ve que si se calculara la constricción hamiltoniana en toda la malla, podríamos creer que no hay la convergencia esperada. Si usamos una máscara que ignore el AH al calcular las constricciones podemos ver que sí hay convergencia al orden correcto.

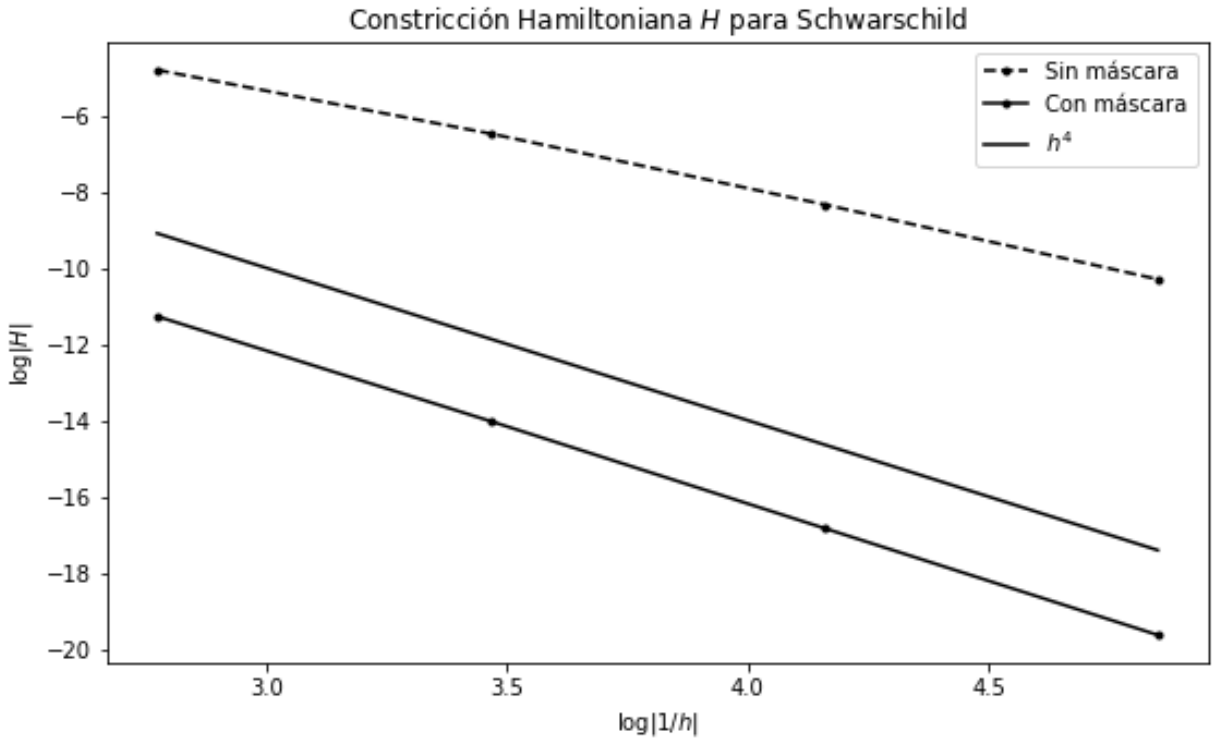

Figura 4.3: Constricción hamiltoniana para datos iniciales de un agujero negro de Schwarschild con $M = 1$. Se ven dos curvas de datos donde en una se ha ignorado el interior del horizonte al calcular las constricciones y en otra no. La recta $\mathcal{O}(h^4)$ está como referencia.

4.2.2. Resultados con datos iniciales de Brill

Habiendo verificado el buen funcionamiento con Schwarschild, ahora se presenta una verificación menos trivial donde ya no se tiene simetría esférica. El ejemplo claro tan sólo por el título del presente trabajo es buscar horizontes en los datos iniciales de Brill obtenidos por el resolvedor elíptico. Este tipo de trabajo se ha hecho en diversos sitios como son [7] y [27]. En particular, en [7], Alcubierre *et. al.* ponen a prueba dos tipos de AHF's usando ondas de Brill con la forma de Holz (2.17) y encuentran un horizonte en los datos iniciales para una amplitud de $a = 12$ para el que también calculan su masa y área.

Análogamente a la sección anterior donde se presentó Schwarschild, se vuelve a trabajar a cuarto orden con resoluciones $h = \{1/16, 1/32, 1/64, 1/128\}$ correspondientes a $N = \{128, 256, 512, 1024\}$ en cada dimensión. En los tres casos, el AHF sí encuentra un horizonte aparente a $t = 0$, mismos que se presenta en la siguiente figura:

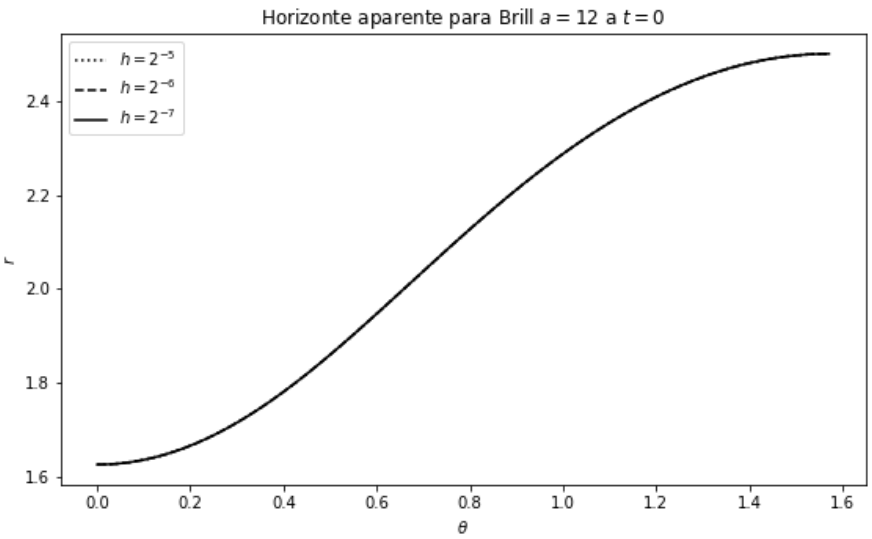

Figura 4.4: Horizonte aparente para datos iniciales de Brill con $a = 12$.

Como ahora el horizonte no es esférico, vale la pena ver la gráfica en forma polar:

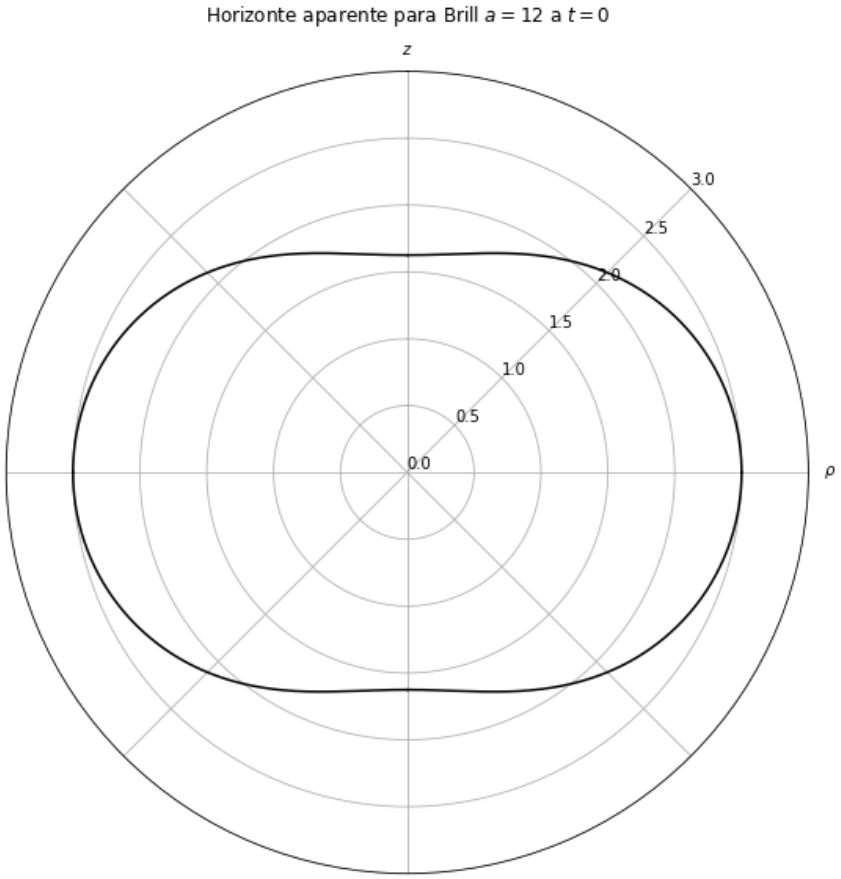

Figura 4.5: Gráfica polar del horizonte aparente para datos iniciales de Brill con $a = 12$.

En esta forma se puede ver de inmediato una similitud con el horizonte de [7]. Además, las masas correspondientes a estas resoluciones son

$$M_{h_0} = 4.58010\ldots\,, \quad M_{h_1} = 4.58000\ldots\,, \quad M_{h_2} = 4.57999\ldots\,, \tag{4.15}$$

donde $h_0 = 1/16$, $h_1 = 1/32$ y $h_2 = 1/64$. Aunque se trata de sólo tres datos, la autoconvergencia

$$\frac{|M_{h_0} - M_{h_1}|}{|M_{h_1} - M_{h_2}|} = 25.624\ldots\,, \tag{4.16}$$

excede el factor esperado de $2^4 = 16$, lo que indica que los valores se están acercando más rápido que $\mathcal{O}(h^4)$ al menos para estas masas calculadas.

De mayor importancia es comparar este número con el valor reportado en [7] de $M \approx 4.5$ que está muy de acuerdo con estas masas anteriores.

Como último detalle, en [7] se afirma que la amplitud crítica a_* a partir de la cual aparece un horizonte en los datos iniciales es de $a_* \in [11.81, 11.82]$. Este encontrador de horizontes confirma esto también al encontrar que

$$a_* \in [11.81594, 11.81595]\,. \tag{4.17}$$

Como conclusión a este capítulo, se puede llegar a que el encontrador de horizontes diseñado funciona muy de acuerdo a lo esperado en casos estáticos. Puesto que es una subrutina de análisis (no modifica ninguna cantidad geométrica de entrada), su uso correcto durante la evolución dependerá del código de evolución. En el Capítulo 6 de resultados se volverá a visitar el comportamiento correcto del AHF usando una corta evolución de Schwarschild para ver si mantiene una masa constante.

Capítulo 5

El código OllinBrill

El código de evolución de este trabajo se llama *OllinBrill*. [1] Dicho código está fuertemente inspirado en el código de José Manuel Torres *OllinAxis* (véase [61]). Sin embargo, este trabajo es totalmente independiente y no consiste en simplemente reescribir el código *OllinAxis*, originalmente escrito en FORTRAN, en C.

La diferencia central de este trabajo es el resolvedor elíptico implementado. Por lo tanto, *OllinBrill* tiene paralelismo en *OpenMP* (véase [23]), a diferencia de *OllinAxis* que trabaja con *MPI* (véase [28]). *OllinAxis* trabaja muy bien con *MPI* porque durante toda la evolución no es necesario que un procesador conozca toda la información. Basta con hacer sincronizaciones en unas ghost zones en momentos clave para que la evolución proceda correctamente. Mientras tanto, *OllinBrill* necesita construir un sistema matricial para toda la malla a la hora de resolver las ecuaciones elípticas. Específicamente la solución de las ecuaciones elípticas es incompatible con *OllinAxis* sin tener que hacer grandes cambios o escribir subrutinas que junten información de los procesadores y luego la vuelvan a distribuir.

Si bien *OllinAxis* también tiene mayor generalidad para resolver más datos iniciales y tratar con el caso de que el espacio no tenga simetría ecuatorial, estas limitantes pueden ser vistas como ventajas para *OllinBrill* ya que el enfoque reducido ha permitido tener mayor control sobre las optimizaciones.

Por último, ambos códigos tienen un público distinto. *OllinAxis* es un código más formal para correr en *clusters* con su estructura de *MPI*, mientras que *OllinBrill* está dirigido para uso personal para nuevos estudiantes en el área que no tienen ni necesitan *hardware* de uso profesional. Para mayor información sobre *OllinBrill* incluyendo cómo bajarlo y compilarlo, se presenta el repositorio [44].

5.1. Subrutinas de geometría

El cálculo de las cantidades geométricas como son los componentes del tensor de Ricci, la divergencia $\hat{\nabla}_k \hat{A}^{ik}$, la segunda derivada covariante del lapso $\nabla_i \nabla_j \alpha$, etc. no debería ser un problema mayor en el código. Sin embargo, debido a que las expresiones se hacen muy largas, la experiencia dice que estas subrutinas son las que más sufren de errores humanos triviales como escribir mal una variable o pasar un argumento equivocado a la hora de llamarla.

Con el fin de evitar estos errores, las expresiones algebraicas para los términos geométricos fueron generadas en *Mathematica* (véase [35]). La libreta usada está incluida en el repositorio [44] y describe cuidadosamente cómo se calculan los términos de la formulación BSSN covariante. Esta

[1] *Ollin* significa 'movimiento' en náhuatl.

misma libreta genera código en C para que se pueda insertar directamente en los archivos del código tras unas modificaciones. Dichas modificaciones pueden ser muy importantes, sobre todo cuando se trata del esquema de regularización detallado en la Sección 1.5.2.

Para demostrar que estas subrutinas hacen un cálculo correcto que converge al orden esperado (segundo o cuarto orden), se usa una métrica conforme $\hat{\gamma}_{ij}$ prueba definida como

$$\begin{aligned} \hat{\gamma}_{\rho\rho} &= a = e^{2q/3}\,, \\ \hat{\gamma}_{zz} &= b = e^{2q/3}\,\left(1+\rho^2\,z^2\,e^{-2r^2-4q/3}\right)\,, \\ \hat{\gamma}_{\varphi\varphi}/\rho^2 &= h = e^{-4q/3}\,, \\ \hat{\gamma}_{\rho z}/\rho &= c = z\,e^{-r^2}\,, \end{aligned} \tag{5.1}$$

donde q es esencialmente la q de Holz (2.17) con $a=1$, i.e. $q=\rho^2\,e^{-r^2}$. Obsérvese que esta métrica es compatible porque cumple todas las condiciones de simetría requeridas para las variables a,b,h,c y que

$$\hat{\gamma} = (ab-\rho^2\,c^2)\,\rho^2\,h = \rho^2 = \mathring{\gamma}\,. \tag{5.2}$$

Naturalmente, esta métrica también tiene una variable de regularización asociada

$$\lambda = \frac{e^{2q/3}-e^{-4q/3}}{\rho^2} \tag{5.3}$$

y termina siendo regular en el eje gracias a la forma de q. También tiene un vector $\hat{\Delta}^i$ calculado via (1.26) que no se escribe aquí debido a que la expresión es muy larga y poco ilustrativa.

Para continuar con la geometría que se va utilizar como prueba, se introduce arbitrariamente una curvatura no trivial. Con el fin de no escribir expresiones largas y poco legibles se opta por dejarla en la forma simbólica

$$\hat{A}_{ij} = \mathring{\gamma}_{ij} - \left(\frac{1}{3}\,\hat{\gamma}^{kl}\,\mathring{\gamma}_{kl}\right)\,\hat{\gamma}_{ij}\,, \tag{5.4}$$

donde $\hat{\gamma}_{ij}$ es la métrica introducida arriba en (5.1) y $\mathring{\gamma}_{ij}$ es la métrica plana diag $\left(1,1,\rho^2\right)$. Nótese que $\hat{A}_{ij}$ no tiene traza, como debería ser y además no resulta díficil ver que $\hat{A}_\lambda$ también es regular en el origen por consecuencia directa de que λ ya lo era. Adicionalmente, se define que la traza $K=0$ para esta geometría.

Se hace énfasis especial en que esta geometría se ha impuesto arbitrariamente sin que signifique algo físicamente. Se trata tan solo de una serie de cantidades que cumplen todos los requisitos de métrica y curvatura y que generan términos no triviales tales que se puede comparar la salida del código contra valores analíticos. Es decir, el objetivo es ver que las subrutinas de geometría y regularización se estén usando correctamente. Al tener expresiones analíticas para la métrica, se puede calcular también de manera analítica las cantidades geométricas como el tensor de Ricci $\hat{R}_{ij}$ y $\hat{\nabla}_k\,\hat{A}^{ik}$. Después se comparan estos resultados contra los valores regresados por las subrutinas. En las siguientes Figuras 5.1 y 5.2 se presenta la norma infinita de restar la variable analítica contra la variable numérica. Se presentan cuatro resoluciones $h=0.08,\,0.04,\,0.02,\,0.01$ con puntos interiores en cada dimensión igual a $N=128,256,512,1024$, respectivamente. El cálculo ha sido ejecutado a cuarto orden.

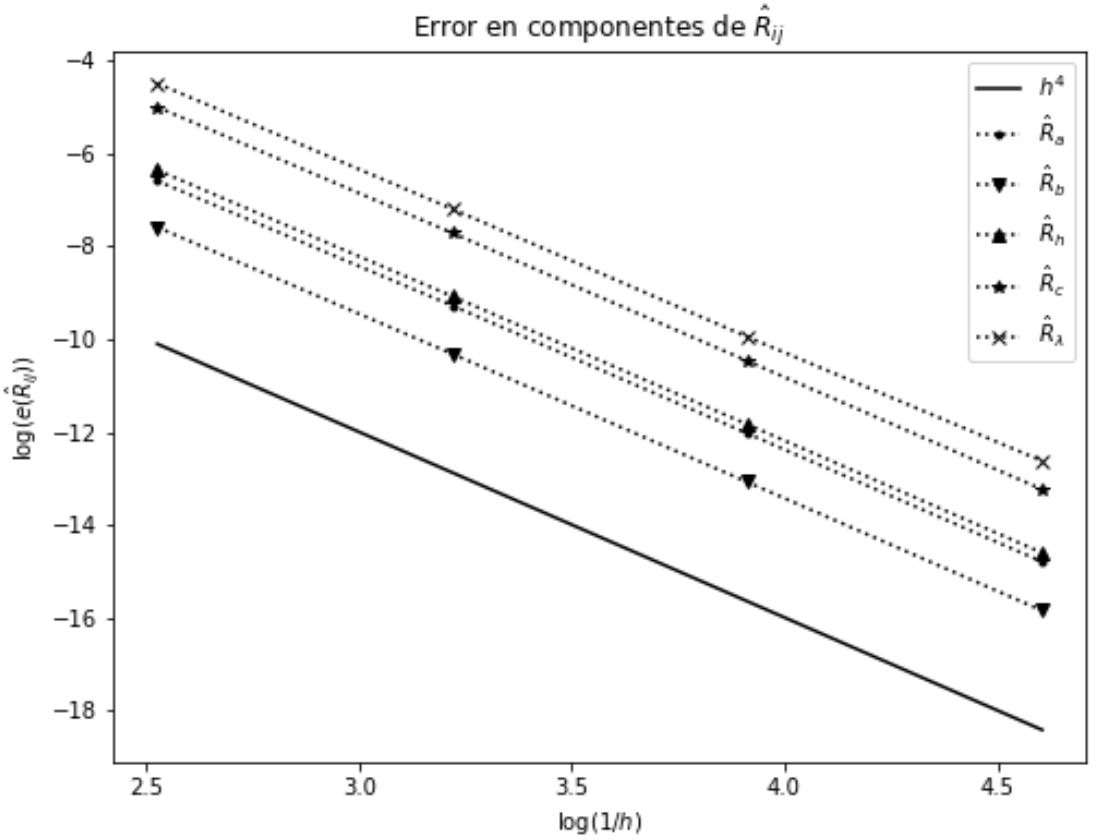

Figura 5.1: Gráfica log vs. log del error en los componentes del tensor de Ricci $\hat{R}_{ij}$ asociados a la métrica prueba. Se han dibujado los cuatro componentes a, b, h, c y el de regularización λ. En adición, la curva sólida dibuja $\mathcal{O}(h^4)$.

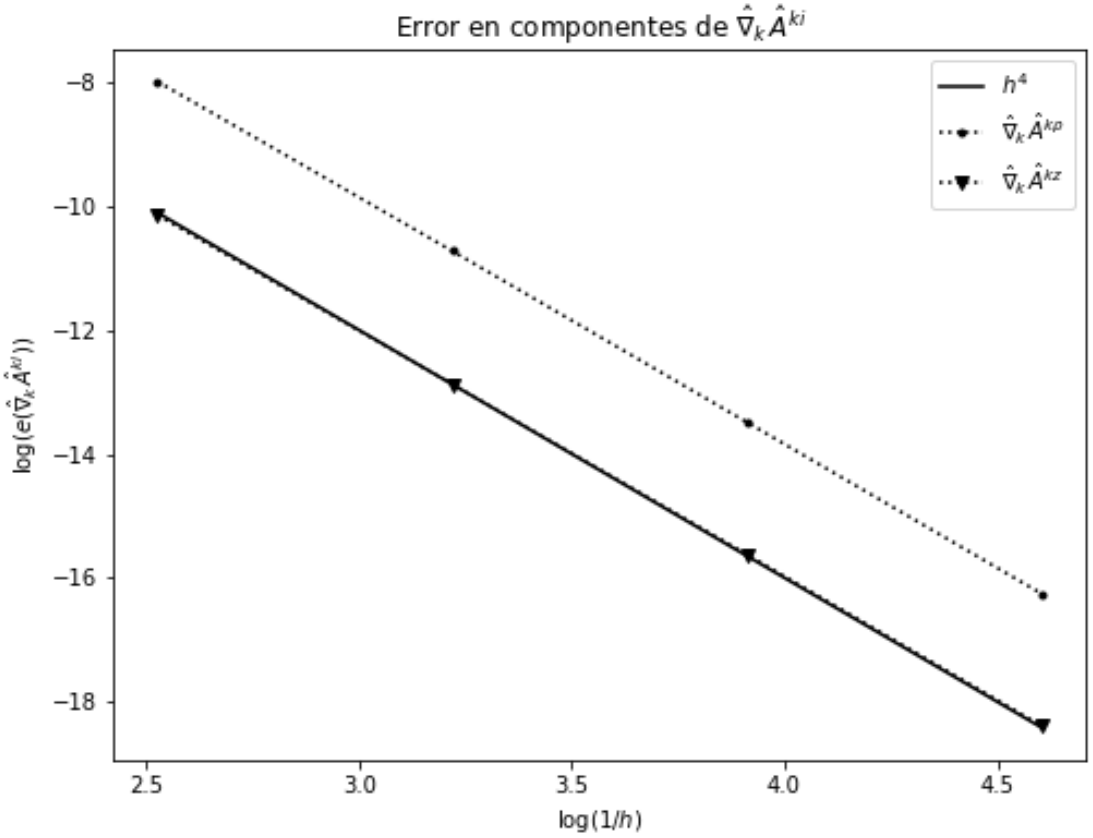

Figura 5.2: Gráfica log vs. log del error en los componentes de la divergencia $\hat{\nabla}_k \hat{A}^{ik}$ asociados a la métrica y curvatura prueba. Se han dibujado los dos componentes ρ, z así como la curva sólida para $\mathcal{O}(h^4)$.

Las dos figuras anteriores demuestran que el cálculo de estas cantidades geométricas (que son la más complejas y requieren mayor regularización) se está realizando de manera correcta y de hecho sí tenemos convergencia a cuarto orden.

La última cantidad compleja resulta ser $\nabla_i \nabla_j \alpha$, para la cual usamos la misma métrica conforme anterior (de hecho esta va a ser la métrica física si se pone que $\phi = 0$ como factor conforme). Luego, el lapso debe ser una cantidad no trivial, como podría ser arbitrariamente

$$\alpha = 1 + \frac{r^2}{1+r^2}\, e^{-r^2}\,. \tag{5.5}$$

Nótese que este lapso es regular en los ejes donde también es par y asintóticamente va a uno. Así que aunque es una cantidad arbitraria, cumple ciertos requisitos comunes a un lapso. Entonces, si se procede de manera análoga (es decir comparando la salida del código contra valores analíticos) se obtiene la Figura 5.3 que demuestra que también hay convergencia al orden esperado.

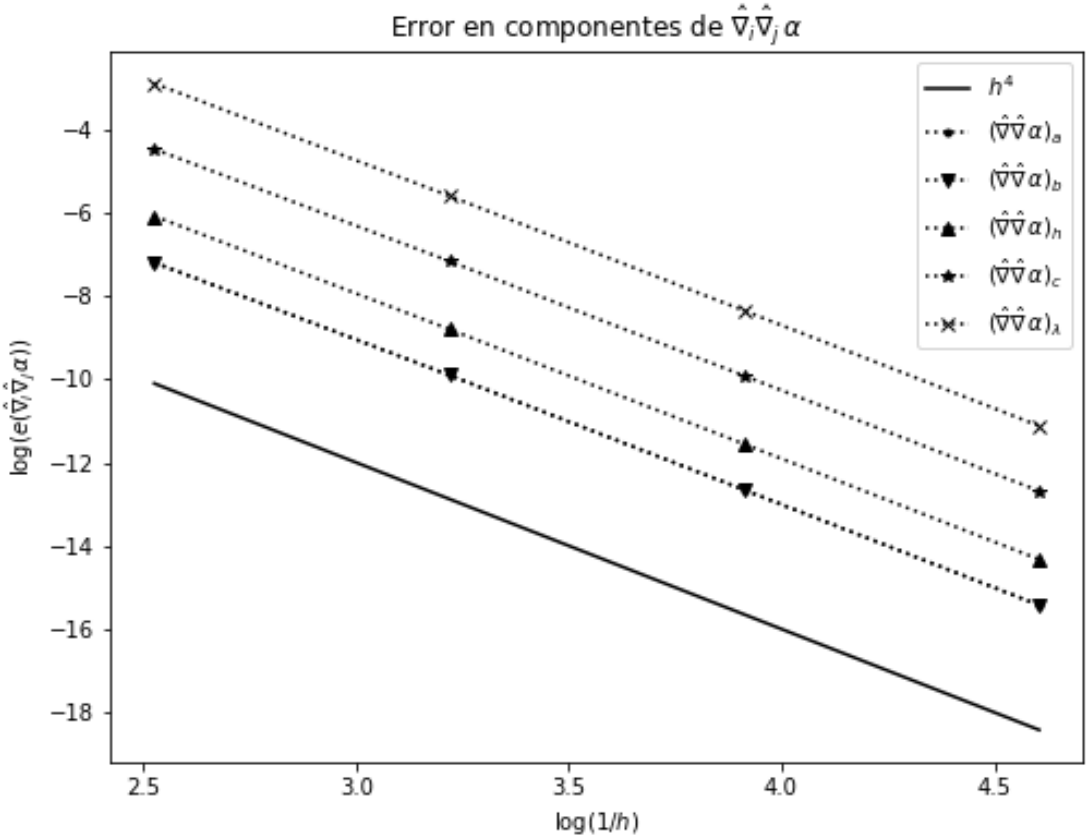

Figura 5.3: Gráfica log vs. log del error en los componentes de la segunda derivada covariante del lapso $\nabla_i \nabla_j \alpha$ asociados a la métrica y lapso prueba. Se han dibujado los cuatro componentes a, b, h, c y el de regularización λ. En adición, la curva sólida dibuja $\mathcal{O}(h^4)$.

5.2. Condiciones de frontera

Como se anticipó desde el capítulo anterior donde se enseña la forma de la malla computacional en la Figura 3.1, el código tiene un tratamiento especial para los puntos de frontera. El problema de las condiciones de frontera suele ser uno de los problemas más complicados de la Relatividad Numérica. Existen varios tipos de condiciones como se pueden consultar en [1]. Sin embargo, de todas las presentadas en la referencia anterior, sólo se utiliza la más sencilla: la condición de frontera radiativa. La razón a la que se debe esto es porque las demás condiciones, i.e. una frontera maximalmente disipativa y una frontera que preserva las constricciones, son muy complicadas al tener que descomponer en eigen-campos (para una discusión de eigen-campos, hiperbolicidad y cómo garantizar que las condiciones de frontera estén bien planteada, se refiere de nuevo a [1]).

La condición de frontera radiativa viene de de asumir que en la frontera de la malla ($r \gg 1$) las variables se comportan como ondas esféricas, lo que significa que

$$f(r) \approx f_\infty + \frac{f_1(r - vt)}{r}\,, \tag{5.6}$$

donde f_∞ es el valor del campo en infinito (usualmente cero o uno) y f_1 es una función en cuyo

argumento está el comportamiento clásico de onda. Es decir, f se comporta como una onda esférica *saliente* con velocidad v. Esta frontera se utiliza mucho en electromagnetismo, donde se llama condición de frontera de tipo Sommerfeld.

Análogamente a la condición de Robin (3.12) tratada antes, la ecuación anterior (5.6) debe reescribirse en términos de las derivadas de f, por lo que se concluye

$$\partial_t f + v\,\partial_r f + v\,\frac{f - f_\infty}{r} \approx 0\,. \tag{5.7}$$

Para implementar esta condición de frontera se asume que f es una variable dinámica del sistema BSSN, por ejemplo K cuya ecuación de evolución (1.37) se reescribe abajo:

$$\partial_t K = \pounds_\beta K - \nabla^2 \alpha + \alpha \left(\hat{A}_{ij}\,\hat{A}^{ij} + \frac{1}{3} K^2 \right) + 4\pi\alpha\,(\rho + S)\,. \tag{5.8}$$

Dicha expresión permite calcular la fuente de K, i.e. $\partial_t K$ en toda la malla, pues todas las cantidades se pueden calcular aunque sea utilizando diferencias finitas ladeadas en la frontera. Sin embargo, para implementar la condición de frontera, en los puntos de frontera no se calcula $\partial_t K$ via el sistema BSSN sino como

$$\partial_t K = -v \left(\partial_r K + \frac{K - K_\infty}{r} \right)\,. \tag{5.9}$$

La derivada radial $\partial_r K$ se puede aproximar en la malla cartesiana via la regla de la cadena como se hizo para Robin (véase (3.14), (3.15) y (3.16)). Ahora sería un buen momento decir quién es K_∞ y v. Si la frontera está suficientemente alejada y lejos se tiene esencialmente una métrica plana, la velocidad de la luz será aproximadamente $v = 1$. Pero esto no es cierto siempre. De hecho, para lapsos de la familia Bona-Massó se puede ver (véase por ejemplo [1]) que las variables se propagan asintóticamente con $v = \sqrt{f(\alpha)}$, así que el código debe tener considerado este hecho por si se está evolucionando con lapso $1 + \log$. El valor de K_∞ también se puede inferir a partir de asumir de que lejos se tiene una situación mayormente estática y asintóticamente plana. Ergo, para las cantidades como son $K, \hat{A}_{ij}, \hat{\Delta}^i$ y los componentes no-diagonales de la métrica tienen valor cero, mientras que la métrica diagonal y el lapso tienen valor uno.

Con base en [1], hay que justificar que se cumplen los tres prerequisitos de la frontera radiativa:

1. *El espacio tiempo es asintóticamente plano.* Esto es sencillo de ver para ondas de Brill y se debe a que el factor conforme $\tilde{\psi}$ tiene como condición de frontera que se comporte como Schwarschild $1 + \frac{2M}{r}$ y se sabe que Schwarschild es asinóticamente plano.

2. *Las fuentes está localizadas en una región específica.* También es cierto para ondas de Brill. Recuérdese que se mencionó en el capítulo de ondas de Brill que tanto la magnitud de la fuente lineal así como su derivada angular tienen valor del orden a 10^{-15} para un radio mayor a $r = 8$. Por lo tanto, se puede asumir que el comportamiento de las variables será el de un frente esférico suficientemente lejos.

3. *El vector de corrimiento es pequeño en la frontera.* Esto se cumple trivialmente si se pone como valor cero constante. Normalmente esto sólo es relevante para corrimientos corrotantes.

Aunque se cumplen todas las condiciones anteriores, es bien conocido que esta condición de frontera viola las constricciones y es natural esperar que metan error al código. De hecho, la violación

a las constricciones es tan grande que si se toma la norma[2] de las constricciones en toda la malla el análisis de convergencia indica que no hay convergencia al tomar distintas resoluciones. Esto se debe a que el ruido de las fronteras empieza a propagarse hacia dentro y su magnitud es independiente a la resolución utilizada en la evolución, lo que hace que el código no converja.

Para ver este fenómeno se presenta a continuación la constricción de momento M^z graficada sobre la diagonal en la Figura 5.4. La evolución[3] se realiza con tres resoluciones h_0, h_1, h_2 con valores de $1/16, 1/32, 1/64$ y puntos internos $N = 128, 256, 512$, respectivamente. De esta manera, la frontera está en aproximadamente $r = 8$, aunque la diagonal se extiende hasta $8\sqrt{2} \approx 11.3$.

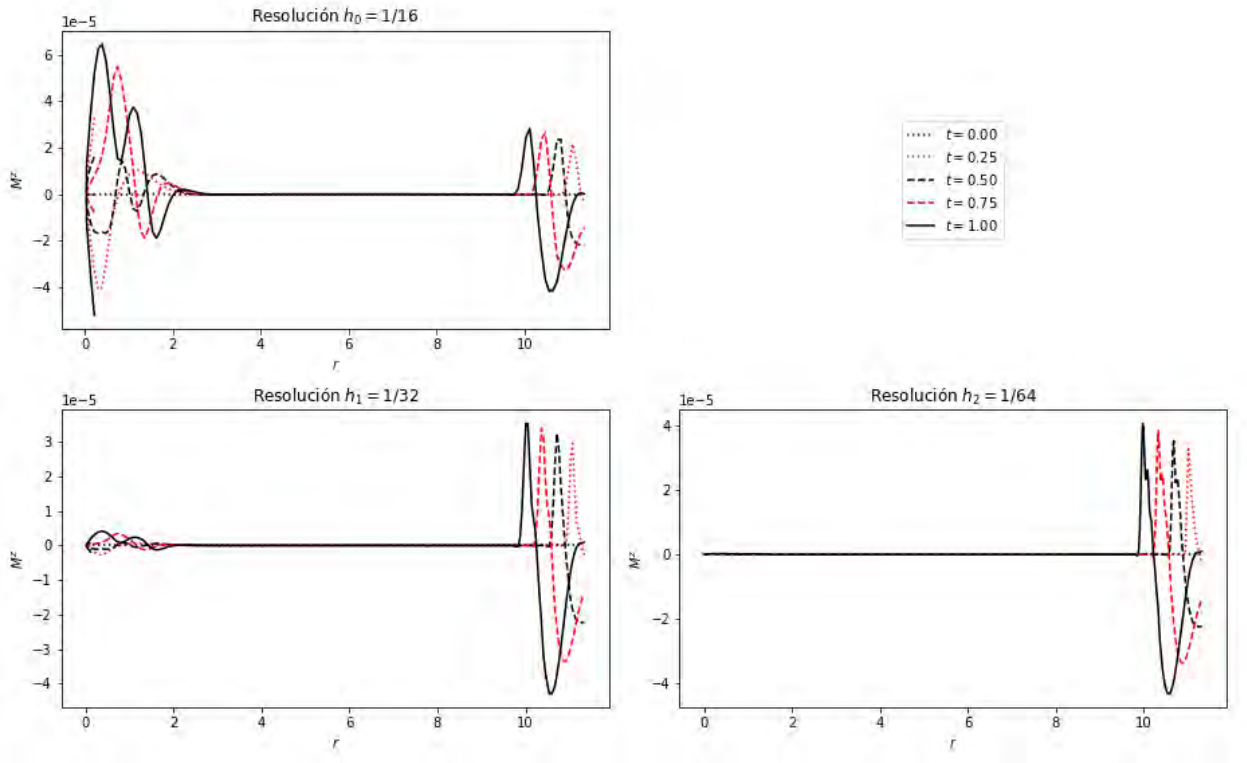

Figura 5.4: Evolución de la constricción de momento M^z para tres resoluciones h_0, h_1, h_2 sobre la diagonal. Se presentan cinco instantáneas correspondientes a $t = 0.0, 0.25, 0.50, 0.75, 1.0$.

La Figura 5.4 respalda por completo la discusión anterior, la violación de las fronteras siempre es del mismo orden $M^z \approx -4 \times 10^{-5}$ mientras que la parte interna sí se está reduciendo. Además, el avance de las constricciones también es evidente y se infiere que si se evoluciona un tiempo suficiente, las violaciones de la frontera terminarán tocando la parte interna.

Por lo tanto, si se va a determinar convergencia con las normas de las constricciones, no es posible hacer esto si se está calculando sobre la malla entera. Con base en este hecho, el código *OllinBrill* tiene dos opciones para lidiar con que el ruido de las fronteras va cayendo hacia dentro de la malla.

[2]Cuando se habla de norma en este trabajo se habla de la norma L_2, $\|u\|_2 = \sqrt{\sum_i^N u_i^2/N}$, salvo que se indique lo contrario.

[3]Por el momento no es tan importante decir que la evolución son datos iniciales de Brill $a = 1$ ya que la sección destinada a discutir los resultados del código es el siguiente Capítulo 6.

- Calcular sobre una malla que se va reduciendo adaptativamente a una velocidad $v = 1$. Es decir, si se empieza con una malla de dimensiones $\rho_{\max} \times z_{\max}$, entonces a tiempo t, el cálculo de las normas se hace sobre una malla $(\rho_{\max} - t) \times (z_{\max} - t)$. Naturalmente, esto no está definido para evolucionar tiempos mayores al mínimo de $\rho_{\max}, z_{\max}$ y de hecho no se permite evolucionar después de $t = \text{mín}(\rho_{\max}, z_{\max})/2$.

- Tomar las normas sobre una malla reducida fija, i.e. $\rho_{\max}/2 \times z_{\max}/2$. Al igual que en la opción anterior, no se recomienda evolucionar después de $t = \text{mín}(\rho_{\max}, z_{\max})/2$.

En la práctica, la segunda opción es más sencilla porque no se necesita estar haciendo cálculos de qué puntos están dentro de la malla adaptativa.

Con cualquiera de estas opciones se encuentra que el código sí converge, pero dicha discusión se tiene en el siguiente Capítulo 6.

5.3. Análisis CFL

Es bien sabido que las ecuaciones hiperbólicas requieren un estudio cuidadoso de la condición CFL (Courant-Friedrichs-Levy [22]) que determina qué tan grande puede ser el paso de tiempo para que no haya violación a la causalidad y el código converja al integrar los pasos de tiempo. Si dicho paso de tiempo se denota como Δt y h es el menor de los pasos de la malla (en el problema bidimensional $h = \text{mín}\,(\Delta\,\rho,\, \Delta\, z)$), se define el factor de Courant como

$$\Delta\, t = C_{\text{CFL}}\, h\,, \tag{5.10}$$

Es bien sabido que $C_{\text{CFL}} = 1$ para la típica ecuación de onda unidimensional con velocidad constante $c = 1$ (véase [38] o incluso [1]). Sin embargo, de manera general y en más dimensiones, la condición CFL determina una restricción sobre la velocidad de la luz c (véase [58]) via

$$c \leq \frac{1}{\Delta\, t} \left(\sum_{i=1}^{n} \frac{1}{\Delta\, x_i} \right)^{-1/2} \leq \frac{h}{\sqrt{n}\, \Delta\, t} = \frac{1}{\sqrt{n}\, C_{\text{CFL}}}\,, \tag{5.11}$$

donde se suma sobre las n dimensiones $i = 1, 2, \ldots n$ y sus pasos asociados. Esta condición se suele reescribir como

$$C_{\text{CFL}} \leq \frac{1}{\sqrt{n}\, c}\,. \tag{5.12}$$

Lo que nos dice la ecuación (5.12) anterior es que si se calcula la velocidad de la luz c localmente sobre la malla y después se conoce su máximo, es posible determinar C_{CFL} para el paso de integración futuro. Aunque el problema es bidimensional, en este cálculo se toma $n = 3$ para tomar en cuenta que el espacio completo es tridimensional.

Antes de continuar, es necesario saber cómo calcular la velocidad de la luz. Específicamente se quiere la velocidad de la luz en una dirección específica. Aquí lo que se hace es calcular la velocidad en las tres direcciones ortogonales ρ, z, φ. Gracias a [1] se puede afirmar que el componente longitudinal de la velocidad de la luz en la dirección i es

$$c_i = \alpha \sqrt{\gamma^{ii}}\,. \tag{5.13}$$

Entonces, para conocer C_{CFL} se calculan las tres velocidades de manera local y luego se determina el máximo de las tres. De esta manera se determina un valor máximo para C_{CFL}.

En este punto el código tiene dos opciones. La primera es que a la hora de empezar la simulación se haya elegido tener una factor de Courant fijo. Si es así y la fase de análisis CFL determina que este factor es demasiado grande, se manda una advertencia a pantalla declarando que se pueden generar inestabilidades. La segunda opción es la más interesante porque permite implementar un paso temporal variable que se calcula a partir de tener un factor de Courant calculado en cada análisis. De esta manera se garantiza que el dominio de dependencia numérica será menor que el dominio de dependencia física y la evolución será estable. Para mayor información se puede consultar el código en [44], específicamente las subrutinas en el archivo *metric_cfl_analysis.cpp*.

El cálculo del factor de Courant es especialmente relevante para el estudio de fenómeno crítico para ondas de Brill, como se verá en Capítulo 6.

5.4. Archivo de parámetros

El repositorio [44] tiene un archivo de parámetros ejemplo comentado y explicando las opciones. Para darle mayor completitud, aquí se explican los parámetros del código *OllinBrill*. El archivo de configuración se procesa con la librería *libconfig* (véase [40]) y los comentarios en este archivo empiezan con el numeral.

Primero se tienen los parámetros de la malla así como el tiempo final de evolución.

```
# PARÁMETROS DE LA MALLA.
# Pasos espaciales en r y z.
dr = 0.0625
dz = 0.0625
# Numero de puntos internos en cada dirección.
NrInterior = 128
NzInterior = 128

# TIEMPO.
# Si se evoluciona con CFL adaptativo se da simplemente el
# tiempo final. Se recomienda un múltiplo del mínimo de dr, dz.
FinalTime = 25.00
# En cambio, si se prefiere un factor de Courant fijo, se
# proporciona su valor dtfac distinto de cero y que se hagan
# Nt pasos de dtfac * min(dr,dz).
dtfac = 0.0
Nt = 0
```

Aquí se especifican los pasos espaciales en las direcciones ρ, z (en el código la letra `r` representa ρ mientras que `rr` representa el radio r). Después se tienen los puntos N_ρ, N_z. Como la malla es lineal, esto determina su extensión. Como se comentó arriba sobre la condición CFL, hay dos opciones. Simplemente decirle al código un tiempo final y que este evolucione con un paso de tiempo adaptativo que procura no violar la condición CFL. Alternativamente, el usuario puede proporcionar un factor C_{CFL} fijo (escrito en el código como `dtfac`) y un número de pasos de tiempo.

La siguiente sección determina el tipo de evolución así como la condición de frontera y hasta dónde se está calculando la norma de las constricciones.

```
# EVOLUCIÓN.
# Orden de evolución: "two", "four" para segundo y cuarto
# orden respectivamente.
order = "four"
# Tipo de integrador: "icn", "rk4".
integrator = "rk4"
# Eta BSSN. Valor estándar es 2.
eta = 2.0
# Tipo de evolución según evolucione o no el determinante.
# "lagrangian": no evoluciona, "eulerian" sí evoluciona.
bssn_flavor = "lagrangian"

# FRONTERA.
# Opciones para frontera plana, estática o radiativa.
# Es decir, "flat", "static", "radiative".
boundtype = "radiative"
# Calcula la norma de las constricciones hasta solo la mitad de la malla,
# i.e. hasta NrInterior/2, NzInterior/2. 0: apagado, 1: habilitado.
# No compatible con adaptive_boundary_norm.
interior_norm = 1
# Reduce la malla sobre la que se calcula las constricciones
# adaptativamente. 0: apagado, 1: encendido.
# No compatible con interior_norm.
adaptive_boundary_norm = 0
```

Aquí se resalta que el 'sabor' BSSN corresponde a (1.29) donde $s = 0$ para una condición Lagrangiana y $s = 1$ para una condición Euleriana. Si se tiene una evolución Lagrangiana, el determinante no evoluciona, lo cual significa que $\hat{\gamma} = \mathring{\gamma} = \rho^2$ durante toda la evolución, cosa que puede simplificar algunas ecuaciones en el código. El resto de las opciones corresponden a la discusión de la sección anterior respectiva a la frontera. Es decir, la opción `interior_norm` sólo calcula las normas en la malla $\rho_{\max}/2 \times z_{\text{máx}}/2$ y la opción `adaptive_boundary_norm` corresponde a la malla adaptativa con velocidad $v = 1$.

Los parámetros que siguen están asociados al lapso y vector de corrimiento.

```
# LAPSO
# Lapsos posibles: 1+log, armónico, maximal. Correspondientes
# a "1+log", "harmonic" y "maximal", respectivamente.
slicing = "maximal"
# Lapso inicial. Constante igual a uno: "one"; precolapsado
# alpha = 1/psi**n con n = 2 ó n = 4 son "precolapsed2" y "precolapsed4";
# isotrópico: "isotropic"; o gaussiano: "gaussian".
# Lapso maximal debe tener ilapse = "one".
ilapse = "one"
# Parámetros asociados al lapso gaussiano.
gl_alpha0 = 0.00
gl_rr0 = 0.00
# Velocidad de norma para el lapso de Bona-Massó.
# 1+log tiene gauge_f = 2.0 y lapso armónico gauge_f = 1.0.
gauge_f = 0.0
```

Es claro que el código tiene tres opciones de lapso correspondientes a armónico[4], $1+\log$ y lapso maximal. El lapso $\alpha(t=0)$ se puede determinar también como constante e igual a uno, precolapsado (véase [1])

$$\alpha(t=0) = \frac{1}{\psi^n}\,, \tag{5.14}$$

con $n = 2, 4$ y de tipo gaussiano. La forma explícita del lapso gaussiano (6.1) se discute en el Capítulo 6 subsecuente.

La siguiente sección del archivo de parámetros es la que determina la geometría inicial del espacio.

```
# DATOS INICIALES
# Ondas de Brill (BrillWave), Minkowski, Schwarschild, TestBed.
idata = "BrillWave"

# Parámetros de Brill con q tipo Holz.
brill_a0 = 11.0
brill_sr0 = 1.0
brill_sz0 = 1.0

# Masa de Schwarschild.
schwar_mass = 0.0
```

Las geometrías válidas son Minkowski, Schwarshild (para quien hay que especificar la masa `schwar_mass`), ondas de Brill con q tipo Holz (2.17) y la geometría de prueba que se utilizó para revisar las subrutinas de geometría (ecuaciones (5.1) y (5.4)).

[4]El lapso armónico es también de la familia Bona-Massó con $f(\alpha) = 1$. Dicho lapso se discute con mayor interés en el siguiente Capítulo 6 donde se evoluciona Minkowksi con lapso armónico.

Para tener un control sobre el resolvedor elíptico hay que establecer los siguientes parámetros:

```
# RESOLVEDOR ELÍPTICO
# Calcular el lapso maximal cada paso de integración ("every"), a dt/2 ("half"),
# o al paso completo dt ("full").
ell_freq = "every"
# Usar la permutación calculada en el primer paso (1). No usarla (0).
use_permutation = 0
# Volver a calcular la permutación después de perm_count_max.
# Si es negativa, no se vuelve a calcular en toda la evolución.
perm_count_max = 0
# Usar precondicionador CGS después del primer paso (1). Si no (0).
# Tolerancia CGS 10**(-precond_L).
precond_L = 0
# Tipo de operador de Robin: 1,2,3.
nrobin = 1
```

Como primera opción está si se debe estar usando el resolvedor en cada paso de integración. La experiencia encontró que para la mayor de los casos, si se está evolucionando a segundo orden, se puede calcular el lapso cada paso completo de integración y conservar la convergencia. Sin embargo, a cuarto orden es necesario calcular el lapso cada paso de integración si se quiere convergencia a cuarto orden. Las demás opciones están relacionadas a optimizar el resolvedor utilizando la permutación y precondicionador explicados en el capítulo del resolvedor elíptico.

Continuando sobre el archivo de parámetro, se tiene el control sobre el enontrador de horizontes.

```
# AHF
# Usar (1) o no usarlo (0) durante la evolución.
ahfind_flag = 1
# Interpolar con bilinear (0) o bicúbica (1).
ahfind_bicubic = 1
# Buscar AH cada t = ahfind_freq * min(dr,dz).
ahfind_freq = 2
# Aplicar máscara del AH sobre las constricciones (1). Si no (0).
ahfind_mask_filter = 1
# Iniciar AHF a tiempo ahfind_start_time.
ahfind_start_time = 0.80
# Radio máximo y mínimo donde se busca el AH.
ahfind_maxr = 1.50
ahfind_minr = 1.20
# Número máximo de iteraciones de bisección utilizadas.
ahfind_bis_iter = 50
```

La última parte del archivo es la que corresponde a imprimir archivos.

```
# SALIDA Y CONSTRICCIONES.
# Nombre del directorio donde se escribe.
dirname = "Brill Collapse Long Evolution"
# Calcula las constricciones cada t = constraintCheck * min(dr, dz).
constraintCheck = 2
# Escribe mallas bidimiensionales cada t = Noutput2D * min(dr, dz).
Noutput2D = 2

# Variables a escribir: apagada (0), encendida (1).
# Constricciones.
ham = 1
mom_r = 1
mom_z = 1
# Lapso.
alpha = 1
# Vector de corrimiento.
beta_r = 0
beta_z = 0
# Métrica.
phi = 1
a = 1
b = 1
h = 1
c = 1
lambda = 0
# Curvatura.
K = 0
A_a = 1
A_b = 1
A_h = 1
A_c = 1
A_lambda = 0
```

Se pueden escribir todas las variables declaradas en el archivo cabezal *arrays.h*. Como son más de 300, no están todas escritas en el ejemplo de arriba.

Capítulo 6

Resultados

En este capítulo se presentan los resultados del código de evolución. Como primera parte se comprueba la convergencia para casos sencillos de Minkowski y Schwarschild. En dichas instancias se examinan lapsos de tipo armónicos, 1+log y maximal. Como se anticipó en el Capítulo 4 del AHF, también se pone énfasis especial en la evolución del horizonte para Schwarschild, específicamente que la masa del horizonte se mantenga cercana a su valor analítico durante toda la evolución.

Habiendo mostrado la convergencia y buen funcionamiento del código de evolución, la segunda parte de este capítulo trata de los resultados con ondas de Brill. Primero se estudia el caso de una onda de amplitud relativamente débil con $a = 1$. En esta evolución se espera que, al poder ser vista casi como perturbación de Minkowski, la evolución termine con el mismo espacio plano de Minkowski.

El extremo contrario que se estudia es el de una onda de amplitud fuerte con $a = 11$ (hay que recordar que para $a \approx 11.816$ ya existe un AH en los datos iniciales). La evolución da paso a que eventualmente haya un colapso a un AH en $t \approx 1$. Haciendo uso de una buena resolución y de las capacidades de cuarto orden del código, se presenta una evolución hasta $t = 25$ con lapso maximal. Esta evolución es sin duda el resultado más importante de este trabajo, ya que pone a prueba todos los aspectos del código. Desde la formulación covariante, su regularización subsecuente, el resolvedor elíptico, el lapso maximal, el AHF y la integración para obtener la masa del horizonte encontrado. La larga evolución enseña que el valor de esta última masa siempre se encuentra debajo de la masa ADM que también se calcula. Como comentario, también en esta parte se presentan resultados que dan peso importante a por qué es tan crucial que la evolución sea a cuarto orden y no a segundo orden que es sin duda más sencilla de implementar y programar.

Como última sección para este capítulo se presentan resultados para una evolución cercana a la amplitud crítica con $a = 4.5$ (Alcubierre *et. al.* estiman en [5] que la amplitud crítica está en $a_* \approx 4.8$). En dicha evolución, el lapso oscila de tal manera que si no se tiene una alta resolución, el código deja de converger al orden esperado de cuarto orden. El código de este trabajo sí puede lidiar con estas oscilaciones y de hecho, al examinar el comportamiento de la velocidad de la luz localmente, se ve que hay una caída fuerte del factor de Courant en el tiempo en el que el lapso está rebotando de una oscilación. Dicha caída se ve sólo para las mayores de las resoluciones, lo que posiblemente explica por qué deja de haber convergencia si no hay una alta resolución. El estudio del fenómeno crítico queda fuera del enfoque de este trabajo pero los resultados presentados sin duda dan la motivación a hacer investigaciones futuras que lo estudien.

6.1. Pruebas de convergencia con Minkowski y Schwarschild

6.1.1. Minkowski con lapso armónico

La prueba más simple que se le puede hacer al código es poner datos iniciales muy sencillos (incluso con simetría esférica y por ende sin términos cruzados como $\hat{\gamma}_{\rho z}$). La evolución se hace con un lapso de la familia Bona-Massó, ya que así se pueden revisar más términos que con una evolución de lapso maximal donde ni K ni ϕ evolucionan.

Por lo tanto, Minkowski se elige como la geometría inicial. Sin embargo, para que haya una evolución no trivial, se necesita una lapso inicial α distinto de uno. Para ello, el lapso inicial puede ser una cáscara esférica gaussiana

$$\alpha(\rho, z) = 1 + \frac{\alpha_0\, r^2}{1 + r^2} \left(e^{-(r+r_0)^2} + e^{-(r-r_0)^2} \right) , \tag{6.1}$$

donde α_0, r_0 son parámetros que caracterizan la amplitud del pulso gaussiano y dónde está centrado. Para esta primera evolución se utilizan $\alpha_0 = 1/100$ y $r_0 = 2$ que se puede ver en la siguiente Figura 6.1.

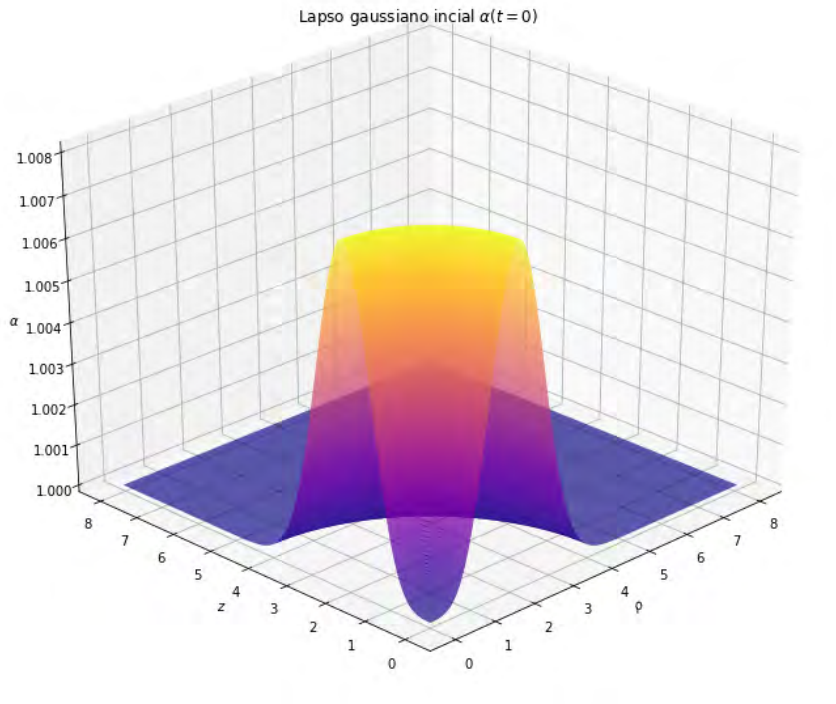

Figura 6.1: Lapso gaussiano inicial.

La evolución procede con un lapso de la familia Bona-Massó. Específicamente un lapso armónico donde

$$\frac{\mathrm{d}\alpha}{\mathrm{d}t} = -\alpha^2\, K\,, \tag{6.2}$$

es decir, $f(\alpha) = 1$ para la expresión (1.44). Este lapso se denomina armónico porque viene de pedir que las coordenadas satisfagan un operador armónico $\Box\, x^\alpha = 0$, lo cual produce la ecuación anterior al traducir a la formulación 3 + 1 (véase por ejemplo [1]).

Al poner estos datos iniciales, se corre la evolución hasta $t = 4$ para tres resoluciones distintas con $h = 1/16, 1/32, 1/64$ y con puntos interiores en cada dimensión iguales a $N = 128, 256, 512$, respectivamente. Esto quiere decir que la frontera queda aproximadamente en $\rho, z = 8$ y se impide que caigan efectos de la frontera a la parte central de evolución al sólo evolucionar hasta $t = 4$. Con esto en mente, se procede a examinar la convergencia del código con la inspección de cómo se comportan las constricciones. Para empezar, tenemos las constricciones más importantes:

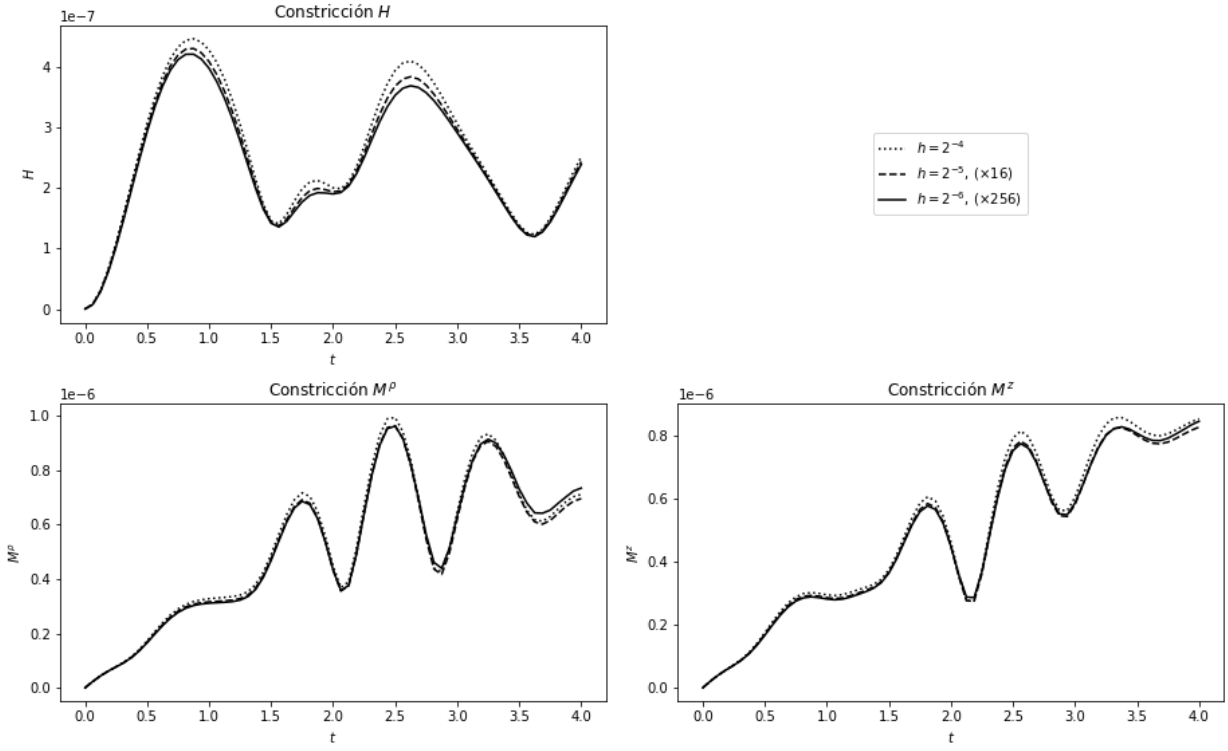

Figura 6.2: Evolución de las constricciones hamiltoniana y de momento para datos iniciales de Minkowski con un lapso inicial gaussiano y lapso armónico. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

En la figura anterior (Figura 6.2) y en todas las posteriores donde se pongan constricciones, se reescalan las curvas de mayor resolución para confirmar si hay convergencia al orden esperado. En este caso, la resolución base es $h_0 = 1/16$. Si esperamos convergencia a cuarto orden, la siguiente resolución con $h_1 = h_0/2$ deberá reducir las constricciones por un factor de $2^4 = 16$. Del mismo modo, $h_2 = h_0/4$ tendría que reducir las constricciones por $4^4 = 256$. Estos factores de 16 y 256 son justamente los que se usan para reescalar las curvas, por lo que se comprueba que hay convergencia a cuarto orden.

Las siguientes gráficas demuestran que también hay convergencia para las otras constricciones de las variables que se promueven a variables independientes en la formulación covariante. Es decir, $\hat{\Delta}^\rho$, $\hat{\Delta}^z$, λ, $\hat{A}_\lambda$.

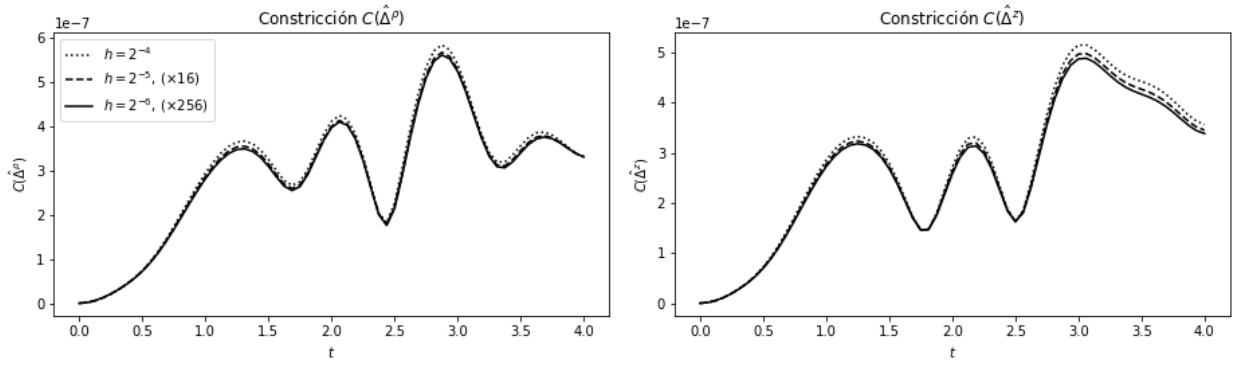

Figura 6.3: Evolución de las constricciones del vector $\hat{\Delta}^i$ para datos iniciales de Minkowski con un lapso inicial gaussiano y lapso armónico. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

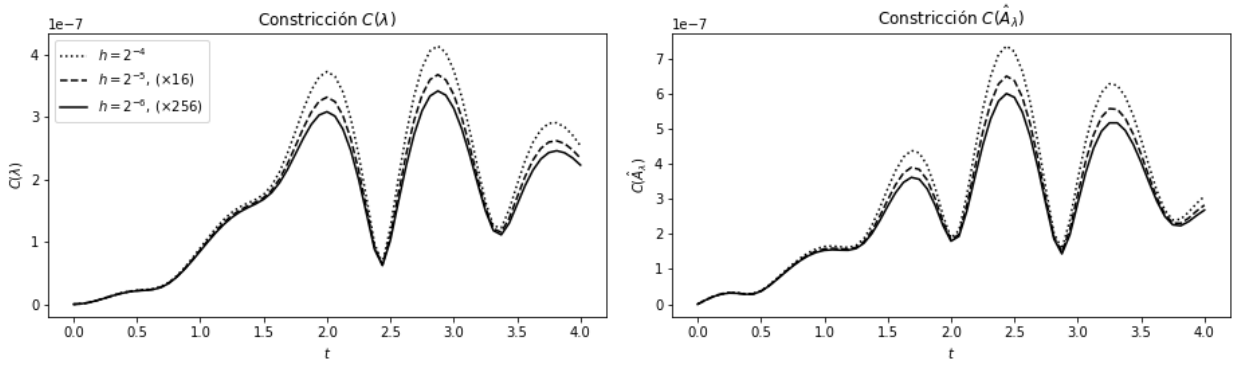

Figura 6.4: Evolución de las constricciones de las variables de regularización λ y $\hat{A}_{\lambda}$ para datos iniciales de Minkowski con un lapso inicial gaussiano y lapso armónico. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

Las dos figuras anteriores (Figuras 6.3 y 6.4) no sólo verifican la convergencia de la simulación sino que comprueban que la formulación covariante está bien fundamentada (por medio del vector $\hat{\Delta}^i$ introducido) y que la regularización también ha funcionado apropiadamente.

Ahora que se ha verificado la convergencia a cuarto orden de la evolución, se puede realmente examinar alguna cantidad física, por ejemplo el lapso. Un hecho bien conocido del análisis de Fourier es que la forma gaussiana del lapso inicial en (6.1) puede descomponerse en componentes con velocidad entrante y otros con velocidad saliente. Por lo tanto, se espera que el pulso inicial se separe en uno que salga al infinito y otro que caiga al origen. Al caer al origen, la condición de paridad y regularidad del lapso nos anticipan que el pulso deberá rebotar de manera suave y regresar al infinito. Esto se comprueba si graficamos el valor del lapso en el punto más cercano al origen (recuérdese que la malla computacional no toca a los ejes ni mucho menos al origen, sin embargo, cuando se hable del origen se estará hablando del punto $(\rho, z) = (h/2, h/2)$ que se acerca más a dicho punto a medida que aumentamos la resolución).

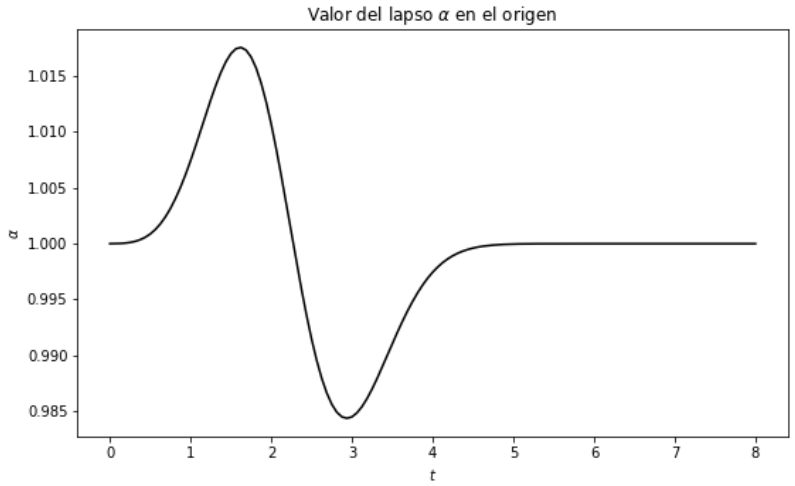

Figura 6.5: Evolución del valor del lapso en el origen.

Dicha Figura 6.5 fundamenta los comentarios anteriores y ergo demuestra como 'bonus' que la condición de paridad está bien implementada. Para obtener una figura con mayor carácter ilustrativo, se grafica ahora el lapso en la diagonal a distintos tiempos (esto se conoce como *snapshots* en inglés o instantáneas en español y podemos aceptarla como la mejor opción para visualizar una cantidad dinámica en una hoja de papel estática).

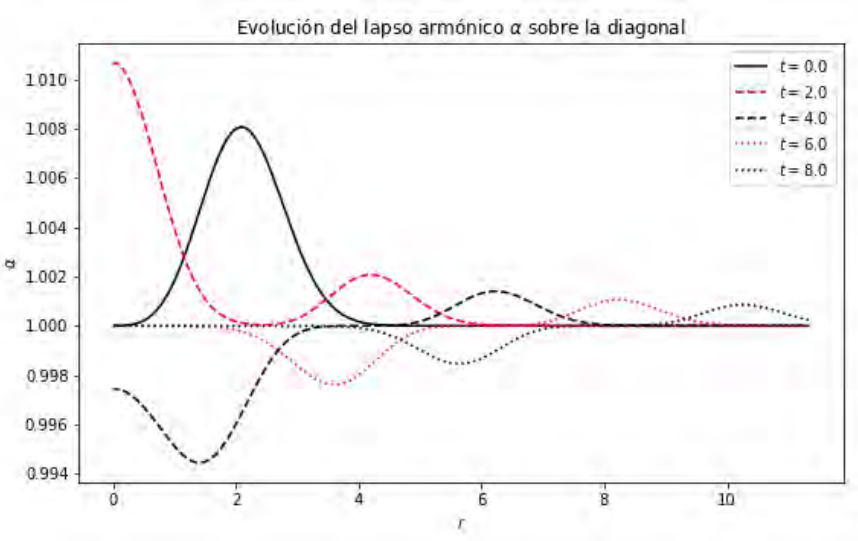

Figura 6.6: Instantáneas del lapso armónico sobre la diagonal. Se presentan cinco tiempos de evolución $t = 0, 2, 4, 6, 8$.

Para interpretar con mayor accesibilidad la Figura 6.6, el color de las curvas cambia de negro a rojo entre las instantáneas sucesivas. Se empieza de la curva negra sólida con la forma gaussiana inicial compuesta de un solo pulso. A $t = 2$ (la curva roja discontinua) se ve la clara separación de los pulsos, con un pulso ya dentro del origen. Después a $t = 4$ (la curva negra discontinua) el lapso ya ha rebotado y está en proceso de dejar el origen, de tal manera que para los futuros tiempos no queda rastro del pulso en el origen y ambos pulsos están alejándose a la velocidad de la luz.

6.1.2. Schwarschild con lapso $1 + \log$

Ahora continuamos con una prueba menos trivial que es la evolución de un agujero negro de Schwarschild con $M = 1$ con la mismas mallas y resoluciones que se usaron en Minkowski. Como se quiere inspeccionar la mayor cantidad de términos por si hubiera algún error, se sigue utilizando un lapso de tipo Bona-Massó pero ahora se trata del lapso $1 + \log$ que además, siguiendo a [1] comienza precolapsado, i.e.

$$\alpha(t = 0) = \frac{1}{\psi^2}\,, \tag{6.3}$$

donde $\psi = 1 + \frac{2M}{r}$ es el factor conforme de Schwarschild. Dicho lapso tiene la siguiente forma sobre la diagonal:

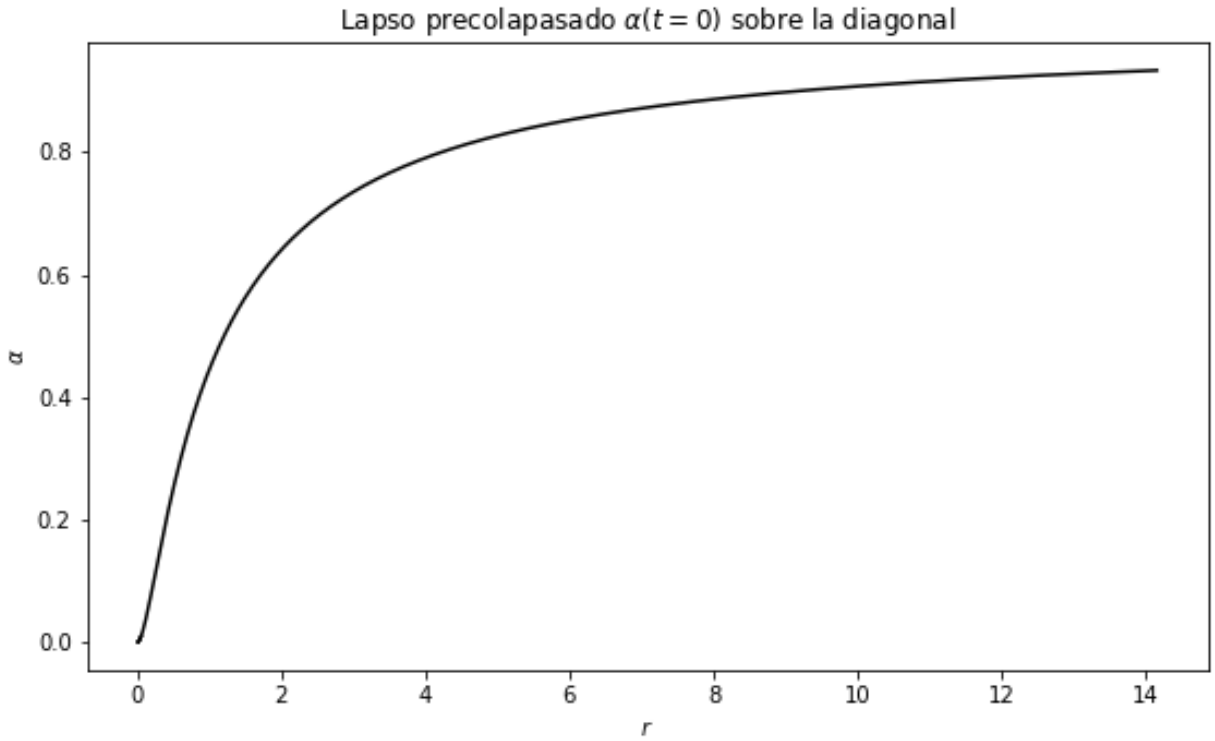

Figura 6.7: Lapso inicial de tipo precolapsado con $\alpha = 1/\psi^2$ para Schwarschild con $M = 1$. Se presenta el lapso sobre la diagonal.

El interés primordial en esta sección es verificar la convergencia de las constricciones y que la masa del AH se mantenga cercana a uno durante la misma evolución. Análogamente a la sección anterior, se presentan primero las gráficas de las constricciones.

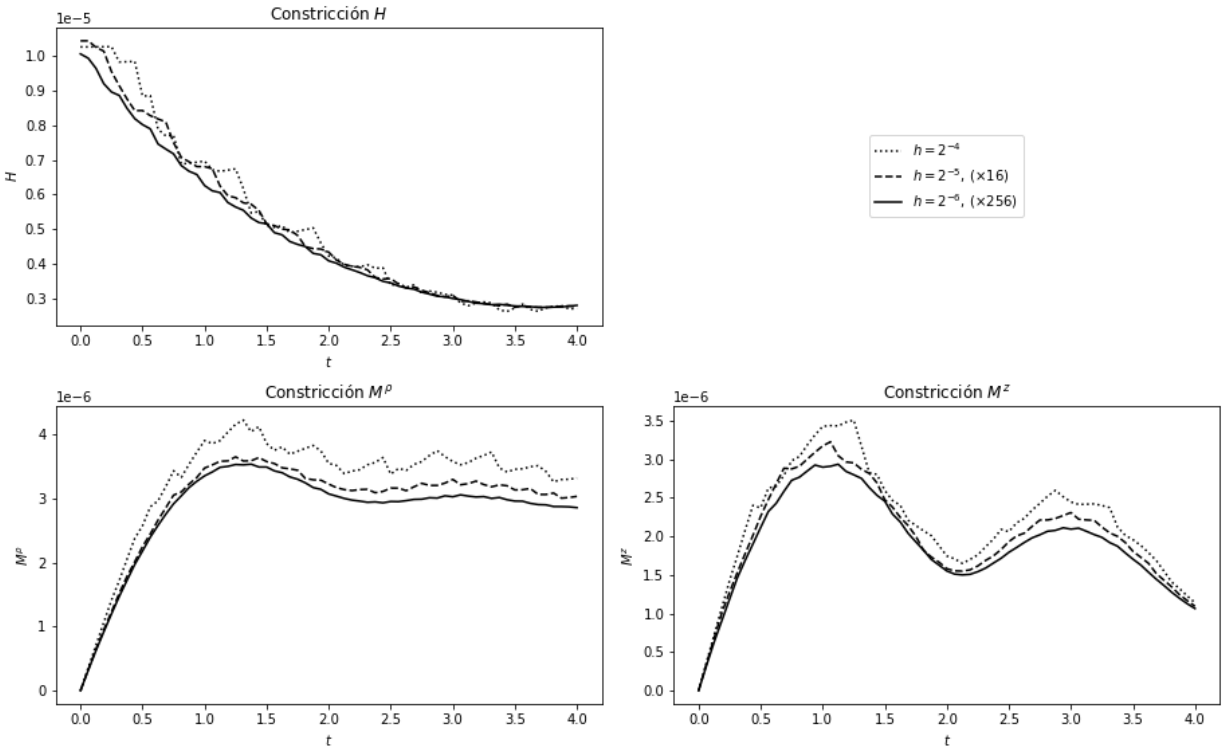

Figura 6.8: Evolución de las constricciones hamiltoniana y de momento para datos iniciales de Schwarschild $M = 1$ con lapso $1 + \log$. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

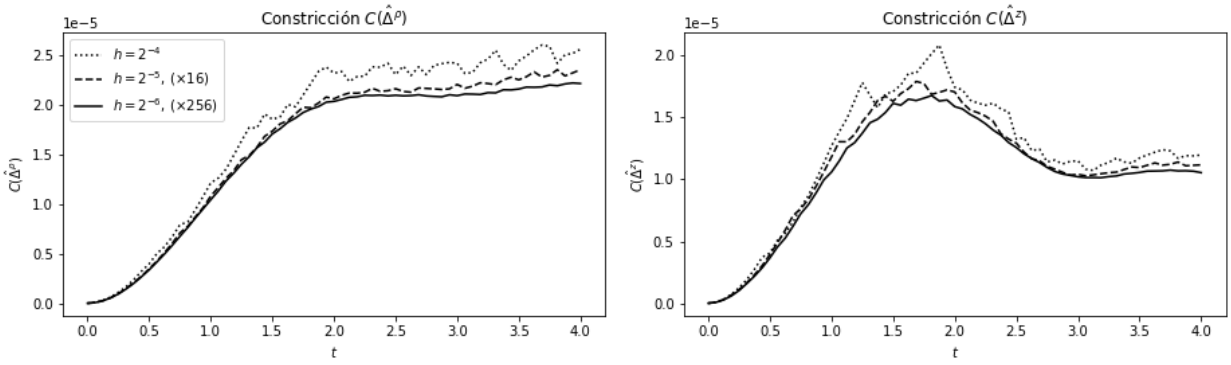

Figura 6.9: Evolución de las constricciones del vector $\hat{\Delta}^i$ para datos iniciales de Schwarschild $M = 1$ con lapso $1 + \log$. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

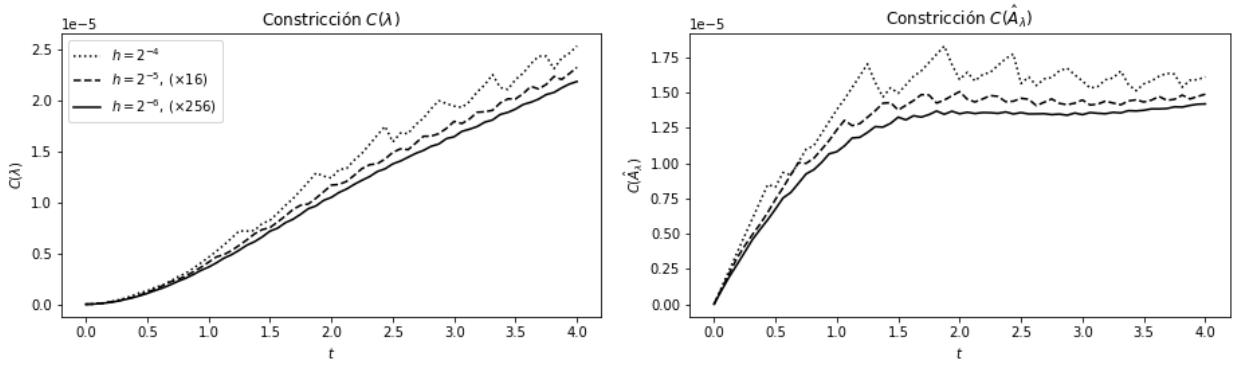

Figura 6.10: Evolución de las constricciones de las variables de regularización λ y $\hat{A}_\lambda$ para datos iniciales de Schwarschild $M = 1$ con lapso $1 + \log$. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

Como resumen de las tres figuras anteriores (Figuras 6.8, 6.9 y 6.10), se puede afirmar nuevamente que sí hay convergencia a cuarto orden aunque ahora se empieza a ver algo de 'rugosidad' en la menor resolución. Esto posiblemente anticipa que se necesita una mejor resolución base para cuando se evolucionen ondas de Brill. También vale la pena recordar el hecho que se anticipó en el capítulo anterior sobre el AHF: se necesita implementar la máscara del AH para que veamos la convergencia anticipada. De otro modo, se corre riesgo de pensar que no hay convergencia si se está considerando la región dentro del AH que contiene la singularidad.

El éxito de esta evolución debe sin duda una parte importante a que el lapso está precolapsado, ya que esto simula muy bien la presencia de la singularidad y rápidamente 'congela' una buena parte de la evolución al interior del AH. Sin embargo, para poder dar cimiento a esta expectativa (y además constatar el funcionamiento del AHF), ahora se presenta la evolución de la masa calculada a partir del AH.

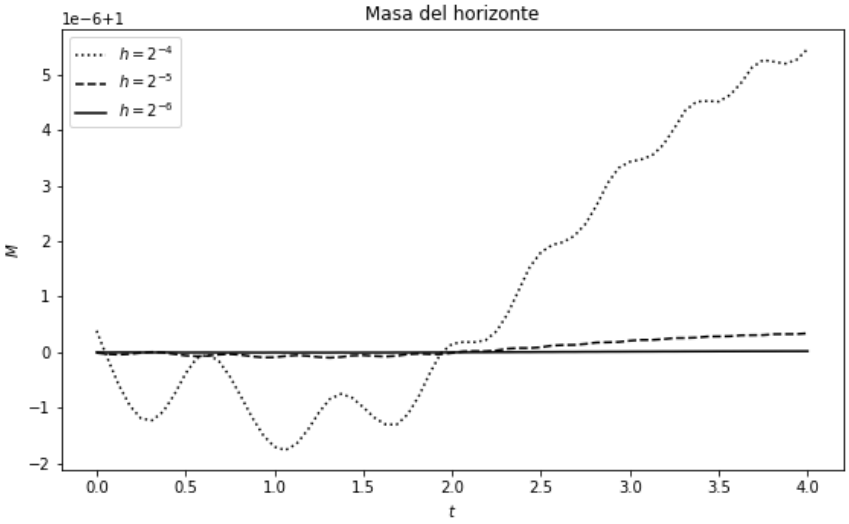

Figura 6.11: Evolución de la masa del AH encontrado por el código para lapso $1 + \log$. Se presentan tres resoluciones $h = 1/16,\ 1/32,\ 1/64$. Nótese que se ha restado el valor analítico $M = 1$ en el eje vertical.

Al ver que se ha restado $M = 1$ al eje vertical, se puede leer de la Figura 6.11 que el error incluso para la menor de las resoluciones ($h_0 = 1/16$) es del orden de 10^{-6}. Sin embargo, también se puede apreciar que las curvas de mayor resolución se van acercando más al valor analítico de $M = 1$ e incluso parecen mantenerlo mucho más constante durante la evolución. Para tener algo más que esta observación textual, se analiza ahora el error de la masa en función de las resoluciones. Esto es análogo como si la masa fuera una constricción (pues conocemos su valor analítico).

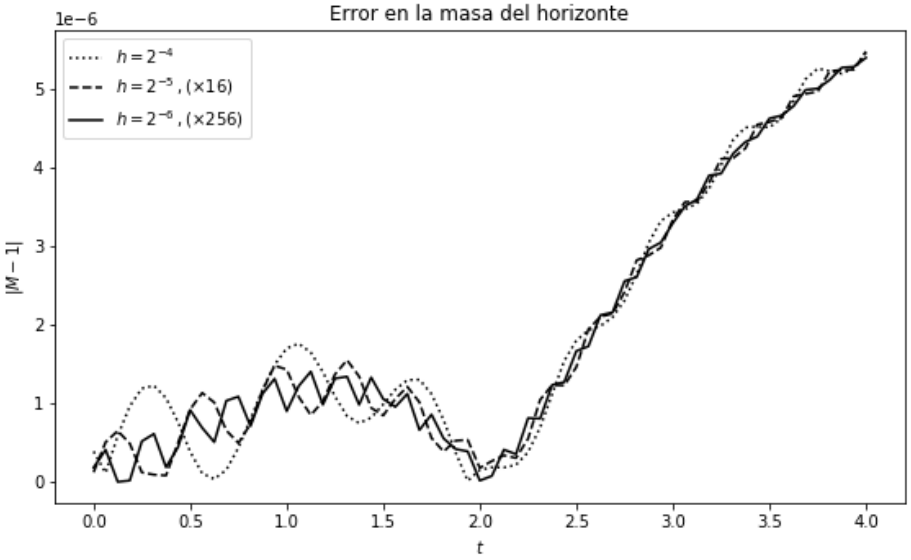

Figura 6.12: Evolución del error de la masa calculada por el AH para lapso $1 + \log$. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

El hecho de que este error también se comporte a cuarto orden genera una buena confianza tanto en el código de evolución así como en el AHF y su subrutina de integración que computa la masa.

6.1.3. Schwarschild con lapso maximal

Todo este análisis de convergencia se ha basado en dar pequeños pasos hacia el problema final de evolución de ondas de Brill con lapso maximal y ahora se presenta un primer paso mayor que es la evolución del mismo agujero negro de Schwarschild pero ahora con lapso maximal. Los parámetros de las mallas son exactamente los mismos, con la diferencia de que el lapso inicial debe ser $\alpha(t=0)=1$ debido a la simetría temporal cuando hay lapso maximal.

Como ya es costumbre, se comienza con las constricciones:

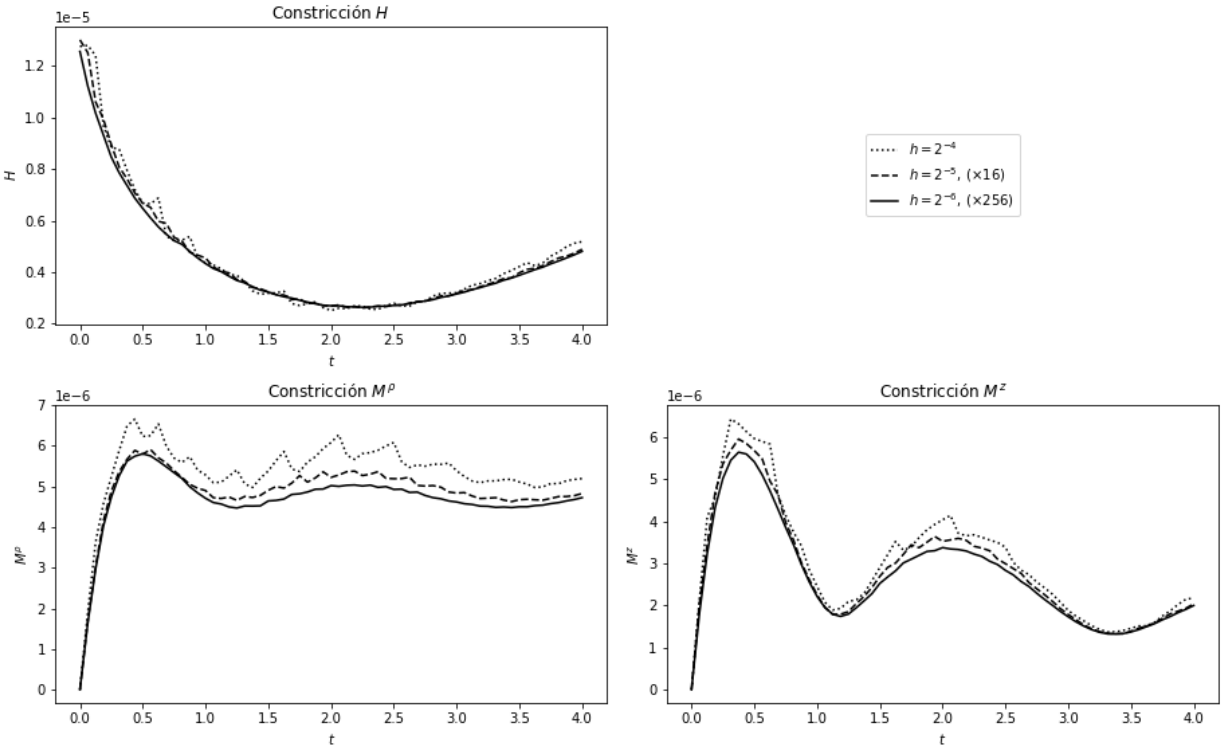

Figura 6.13: Evolución de las constricciones hamiltoniana y de momento para datos iniciales de Schwarschild $M=1$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

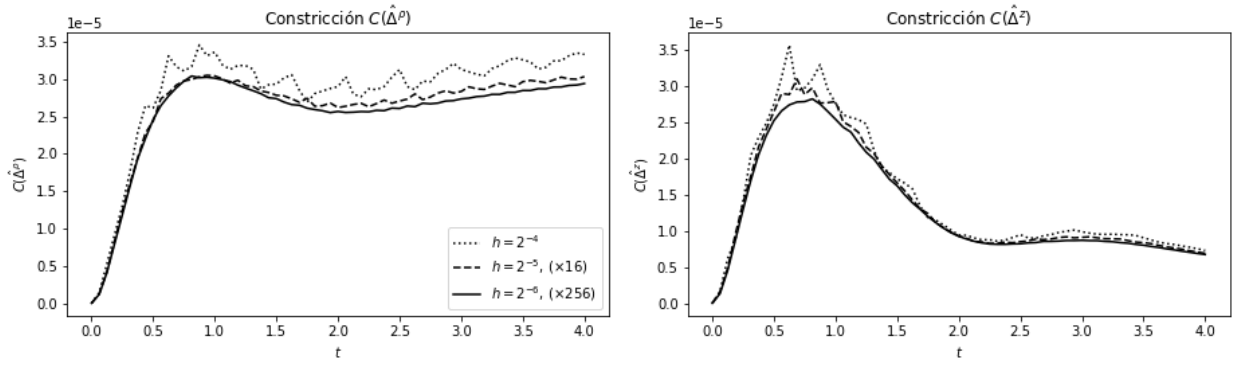

Figura 6.14: Evolución de las constricciones del vector $\hat{\Delta}^i$ para datos iniciales de Schwarschild $M = 1$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

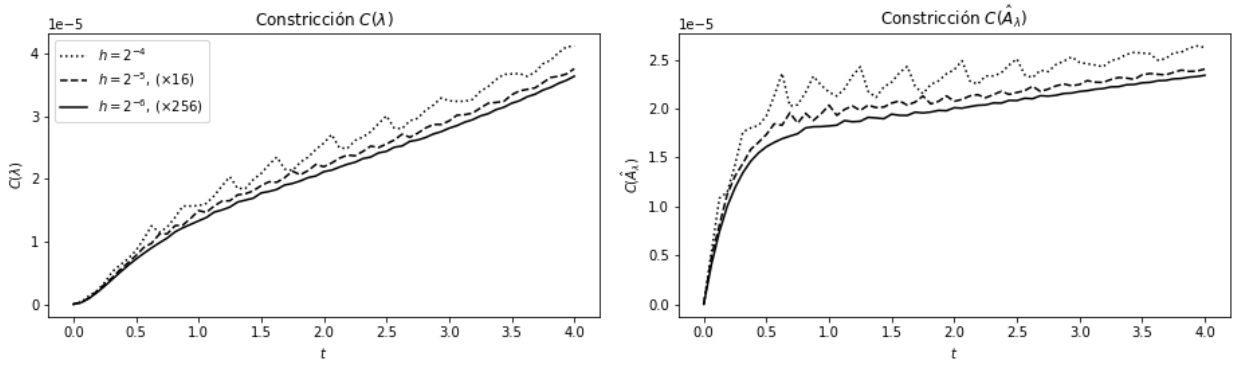

Figura 6.15: Evolución de las constricciones de las variables de regularización λ y $\hat{A}_\lambda$ para datos iniciales de Schwarschild $M = 1$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

Las Figuras 6.13, 6.14 y 6.15 verifican la convergencia a cuarto orden y además tienen una similitud importante con la simulación anterior que utiliza lapso 1+log precolapsado, i.e. las Figuras 6.8, 6.9 y 6.10. De hecho, la forma y orden de las curvas es prácticamente igual si no es porque las curvas de lapso maximal parecen extenderse más. Esto debe de venir del hecho de que en lapso maximal no se empieza con un lapso precolapsado, lo que causa que el tiempo se 'estire' para los observadores con lapso maximal contra los observadores con lapso $1 + \log$.

Antes de pasar a la evolución de la masa del AH, podemos verificar el colapso del lapso sobre la diagonal en la siguiente Figura 6.16.

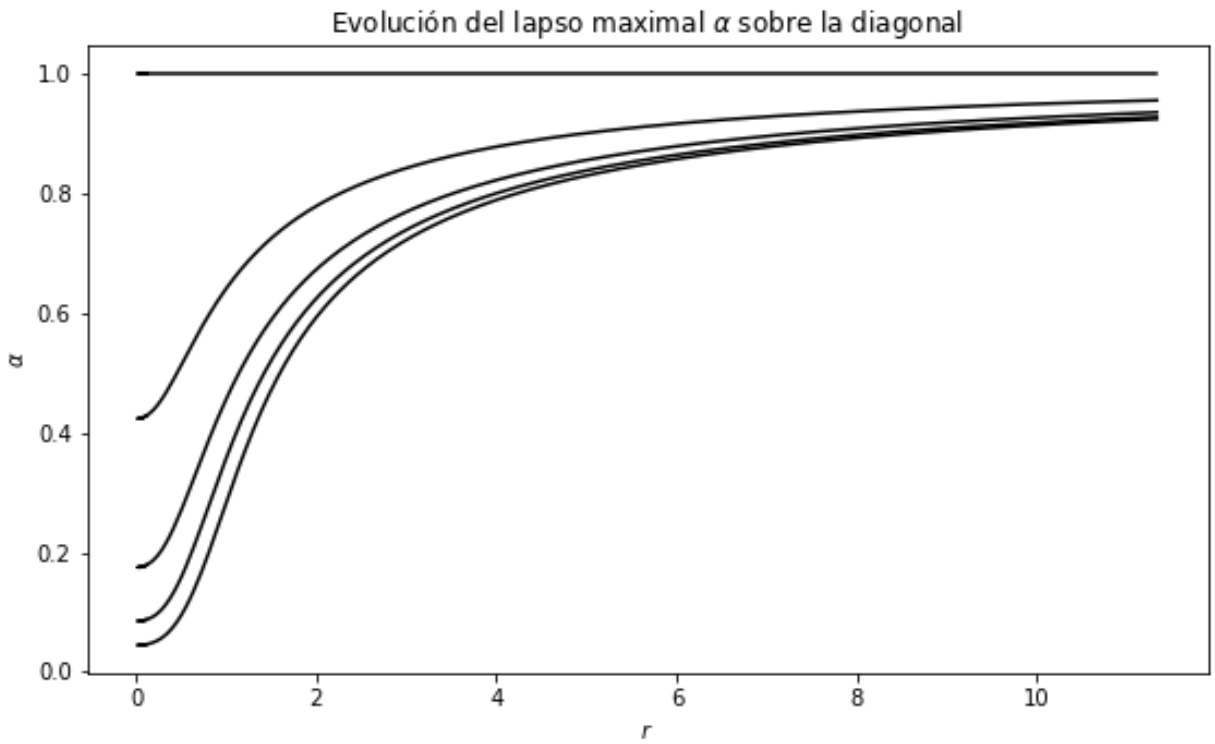

Figura 6.16: Instantáneas del lapso maximal sobre la diagonal para Schwarschild $M = 1$ tomadas a intervalos de $t = 1$. El lapso inicial es la constante $\alpha = 1$ y las instantáneas subsecuentes van cayendo una a una por debajo de la anterior.

El colapso del lapso es evidente y sirve además para entender un poco por qué un lapso precolapsado como el de la Figura 6.7 es tan robusto con una evolución $1+\log$ (para una mayor discusión sobre la conexión entre el lapso maximal y el lapso $1 + \log$ se puede consultar [1]).

Siguiendo la lógica de la sección anterior, ahora se verifica la evolución de la masa del horizonte, así como el comportamiento del error de dicha masa al valor analítico.

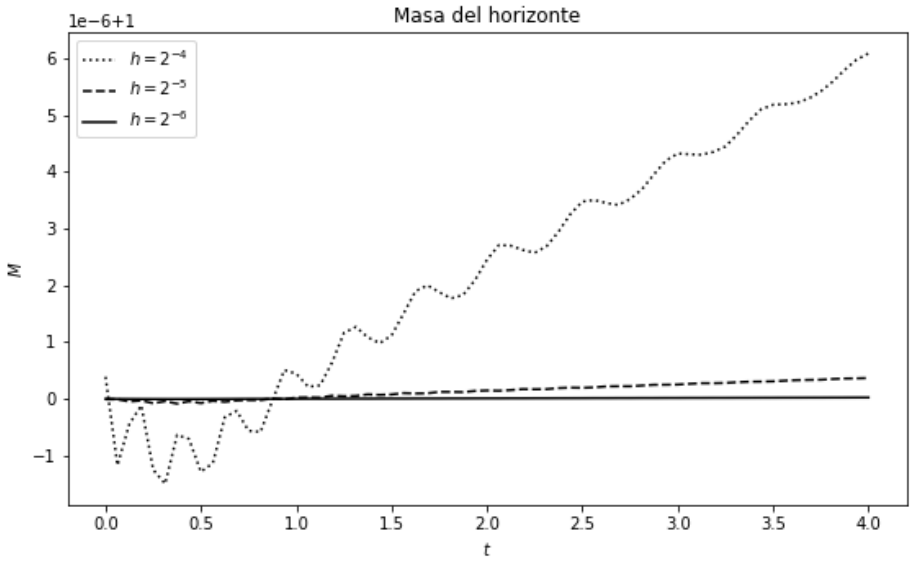

Figura 6.17: Evolución de la masa del AH encontrado por el código para lapso maximal. Se presentan tres resoluciones $h = 1/16,\ 1/32,\ 1/64$. Nótese que se ha restado el valor analítico $M = 1$ en el eje vertical.

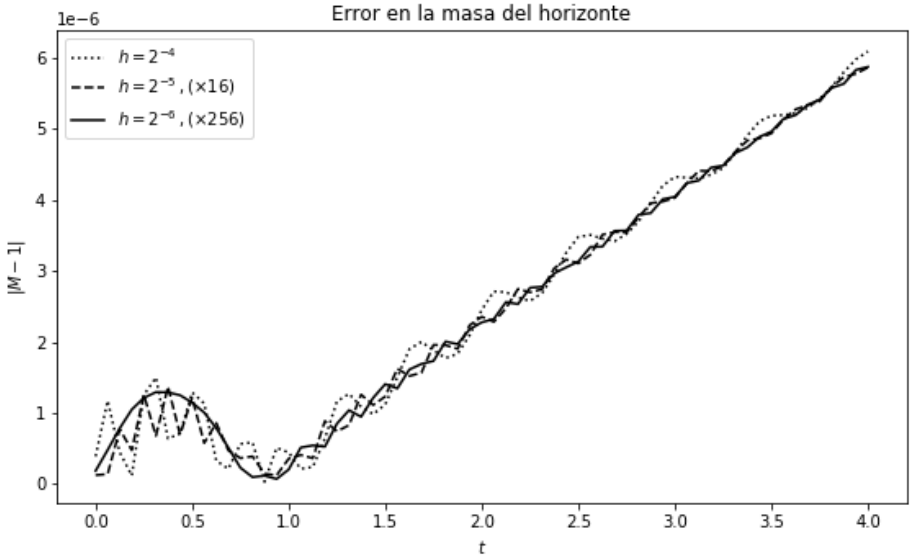

Figura 6.18: Evolución del error de la masa calculada por el AH para lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

Al igual que en la sección anterior, estas dos figuras afirman la validez del código con la adición de que ahora se ha probado también las subrutinas de lapso maximal que incluyen toda la parte del resolvedor elíptico.

6.2. Ondas de Brill

Es posible que toda la discusión de la sección anterior con respecto a Minkwoski y Schwarschild haya sido algo superflua o rutinaria para el lector. Sin embargo, dentro de todas las pruebas anteriores quedan implícitas todas las docenas de veces que sirvieron para encontrar *bugs* o errores en el código. Naturalmente, el resultado final consiste en cuando todas las subrutinas ya no tienen error que se pueda ver en estas pruebas pero de todos modos, el valor de estas simulaciones con geometrías sencillas no puede ser subestimado. Gracias a ellas, ahora estamos en una posición de analizar la primera onda de Brill de este trabajo.

6.2.1. Onda débil $a = 1$

El primer caso a tratar es el de una onda de Brill con amplitud relativamente débil. En este caso, los resultados de [5] hacen esperar que este fenómeno sea una perturbación de Minkowski. Pero antes de hacer una discusión de estos resultados, hay que verificar que el código converja correctamente al ver las constricciones. Se hace notar que las dimensiones de la malla y las resoluciones son las mismas que se han usado para Schwarschild y Minkowski. Es decir, $h = 1/16,\ 1/32,\ 1/64$, $N = 128,\ 256,\ 512$ con evolución hasta $t = 4$.

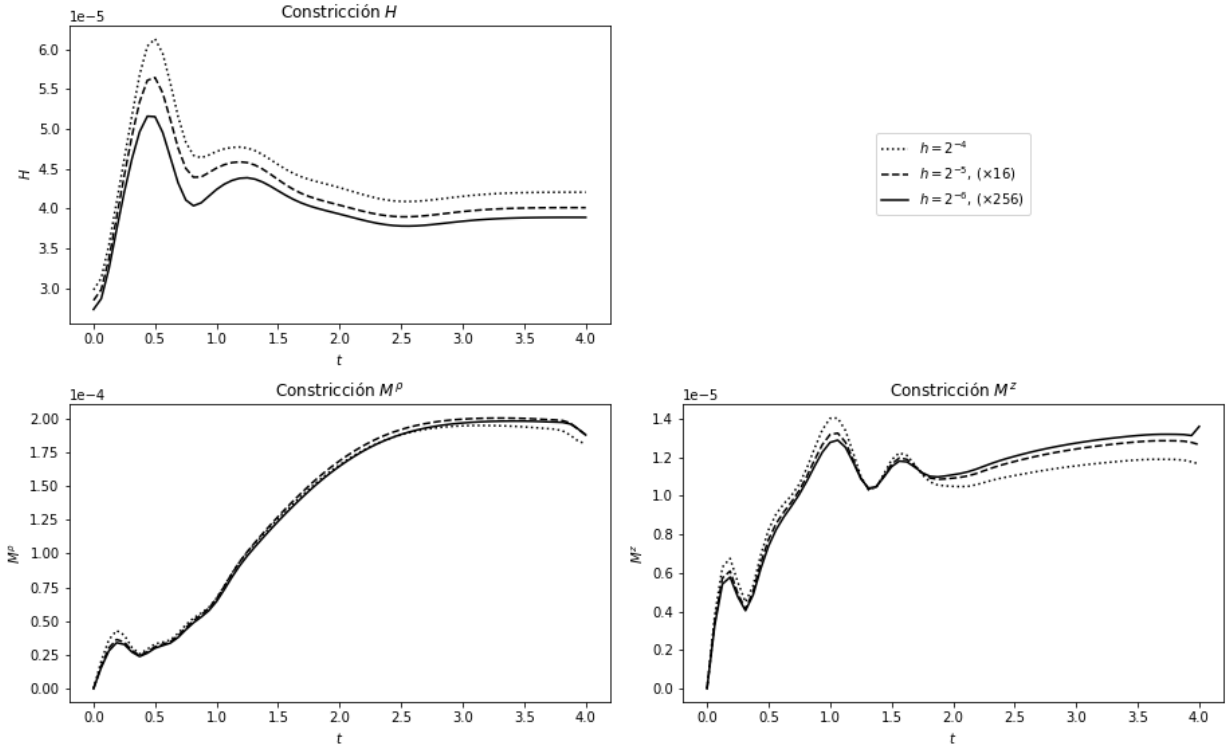

Figura 6.19: Evolución de las constricciones hamiltoniana y de momento para datos iniciales de Brill $a = 1$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

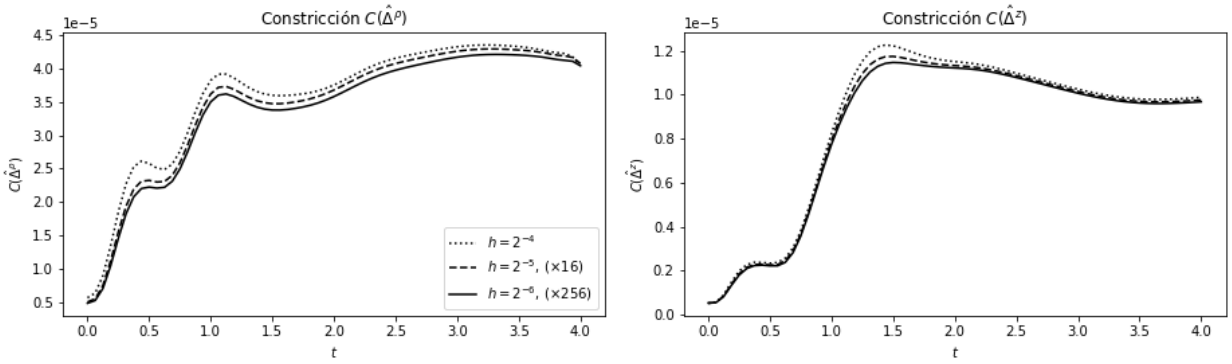

Figura 6.20: Evolución de las constricciones del vector $\hat{\Delta}^i$ para datos iniciales de Brill $a = 1$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

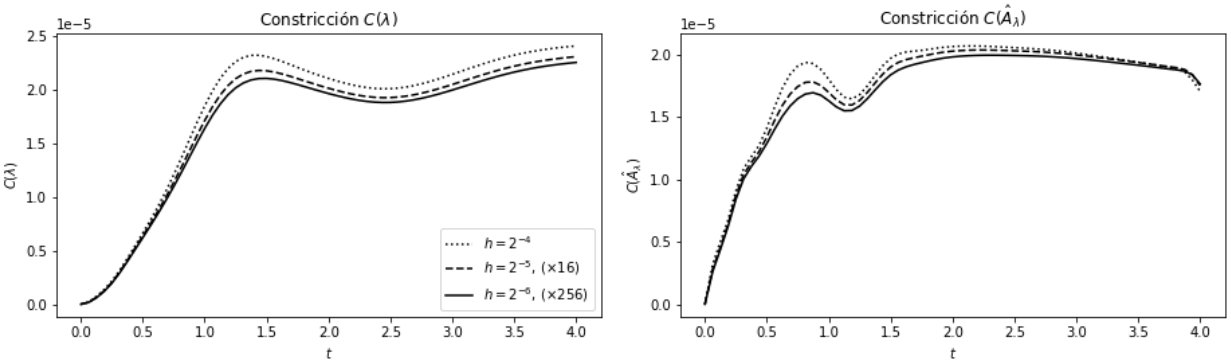

Figura 6.21: Evolución de las constricciones de las variables de regularización λ y $\hat{A}_\lambda$ para datos iniciales de Brill $a = 1$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

El reescalamiento en las Figuras 6.19, 6.20 y 6.21 demuestra que el código sí converge a cuarto orden con la observación adicional de que ahora se está poniendo a prueba el resolvedor elíptico tanto en la modalidad en que calcula los datos iniciales, así como en la modalidad en la que calcula el lapso maximal a cada paso de integración.

Habiendo probado el buen comportamiento de las constricciones y su convergencia, se realiza ahora una evolución más larga hasta $t = 10$ con una malla con resolución $h = 0.04$ y $N = 500$ para poner la frontera más allá de $r = 20$. Para respaldar dicha corrida podemos ver las constricciones de Einstein en la Figura 6.22.

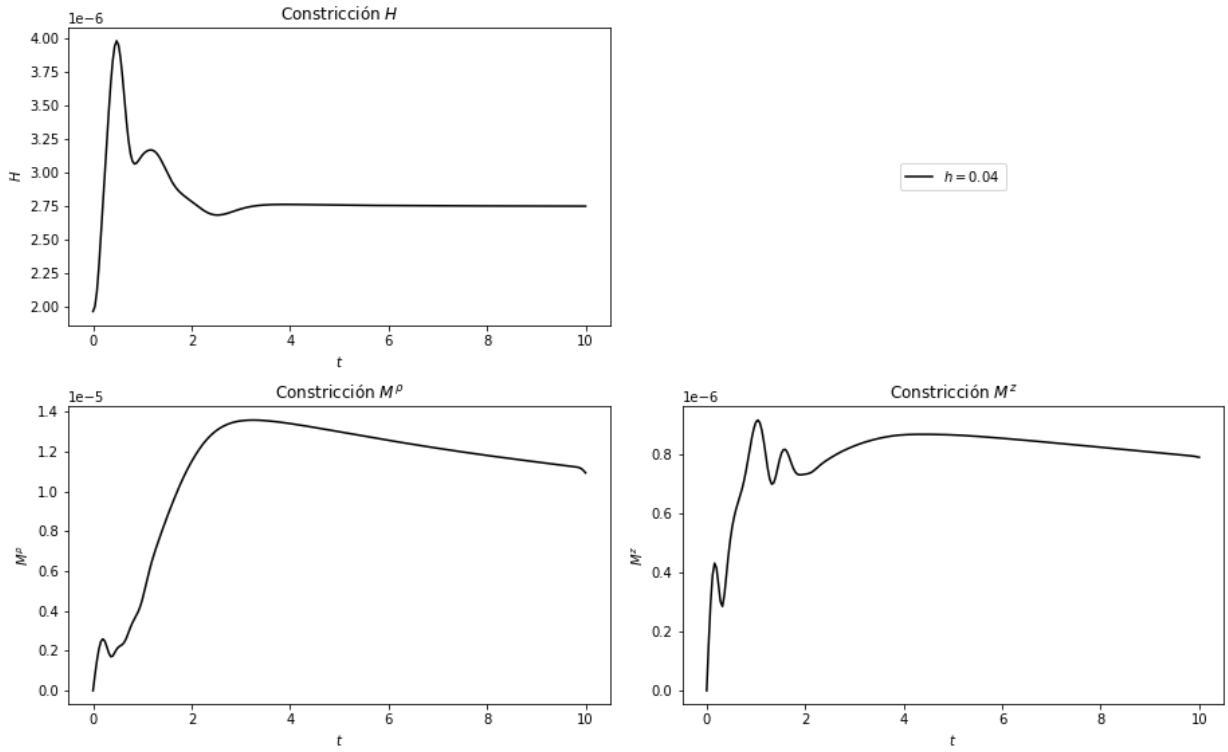

Figura 6.22: Evolución de las constricciones hamiltoniana y de momento para datos iniciales de Brill $a = 1$ con lapso maximal. La evolución se hace ahora hasta $t = 10$.

Es fácil ver que las constricciones siguen comportándose de manera muy regular, sobre todo en comparación a la contraparte, i.e. la Figura 6.19. En particular, para la constricción hamiltoniana H, se ve claramente que a tiempos tardíos, se asienta en un valor relativamente constante. Esto anticipa que ya no hay realmente evolución en el espacio-tiempo y al analizar otras cantidades en un momento, esto da fuerza particular a que el espacio regresa a Minkowski.

Ahora se puede también examinar el lapso maximal calculado, ya que la magnitud de su caída puede a veces ser muy ilustrativa.

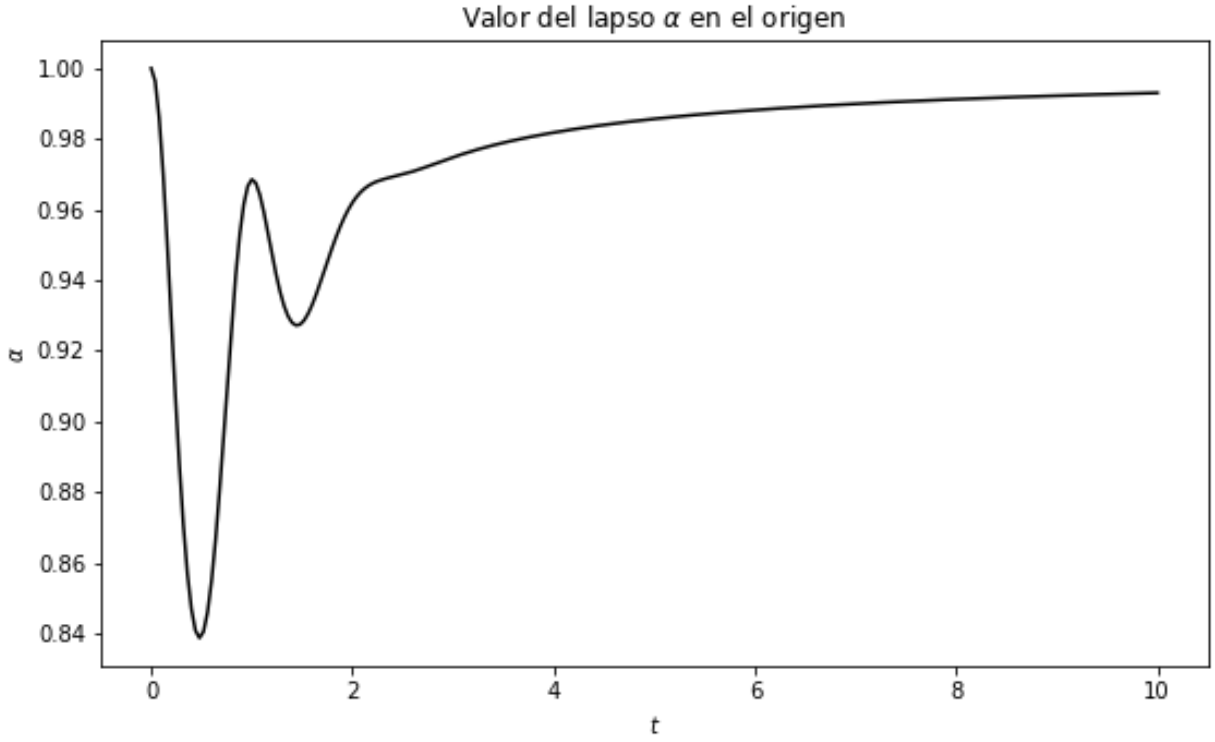

Figura 6.23: Evolución del valor del lapso en el origen para datos iniciales de Brill $a = 1$ con lapso maximal.

En la Figura 6.23 se ve que el valor del lapso no sólo baja a un valor de $\alpha \approx 0.84$ sino que después oscila hasta que a tiempos tardíos empieza a regresar al valor constante $\alpha = 1$. Con base en el trabajo presentado en [1], donde se expone un análisis cuidadoso para lapso maximal de Schwarschild, así como una recolección de resultados empíricos, se puede concluir que esta caída a $\alpha = 0.84$ no es realmente fuerte y mucho menos indicativa de que se puede llegar a formar un AH. Además el hecho de que el lapso está regresando a $\alpha = 1$, también puede ser un indicador de que la evolución está deteniéndose.

Para fundamentar este último hecho sobre la evolución, podemos analizar dos variables dinámicas: $\hat{\gamma}_{\rho\rho}$ y $\hat{A}_{\rho\rho}$. Las ecuaciones de evolución, (1.34) y (1.36), se reducen en el caso de lapso maximal a

$$\partial_t \hat{\gamma}_{\rho\rho} = -2\alpha \hat{A}_{\rho\rho} , \tag{6.4}$$

$$\partial_t \hat{A}_{\rho\rho} = e^{-4\phi} \left(-\nabla_\rho \nabla_\rho \alpha + \alpha R_{\rho\rho}\right) , \tag{6.5}$$

donde se ha usado que $\nabla^2 \alpha = R\alpha$ para el lapso maximal, así como la afirmación de que en toda estas evoluciones no se ha usado nunca un vector de corrimiento β^i. En resumen, estas ecuaciones dicen que la fuente de $\hat{\gamma}_{\rho\rho}$ es proporcional a $\hat{A}_{\rho\rho}$ y que la fuente de la curvatura viene de un término que analiza cómo varía el lapso espacialmente y del escalar de Ricci. Por lo tanto, para que se vuelvan cantidades constantes, se requiere que $\hat{A}_{\rho\rho}$ vaya a cero a tiempos tardíos y que sea constante en dicho valor, lo que sería consecuencia lógica de que el escalar de Ricci regrese a cero y que el lapso se asiente en la constante $\alpha = 1$. Con base en este breve análisis, se presentan ahora tres figuras que dan el valor de $\hat{\gamma}_{\rho\rho}$, $\hat{A}_{\rho\rho}$ y R en el origen.

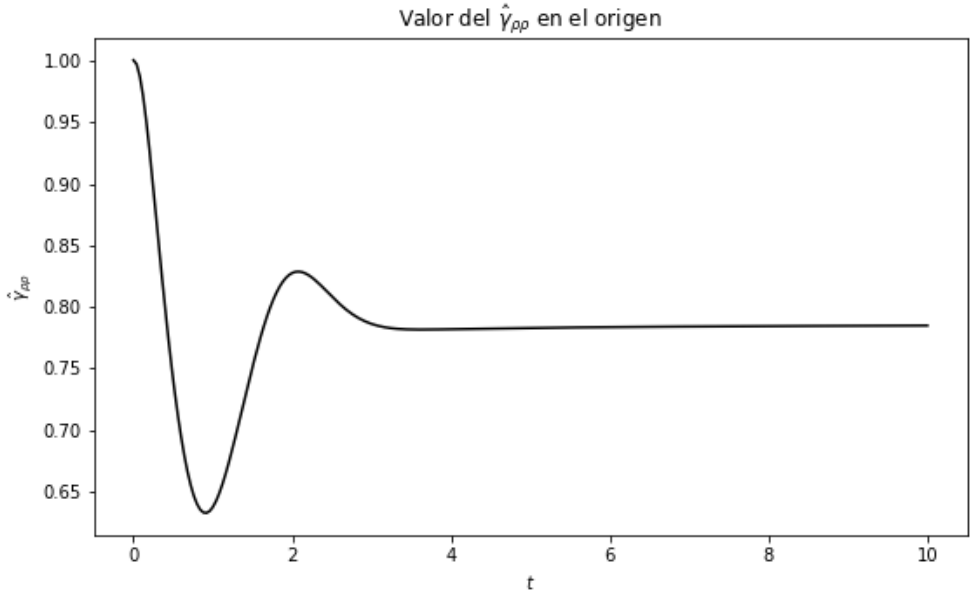

Figura 6.24: Evolución del valor de $\hat{\gamma}_{\rho\rho}$ en el origen para datos iniciales de Brill $a = 1$ con lapso maximal.

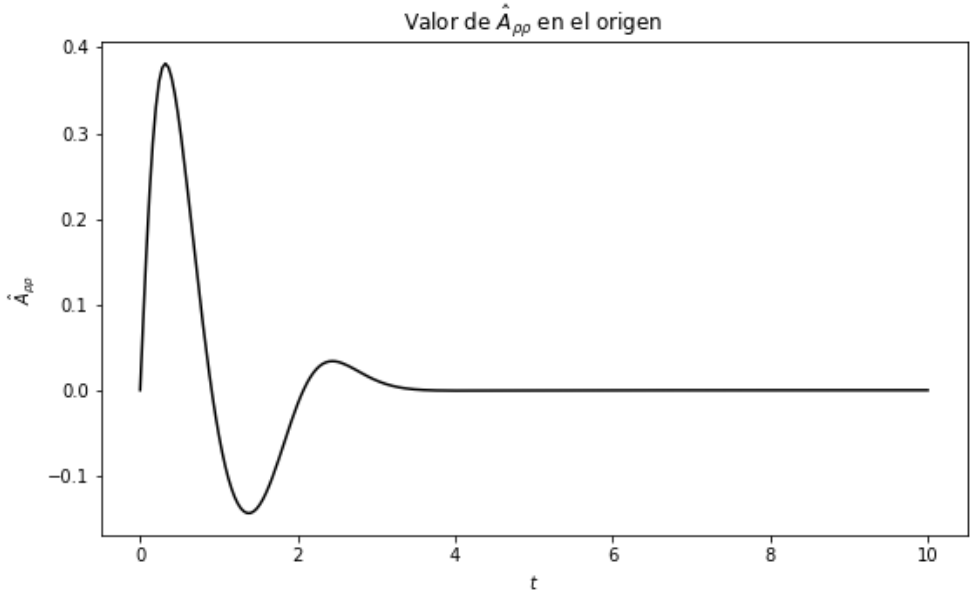

Figura 6.25: Evolución del valor de $\hat{A}_{\rho\rho}$ en el origen para datos iniciales de Brill $a = 1$ con lapso maximal.

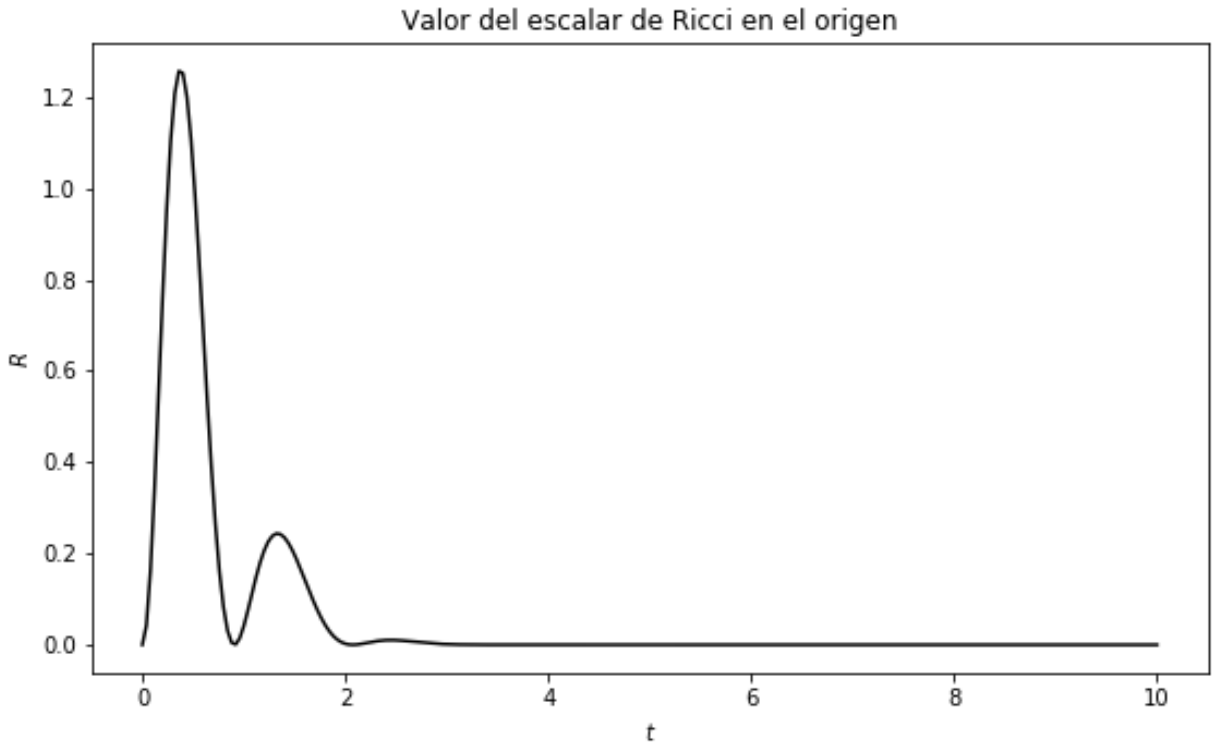

Figura 6.26: Evolución del valor del escalar de Ricci en el origen para datos iniciales de Brill $a = 1$ con lapso maximal.

Es claro que para al menos el caso del origen queda exhibido el comportamiento esperado para que el espacio regrese a Minkowski y se quede estático. La Figura 6.24 detalla cómo esta cantidad métrica se queda congelada en un valor a tiempos tardíos; la Figura 6.25 comprueba cómo la métrica ya no evoluciona debido a que la curvatura se ha ido a cero; y finalmente, la Figura 6.26 dibuja la caída del escalar de Ricci que en este caso implica que el espacio regresa a ser plano.

Para tener un estudio que no sea solamente local en el origen, se presenta ahora la norma (como siempre, la norma L_2) de $\hat{A}_{\rho\rho}$ y R.

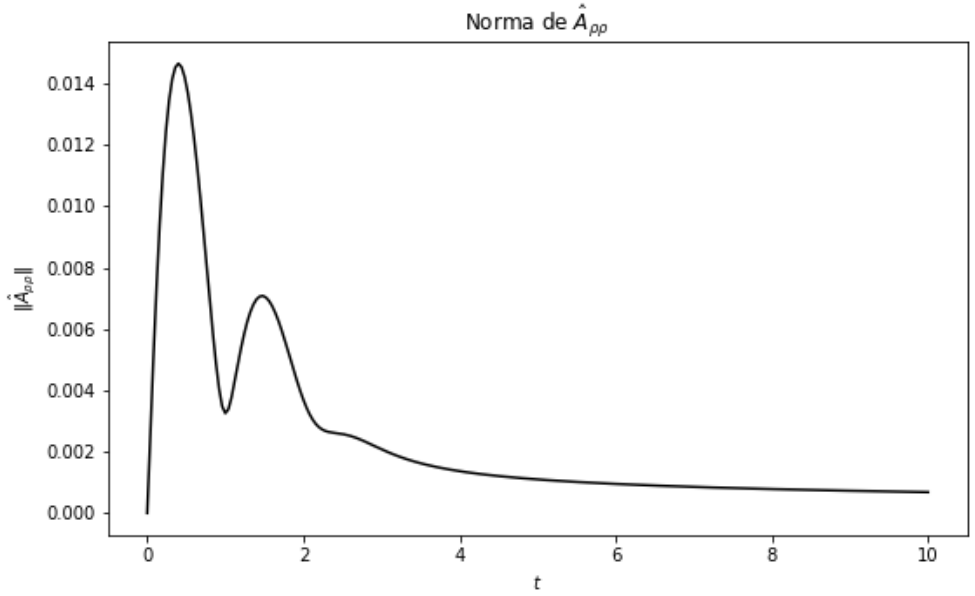

Figura 6.27: Evolución de la norma de la curvatura $\hat{A}_{\rho\rho}$ para datos iniciales de Brill $a = 1$ con lapso maximal.

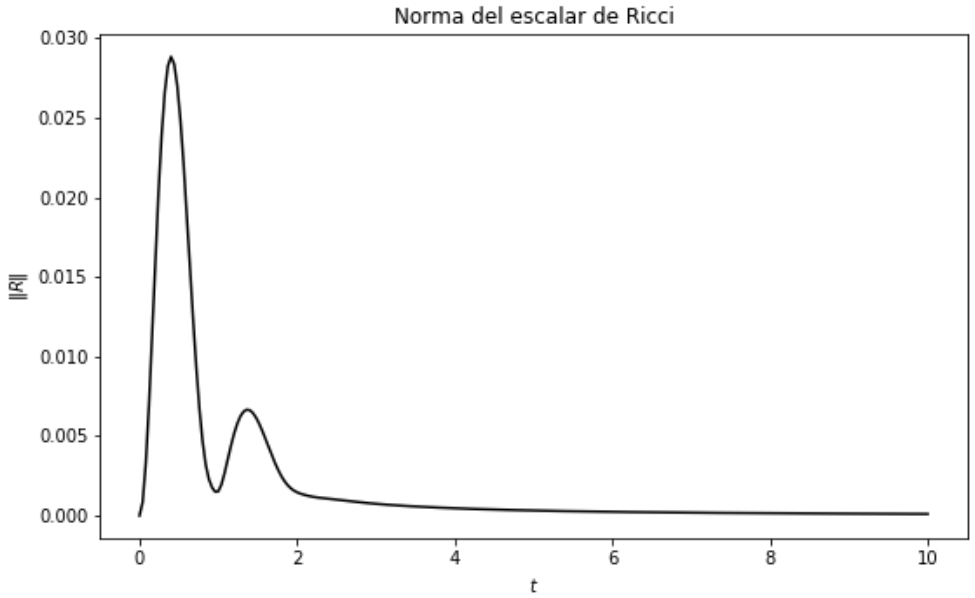

Figura 6.28: Evolución de la norma del escalar de Ricci para datos iniciales de Brill $a = 1$ con lapso maximal.

Las Figuras 6.27 y 6.28 anteriores dan un carácter global a las afirmaciones anteriores y dejan evidencia muy sólida a que la evolución de ondas de Brill regresa a Minkowski para amplitudes débiles.

6.2.2. Onda fuerte $a = 11$

En la sección anterior se trató el problema para una onda con amplitud débil y se halló que el código pudo tratarla con bastante éxito y con la convergencia a cuarto orden esperada. El caso de una amplitud fuerte con $a = 11$ no es tan sencilla como simplemente poner en el archivo de parámetros la amplitud deseada y poner la simulación a correr con las mismas mallas que se han usado en todos los casos anteriores. De hecho, lo que se encuentra es que para una amplitud fuerte, las derivadas de varias cantidades pueden ser tan intensas que no cualquier resolución da una convergencia correcta, independientemente de que el código esté bien escrito. Para ser concretos, la resolución base de $h = 1/16$ resulta ser demasiado baja para este problema, hecho que también es confirmado en [5] donde también hallan que se debe adaptar la resolución dependiendo de la amplitud del problema. Al hacer unas corridas preliminares, se decide usar ahora las resoluciones $h = 1/25$, $\sqrt{2}/50$, $1/50$, es decir $h = 0.04$, $0.02828\ldots$, 0.02. Además ahora el tamaño de las mallas es de $N = 250$, 384, 500, con la intención de poner la frontera en aproximadamente $r = 10$ y evolucionar hasta $t = 5$. Hay que hacer la observación crucial de que ahora las resoluciones son tal que $h_0 = \sqrt{2}\, h_1 = 2\, h_2$, así que si se espera convergencia a cuarto orden, los factores de reescalamiento serán 4 y 16.

Al tener una amplitud muy cercana a la que se halló en la sección anterior como la amplitud necesaria a que existe un AH dentro de los datos iniciales, se justifica esperar que se pueda formar un AH durante la evolución. Esto es más obvio si se toma en cuenta el valor de amplitud crítica reportada en [5], i.e. el valor a_* tal que se forma un AH después de una evolución. Dicho valor es cercano a $a_* \approx 4.85$ y es evidente que la amplitud $a = 11$ está muy por encima de dicha amplitud crítica. Inicialmente, la búsqueda del AH empieza desde $t = 0$ y se hace buscando desde $r = 10$ hasta $r = 10\, h_i$ ($i = 0, 1, 2$ según la resolución que se trate). Haciendo unas corridas preliminares, el horizonte se encuentra en $t \approx 1$ y en $r \approx 1$. Ahora se puede refinar la búsqueda y optimizar el AHF para que empiece a buscar a partir de $t = 0.80$ en el rango $r \in [1.25, 0.75]$. Las resoluciones h_0, h_2 están de acuerdo que el horizonte se encuentra en $t = 0.96$, mientras que h_1 lo pone en $t = 1.02$. El desfase se debe a que h_1 no integra en los mismos tiempos que h_0 y h_2 gracias al factor de $\sqrt{2}$ en $h_0 = \sqrt{2}\, h_0$. En todo caso, después de que se forma el AH, se aplica la máscara que hacer cero las constricciones dentro del mismo. Por lo tanto, a la hora de analizar las constricciones graficadas, son necesarias dos gráficas: una antes del colapso y una después. Por ejemplo, las constricciones de Einstein previo al colapso son la siguiente Figura 6.29.

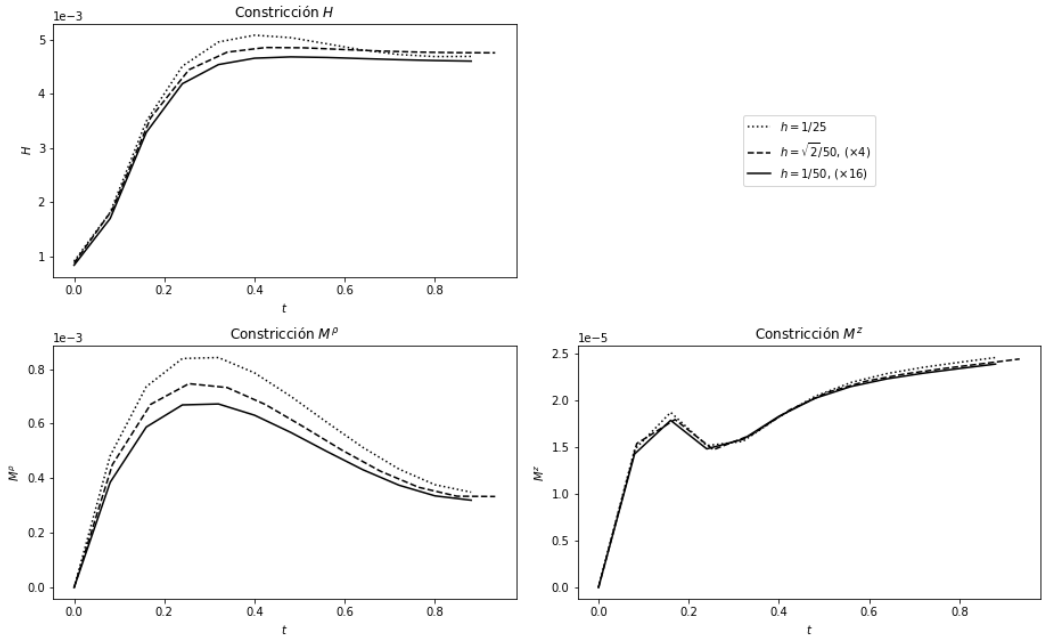

Figura 6.29: Evolución de las constricciones hamiltoniana y de momento para datos iniciales de Brill $a = 11$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden. La evolución es *antes* del tiempo en que hay un colapso a un AH.

Después se presentan las constricciones después del colapso en la Figura 6.30

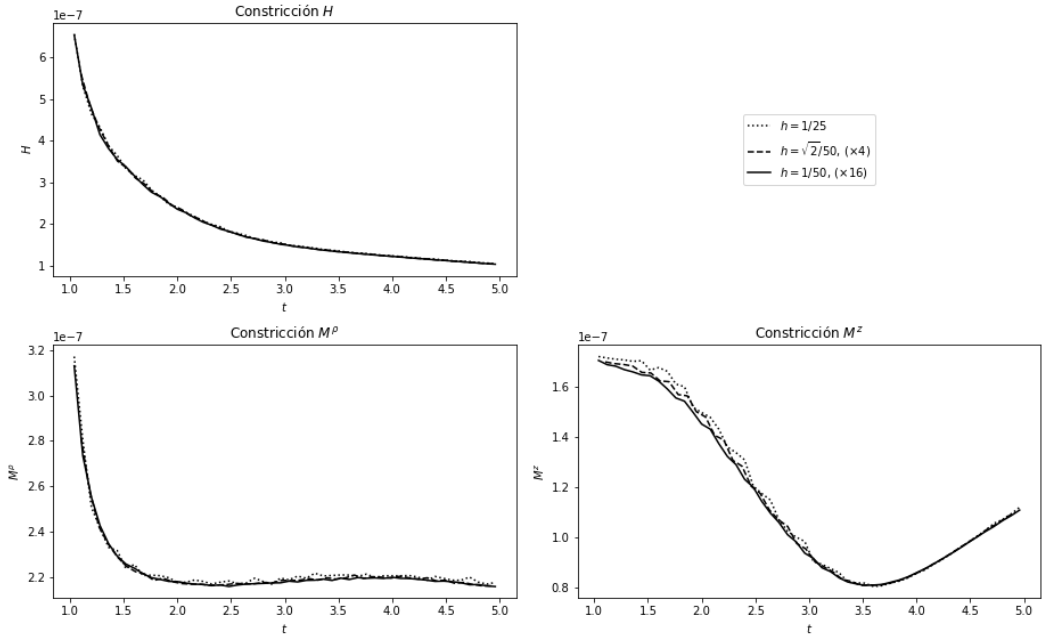

Figura 6.30: Evolución de las constricciones hamiltoniana y de momento para datos iniciales de Brill $a = 11$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden. La evolución es *después* del tiempo en que hay un colapso a un AH.

Análogamente, se pueden ver las otras constricciones antes y después del colapso.

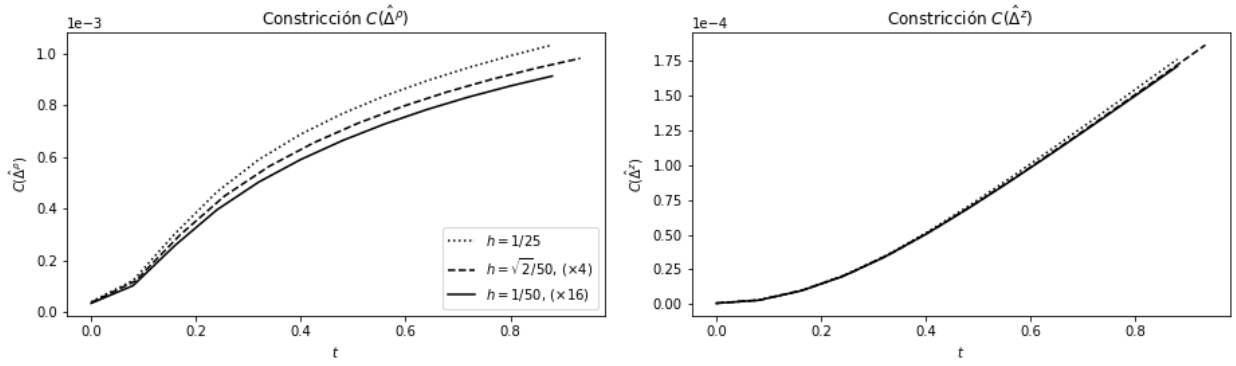

Figura 6.31: Evolución de las constricciones del vector $\hat{\Delta}^i$ para datos iniciales de Brill $a = 11$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden. La evolución es *antes* del tiempo en que hay un colapso a un AH.

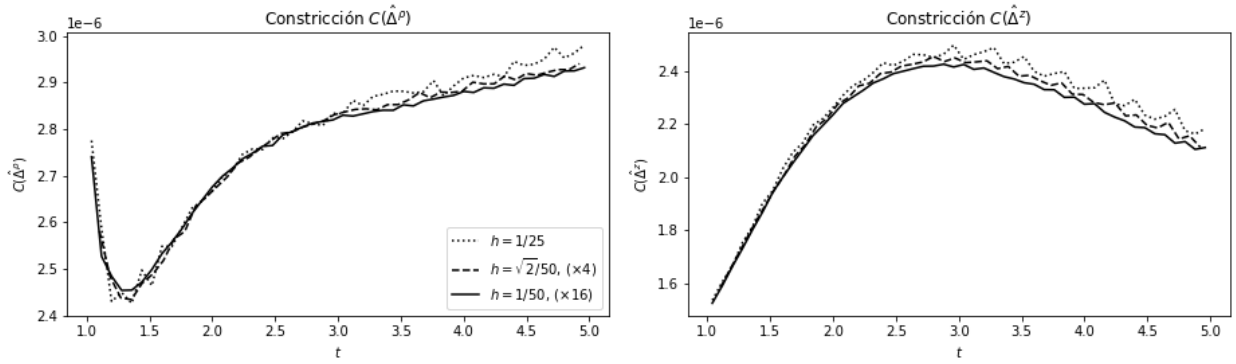

Figura 6.32: Evolución de las constricciones del vector $\hat{\Delta}^i$ para datos iniciales de Brill $a = 11$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden. La evolución es *después* del tiempo en que hay un colapso a un AH.

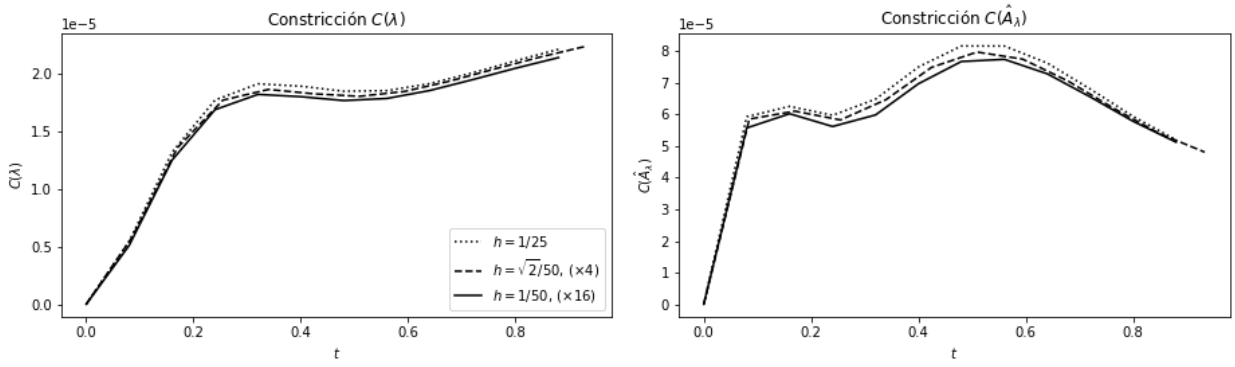

Figura 6.33: Evolución de las constricciones de las variables de regularización λ y $\hat{A}_\lambda$ para datos iniciales de Brill $a = 11$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden. La evolución es *antes* del tiempo en que hay un colapso a un AH.

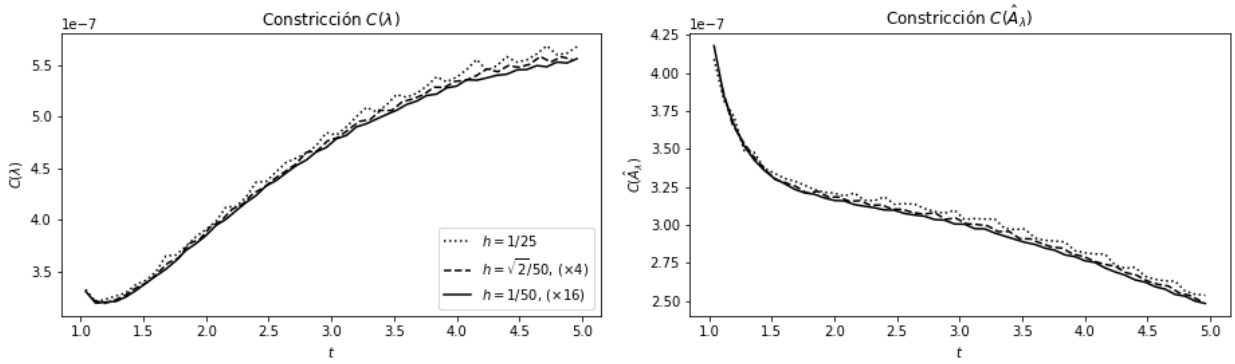

Figura 6.34: Evolución de las constricciones de las variables de regularización λ y $\hat{A}_\lambda$ para datos iniciales de Brill $a = 11$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden. La evolución es *después* del tiempo en que hay un colapso a un AH.

Considerando la observación anterior de que ahora los factores de convergencia a cuarto orden son 4 y 16 para estas resoluciones, las figuras anteriores demuestran que el código está convergiendo a cuarto orden antes y después del colapso a un AH. Esto justifica que ahora se puedan examinar los resultados de las simulaciones, como pueden ser el lapso y la masa del AH.

Empezando por el lapso, se confirma que el valor del mismo en el origen cae rápidamente y el colapso del lapso es muy análogo al caso de Schwarschild en la Figura 6.16.

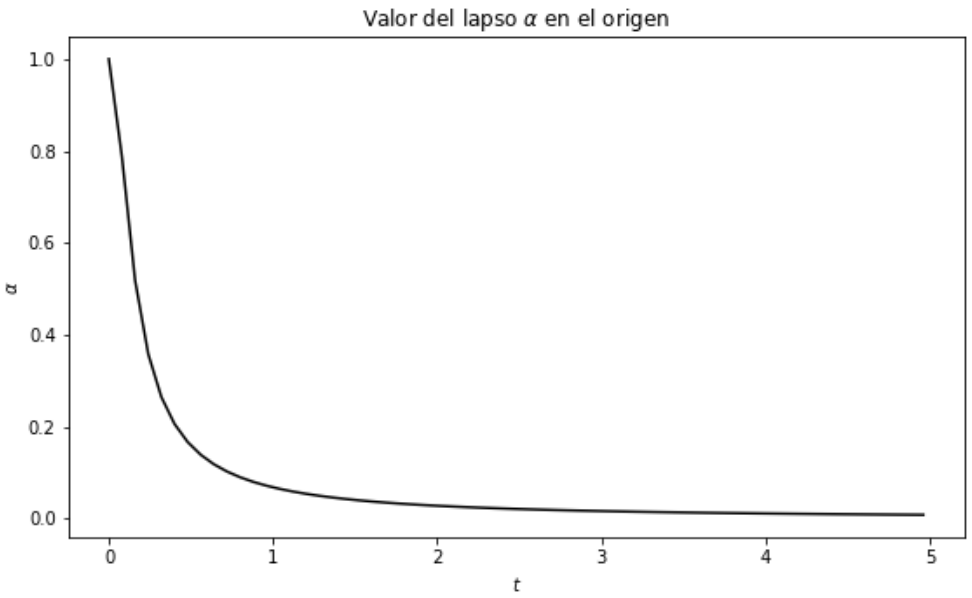

Figura 6.35: Evolución del valor del lapso en el origen para datos iniciales de Brill $a = 11$ con lapso maximal.

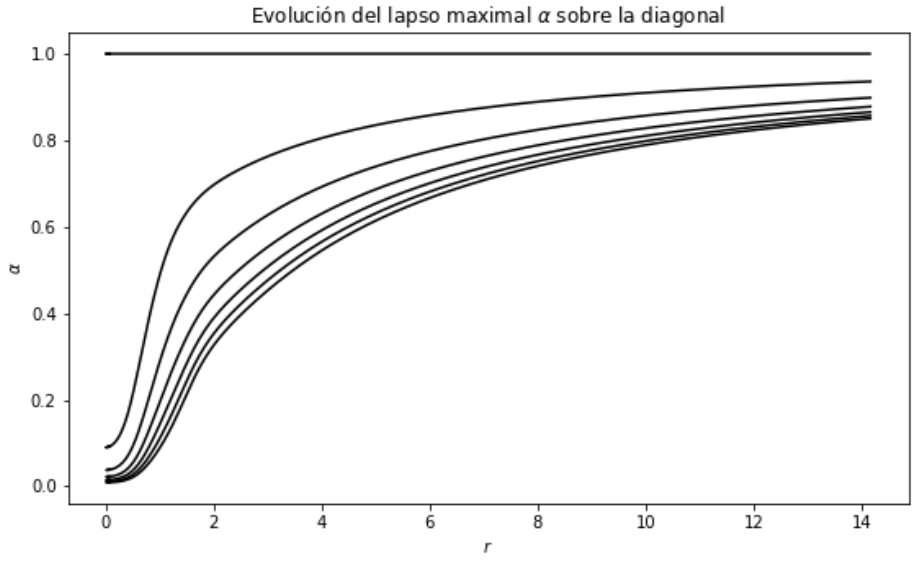

Figura 6.36: Instantáneas del lapso maximal sobre la diagonal para ondas de Brill $a = 11$ tomadas a intervalos de $t = 0.8$. El lapso inicial es la constante $\alpha = 1$ y las instantáneas subsecuentes van cayendo una a una por debajo de la anterior.

Vale la pena constrastar con la Figura 6.23 que da el valor del lapso en el origen para una onda de amplitud débil y ahora queda claro que no hay ninguna oscilación del lapso sino que cae directamente al AH.

Siguiendo la metodología que se hizo en el caso de Schwarschild, ahora se presenta la evolución de la masa del AH.

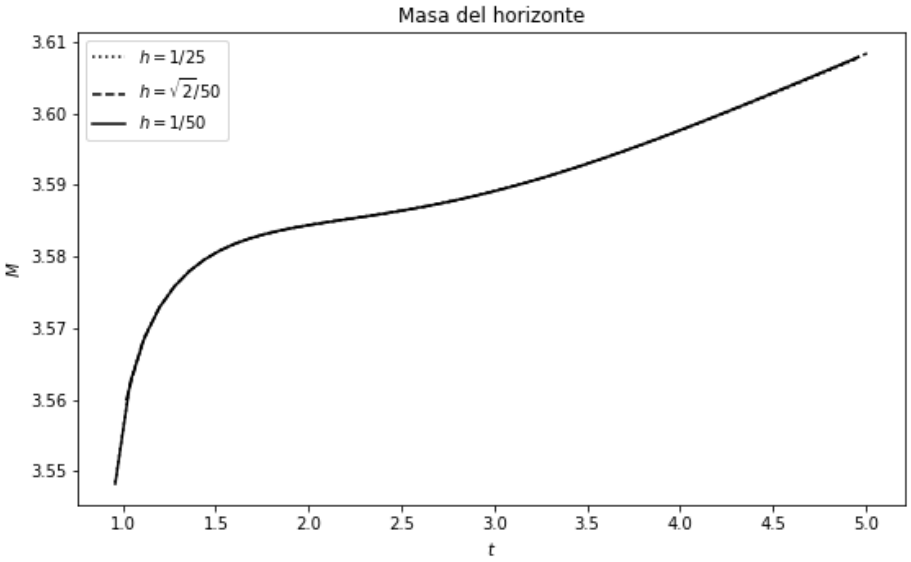

Figura 6.37: Evolución de la masa del AH encontrado por el código para lapso maximal con ondas de Brill $a = 11$.

Aunque las tres curvas correspondientes a las tres resoluciones son básicamente indistinguibles en la Figura 6.37 anterior, el comportamiento es muy diferente a la Figura 6.17 que enseñaba la evolución de la masa del AH para Schwarschild. Ahora la masa del horizonte parece estar creciendo sin aparentar asentarse en un valor constante durante la duración de la evolución a $t = 5$. Sería muy bueno encontrar si existe un valor para el que se asienta la masa, pero antes de empezar a correr una simulación mucho más larga, sin duda sirve tener una referencia de qué valor debería ser dicha masa.

La masa ADM se puede calcular de diversas maneras, sin embargo, siguiendo a [1] y [7] se elige usar dos métodos basados en que el espacio se vuelve esencialmente Schwarschild a medida que r va a infinito. Esto se debe a que el factor conforme de Brill se comporta $\tilde{\psi} \approx 1 + k/r$ en infinito, i.e. equivalente a la condición de Robin que se ha impuesto al momento de resolver los datos iniciales. De los dos métodos para calcular la masa, el primero viene de simplemente hacer el factor k anterior igual al factor $2M$ en el factor conforme de Schwarschild. Esta se denota sencillamente como la masa de Schwarschild.

$$M = 2r\,(\tilde{\psi} - 1)\,, \tag{6.6}$$

El segundo viene ahora de identificar la misma masa de Schwarschild pero ahora sobre la métrica γ_{rr} y se denomina la psuedomasa de Schwarschild.

$$M = \frac{r}{2}\left(1 - \frac{1}{\gamma_{rr}}\right)\,. \tag{6.7}$$

Ahora se puede tomar el factor conforme y la métrica inicial (que para Brill es analítica (2.1)) y calcular dichas masas. Primero puede ser valioso ver una proyección de la forma del factor conforme que se ha calculado aquí con fronteras $\rho, z = 20$ y con alta resolución de $h = 0.01$:

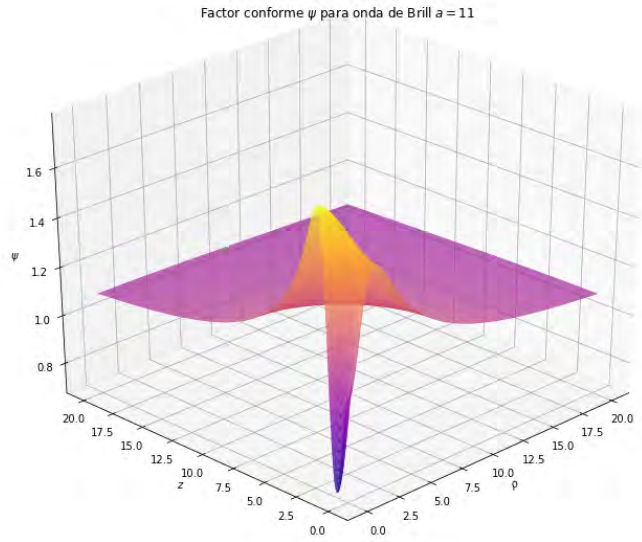

Figura 6.38: Factor conforme para datos iniciales de onda de Brill $a = 11$.

Consecuentemente se calculan ambas masas. Siguiendo a [1] y [7] nuevamente, las masas se estiman en tres sitios: sobre la diagonal y sobre los ejes ρ, z. En seguida se grafican las masas calculadas.

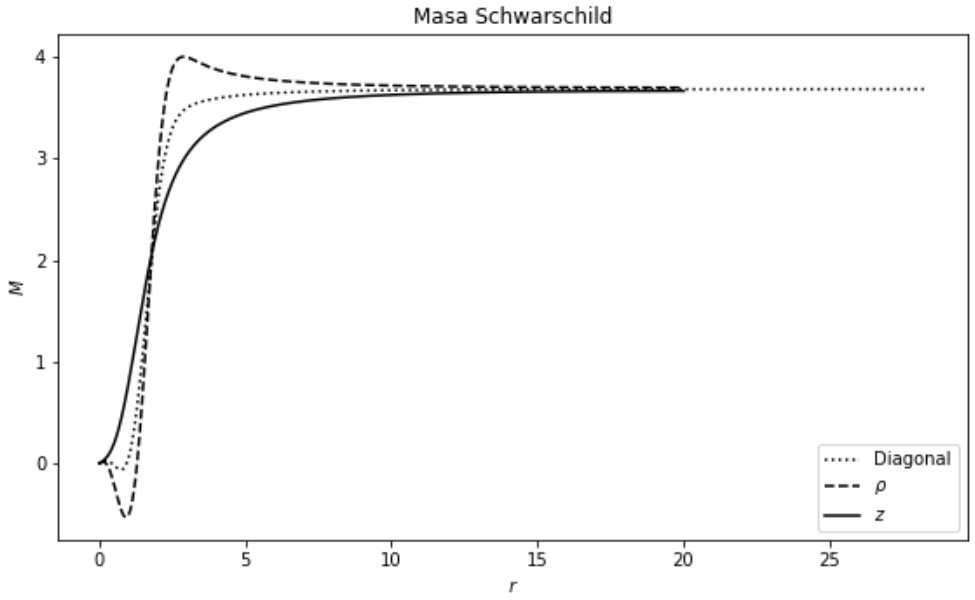

Figura 6.39: Masa de Schwarschild calculada para los datos iniciales de onda de Brill $a = 11$ en función de r.

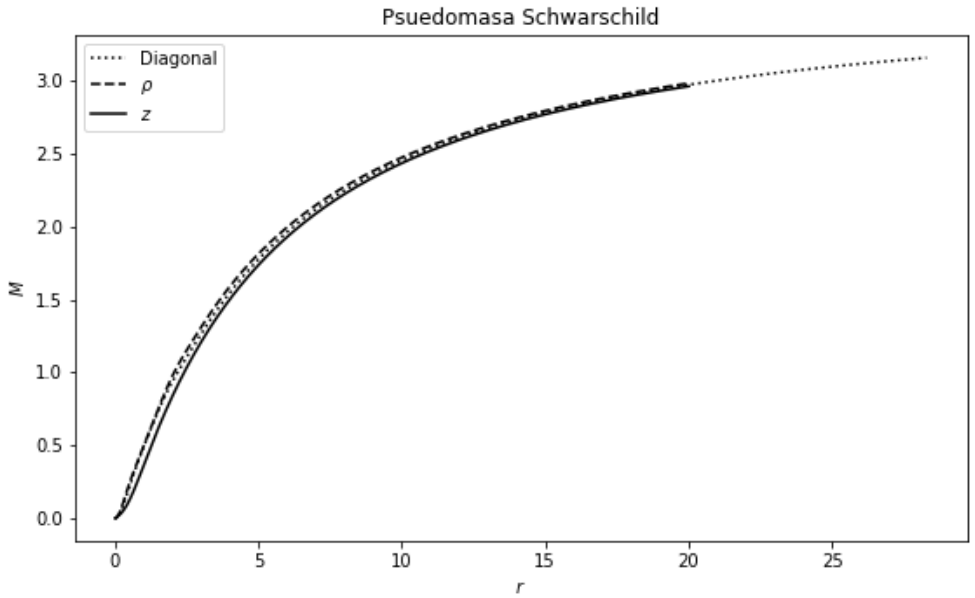

Figura 6.40: Psuedomasa de Schwarschild calculada para los datos iniciales de onda de Brill $a = 11$ en función de r.

Debido a la naturaleza de la malla cuadrada, en ambas Figuras 6.39 y 6.40 el cálculo se extiende más lejos para la diagonal. Adicionalmente, se confirma la afirmación hecha en [7] de que la pseudomasa converge mucho más lento que la masa de Schwarschild en sí, donde para $r > 15$ ya se ve un valor muy constante. Por lo tanto, para el cálculo de la masa ADM se usará la masa de Schwarschild y no la pseudomasa debido a su lenta convergencia.

El hecho de tener estimaciones para la masa en tres sitios permite estimar un error a la hora de reportarla, de donde

$$M_{ADM} = 3.685 \pm 0.016\,. \tag{6.8}$$

Equipados con este valor podemos comparar con la Figura 6.37 donde la masa llega a un valor aproximadamente de $M = 3.61$ que todavía queda un 2 % por debajo del valor (6.8). Esto confirma que no hay violación a la desigualdad de Penrose [45] que nos dice que la masa del horizonte debe estar permanentemente debajo de la masa ADM. Sin embargo, visualmente, la misma Figura 6.37 muestra una tendencia a que esta masa siga creciendo y con una evolución hasta $t = 5$ no hay información para saber si la desigualdad se termina violando a un tiempo futuro o si hay algún comportamiento inesperado. Esto da pie para realizar una simulación con tiempo final en $t = 25$.

Dicha empresa está completamente fuera de proporción con las demás simulaciones realizadas hasta ahora que han sido mallas no mayores a los 512 puntos en cada dimensión, es decir: mallas con aproximadamente 250000 puntos (lo que implica que las matrices a resolver son de 250000×250000). El tiempo de ejecución de las mismas no pasa de cuatro horas en un procesador *Intel i5-4690K* de uso personal y 5GB de memoria. Sin embargo, si se desea seguir la lógica de evolucionar hasta un tiempo máximo igual a un medio de la distancia a la frontera, realizar la simulación hasta $t = 25$ requiere poner la frontera en $r = 50$. Por lo que ya se ha comentado de que no se puede poner una resolución tan grande como $h = 1/16$, para la resolución base de $h = 0.04$ la malla resulta ser de 1250 en cada dirección y ahora cada variable tiene un millón y medio de puntos.

Antes de seguir esta discusión, vale la pena contestar la pregunta a si es tan terrible no poner

la frontera suficientemente lejos y dejar que las violaciones de la frontera caigan al interior. El contraejemplo está en la siguiente simulación con tiempo final $t = 25$, resolución $h = 1/32$ y $N = 320$ puntos internos en cada dirección para poder poner la frontera en $r = 10$. La Figura 6.41 de abajo enseña el comportamiento de las constricciones de Einstein después del colapso a un AH.

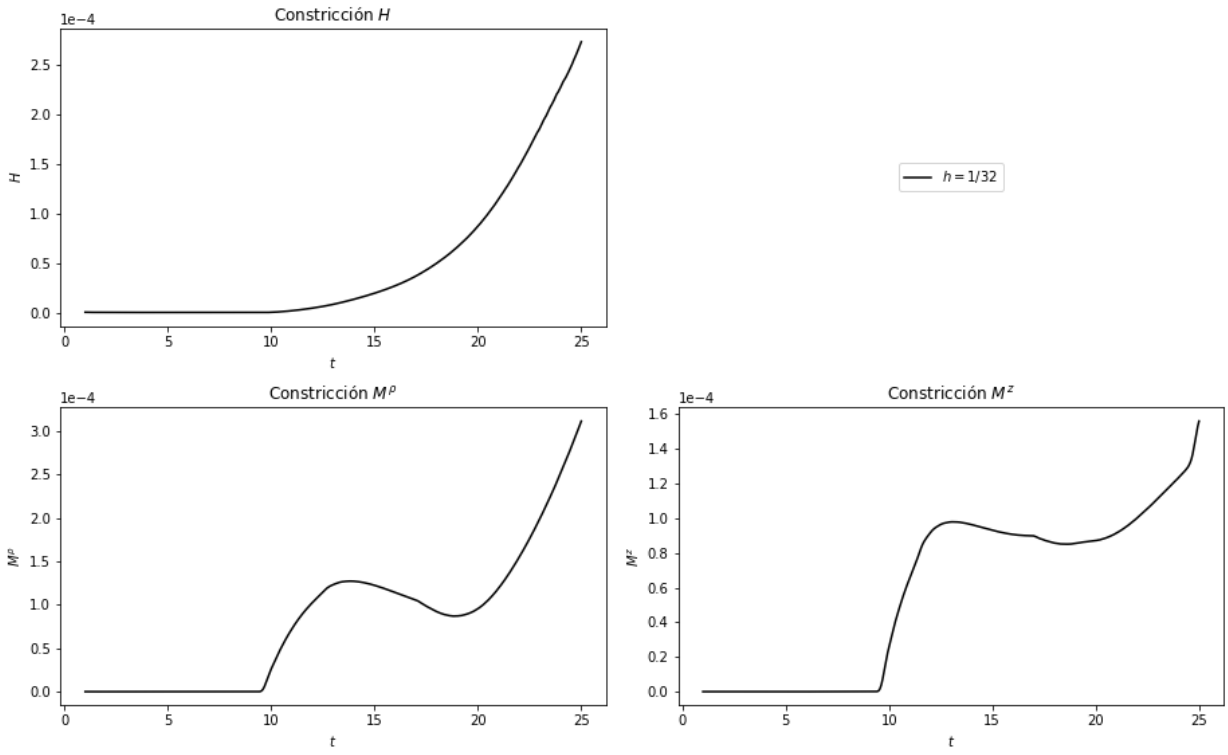

Figura 6.41: Constricciones hamiltoniana y de momento después del colapso a un AH para la simulación detallada con frontera en $r = 10$ y tiempo final $t = 25$.

Es claro que las constricciones explotan para $t \approx= 10$ donde las violaciones de la frontera no sólo ya están adentro de la malla reducida a la mitad sino que ya han llegado al origen. El hecho de que no se ve realmente una violación importante a $t \approx 5$ parece indicar que las violaciones de la frontera no generan tanto problema al caer en la malla reducida. Sin embargo, no hay duda de que la malla debe tener la frontera al menos en el tiempo máximo de evolución.

Otro indicador del comportamiento extraño y de que no hay convergencia se obtiene al graficar la masa del AH encontrado.

Figura 6.42: Evolución de la masa del AH encontrado por el código para lapso maximal con ondas de Brill $a = 11$ usando una malla con frontera reducida en $r = 10$ y tiempo final $t = 25$. Se ha graficado la masa ADM calculada como una línea negra sólida.

Al observar la Figura 6.42 inmediatamente resalta de manera atípica cómo la masa cae en $t \approx 16$. Aunque se alcanza a formar una idea más íntegra de la forma de la evolución de la masa comparándola con la Figura 6.37 donde sólo se evoluciona hasta $t = 5$, todavía no se alcanza a ver que la masa se estabilice en un valor por debajo de la masa ADM.

Toda esta última discusión ha servido para fundamentar que para tener la mayor previsión posible y evitar el ruido de las violaciones de la frontera, se debe seguir utilizando el esquema de sólo evolucionar hasta un tiempo igual a la mitad de la frontera espacial. En todo el análisis antecedente para decir a qué velocidad caen las violaciones de la frontera se ha hecho la simplificación de que la velocidad de la luz es uno en toda la malla. Esto es obviamente falso, ya que la velocidad de la luz es una cantidad que se puede calcular a partir de la métrica y el lapso α localmente en cada punto de la malla (véase la expresión (5.13)). Experimentalmente, por medio de algunas corridas, se encuentra que la información de la frontera cae un poco después de lo esperado al interior de la malla debido al colapso del lapso de tal manera que se puede justificar poner la frontera exterior en $r = 40$ en vez de en $r = 50$. Así resulta la malla final de 1000×1000 puntos y resolución $h = 0.04$. El código de este trabajo terminó dicha simulación en aproximadamente 52 horas con el mismo procesador *Intel i5-4690K* usando aproximadamente 20 Gb de memoria.

Las siguientes figuras se deben comparar directamente con las figuras de la implementación 'ingenua' donde la frontera se puso mucho más cerca. Primero se tiene el comportamiento de las constricciones:

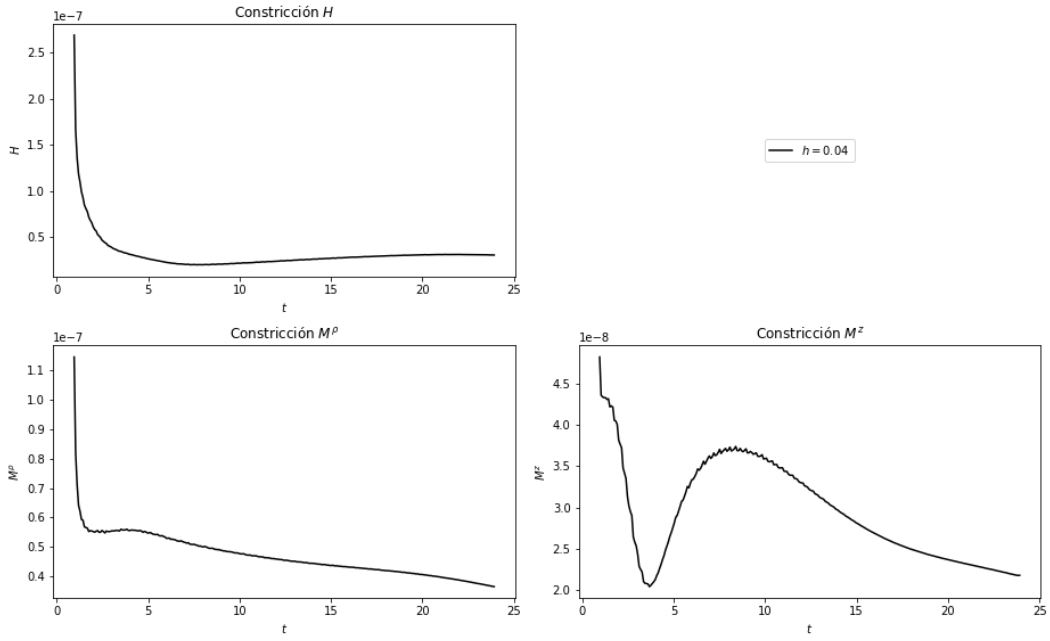

Figura 6.43: Constricciones hamiltoniana y de momento después del colapso a un AH para una onda de Brill con $a = 11$.

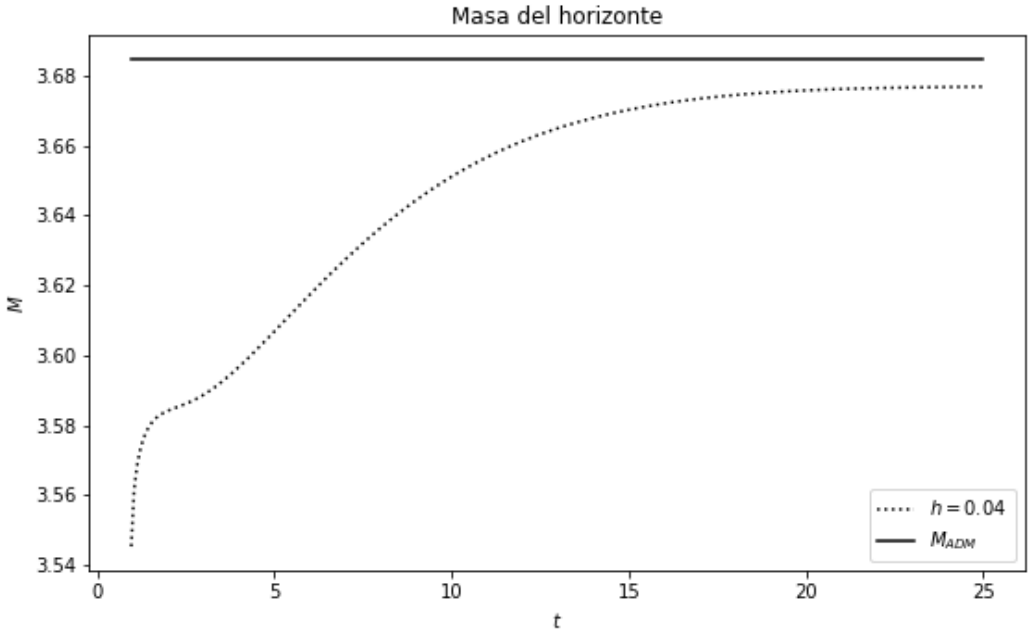

Figura 6.44: Evolución de la masa del AH encontrado por el código para lapso maximal con ondas de Brill $a = 11$. Se ha graficado la masa ADM calculada como una línea negra sólida.

La diferencia es dramática en el sentido de que ahora las constricciones se mantienen muy bien comportadas durante toda la evolución. Sin embargo, una advertencia: al observar con cuidado la gráfica, se puede ver que no se muestra toda la evolución hasta $t = 25$. De hecho se ha cortado en $t = 24$ ya que después de este tiempo se presenta un pico en cada una de las gráficas que sin duda viene de que a este tiempo ya se han introducido violaciones de las fronteras (recuérdese que se optó por poner la frontera en $r = 40$ en vez de en $r = 50$ para reducir el tamaño).

La Figura 6.44 ahora sí demuestra cómo la masa evoluciona de manera exhaustiva: desde cuando se encuentra el AH, cómo va subiendo durante la evolución, y finalmente, cómo se asienta en un valor por debajo de la masa ADM. Específicamente, el valor en $t = 25$, $M = 3.677$ difiere del valor calculado en (6.8) por menos de 0.2 %, que es más pequeño que el error reportado.

En conclusión, tomando todos los resultados de esta sección, se puede afirmar que el código equipado con lapso maximal puede evolucionar exitosamente ondas de Brill de pequeña y alta amplitud hasta tiempos tardíos. Como complemento a esta discusión y como se anticipó en la introducción de este capítulo, se responde ahora a la pregunta de por qué es tan importante la evolución a cuarto orden.

Básicamente, lo que sucede es que los fuertes gradientes que se forman en las variables exigen una alta resolución, sobre todo en las regiones donde el lapso colapsa violentamente. Para el caso de segundo orden, se encuentra que hay convergencia para los tiempos menores a la aparición del AH, pero después no pasa mucho para que deje de haber convergencia y en algunos casos el código llega a pararse por completo gracias a que en algunas variables se producen NaN's. La mejor manera de enseñar la diferencia drástica entre segundo y cuarto orden es graficar la evolución de la masa del AH. Dicha simulación ahora toma como amplitud $a = 6$ que es justamente una de las detalladas en [5].

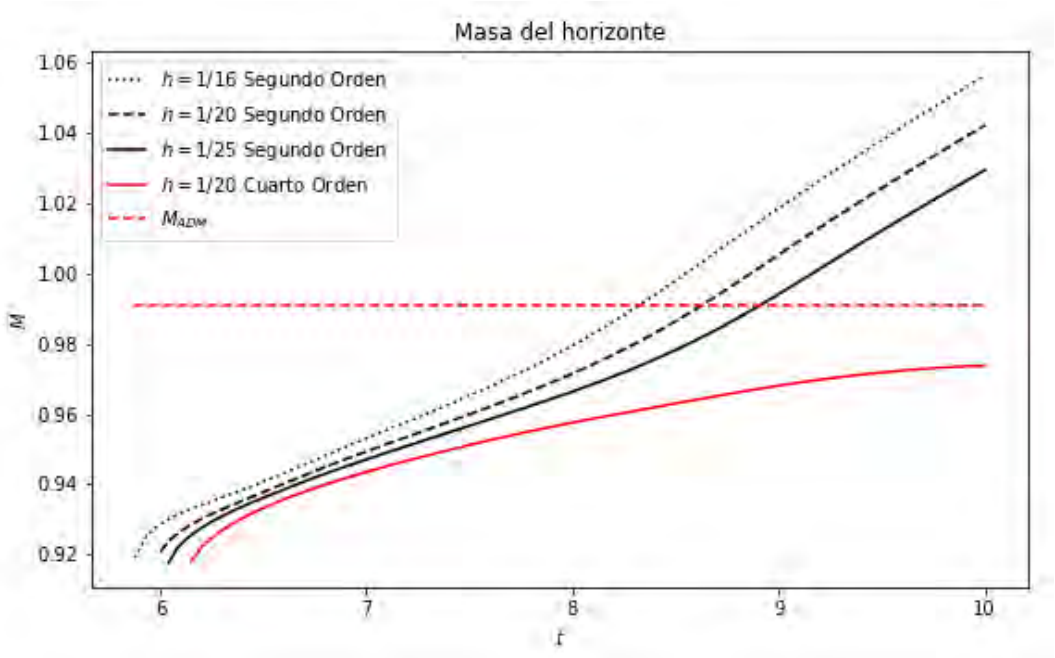

Figura 6.45: Evolución de la masa del AH encontrado por el código para lapso maximal con ondas de Brill $a = 6$. Las curvas negras denotan distintas resoluciones a segundo orden mientras que la curva roja sólida denota una evolución a cuarto orden y la roja punteada es el valor de la masa ADM.

La Figura 6.45 demuestra como las simulaciones a segundo orden pierden rápidamente sentido y la masa del AH sube por encima de la masa ADM. En cambio, la evolución a cuarto orden mantiene un valor mucho más esperado y reminiscente de la Figura 6.44 donde la masa converge a un valor cercano a la masa ADM. Aunque las curvas de segundo orden no presentan en lo absoluto este comportamiento, se puede ver empíricamente que al incrementar la resolución, las curvas se empiezan a pegar al valor correcto de la curva de cuarto orden. Posiblemente al implementar una resolución muy alta de $h = 1/100$ se pueda tener una curva que produzca los mismos resultados a cuarto orden, sin embargo, el costo computacional de tener mallas más grandes a segundo orden es mucho mayor que el que tener la misma resolución y evolucionar a cuarto orden. Puesto que el costo principal de la evolución del código es el sistema lineal que se está resolviendo, vale la pena recalcar de qué tamaño son dichos sistemas para segundo vs. cuarto orden. Al conocer de [37] que el tiempo de ejecución del resolvedor *PARDISO* va como $\mathcal{O}(n^2)$ (donde $n \approx N_\rho N_z$ es el número de ecuaciones) para sistemas lineales comparables a los que se están resolviendo (i.e. $n > 10^5$) se deduce que la implementación a cuarto orden es mucho más preferible si para hacer una simulación a segundo orden se requiere tener una malla 16 o 64 veces más grande.

La programación y la revisión de errores a cuarto orden es mucho más involucrada tanto en la etapa de escritura de los algoritmos así como en la etapa de revisión de convergencia y *debugging* pero para este trabajo se puede ver que el esfuerzo adicional ha valido la pena para poder hacer simulaciones exitosas y estables.

6.2.3. Onda cercana a la amplitud crítica $a = 4.5$

La última sección de este capítulo (y efectivamente así, también de este trabajo) es una especie de preámbulo al fenómeno crítico en las ondas de Brill. Alcubierre *et al.* reportan en [5] que al intentar hacer estudios en la región crítica tuvieron que implementar muy altas resoluciones para tener convergencia. Aún así determinaron que la región crítica está en $a_* = 5 \pm 1$ y con más estudio $a_* = 4.85 \pm 0.15$. Con base en esto, se prueba ahora el código con $a = 4.5$, tres resoluciones $h = 1/25, \sqrt{2}/50, 1/50$ y evolución hasta $t = 6$. La gráfica de las constricciones está abajo en la Figura 6.46.

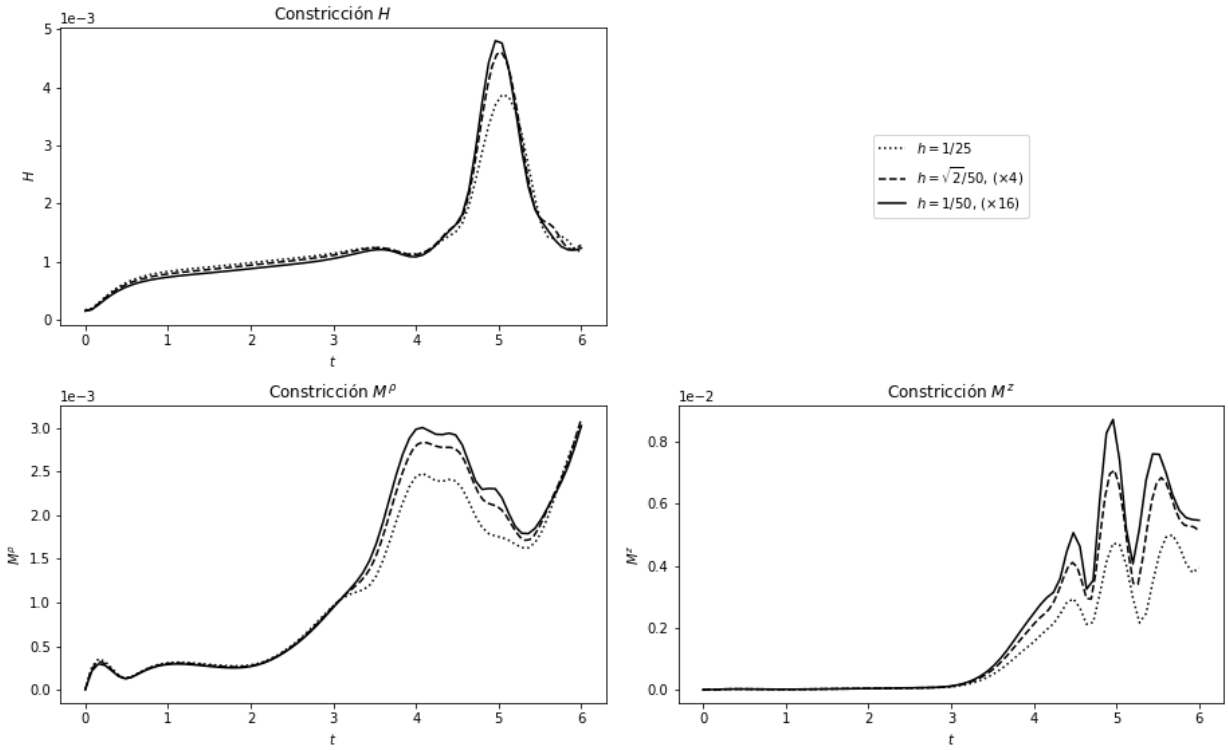

Figura 6.46: Evolución de las constricciones hamiltoniana y de momento para datos iniciales de Brill $a = 4.5$ con lapso maximal. Se presentan tres resoluciones y se reescalan las curvas por factores para una convergencia a cuarto orden.

Se puede visualizar que las constricciones suben de manera importante después de $t \approx 4$ y que aunque ahora los factores de convergencia $4, 16$ no son perfectos, sigue habiendo convergencia aunque un poco menor a cuarto orden. Con base en esta observación, es interesante saber qué está sucediendo a este tiempo y observar el lapso en la Figura 6.47 debajo.

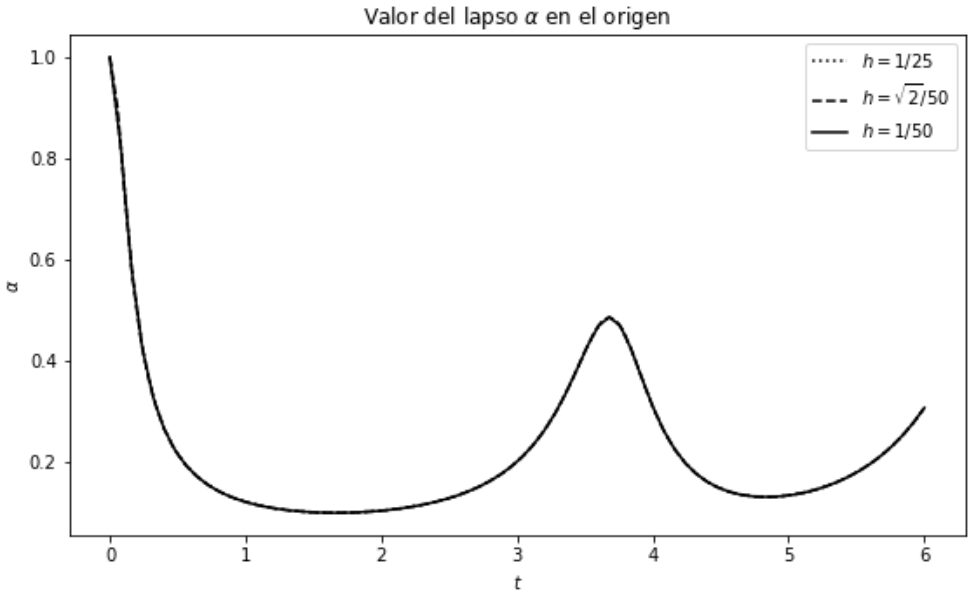

Figura 6.47: Evolución del valor del lapso en el origen para datos iniciales de Brill $a = 4.5$ con lapso maximal.

Dicha figura muestra que el lapso rebota con bastante fuerza. Para ser más precisos, en el caso de amplitud $a = 1$, la Figura 6.23 muestra también una oscilación en el lapso pero ahora tenemos una caída del lapso menor a $\alpha = 0.1$ que regresa a un valor de $\alpha \approx 0.5$ para luego caer inmediatamente después. El aumento de las constricciones y una discusión de estabilidad puede ilustrarse si se observa el factor de Courant calculado a partir de calcular la velocidad de la luz.

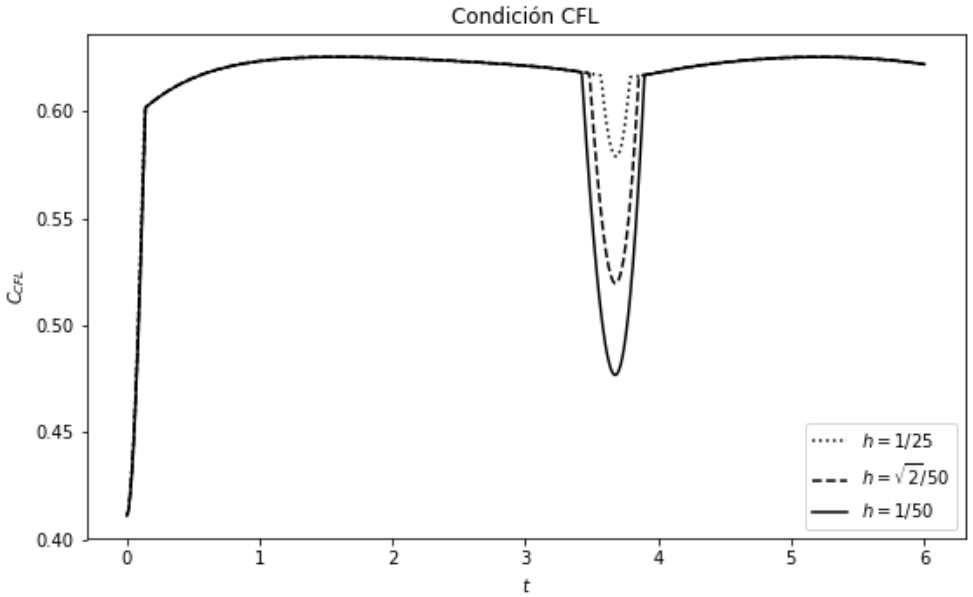

Figura 6.48: Evolución del factor de Courant para datos iniciales de Brill con $a = 4.5$.

Con la Figura 6.48 se puede apreciar que hay una caída drástica en el factor de Courant a $t \approx 4$ que es justamente donde el lapso está rebotando. Esta caída es de mayor magnitud a mayor resolución, hecho que puede dar luz a por qué se requieren grandes resoluciones para hacer una evolución exitosa. Como ya se mencionó, el código de evolución está equipado con un paso de tiempo variable para tomar en cuenta estas caídas del factor de Courant, pero es posible que se requiera implementar más mecanismos de control en estas secciones como puede ser poner disipación artificial o reducir el paso de tiempo aún más. El estudio del fenómeno crítico con este código sería un trabajo interesante a futuro interesante pero por ap:hora, el objetivo de esta sección ha sido demostrar que aún en para este tipo de simulaciones el código puede dar unos resultados valiosos en el sentido de que hay convergencia.

Capítulo 7

Conclusión

En este trabajo se han evolucionado ondas de Brill en un espacio axisimétrico con una formulación BSSN para coordenadas curvilíneas. Para tener un formalismo adecuado a las coordenadas curvilíneas ρ, z, φ asociadas al espacio cilíndrico, en el Capítulo 1 se desarrolló el formalismo tipo BSSN originario del trabajo [3]. Este formalismo es ideal para coordenadas curvilíneas ya que es explícitamente covariante, a diferencia del BSSN usual que tiene varias variables que no lo son. La discusión de este capítulo continuó para precisar los tipos de lapso que usaron en este código: el lapso maximal y el lapso $1 + \log$ de la familia Bona-Massó. Finalmente, se cerró el capítulo discutiendo sobre el proceso de regularización necesario para la evolución en axisimetría.

En el Capítulo 2 se introdujo la geometría de estudio principal: las ondas de Brill. Después de resumir algunos requisitos para las ondas, se explicó cómo surge la ecuación elíptica para el factor conforme y qué tipo de condiciones de frontera debe obedecer dicho factor. Pasando esto, se describió el proceso para introducir los datos de manera compatible al formalismo desarrollado en el Capítulo 1 y se concluyó el capítulo dando un resumen de resultados previos en el cálculo de las masas ADM, horizontes aparentes y evoluciones. En particular, se destacaron los trabajos [7] y [5] como las referencias centrales que sirvieron para constatar el buen funcionamiento del encontrador de horizontes en el Capítulo 4 y del código de evolución en el Capítulo 6.

La herramienta principal de este trabajo fue sin duda el resolvedor elíptico que se usó tanto para obtener datos iniciales de Brill como para resolver a cada paso de integración el lapso maximal. El Capítulo 3 desarrolló los aspectos técnicos y teóricos detrás del resolvedor elíptico. Se explicó primero la forma de la malla computacional utilizada en este trabajo y cómo se hicieron las discretizaciones en diferencias finitas. Después se discutieron las posibles condiciones de frontera, con particular atención a la frontera de Robin, ya que es esta la que se utilizó tanto en los datos iniciales de Brill, así como en el lapso maximal. Para dicha condición de Robin se siguió el trabajo de [16] para dar estimaciones del error introducido al no poner la frontera suficientemente lejos. La segunda parte del Capítulo 3 ahondó sobre las matrices *sparse* y de cómo se escribieron los sistemas lineales en formato CSR para después resolverse en el resolvedor directo PARDISO ([46] y [47]). La discusión concluyó poniendo atención particular en cómo optimizar el resolvedor elíptico, en particular cuando se están resolviendo sistemas lineales cada paso de integración (como es el lapso maximal) con estructura *sparse* idéntica.

Algunas amplitudes para las ondas de Brill puede ser suficientemente fuertes como para colapsarse en un horizonte aparente. Por lo tanto, desarrollar un encontrador de horizontes fue una parte central de este trabajo y fue esclarecida en el Capítulo 4. Ahí se mostró la teoría detrás de la condición geométrica de tener un horizonte aparente. Luego se resumió la forma del algoritmo implementado y para demostrar su uso correcto se hicieron pruebas contra la geometría analítica de Schwarschild, así como para ondas de Brill con suficiente amplitud tales que ya presentan el

horizonte en los datos iniciales.

El Capítulo 5 trató sobre el código computacional escrito para este trabajo: *OllinBrill*. Se discutió sobre sus similitudes y diferencias contra el código hermano *OllinAxis* de José Manuel Torres [61]. Puesto que las subrutinas de geometría pueden ser algo complejas (sobre todo si utilizan las técnicas de regularización descritas en el Capítulo 1), para constatar su buen funcionamiento se presentaron pruebas de convergencia. Después se habló de las condiciones de frontera implementadas en el código, de cómo llegan a meter ruido a la evolución y cómo inspeccionar las constricciones sin que la convergencia sea arruinada por las violaciones metidas por la frontera. También se hizo un breve tratamiento de la estabilidad de la condición CFL y del modo en que el código usa un paso adaptativo de tiempo para no tener violaciones de dicha condición. Finalmente para este Capítulo 5, se dio un resumen del archivo de parámetros del código, con el fin de que futuros usuarios puedan correrlo específicamente para los problemas que tienen en mente.

Como última parte de este trabajo, se presentaron los resultados principales en el Capítulo 6. Primero se hicieron pruebas de convergencia para las geometrías sencillas de Minkowski y de Schwarschild. Ambas demostraron la convergencia a cuarto orden del código y cimentaron las bases para correr el código con ondas de Brill. Aquí se trataron tres casos, todos los cuales convergieron a cuarto orden:

- El primero fue usar una onda de amplitud débil $a = 1$ donde se evolucionó hasta $t = 10$ con lapso maximal y se encontró que con esta amplitud, el espacio regresa a Minkowski ya que se detiene la evolución y el escalar de Ricci se reduce a cero.

- El segundo caso fue el opuesto al anterior, i.e. un onda de amplitud fuerte con $a = 11$. Aquí nuevamente se comprobó la convergencia a cuarto orden y se pudo encontrar un horizonte aparente a $t \approx 1$ para el cual también es posible calcularle la masa. Con dicha masa, se verificó que no se viola la desigualdad de Penrose [45] y de hecho, al evolucionar hasta $t = 25$, el valor de la masa del horizonte ya se ha estabilizado y difiere en menos de $0.2\,\%$ de la masa ADM calculada. Después de este resultado se explicó por qué es tan importante evolucionar a cuarto orden en vez de a segundo orden ya que segundo orden no es suficiente para algunos gradientes que se forman en la evolución.

- Como último caso se evolucionó una onda de Brill cercana a la amplitud crítica reportada en [5] de $a_* \approx 4.8$. La onda evolucionada fue de $a = 4.5$ y también mostró convergencia a cuarto orden aunque se visualiza que el orden de las constricciones sube notablemente debido a las fuertes oscilaciones. Dichas oscilaciones se ven claramente en el lapso y en la condición CFL. En particular la condición CFL mostró una fuerte caída del factor de Courant que se acentúa a mayor resolución, lo que posiblemente da una pista a por qué se necesita tan alta resolución en este tipo de evoluciones. Con base en estos resultados, se propone hacer un trabajo a futuro que estudie el fenómeno crítico en ondas de Brill.

Apéndice A

Uso de memoria, escalamiento y perfilado

Como apéndice a este trabajo se listan algunos detalles técnicos particulares para *OllinBrill*. Para hacer estos análisis, se compiló el ejecutable con el mayor nivel de optimizaciones (-O3) y se corrieron las pruebas de tiempo, memoria y perfilado con la aplicación *Intel VTune Amplifier* [21]. Se escogió esta aplicación debido a que da información mucho más completa sobre los hilos de *OpenMP* que otro perfilador como podría ser la alternativa de GNU: *gprof*.

A.1. Memoria

El uso de memoria suele ser fácil de predecir ya que la inmensa mayoría de la memoria usada por el programa es asignada explícitamente. En la siguiente Tabla A.1 se lista el uso de memoria para simulaciones de mallas cuadradas (es decir, $N_\rho = N_z$) usando lapso maximal.

$N_\rho = N_z$	Memoria
64	74Mb
128	296Mb
256	1Gb
512	5Gb
1024	22Gb

Tabla A.1: Uso de memoria para *OllinBrill*.

Es evidente que la memoria se cuadruplica al duplicar la extensión lineal en cada dimensión, debido a que $(2N_\rho)(2N_z) = 4N_\rho\, N_z$. Pese a que toda esta memoria se asigna sobre el *heap*, el espacio de memoria estará limitado por la memoria RAM física y la memoria virtual. Empíricamente, para una computadora de uso personal con memoria RAM entre $8 - 16$Gb y espacio de disco de 1Tb, *OllinBrill* no puede correr con mallas mayores a los 2048×2048. Para un cluster esto puede ser muy diferente ya que un solo nodo puede poseer memoria suficiente para correr la simulación de 2048×2048 en memoria RAM ($\sim$ 100Gb). Dependiendo de la configuración de memoria virtual y espacio de disco intrínseco al cluster, potencialmente se pueden correr mallas mucho más grandes.

A.2. Escalabilidad

Sin embargo, la limitación principal de *OllinBrill* no es el uso de memoria sino los largos tiempos del resolvedor elíptico. Al haber escrito el código en *OpenMP*, se espera que haya una mejora en el tiempo si se usan más hilos. Si corremos una simulación prueba de tamaño fijo 256×256 en la que variamos el número de hilos, se puede visualizar el comportamiento entre tiempo de ejecución t y número de hilos n (no se cuenta el tiempo de escritura de archivos en estas simulaciones, debido a que estas se hacen en serie):

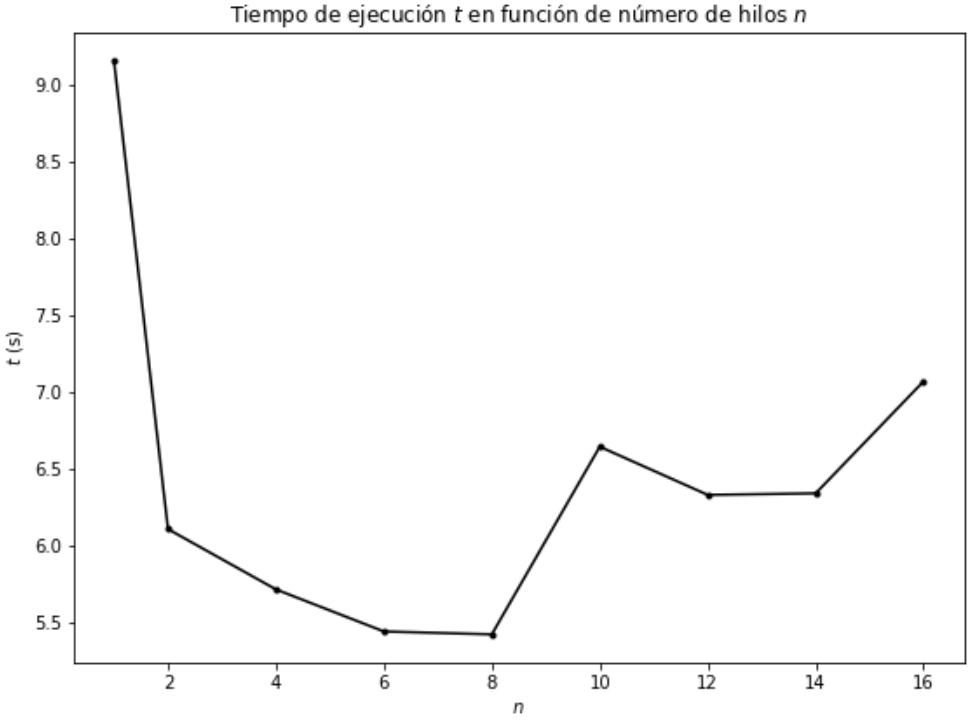

Figura A.1: Tiempos de ejecución de *OllinBrill* en una computadora personal con ocho núcleos.

Esta gráfica es para una computadora personal en la que el número de núcleos son ocho, lo que explica por qué el mejor comportamiento se obtiene en $n = 8$ y arriba de este número no hay mejora (en realidad, empeora el tiempo). Lo que es importante destacar es que el esfuerzo de paralelizar el código sí trae buenos resultados.

Para continuar, podemos ahora escalar a más hilos corriendo en el cluster del Instituto de Ciencias Nucleares de la UNAM, llamado *Xook*. Para correr en *OpenMP* sin tener que distribuir memoria, estamos limitados a hacer la ejecución en un solo nodo, pero dentro del nodo podemos pedir al cluster hasta 32 hilos, lo cual nos permite hacer el mismo análisis pero ahora hasta $n = 32$:

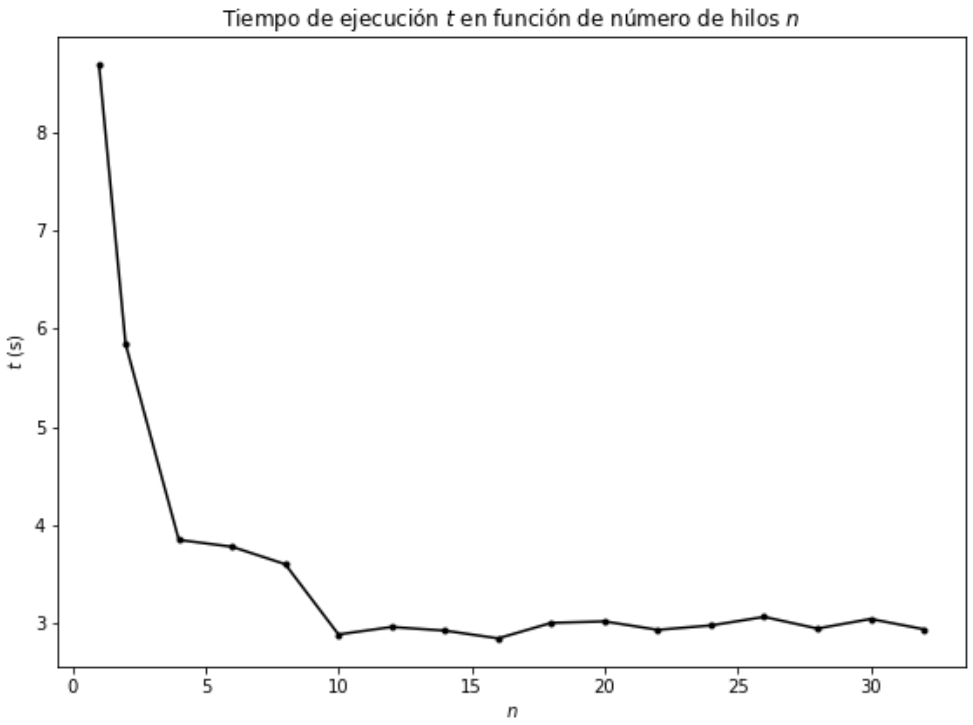

Figura A.2: Tiempos de ejecución de *OllinBrill* sobre el cluster *Xook* del ICN.

Nuevamente se comprueba que el aumento de hilos baja el tiempo de ejecución dentro de la región $n \in [1, 16]$. De manera análoga a la máquina de uso personal, hay un valor para el cual no hay realmente mejora si se aumenta el número de hilos. La explicación a por qué este número es aproximadamente 16 está fuera del enfoque de este trabajo. Sin embargo, sí se puede usar como justificación a que otra posible línea de trabajo a futuro con este código sea implementar un sistema de memoria distribuida en *MPI* para poder correr con muchos más hilos y procesadores.

A.3. Perfilado

Como última parte de este apéndice se detalla sobre el perfilado (o *profiling* en inglés) de *OllinBrill* corriendo con lapso maximal. Este perfilado se hizo en una máquina de uso personal corriendo con ocho hilos de *OpenMP*. La malla fue de tamaño 256×256 y se hizo una evolución de 50 pasos de integración escribiendo a archivos.

De entre los resultados, podemos ver el uso efectivo de CPU's en la siguiente Figura A.3

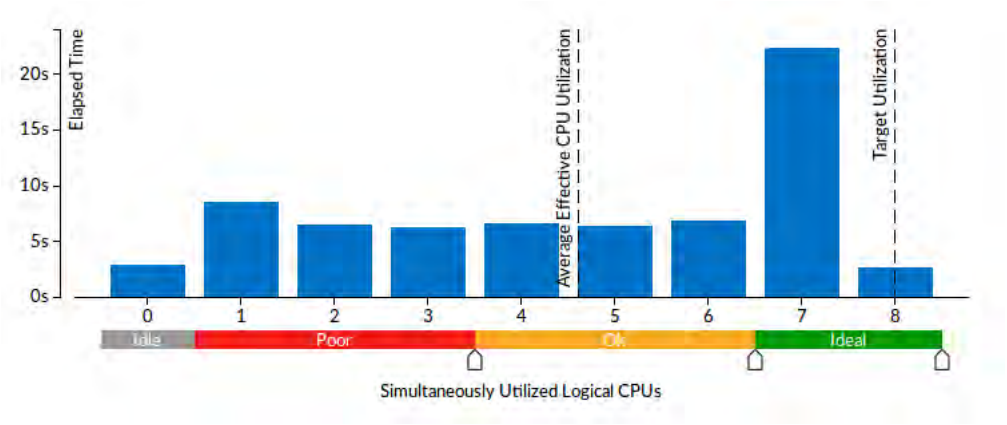

Figura A.3: Resultado de perfilado de *Intel VTune Amplifer* para *OllinBril*.

Aquí se ve que el programa está corriendo la buena parte del tiempo con siete de los ocho hilos efectivos. Esto puede considerarse bueno y podemos explicar que el programa corre con menor número cuando está esperando a que otros hilos terminen su ejecución (es decir, cuando hay sincronizaciones). De hecho, esto se comprueba cuando vemos el uso porcentual de tiempo correspondiente a subrutinas que se puede resumir en la siguiente tabla.

Subrutina	Tiempo de Ejecución (%)
Resolvedor elíptico.	70 %
Cálculo geometría.	12 %
Sincronizaciones.	12 %
Copias y acumulaciones.	2 %
Datos iniciales.	1 %
Escritura a archivos.	0.5 %
Otras.	2.5 %

Tabla A.2: Perfilado para *OllinBrill*.

Entre los resultados de esta tabla, se puede resaltar que la espera en sincronizaciones de las áreas paralelas tiene un costo importante, tanto así que empata como la segunda área de mayor tiempo junto con las subrutinas de geometría (derivadas, Ricci, etc.). Sin embargo, con base en la sección anterior sobre la disminución de tiempo al aumentar el número de hilos, se ve que el beneficio de esperar estas sincronizaciones supera por mucho al de correr con un solo hilo. El resto de las subrutinas tienen muy poco impacto en el tiempo total de ejecución, cosa que puede ser sorprendente para la escritura a archivos que se realiza explícitamente en serie.

Como conclusión, se puede afirmar que *OllinBrill* adquiere beneficios importantes al paralelizarse y que esta paralelización tiene una distribución de trabajo balanceada. De manera complementaria, la información del perfilado indica que las optimizaciones se deben hacer primero sobre el resolvedor elíptico (posiblemente con un Full MultiGrid para mallas más grandes) y usando un mayor número de procesadores al implementar un sistema de memoria distribuida.

Bibliografía

[1] M. Alcubierre. *Introduction to 3 + 1 Numerical Relativity*. Oxford University Press, 2008.

[2] M. Alcubierre y J. A. González. «Regularization of spherically symmetric evolution codes in numerical relativity». En: *Computer Physics Communications* 167.2 (2005), págs. 76-84. ISSN: 0010-4655.

[3] M. Alcubierre y M. D. Méndez. «Formulations of the 3+1 evolution equations in curvilinear coordinates». En: *General Relativity and Gravitation* 43.10 (jun. de 2011), pág. 2769.

[4] M. Alcubierre y col. «Gauge conditions for long-term numerical black hole evolutions without excision». En: *Phys. Rev. D* (2003).

[5] M. Alcubierre y col. «Gravitational collapse of gravitational waves in 3D numerical relativity». En: *Phys. Rev. D* 61 (4 ene. de 2000), pág. 041501.

[6] M. Alcubierre y col. «Symmetry without symmetry: Numerical simulation of axisymmetric systems using cartesian grids». En: *International Journal of Modern Physics D* 10.03 (2001), págs. 273-289.

[7] M. Alcubierre y col. «Test-beds and applications for apparent horizon finders in numerical relativity». En: *Classical and Quantum Gravity* 17.11 (2000), pág. 2159.

[8] Miguel Alcubierre y col. «Symmetry without Symmetry: Numerical Simulation of Axisymmetric Systems using Cartesian Grids». En: *Int. J. Mod. Phys. D* 10 (2001), págs. 273-290. DOI: 10.1142/S0218271801000834. eprint: arXiv:gr-qc/9908012.

[9] P. R. Amestoy y col. «A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling». En: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), págs. 15-41.

[10] P. R. Amestoy y col. «Hybrid scheduling for the parallel solution of linear systems». En: *Parallel Computing* 32.2 (2006), págs. 136-156.

[11] G.B. Arfken y H.J. Weber. *Mathematical Methods for Physicists*. Mathematical Methods for Physicists. Elsevier, 2005. ISBN: 9780120598762.

[12] R. Arnowitt, S. Deser y C. W. Misner. «The dynamics of general relativity». En: *Gravitation: An introduction to current research*.

[13] Satish Balay y col. *PETSc Web page*. http://www.mcs.anl.gov/petsc. 2018. URL: http://www.mcs.anl.gov/petsc.

[14] J. M. Bardeen y T. Piran. «General relativistic axisymmetric rotating systems: Coordinates and equations». En: *Physics Reports* 96.4 (1983), págs. 205-250. ISSN: 0370-1573.

[15] T.W. Baumgarte y S. L. Shapiro. «On the numerical integration of Einstein's field equations». En: *Phys. Rev. D.* (1998).

[16] Alvin Bayliss, Max Gunzburger y Eli Turkel. «Boundary Conditions for the Numerical Solution of Elliptic Equations in Exterior Regions». En: *SIAM Journal on Applied Mathematics* 42.2 (1982), págs. 430-451. DOI: 10.1137/0142032. eprint: https://doi.org/10.1137/0142032. URL: https://doi.org/10.1137/0142032.

[17] C. Bona y col. «New formalism for numerical relativity». En: *Phys. Rev. Lett.* (1995).

[18] D. R. Brill. «On the positive definite mass of the Bondi-Weber-Wheeler time-symmetric gravitational waves». En: *Annals of Physics* (1959).

[19] J. D. Brown. «Covariant formulations of BSSN and the standard gauge». En: *Phys. Rev. D* (2009).

[20] Intel Corportation. *Intel MKL PARDISO - Parallel Direct Sparse Solver Interface*. 2018. URL: https://software.intel.com/en-us/mkl-developer-reference-fortran-intel-mkl-pardiso-parallel-direct-sparse-solver-interface (visitado 17-04-2018).

[21] Intel Corportation. *Intel VTune Amplifier: Modern Processor Performance Analysis*. 2018. URL: https://software.intel.com/en-us/intel-vtune-amplifier-xe (visitado 17-04-2018).

[22] Richard Courant, Kurt Friedrichs y Hans Lewy. «On the partial difference equations of mathematical physics». En: *IBM journal of Research and Development* 11.2 (1967), págs. 215-234.

[23] Leonardo Dagum y Ramesh Menon. «OpenMP: An Industry Standard API for Shared Memory Programming». En: *IEEE Comput. Sci. Eng.* 5.1 (ene. de 1998), págs. 46-55. ISSN: 1070-9924. DOI: 10.1109/99.660313. URL: https://doi.org/10.1109/99.660313.

[24] Intel Corportation: Software Solutions Group - Developer Products Division. «Intel MKL Sparse Solvers». 2007. URL: https://software.intel.com/sites/default/files/managed/de/87/MKL_SparseSolvers.pdf.

[25] Iain S. Duff. «Direct Methods for Solving Sparse Systems of Linear Equations». En: *SIAM Journal on Scientific and Statistical Computing* 5.3 (1984), págs. 605-619. DOI: 10.1137/0905043. eprint: https://doi.org/10.1137/0905043. URL: https://doi.org/10.1137/0905043.

[26] Albert Einstein. «Die Feldgleichungen der Gravitation». En: *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften (Berlin), Seite 844-847.* (1915).

[27] K. Eppley. «Evolution of time-symmetric gravitational waves: Initial data and apparent horizons». En: *Phys. Rev. D* 16 (6 sep. de 1977), págs. 1609-1614.

[28] Message P Forum. *MPI: A Message-Passing Interface Standard*. Inf. téc. Knoxville, TN, USA, 1994.

[29] Nicholas IM Gould, Jennifer A Scott y Yifan Hu. «A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations». En: *ACM Transactions on Mathematical Software (TOMS)* 33.2 (2007), pág. 10.

[30] A. Gupta y V. Kumar. «Parallel Algorithms for Forward and Back Substitution in Direct Solution of Sparse Linear Systems». En: *Proceedings of the IEEE/ACM SC95 Conference*. 1995, págs. 74-74. DOI: 10.1109/SUPERC.1995.242069.

[31] Susan G Hahn y Richard W Lindquist. «The two-body problem in geometrodynamics». En: *Annals of Physics* 29.2 (1964), págs. 304-331. ISSN: 0003-4916. DOI: https://doi.org/10.1016/0003-4916(64)90223-4. URL: http://www.sciencedirect.com/science/article/pii/0003491664902234.

[32] Peter Hansbo. «Nitsche's method for interface problems in computational mechanics». En: 28 (nov. de 2005).

[33] S.W. Hawking y G.F.R. Ellis. *The Large Scale Structure of Space-Time*. Cambridge Monographs on Mathematical Physics. Cambridge University Press, 1975. ISBN: 9781139810951.

[34] D. E. Holz y col. «Coalescence of Primal Gravity Waves to Make Cosmological Mass Without Matter». En: *Directions in General Relativity: Proceedings of the 1993 International Symposium, Maryland: Papers in Honor of Dieter Brill*. Ed. por B. L. Hu y T. A.Editors Jacobson. Vol. 2. Cambridge University Press, 1956, págs. 339-358. DOI: 10.1017/CBO9780511524653.028.

[35] Wolfram Research Inc. *Mathematica, Version 11.3*. Champaign, IL, 2018.

[36] J.D. Jackson. *Classical Electrodynamics*. Wiley, 2012. ISBN: 9788126510948.

[37] J Kwack, G Bauer y Seid Koric. «Performance Test of Parallel Linear Equation Solvers on Blue Waters–Cray XE6/XK7 system». En:

[38] R.J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Other Titles in Applied Mathematics. Society for Industrial y Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2007. ISBN: 9780898717839.

[39] A. Lichnerowicz. «L'intégration des équations de la gravitation relativiste et la problème des n corps». En: *J. Math. Pures et Appl.* (1944).

[40] Mark Lidner. *libconfig C/C++ library for processing configuration files*. 2018. URL: https://hyperrealm.github.io/libconfig/ (visitado 17-04-2018).

[41] Frank Löffler y col. «The Einstein Toolkit: A Community Computational Infrastructure for Relativistic Astrophysics». En: *Class. Quantum Grav.* 29.11 (2012), pág. 115001. DOI: doi:10.1088/0264-9381/29/11/115001. eprint: arXiv:1111.3344[gr-qc].

[42] Nora Lüthen, Mika Juntunen y Rolf Stenberg. «An improved a priori error analysis of Nitsche's method for Robin boundary conditions». En: *Numerische Mathematik* 138.4 (abr. de 2018), págs. 1011-1026. ISSN: 0945-3245. DOI: 10.1007/s00211-017-0927-1. URL: https://doi.org/10.1007/s00211-017-0927-1.

[43] Niall Ó Murchadha. *Brill Waves*. 1993. eprint: arXiv:gr-qc/9302023.

[44] Santiago. Ontañón. *OllinBrill: Brill Wave Simulation in Axisymmetric Space*. https://github.com/sontanon/OllinBrill. 2018.

[45] Roger Penrose. «Naked Singularities». En: *Annals of the New York Academy of Sciences* 224.1 (), págs. 125-134. DOI: 10.1111/j.1749-6632.1973.tb41447.x.

[46] C. G. Petra, O. Schenk y M. Anitescu. «Real-Time Stochastic Optimization of Complex Energy Systems on High-Performance Computers». En: *Computing in Science Engineering* 16.5 (sep. de 2014), págs. 32-42. ISSN: 1521-9615. DOI: 10.1109/MCSE.2014.53.

[47] Cosmin G. Petra y col. «An Augmented Incomplete Factorization Approach for Computing the Schur Complement in Stochastic Optimization». En: *SIAM Journal on Scientific Computing* 36.2 (2014), págs. C139-C162. DOI: 10.1137/130908737. eprint: https://doi.org/10.1137/130908737. URL: https://doi.org/10.1137/130908737.

[48] William H. Press y col. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1992. ISBN: 0-521-43108-5.

[49] Oliver Rinne y Stewart J. M. «A strongly hyperbolic and regular reduction of Einstein's equations for axisymmetric spacetimes». En: *Classical and Quantum Gravity* (2005).

[50] M. Ruiz, M. Alcubierre y D. Núñez. «Regularization of spherical and axisymmetric evolution codes in numerical relativity». En: *General Relativity and Gravitation* 40.1 (ene. de 2008), págs. 159-182.

[51] Y. Saad. *Iterative Methods for Sparse Linear Systems: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial y Applied Mathematics, 2003. ISBN: 9780898715347.

[52] Olaf Schenk y Klaus Gärtner. *PARDISO User Guide*. English. Ver. Version 6.1.0. PARDISO Project. 6 de ene. de 2018. 2018-04-18.

[53] S. L. Shapiro y T. W. Baumgarte. *Numerical Relativity: Solving Einstein's Equations on the Computer*. Cambridge University Press, 2010.

[54] M. Shibata y T. Nakamura. «Evolution of three-dimensional gravitational waves: Harmonic slicing case». En: *Phys. Rev. D.* (1995).

[55] L. Smarr y J. W. York. «Kinematical conditions in the construction of spacetime». En: *Phys. Rev. D* (1978).

[56] Peter Sonneveld. «CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear systems». En: *SIAM Journal on Scientific and Statistical Computing* 10.1 (1989), págs. 36-52. DOI: 10.1137/0910004. eprint: https://doi.org/10.1137/0910004. URL: https://doi.org/10.1137/0910004.

[57] *Sparse Matrices*. Mathematics in Science and Engineering. Elsevier Science, 1973.

[58] G. Strang. *Computational Science and Engineering*. Wellesley-Cambridge Press, 2007. ISBN: 9780961408817.

[59] J. Thornburg. «Event and Apparent Horizon Finders for 3+1 Numerical Relativity». En: *ArXiv General Relativity and Quantum Cosmology e-prints* (dic. de 2005). eprint: gr-qc/0512169.

[60] J. Thornburg. «Event and Apparent Horizon Finders for 3+1 Numerical Relativity». En: *ArXiv General Relativity and Quantum Cosmology e-prints* (dic. de 2005). eprint: gr-qc/0512169.

[61] J. M. Torres. «Dinámica de materia cargada en relatividad general». Tesis doct. Universidad Nacional Autónoma de México, Instituto de Ciencias Nucleares.

[62] J. M. Torres. «Regularization of the generalized BSSN formulation for axisymmetric spacetimes». En: *AIP Conference Proceedings* 1473.1 (2012), págs. 37-42.

[63] J. W. York. «Kinematics and dynamics of general relativity». En: *Sources of gravitational radiation*.