Universidad Nacional Autónoma de México

Posgrado en Ciencias Físicas

CÓDIGO COMPUTACIONAL PARA RESOLVER ECUACIONES ELÍPTICAS EN RELATIVIDAD NUMÉRICA

T E S I S

QUE PARA OBTENER EL TÍTULO DE

Maestro en ciencias

PRESENTA

JUAN PABLO GALAVIZ VILCHIS

DIRECTOR DE TESIS

Dr. Darío Núñez Zúñiga

2007

Sinodales:

Propietario: Darío Núñez Zúñiga

Propietario: Francisco Guzmán Murillo

Propietario: Jorge Luis Cervantes Cota

Suplente: Miguel Alcubierre Moya

Suplente: Mario Alberto Rodríguez Meza

Agradecimientos.

Esta tesis esta dedicada con mucho cariño a las siguientes personas:

A mi padres Alicia y Manuel por su gran cariño y apoyo, a mis hermanos Víctor y Maricela y mis sobrinos Daniel, Braulio y Bruno. A mi tía Lupita y Rodofo.

A Mariana Vargas y su familia Germán Vargas, Rosa María Magaña, Rosita y German con mucho cariño.

A mi comité tutoral Darío Núñez, Miguel Alcubierre y Tonatiuh Matos por todo el apoyo, enseñanzas y amistad que me han brindado. Un agradecimiento en especial a Francisco Guzmán Murillo por la ayuda con los códigos y por los comentarios.

A mi segunda familia mi tío Marco Antonio, mi tía Pola y mis primos Martha, Alba, Carlos Ramón, Mode y Sergio, sin su ayuda esto no sería posible.

A mis amigos y familiares Tannya, Nico, Ali, Adrian, Toño, Esteban, Alejandro, Paty, Milton y una larga lista que no aparece aquí, gracias por su amistad.

Índice general

Capítulo 1

Introducción

Las ecuaciones diferenciales son el pilar de las herramientas matemáticas utilizadas por las teorías físicas. Las ecuaciones diferenciales parciales lineales se clasifican en tres tipos: hiperbólicas, elípticas y parabólicas, esta clasificación es motivada por la clasificación de las secciones cónicas en geometría analítica[1]. Un ejemplo típico de ecuación hiperbólica es la ecuación de onda:

$$\frac{\partial^2 \psi}{\partial t^2} = v^2 \nabla^2 \psi, \tag{1.1}$$

en donde $\psi(\vec{x}, t)$ es la amplitud de la onda y v es la velocidad de propagación de la misma y depende de la densidad del medio en donde se propaga la onda. La ecuación de onda es una ecuación de evolución, es decir, determina la forma de la onda para todo tiempo. Para obtener la solución de este tipo de ecuación hay que dar el valor de ψ en el espacio, su primer derivada temporal para $t = 0$ y condiciones de frontera.

Por otro lado, una ecuación elíptica es la ecuación de Poisson:

$$\nabla^2 \phi = \rho, \tag{1.2}$$

en donde $\phi(\vec{x})$ es por ejemplo, el valor del potencial eléctrico y $\rho(\vec{x})$ es una función que representa la densidad de carga en un medio. La ecuación de Poisson es una ecuación que no depende del tiempo, por lo que la solución no evoluciona y solo depende de los valores de ϕ en la frontera.

Una ecuación parabólica es la ecuación de calor:

$$\frac{\partial \Theta}{\partial t} = \kappa \nabla^2 \Theta, \tag{1.3}$$

en donde $\Theta(\vec{x}, t)$ representa la temperatura en un medio y κ es una constante que depende de la conductividad térmica del material, su densidad y capacidad calorífica. La ecuación de calor es una ecuación de evolución que tiende a una solución estacionaria. La solución estacionaria de la ecuación de Calor es una solución de la ecuación de Laplace[2]. Como se verá más adelante, la idea

[1] La forma general de una ecuación diferencial parcial de segundo orden lineal en dos variables es: $Au_{,xx} + 2Bu_{,xy} + Cu_{,yy} + Du_{,x} + Eu_{,y} + Fu + G = 0$, que es equivalente a la forma de las ecuaciones de las secciones cónicas en geometría analítica: $Ax^2 + 2Bxy + Cy^2 + Dx + Ey + F = 0$. En los dos casos de acuerdo con el signo del discriminante $d = AC - B^2$ las ecuaciones se clasifican como: elípticas si $d > 0$, hiperbólicas si $d < 0$ y parabólicas si $d = 0$. Ver sección 4 para más detalles.

[2] La ecuación de Laplace es una ecuación de Poisson con fuente $\rho = 0$

de obtener la solución de una ecuación elíptica utilizando la solución estacionaria de una ecuación parabólica es una de las ideas centrales de los métodos iterativos.

En la siguiente tabla se resume el tipo de operadores diferenciales y el comportamiento genérico de las soluciones. Cabe mencionar que a pesar de que la clasificación se hace para operadores lineales, muchas de las propiedades son heredadas por operadores semi-lineales y en ocasiones por operadores cuasi-lineales.

Tipo de Ecuación	**Comportamiento de la solución**
Elíptica.	Función estacionaria y cuya forma cumple propiedades de minimización.[3]
Parabólica.	Función muy suave y que representa flujos que se dispersan.
Hiperbólica.	Función ondulatoria y que representa perturbaciones que se preservan.

Es común en los sistemas físicos y en particular en la teoría de Relatividad General, que las ecuaciones que describen la evolución de los sistemas sean ecuaciones de tipo hiperbólico, mientras que las ecuaciones elípticas definen constricciones al sistema. Un ejemplo es la electrodinámica, en donde las ecuaciones de evolución están dadas por la ley de Faraday y la de Ampère-Maxwell:

$$\frac{\partial \vec{E}}{\partial t} = -\nabla \times \vec{E}, \tag{1.4}$$

$$\frac{\partial \vec{D}}{\partial t} = \nabla \times \vec{H} - 4\pi \vec{J}, \tag{1.5}$$

y las ecuaciones de constricción por la ley de Gauss eléctrica y magnética:

$$\nabla \cdot \vec{D} = 4\pi\rho, \tag{1.6}$$

$$\nabla \cdot \vec{B} = 0, \tag{1.7}$$

en término de los potenciales, estas ecuaciones exhiben la naturaleza hiperbólica y elíptica de las mismas.

Una gran cantidad de problemas físicos requieren de soluciones a ecuaciones elípticas para las cuales no existen métodos de solución que den una forma analítica cerrada, es por ello que muchas veces es necesario obtener la solución de forma numérica. Sin embargo, los métodos numéricos más simples para resolver ecuaciones elípticas son ineficientes para obtener soluciones precisas a problemas tridimensionales. Por ello es necesario implementar técnicas más complicadas pero que aumentan la eficiencia de los métodos. Como se verá más adelante en relatividad numérica es indispensable contar con este tipo de *resolvedores elípticos*[4], capaces de obtener soluciones para una amplia gama de ecuaciones elípticas y que además sean muy eficientes.

A lo largo de varios años se han desarrollado códigos computacionales para resolver ecuaciones elípticas. Existen diversas bibliotecas numéricas y una gran cantidad de libros y artículos de investigación que plantean métodos de solución para ecuaciones elípticas que aparecen en diversos contextos. Sin embargo, por el momento no existen "códigos universales", que se apliquen de forma general o que se puedan adaptar fácilmente, es por ello que en la experimentación numérica, es más común desarrollar los códigos computacionales para problemas específicos y adaptar códigos de terceros cuando la implementación de las bibliotecas resulta más conveniente, por ejemplo en

[3] Según el problema se minimiza la energía, el área, el volumen, etc.

[4] Por resolvedor elíptico se entenderá el código computacional que implementa algún algoritmo numérico para resolver ecuaciones elípticas.

el código CACTUS del cual hablaré más adelante se pueden utilizar varias bibliotecas como MPI, Lapack, BLAS y PETSc[5].

Para esta tesis se desarrolló un código que implementa el método iterativo de Gauss-Seidel y el método Multigrid para resolver ecuaciones elípticas en relatividad numérica. Es común que los programas tengan un nombre que los identifique y que exhiban alguna característica particular, en este caso nombré al código Olliptic, que es derivado del nombre del grupo de Relatividad Numérica del ICN (el grupo Ollin) y de la palabra elíptico. El código Olliptic fue construido como parte de las bibliotecas del código CACTUS [1, 2] de relatividad numérica. Esta tesis trata sobre los fundamentos matemáticos necesarios para desarrollar el código Olliptic y sobre las aplicaciones del mismo. La organización de la tesis es la siguiente:

- En lo que resta de este capítulo se darán algunos antecedentes, principalmente del trabajo numérico que se ha realizado para dotar al código CACTUS de bibliotecas para resolver ecuaciones elípticas. Se explicará la motivación de la tesis y sus aplicaciones.

- En el capítulo 2 se da una breve introducción a relatividad numérica, en donde se enfatiza en la clasificación de las ecuaciones en dos tipos: las de evolución y las de constricción. Se hacen algunas analogías con la mecánica clásica para entender el papel que juegan las constricciones y las ecuaciones dinámicas.

- Una de las principales aplicaciones de las soluciones numéricas de ecuaciones elípticas en relatividad numérica es la generación de datos iniciales. En el capítulo 3 se hace un resumen de algunos métodos para obtener ecuaciones elípticas que generan datos iniciales y se dan algunos ejemplos.

- En el capítulo 4 se estudian algunos métodos de solución de ecuaciones elípticas. Primero se da la clasificación formal y se mencionan algunos métodos analíticos. Posteriormente se procede a la parte central del trabajo que son los método numéricos. Se definen algunos conceptos útiles para estimar el error de aproximación y la convergencia. Se da una introducción a la discretización de operadores usando diferencias finitas y se describen algunos métodos aplicados a ecuaciones elípticas usando diferencias finitas; en particular se estudia el método Multigrid, que es el método más eficiente y que se implementa en el código computacional. Se describe el método seudo-espectral, que es el más preciso y se plantean algunas ideas para su implementación futura en el código. Por último, se describen las condiciones de frontera utilizadas en el código.

- En el capítulo 5 se describen las características del código Olliptic y su implementación dentro del código CACTUS de relatividad numérica. Se mencionan posibles extensiones y mejoras al código que quedan fuera del alcance de este trabajo. Se realizan pruebas al código utilizando ecuaciones modelo. Se hacen algunos experimentos numéricos para probar convergencia, precisión y eficiencia del código. Se estudian dos ecuaciones: la ecuación de Poisson y una ecuación elíptica lineal.

- En el capítulo 6 se hacen dos experimentos numéricos, el primero es la solución del factor conforme para obtener datos iniciales para ondas de Brill. El segundo aplicado al sistema Schrödinger-Poisson que describe la evolución de un campo escalar en el límite post-Newtoniano.

[5] En particular PETSc es una biblioteca numérica que implementa resolvedores elípticos, pero que por el momento no se ha podido adaptar completamente en aplicaciones de relatividad numérica.

- El capítulo 7 está dedicado a las conclusiones y a dar un panorama general de las posibles extensiones y mejoras que se pueden hacer al código.
- Por ultimo, en los apéndices se dan algunos detalles de las técnicas de programación utilizadas en el código.

1.1. Antecedentes

Las ecuaciones del campo gravitacional de Einstein:

$$R_{ab} - \frac{1}{2} g_{ab} R = 8\pi T_{ab}, \tag{1.8}$$

conforman un sistema de ecuaciones diferenciales parciales acopladas y no lineales con cientos de términos. Además, las ecuaciones están escritas en forma covariante, lo cual implica que es posible seleccionar cualquier condición de norma para obtener la solución. Esta libertad de norma es útil para obtener soluciones a las ecuaciones, sin embargo, también complica la interpretación física de las mismas. Para campos gravitacionales fuertes y dinámicos, es prácticamente imposible obtener soluciones analíticas a las ecuaciones de Einstein, es por ello que surge la Relatividad Numérica como un campo de estudio independiente, que tiene como principal objetivo estudiar este tipo de sistemas.

A principios de la década de los 60's Arnowitt, Deser y Misner [3], iniciaron los primeros intentos por reformular la teoría de la Relatividad General, rompiendo la covariancia para poder obtener una formulación Hamiltoniana de la Relatividad General. Sin embargo, los primeros trabajos exitosos en esta área se llevaron a cabo a mediados de la década de los 70's por Smarr [4] y Eppley [5]. Estos trabajos se orientaron en el problema de colisión de dos agujeros negros, sin embargo fue hasta años recientes que el problema se resolvió en la forma más general. [6, 7, 8]

El problema de la colisión de agujeros negros fue un problema que impulsó el desarrollo de la relatividad numérica, la formación de grupos de investigación en esa área y el desarrollo de códigos computacionales. Dentro de esos esfuerzos se conformó el código CACTUS[6] que originalmente se diseñó para ser un entorno de desarrollo para relatividad numérica pero que en la actualidad tiene propósitos más generales con aplicaciones en biología y economía.

Como se verá en el capítulo 3, para poder hacer una simulación en relatividad numérica es indispensable poder resolver primero las ecuaciones de constricción que son de tipo elíptico y en algunos casos también es necesario resolver ecuaciones elípticas durante la evolución. Por ello es indispensable obtener soluciones de forma eficiente. Como se describe mas adelante, el código CACTUS contiene varias bibliotecas o "espinas"[7] que desempeñan diversas tareas dentro de las simulaciones en relatividad numérica. Una de esas biblioteca es BAM-Elliptic, un código computacional para resolver ecuaciones elípticas que usa el método Multigrid y que es además un generador de datos iniciales para diversos sistemas. BAM-Elliptic fue originalmente desarrollado por Bernd Bruegmann y posteriormente actualizado por Denis Pollney. Es un código escrito en lenguaje C que ha tenido poco desarrollo reciente. Posteriormente se han implementado otras funcionalidades a CACTUS para poder tener una interfase común para implementar resolvedores elípticos conocida como EllBase. Para esta interfase se ha implementado un código llamado EllSOR que implementa

[6] Para más detalles al respecto ver la sección 5.2

[7] En el código CACTUS el desarrollador programa módulos o bibliotecas y el código principal se encarga de la interacción entre las mismas, por esa característica se nombró al código CACTUS y a las bibliotecas o módulos se les conoce como espinas, para más detalles consultar la sección 5.1.

un método de sobre relajación sucesiva (ver sección 4.3), sin embargo, aún no se cuenta con implementaciones más eficientes que usen esa interfase. En un futuro el código Olliptic tendrá una interfase común con EllBase, por el momento tiene una interfase independiente en donde se pueden programar fácilmente rutinas para generar condiciones iniciales y comunicación con otras "espinas" de Cactus para resolver ecuaciones elípticas.

Como ya se mencionó, CACTUS es un código modular en el que se pueden programar "espinas" que desempeñan diversas funciones. Una de las funciones centrales la desempeña el controlador de mallas o "driver" que hace la partición del dominio, dividiéndolo en regiones asignadas a cada procesador (en caso de que se ejecute el programa en varios procesadores) y discretizando las funciones, para poder implementar métodos en diferencias finitas. El controlador que se utiliza por defecto o de manera estándar en Cactus se conoce como PUGH y es un controlador uni-malla, es decir hace particiones homogéneas del dominio. Técnicas más avanzadas en diferencias finitas requieren el uso de controladores multi-mallas, la idea básica es poder hacer particiones inhomogéneas del dominio dotando a ciertas zonas de mayor resolución y como consecuencia obteniendo una mejor aproximación en dichas zonas. En cactus se ha desarrollado recientemente un controlador llamado CARPET [9], que implementa subdominios o "cajas" en las que se definen particiones homogéneas de diferentes tamaños, lo cual permite administrar mejor la memoria y el trabajo de cálculo dotando de mayor resolución a la zonas del dominio global que requieren solución con una mayor precisión. Por el momento no existe un código para resolver ecuaciones elípticas que implemente metodos Multigrid que funcione con CARPET, el código BAM-Elliptic ha quedado en ese sentido obsoleto y modificarlo para que funcione con otro controlador de mallas puede resultar sumamente complicado. Por otro lado, el código Olliptic ha sido diseñado de forma modular y al igual que CARPET esta programado en lenguaje C++, lo cual facilita la implementación directa de muchas características. Como proyecto futuro se plantea la posibilidad de adaptar Olliptic al código CARPET para generar datos iniciales y para resolver ecuaciones elípticas durante la evolución numérica de algún sistema que así lo requiera. Por el momento Olliptic resuelve únicamente operadores elípticos en espacios planos, es necesario también en un futuro implementar en el código la opción de resolver las ecuaciones usando una métrica arbitraria. Sin embargo, esas funcionalidades quedan fuera del alcance de esta tesis.

1.2. Motivación de la tesis

El principal objetivo de esta tesis es dotar al grupo de relatividad numérica del Instituto de Ciencias Nucleares de la UNAM y a la comunidad que utiliza como base CACTUS, de un código capaz de resolver de forma eficiente ecuaciones elípticas dentro del contexto de relatividad numérica. Como se mencionó anteriormente el código desarrollado se llama Olliptic y se ha diseñado para ser fácilmente modificado, para ello se han empleado técnicas modernas de programación. Con la finalidad de que el programa sea al mismo tiempo eficiente y fácil de modificar, se escribió el mismo usando el lenguaje de programación C++ y utilizando programación orientada a objetos, que ha demostrado ser la opción más viable para grandes proyectos científicos.[8] El resultado es un código muy flexible, que podrá convertirse en el remplazo de BAM-Elliptic. Debido a que CACTUS es un código que permite programar bibliotecas en diversos lenguajes de programación, no ha sido

[8] El principal ejemplo es el código ROOT [10] del laboratorio CERN. El laboratorio CERN se ha caracterizado por mantenerse a la vanguardia en el uso de infraestructura computacional, fue ahí en donde se desarrolló el protocolo web. El código ROOT es un proyecto para desarrollar un sistema base, similar a CACTUS, para el análisis de datos experimentales y que utiliza programación orientada a objetos para hacer un sistema modular, fácil de extender y modificar.

diseñado para implementar la programación orientada a objetos de forma natural, es por ello que se ha tenido que hacer una interfase para hacer a Olliptic una biblioteca de CACTUS, los detalles al respecto se encuentran en la sección 5.2.

Para probar el código Olliptic se han planteado dos problemas físicos a resolver. El primero de ellos es la generación de datos iniciales para ondas de Brill, que es un trabajo que ya se ha hecho pero que permite comparar las soluciones obtenidas con las generadas por otros códigos. Dichas ondas son una solución axial simétrica de las ecuaciones de Einstein.

El segundo problema físico planteado es el sistema Schrödinger-Poisson. Dicho sistema es una aproximación de campo débil de las ecuaciones de Einstein para un campo escalar, que se utiliza como modelo de materia obscura en halos galácticos [11, 12]. Se han realizado algunos estudios con simetría esférica [13] y con simetría axial [14, 15]. Sin embargo, debido a que los métodos más simples de solución de ecuaciones elípticas no son eficientes para la evolución de un sistema en 3 dimensiones sin simetrías, se ha requerido utilizar Olliptic para realizar estos experimentos numéricos. En esta tesis se reproducen los resultados obtenidos para una configuración de equilibrio con simetría esférica y se realizan algunas evoluciones para un sistema 3 dimensional sin simetrías. En particular se estudian dos distribuciones de campo escalar que orbitan y colisionan debido a la mutua interacción gravitacional y se analizan en el contexto astrofísico.

Capítulo 2

Relatividad numérica

La Relatividad General es una teoría formulada en el lenguaje matemático de la Geometría Diferencial en la que no se hace una distinción entre el espacio y el tiempo; las tres coordenadas espaciales y el tiempo desempeñan el mismo papel. Sin embargo, existen problemas físicos para los cuales nos gustaría poder evolucionar el sistema en el tiempo, o simplemente poder distinguir entre espacio y tiempo. Como problema modelo de la Relatividad Numérica podemos mencionar la colisión de agujeros negros, que es un sistema dinámico simple, por tratarse de una solución de vacío, pero que requiere una descripción relativista por generar una campo gravitacional intenso. Sin embargo, existen otros motivos por los cuales es necesario distinguir entre coordenadas espaciales y temporales, por ejemplo, para definir una formulación Hamiltoniana que puede ser usada como punto de partida de una teoría de cuantización de la gravedad.

Una de las formas más comunes de hacer la separación del espacio-tiempo en relatividad general se conoce como formulación 3+1 y es la representacion que se usará en esta tesis, sin embargo, podemos mencionar al menos dos más. El *formalismo característico* consiste en separar el espacio-tiempo en conos de luz que emanan de un tubo de mundo tipo tiempo, los detalles se pueden encontrar en [16]. El *formalismo conforme* consiste en separar el espacio-tiempo en hipersuperficies tipo espacio que intersectan el infinito nulo haciendo una transformación conforme para traer las fronteras a una región finita, para más detalles consultar [17].

En este capítulo se presenta de forma breve la formulación 3+1 de la relatividad general y un análisis de las ecuaciones resultantes distinguiendo entre ecuaciones de constricción y ecuaciones de evolución.

Los primeros estudios de esta formulación fueron realizados por Arnowitt, Deser y Misner [3], las ecuaciones originalmente derivadas fueron reescritas de manera no trivial por York [18] y fue en esa forma que se utilizaron originalmente en relatividad numérica. Trabajos posteriores han derivado en reformulaciones que aseguran que el problema de valores iniciales es “bien planteado”. Una discusión detallada de la formulación 3+1 y de relatividad numérica se puede encontrar en [19], en este capítulo únicamente se dará un panorama general.

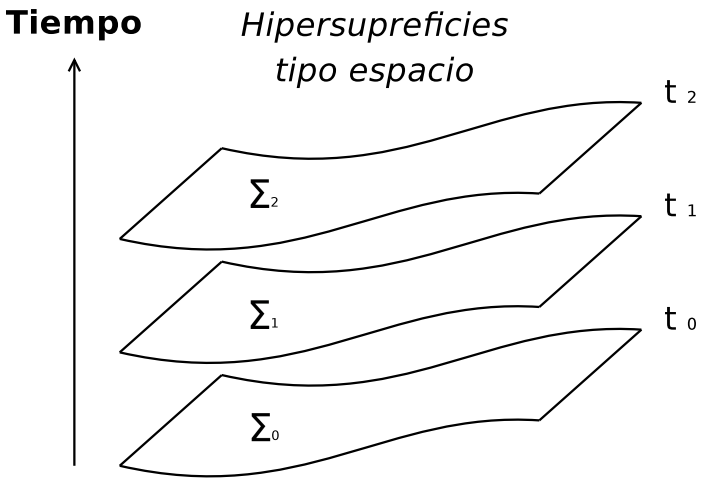

Figura 2.1: Foliación del espacio. En el esquema se representa la descomposición del espacio (conocida como foliación del espacio-tiempo), en hipersuperficies 3 dimensionales tipo espacio parametrizadas por una coordenada temporal.

2.1. Formulación 3+1

La formulación 3+1 de la Relatividad General se basa en la descomposición de las ecuaciones de Einstein:[1]

$$G_{ab} = 8\pi T_{ab}, \tag{2.1}$$

en ecuaciones que describen la física sobre hipersuperficies de tiempo contante Σ (rebanadas de la variedad para t constante). En la figura 2.1 se muestra un esquema de la descomposición. Las hipersuperficies están parametrizadas por t de modo que se puede definir un vector normal unitario ($n^a n_a = -1$) a la hipersuperficie:

$$n^a := -\alpha \nabla^a t, \tag{2.2}$$

en donde α se conoce como la función de lapso y determina el ritmo en el que el parámetro temporal avanza; adicionalmente podemos definir un vector tiempo:

$$t^a := \alpha n^a + \beta^a, \tag{2.3}$$

en donde β^a se conoce como vector de corrimiento y cumple:

$$\beta^a n_a \equiv 0, \tag{2.4}$$

por lo tanto β^a es un vector que solo tiene componentes sobre Σ. Gracias a la libertad de norma podemos escoger un sistema de coordenadas especialmente adaptado a las descomposición de la variedad en hipersuperficies. Sean x^i, con $i \in \{1, 2, 3\}$ las componentes espaciales, y t la cuarta coordenada que parametriza la rebanada, si γ_{ij} denota al tensor métrico sobre Σ, entonces el elemento de linea en ese sistema coordenado toma la forma:

$$\mathrm{d}s^2 = -\alpha^2 \mathrm{d}t^2 + \gamma_{ij}(\mathrm{d}x^i + \beta^i \mathrm{d}t) \otimes (\mathrm{d}x^j + \beta^j \mathrm{d}t). \tag{2.5}$$

[1]Notación: en adelante se usan como índices que corren de 0 a 3 las primeras letras del alfabeto hasta la h y de la i en adelante para los índices espaciales que van de 1 a 3

El elemento de linea escrito de esta forma tiene una interpretación geométrica que se representa en la figura 2.2, se presentan en el diagrama dos hipersuperficies separadas por un tiempo dt y se muestra para un punto de la hipersuperficie dos posibles trayectorias. La primera es normal a la hipersuperficie y la segunda con un desplazamiento dado por el vector β^i.

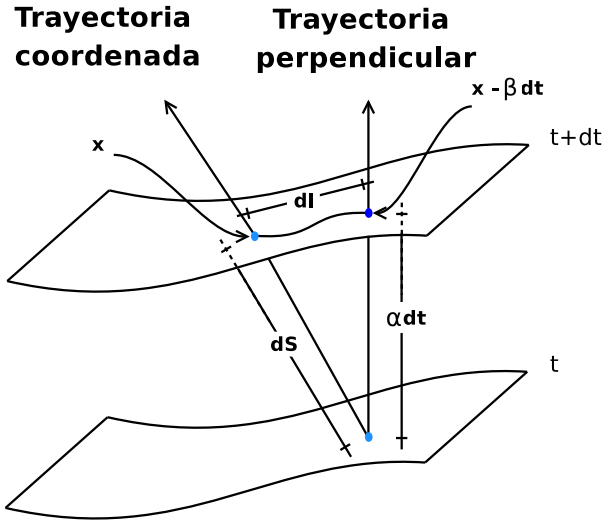

Figura 2.2: Representación del elemento de linea ds. El vector β^i determina el desplazamiento del sistema de coordenadas respecto a la trayectoria perpendicular a la hipersuperficie, α el ritmo al que se recorre el parámetro t y γ^{ij} es el tensor métrico con el que se calculan las distancias sobre Σ. En la figura $dl = \gamma_{ij}(\mathrm{d}x^i + \beta^i \mathrm{d}t) \otimes (\mathrm{d}x^j + \beta^j \mathrm{d}t)$

Adicionalmente se puede definir un tensor que caracteriza la curvatura de Σ que normalmente se denomina *tensor de curvatura extrínseca*:

$$K_{ij} := -\frac{1}{2}\mathcal{L}_{\vec{n}}\gamma_{ij}, \tag{2.6}$$

en donde $\mathcal{L}_{\vec{n}}$ denota la derivada de Lie a lo largo de la curva cuyo vector tangente es n^a, K_{ij} es un tensor que representa la variación del vector normal a la hipersuperficie dado un transporte paralelo del mismo. En términos de los elementos mencionados anteriormente es posible encontrar la forma de las ecuaciones de Einstein, para ello se utiliza el operador de proyección:

$$P^a_{\ b} = \delta^a_{\ b} - n^a n_b. \tag{2.7}$$

Con este operador y el vector normal n^a es posible hacer tres tipos de proyecciones independientes. La proyección sobre la hipersuperficie:

$$P[G_{ab} - 8\pi T_{ab}] = 0, \tag{2.8}$$

en donde $P := P^c_{\ a} P^d_{\ b}$, nos proporciona 6 ecuaciones, que de forma explicita son:

$$\begin{aligned} \partial_t K_{ij} = \alpha[R^{(3)}_{ij} - 2K_{il}K^l_{\ j} + KK_{ij} - 8\pi S_{ij} + 4\pi\gamma_{ij}(S-\rho)] \\ - \nabla^{(3)}_i \nabla^{(3)}_j \alpha + \beta^l \nabla^{(3)}_l K_{ij} + K_{il}\nabla^{(3)}_j \beta^l + K_{jl}\nabla^{(3)}_i \beta^l, \end{aligned} \tag{2.9}$$

en donde $\nabla_i^{(3)}$ es el operador de derivada covariante asociado a la métrica espacial, $R_{ij}^{(3)}$ es el tensor de Ricci asociado a la métrica espacial, $K := K^i_{\ i}$ es la traza de la curvatura extrínseca, $\rho := n^a n^b T_{ab}$ es la densidad de materia, S_{ij} es el tensor de esfuerzo de materia[2] y $S := S^i_{\ i}$ es su traza.

La proyección en la dirección normal a Σ es la segunda proyección posible:

$$n^a n^b [G_{ab} - 8\pi T_{ab}] = 0, \tag{2.10}$$

de ella obtenemos 1 ecuación, que tiene las siguiente forma:

$$R^{(3)} + K^2 - K_{ij}K^{ij} = 16\pi\rho. \tag{2.11}$$

Esta ecuación mantiene la estructura de las ecuaciones de Einstein, del lado izquierdo permanecen los elementos geométricos (en este caso los escalares de la curvatura y de la curvatura extrínseca) y de forma separada en el lado derecho de la ecuación la componente de materia dada por la densidad de energía.

Por último haciendo una proyección mixta, primero normal y luego paralela a Σ se obtienen las 3 ecuaciones restantes:

$$P[n^a(G_{ab} - 8\pi T_{ab})] = 0. \tag{2.12}$$

La forma explícita de estas ecuaciones es la siguiente:

$$\nabla_j^{(3)}[K^{ij} - \gamma^{ij}K] = 8\pi j^i, \tag{2.13}$$

en donde $j^i := -P^{ia}n^b T_{ab}$ es el flujo de momento medido por los observadores que se desplazan de forma perpendicular a Σ. Nuevamente se presenta la estructura de las ecuaciones de Einstein en donde no se mezclan las componentes geométricas con las términos de materia.

2.2. Ecuaciones dinámicas

En un sistema mecánico clásico, la trayectoria de una partícula de masa m sujeta a una fuerza $\vec{F}$ esta descrita por el sistema:

$$m\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = \vec{F}, \tag{2.14}$$

$$\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = \vec{v}. \tag{2.15}$$

La primer ecuación es la segunda ley de Newton y determina la dinámica del sistema. La segunda es la definición de velocidad y determina la cinemática. En el caso de la descomposición que se ha hecho de las ecuaciones de Einstein se observa que existen ecuaciones que juegan un papel análogo al caso mecánico. De la definición de curvatura extrínseca (2.6), se observa que K^{ij} es la "velocidad" de la métrica espacial, esta definición es una ecuación cinemática. Por otro lado, observamos que solo la proyección sobre Σ genera ecuaciones que contienen derivadas respecto del tiempo. La ecuación (2.9) determina la dinámica del sistema y juega el papel de la ley de Newton en el caso del ejemplo mecánico.

[2]El tensor de esfuerzos y la densidad de materia, corresponde a las magnitudes que miden los observadores de Euler, es decir, los que se desplazan de forma normal a la hipersuperficie.

La mecánica clásica también nos proporciona otras analogías. Para poder integrar el sistema es necesario proporcionar como dato inicial, la posición y velocidad. En el caso relativista debemos proporcionar la forma de la métrica espacial y de la curvatura extrínseca como dato inicial, es decir debemos proporcionar el conjunto de valores: $\{\gamma_{ij}, K_{ij}\}$. Como se verá en la siguiente sección, no somos completamente libres de dar esos dos conjuntos de parametros[3] si queremos realmente resolver las ecuaciones de Einstein, las soluciones a las ecuaciones del campo gravitacional están constreñidas en un subespacio.

Por último hay que mencionar que en la práctica no se utilizan las ecuaciones (2.6) y (2.9) para evolucionar sistemas numéricamente, esto debido a que primero se observó "experimentalmente" que las ecuaciones no conducían a sistemas estables y posteriormente se determinó que las ecuaciones así escritas son débilmente hiperbólicas, lo cual hace que el sistema no este "bien puesto" matemáticamente[4]. Sin embargo, se encontró una forma de hacer el sistema de evolución fuertemente hiperbólico[5], lo cual permitió hacer evoluciones más estables, a esa reformulación de las ecuaciones se le conoce como sistema BSSN(OK)[6]. En esta tesis no se pretende analizar en detalle las ecuaciones de evolución por lo que simplemente se mostrarán las relaciones finales de la formulación BSSN(OK). Los detalles se pueden consultar en [22].

El sistema BSSN(OK) consta de las variables: K, $\tilde{A}_{ij}$, ϕ, $\tilde{\gamma}_{ij}$ y $\tilde{\Gamma}^i$, que en términos de las variables ADM están definidas por:

$$\begin{aligned}
\tilde{A}_{i,j} &:= \mathrm{e}^{-4\phi} A_{ij}, && (2.16)\\
A_{ij} &:= K_{ij} - \frac{1}{3}\gamma_{ij} K, && (2.17)\\
\phi &:= \frac{1}{12}\ln(\det \gamma_{ij}), && (2.18)\\
\tilde{\gamma}_{ij} &:= \mathrm{e}^{-4\phi}\gamma_{ij}, && (2.19)\\
\tilde{\Gamma}^i &:= \tilde{\gamma}^{jk}\tilde{\Gamma}^i{}_{jk} = -\partial_j \tilde{\gamma}^{ij}, && (2.20)
\end{aligned}$$

en donde en la última expresión $\tilde{\Gamma}^i{}_{jk}$ se refiere a los coeficientes de conexión asociados con $\tilde{\gamma}^{ij}$.

Las ecuaciones de evolución para las variables BSSN(OK) son:

$$\begin{aligned}
\frac{\mathrm{d}K}{\mathrm{d}t} &= -\gamma^{ij}\nabla^{(3)}_i\nabla^{(3)}_j\alpha + \alpha\left(\tilde{A}_{ij}\tilde{A}^{ij} + \frac{1}{3}K^2\right) + 4\pi\alpha(\rho + S),\\
\frac{\mathrm{d}\tilde{A}_{ij}}{\mathrm{d}t} &= \mathrm{e}^{-4\phi}\left\{-\nabla^{(3)}_i\nabla^{(3)}_j\alpha + \alpha R^{(3)}_{ij} + 4\pi\alpha\left[\gamma_{ij}(S-\rho) - 2S_{ij}\right]\right\}^{\mathrm{TF}}\\
&\quad +\alpha(K\tilde{A}_{ij} - 2\tilde{A}_{ik}\tilde{A}^k{}_j),\\
\frac{\mathrm{d}\phi}{\mathrm{d}t} &= -\frac{1}{6}\alpha K,\\
\frac{\mathrm{d}\tilde{\gamma}_{ij}}{\mathrm{d}t} &= -2\alpha\tilde{A}_{ij},\\
\frac{\mathrm{d}\tilde{\Gamma}^i}{\mathrm{d}t} &= \tilde{\gamma}^{ij}\partial_j\partial_k\beta^i + \frac{1}{3}\tilde{\gamma}^{ij}\partial_j\partial_k\beta^k - 2\left(\alpha\partial_j\tilde{A}^{ij} + \tilde{A}^{ij}\partial_j\alpha\right),
\end{aligned}$$

[3] En total se tienen 12 cantidades linealmente independientes, 6 componentes de γ_{ij} y 6 de K_{ij}

[4] En el capítulo 4 se discute un poco más sobre las condiciones para que un problema este "bien puesto"

[5] Lo cual implica que al reescribir las ecuaciones de segundo orden en términos de un sistema de ecuaciones de primer orden se obtengan valores propios reales y un conjunto completo de vectores propios para la matriz asociada al sistema de ecuaciones.

[6] Por las siglas de los que llegaron inicialmente a esas ecuaciones, de forma cronológica según los artículos: Nakamura, Oohara y Kojima [20]; posteriormente Shibata y Nakamura [21]; y finalmente Baumgarte y Shapiro [22]

en donde $\frac{\mathrm{d}}{\mathrm{d}t} := \partial_t - \mathcal{L}_{\vec{\beta}}$ y TF denota la parte libre de traza de la expresión, por ejemplo para el tensor de Ricci $R_{ij}^{\mathrm{TF}} := R_{ij} - \frac{1}{3}\gamma_{ij}R$. Como se observa, solo las dos primeras expresiones tienen componentes físicas de materia, esas dos relaciones determinan la dinámica, el resto son relaciones cinemáticas. Con estas ecuaciones es posible evolucionar un sistema garantizando que el problema es "bien puesto" desde el punto de vista matemático. Por último hay que notar que la métrica $\tilde{\gamma}_{ij}$ y ϕ juegan el papel de "coordenada" en la analogía con el sistema mecánico y la curvatura extrínseca $\tilde{A}_{ij}$ y la traza K juega el papel de la "velocidad".

2.3. Ecuaciones de constricción

Regresando a los sistemas mecánicos, tomemos ahora como problema modelo el péndulo simple. La función Lagrangiana que describe el problema es:

$$L = \frac{1}{2}mL^2\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2 + mgL\cos\theta, \tag{2.21}$$

con la cual se obtiene la ecuación de movimiento:

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \frac{g}{L}\sin\theta = 0, \tag{2.22}$$

la solución de esta ecuación no tiene una forma cerrada en términos de funciones analíticas, sin embargo, es claro que la solución es una función solo de θ. Si pensamos que el sistema se encuentra inmerso en un espació tres dimensional, es decir, es un péndulo de laboratorio, existen dos coordenadas que están constreñidas: $\phi = 0$ y $r = L$. Si no se aplican fuerzas en la dirección ϕ, entonces:

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = 0, \tag{2.23}$$

y si el péndulo esta sujeto por una barra rígida:

$$\frac{\mathrm{d}r}{\mathrm{d}t} = 0. \tag{2.24}$$

El espacio fase para este sistema tiene 6 dimensiones $(\theta, \phi, r, \dot{\theta}, \dot{\phi}, \dot{r})$, sin embargo las constricciones restringen las trayectorias solución a un espacio bidimensional. Analizando la ecuación en el espacio fase como un sistema dinámico es posible determinar las trayectorias en el subespacio fase $(\theta, \dot{\theta})$, se obtienen soluciones periódicas y por la periodicidad de θ es posible determinar que el espacio en el que se encuentra la solución es un cilindro (ver figura 2.3).

Separando la ecuación de segundo orden (2.22) en un sistema de dos ecuaciones ordinarias:

$$\frac{\mathrm{d}\theta}{\mathrm{d}\tau} = \Omega, \tag{2.25}$$

$$\frac{\mathrm{d}\Omega}{\mathrm{d}\tau} = -\sin\theta, \tag{2.26}$$

en donde $\tau := \sqrt{\frac{g}{L}}t$ y $\Omega := \frac{\mathrm{d}\theta}{\mathrm{d}\tau}$, se pueden obtener las trayectorias en el espacio fase integrando numéricamente. La forma más adecuada de visualizar las trayectorias en un plano es hacer un mapeo del cilindro con la transformación:

$$\xi = \mathrm{e}^{-\frac{\Omega}{2}}\cos\theta, \tag{2.27}$$

$$\eta = \mathrm{e}^{-\frac{\Omega}{2}}\sin\theta, \tag{2.28}$$

esta transformación lleva la tapa superior (que se encuentra en $\Omega \to \infty$) al origen y la tapa inferior (que se encuentra en $\Omega \to -\infty$) a una circunferencia de radio $r \to \infty$. La gráfica de las trayectorias del sistema 2.25 se muestran en la figura 2.3. Con las transformación dada se muestra que las trayectorias de la solución en el espacio fase cubren completamente un espacio $\mathbb{R}^2$ y toda solución de las ecuaciones están contenidas en ese espacio, algo similar ocurre en relatividad general con las ecuaciones de Einstein.

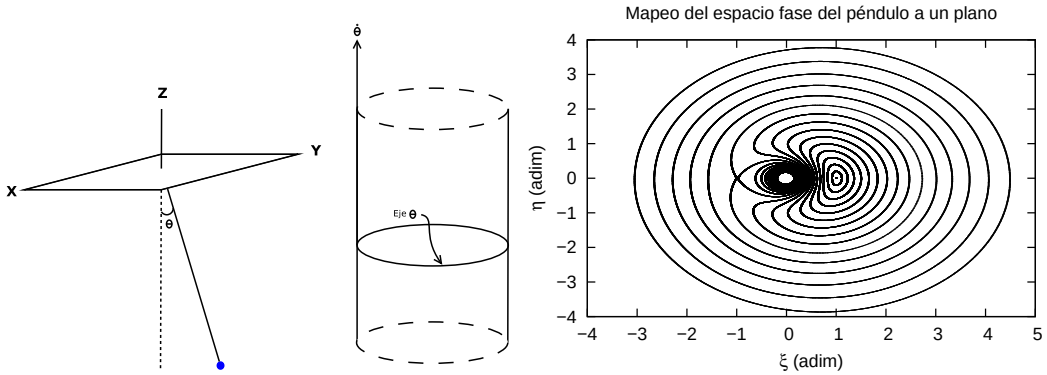

Figura 2.3: Péndulo simple. Se muestra un diagrama del sistema físico (izquierda), el subespacio fase de soluciones $(\theta, \dot{\theta})$, el cual es un cilindro inmerso en $\mathbb{R}^6$ (centro) y una gráfica del espacio fase mapeado del cilindro a un plano $\mathbb{R}^2$ (derecha). Este sistema es un ejemplo mecánico de sistema con constricciones.

Las constricciones en mecánica dependen del problema físico, sin embargo, con las ecuaciones de Einstein escritas usando el formalismo 3+1 tenemos un sistema con constricciones que se aplica a todo problema físico. Las ecuaciones:

$$R^{(3)} + K^2 - K_{ij}K^{ij} = 16\pi\rho, \tag{2.29}$$

$$\nabla_j^{(3)}[K^{ij} - \gamma^{ij}K] = 8\pi j^i, \tag{2.30}$$

constituyen un sistema de ecuaciones de constricción, ya que son ecuaciones que no contienen derivadas temporales y no depende de la elección de norma (no contienen términos con α o β^i). La ecuación (2.29) se conoce como *constricción Hamiltoniana* y las ecuaciones (2.30) se conocen como *constricciones de momento*. Estas 4 ecuaciones son una restricción en la elección arbitraria de datos iniciales para las 12 cantidades a evolucionar $\{\gamma_{ij}, K_{ij}\}$. Sin embargo, la distinción entre magnitudes libres y constreñidas no es clara, en el capítulo 3 se estudiarán algunos procedimientos para generar datos iniciales y hacer elecciones adecuadas de las variables a resolver. Las ecuaciones de constricción, restringen al igual que en el caso del péndulo, el espacio de soluciones a un subespacio. Las soluciones en ese subespacio, son soluciones físicas. Adicionalmente cada problema físico puede imponer constricciones adicionales, como por ejemplo, en las simetrías del sistema.

En el caso de la relatividad general, las identidades de Bianchi:

$$\nabla_{[a}R_{bc]de} = 0, \tag{2.31}$$

garantizan que una solución que cumple las constricciones al tiempo $t = 0$, seguirá cumpliendolas al tiempo $t > 0$. Sin embargo, eso solo es válido para soluciones exactas, al resolver numéricamente

las constricciones, y durante la evolución se generan pequeños errores. Si el sistema es bien puesto la solución numérica no se alejará demasiado del subespacio de soluciones físicas. Adicionalmente, es posible realizar correcciones a la solución numérica, resolviendo las constricciones a cada paso de tiempo, lo cual mantiene a la solución numérica cerca del subespacio de soluciones físicas. Las ecuaciones de constricción constituyen un sistema de ecuaciones diferenciales parciales de tipo elíptico. El propósito de esta tesis ha sido el estudio de este tipo de ecuaciones y el desarrollo del código Olliptic, el cual tiene como finalidad resolver numéricamente ecuaciones elípticas con aplicación a Relatividad Numérica, en particular ecuaciones de constricción para generación de datos iniciales y durante la evolución. Un código que cumple con estas características debe ser eficiente, principalmente en la etapa de evolución debido a que, en general, resolver ecuaciones elípticas requiere una gran cantidad de tiempo de computo, en comparación con una ecuación hiperbólica.

Capítulo 3

Datos Iniciales en relatividad numérica

En el contexto de la Relatividad General las ecuaciones de Einstein conforman un sistema de 10 ecuaciones diferenciales parciales, fuertemente acopladas con 10 variables métricas independientes g_{ab}, por lo que se tiene el mismo número de ecuaciones e incógnitas. Como sabemos las ecuaciones de Einstein en la descomposición 3+1 están representadas por 6 ecuaciones de evolución y 4 ecuaciones de constricción. Por las identidades de Bianchi sabemos que

$$\nabla_a G^{ab} = 0, \tag{3.1}$$

lo cual garantiza que al cumplirse las ecuaciones de constricción inicialmente, durante la evolución seguirán cumpliéndose.

Se entiende por **Problema de Cauchy** a un problema físico o matemático planteado usando un sistema de ecuaciones diferenciales ordinarias o parciales de orden m, y en donde se dan los valores de la solución y sus $m-1$ derivadas en una superficie dada.

El problema de Cauchy en relatividad General consiste en dar una solución a las ecuaciones de Einstein en una hipersuperficie espacial (a un tiempo inicial t_0), tal que cumple las ecuaciones de constricción y a partir de esa condición inicial generar la evolución del sistema usando las ecuaciones dinámicas.

En las siguientes secciones se da un resumen de los métodos para generar datos iniciales en relatividad numérica. Una discusión más detallada se puede encontrar en [23, 19].

3.1. Descomposición conforme

La forma más común para obtener una transformación del sistema de constricciones que permite generar datos iniciales es haciendo una descomposición conforme. La primer descomposición que se presenta se conoce como *descomposición de York-Lichnerowicz*. La idea básica es, suponer que la métrica espacial se puede expresar como un factor que multiplica a una métrica auxiliar (conocida como métrica conforme o de fondo):

$$\gamma_{ij} = \psi^4 \bar{\gamma}_{ij}, \tag{3.2}$$

con (3.2) la constricción Hamiltoniana (2.11) se reescribe en la forma:

$$\bar{\nabla}^2\psi - \frac{1}{8}\bar{R}\psi + \frac{1}{8}(K_{ij}K^{ij} - K^2 + 16\pi\rho)\psi^5 = 0, \tag{3.3}$$

en donde $\bar{\nabla}^2$ y $\bar{R}$ denotan respectivamente el Laplaciano covariante[1] y el escalar de Ricci asociados a la métrica $\bar{\gamma}_{ij}$. La curvatura extrínseca se separa también en su traza y en la parte libre de traza:

$$A^{ij} = K^{ij} - \frac{1}{3}\gamma^{ij}K, \tag{3.4}$$

entonces, sustituyendo esta expresión en (3.3) obtenemos:

$$\bar{\nabla}^2\psi - \frac{1}{8}\bar{R}\psi + \frac{1}{8}\left(A_{ij}A^{ij} - \frac{2}{3}K^2 + 16\pi\rho\right)\psi^5 = 0. \tag{3.5}$$

La ecuación (3.5) es una ecuación elíptica cuasi-lineal (ver capítulo 4) para el factor ψ. La constricción de momento (2.13) en términos de A^{ij} tiene la forma:

$$\nabla_j A^{ij} - \frac{2}{3}\nabla^i K - 8\pi j^i = 0. \tag{3.6}$$

Con a finalidad obtener un sistema de tres ecuaciones con tres incógnitas para la constricción de momento, se procede a utilizar un resultado general de álgebra lineal. Cualquier tensor simétrico y con traza cero S^{ij} puede expresarse de la siguiente manera:

$$S^{ij} = T^{ij} + (\mathbb{L}X)^{ij}, \tag{3.7}$$

en donde T^{ij} es un tensor simétrico, con traza cero y transverso (es decir, con divergencia cero $\nabla_i T^{ij} = 0$), X^i es un vector y $\mathbb{L}$ es el operador:

$$(\mathbb{L}X)^{ij} := \nabla^i X^j + \nabla^j X^i - \frac{2}{3}\gamma^{ij}\nabla_k X^k. \tag{3.8}$$

Hay que destacar que es posible usar cualquier métrica para definir al operador $\mathbb{L}$, se estudiarán dos posibilidades: Utilizando la métrica conforme o utilizando la métrica física.

Como se mencionó anteriormente no es la intención de esta tesis estudiar en detalle la generación de datos iniciales, a pesar de que será esta una de las aplicaciones más importantes del código, los métodos de solución de ecuaciones elípticas se aplicarán en general a varios casos en relatividad numérica. Es necesario también, presentar estos resultados debido a que serán utilizados más adelante, en los casos físicos en los que se aplicó el código.

Descomposición transversa conforme

Si se define el operador $\mathbb{L}$ usando la métrica conforme se obtiene el siguiente sistema de ecuaciones acopladas:

$$\bar{\nabla}^2\psi - \frac{1}{8}\bar{R}\psi + \frac{1}{4}\left(8\pi\rho - \frac{1}{3}K^2\right)\psi^5 + \frac{1}{8}\bar{A}_{ij}\bar{A}^{ij}\psi^{-7} = 0, \tag{3.9}$$

$$\bar{\nabla}^2 V^i + \frac{1}{3}\bar{\nabla}^i\bar{\nabla}_j V^j + \bar{R}^i{}_j V^j = \sigma^i, \tag{3.10}$$

[1] En lo que sigue, siempre que no cause confunción, se omitirá el superíndicie (3) en expresiones como $\nabla_i^{(3)}$ para denotar magnitudes sobre la hipersuperficie espacial.

en donde σ^i es un término de fuente que se presentará en seguida y $\bar{A}^{ij}$ se ha construido utilizando (3.7), explicitamente estas cantidades están dadas por:

$$\bar{A}^{ij} = (\bar{\mathbb{L}}\bar{V})^{ij} + \bar{M}^{ij}, \tag{3.11}$$

$$\sigma^i = \frac{2}{3}\psi^6\bar{\nabla}^i K - \bar{\nabla}_j\bar{M}^{ij} + 8\pi j^i\psi^{10}. \tag{3.12}$$

Las ecuaciones (3.9) y (3.10) tienen como incógnitas ψ y $\bar{V}^i$ y los datos que somos libres de proporcionar son: la métrica de fondo $\bar{\gamma}^{ij}$, un tensor simétrico y con traza cero $\bar{M}^{ij}$, la traza de la curvatura extrínseca K y los términos de materia ρ y j^i. Las variables físicas se recuperan usando:

$$\gamma_{ij} = \psi^4\bar{\gamma}_{ij}, \tag{3.13}$$

$$K^{ij} = \psi^{-10}\bar{A}^{ij} + \frac{1}{3}\psi^{-4}\bar{\gamma}^{ij}K. \tag{3.14}$$

Normalmente es posible simplificar el problema seleccionando K constante, es decir, tomando una curvatura espacial promedio constante en la hipersuperficie. Si no existen densidades de momento, la constricción Hamiltoniana (3.9) se desacopla de las constricciones de momento (3.10); en ese caso podemos proceder a obtener V^i de la constricción de momento (3.10), para después construir el tensor $\bar{A}^{ij}$ usando (3.11) y por último resolver la constricción Hamiltoniana (3.9).

Descomposición transversa física

Si se utiliza la métrica física para definir al operador $\mathbb{L}$ se llega a un sistema similar al anterior:

$$\bar{\nabla}^2\psi - \frac{1}{8}\bar{R}\psi + \frac{1}{8}\left(A_{ij}A^{ij} + 16\pi\rho - \frac{2}{3}K^2\right)\psi^5 = 0, \tag{3.15}$$

$$\bar{\nabla}^2\bar{V}^i + \frac{1}{3}\bar{\nabla}^i\bar{\nabla}_j\bar{V}^j + \bar{R}^i{}_j\bar{V}^j + 6(\partial_j\ln\psi)(\bar{\mathbb{L}}\bar{V})^{ij} = \sigma^i, \tag{3.16}$$

en donde en este caso

$$A^{ij} = \psi^{-4}(\bar{\mathbb{L}}\bar{V})^{ij} + \psi^{-10}\bar{M}^{ij}, \tag{3.17}$$

$$\sigma^i = \frac{2}{3}\bar{\nabla}^i K - \psi^{-6}\bar{\nabla}_j\bar{M}^{ij} + 8\pi j^i\psi^4, \tag{3.18}$$

las variables a resolver son ψ y $\bar{V}^i$. Los parámetros que debemos proporcionar son los mismos que en el caso anterior: $\bar{\gamma}^{ij}$, $\bar{M}^{ij}$, K y los términos de materia ρ y j^i. Se reconstruyen las variables físicas con:

$$\gamma_{ij} = \psi^4\bar{\gamma}_{ij}, \tag{3.19}$$

$$K^{ij} = A^{ij} + \frac{1}{3}\gamma^{ij}K. \tag{3.20}$$

Descomposición transversa ponderada

Además de las descomposiciones usando la métrica conforme y la métrica física es posible hacer una descomposición un poco más general tomando:

$$A^{ij} = A^{ij}_* + \frac{1}{\eta}(\mathbb{L}W)^{ij}, \tag{3.21}$$

las ecuaciones que se obtienen en este caso son:

$$\bar{\nabla}^2\psi - \frac{1}{8}\bar{R}\psi + \frac{1}{4}\left(8\pi\rho - \frac{1}{3}K^2\right)\psi^5 + \frac{1}{8}\bar{A}_{ij}\bar{A}^{ij}\psi^{-7} = 0, \tag{3.22}$$

$$\bar{\nabla}_j\left[\frac{1}{\eta}(\bar{\mathbb{L}}\bar{V})^{ij}\right] = \sigma^i, \tag{3.23}$$

en donde:

$$\bar{A}^{ij} = \frac{1}{\eta}(\bar{\mathbb{L}}\bar{V})^{ij} + \bar{M}^{ij}, \tag{3.24}$$

$$\sigma^i = \frac{2}{3}\psi^6\bar{\nabla}^i K - \bar{\nabla}_j\bar{M}^{ij} + 8\pi j^i\psi^{10}, \tag{3.25}$$

$$\bar{\eta} = \psi^{-6}\eta. \tag{3.26}$$

Los datos que debemos proporcionar en este caso son: la métrica de fondo $\bar{\gamma}_{ij}$, el tensor simétrico y de traza cero $\bar{M}^{ij}$, la traza de la curvatura extrínseca K, las densidades de energía y momento ρ y j^i, además de la función de peso η.

3.2. Descomposición del "sándwich delgado"

Para complementar las discusión sobre los métodos para generar datos iniciales se mencionará por último otra forma de descomposición de las ecuaciones de constricción, introducida inicialmente por York y que se conoce como descomposición del "sándwich delgado" [24]. La idea básica es dar valores para la métrica conforme en dos hipersuperficies cercanas (de ahí el nombre sándwich delgado) o de forma equivalente se puede dar la métrica conforme y su derivada temporal en una hipersuperficie.

Como en el resto de las descomposiciones se dan únicamente el sistema de ecuaciones resultante de estas formulación. Las ecuaciones son:

$$\bar{\nabla}^2\psi - \frac{1}{8}\bar{R}\psi + \frac{1}{4}\left(8\pi\rho - \frac{1}{3}K^2\right)\psi^5 + \bar{A}_{ij}\bar{A}^{ij}\psi^{-7} = 0, \tag{3.27}$$

$$\bar{\nabla}_j\left[\frac{1}{2\bar{\alpha}}(\bar{\mathbb{L}}\beta)^{ij}\right] = \sigma^i, \tag{3.28}$$

en donde:

$$\sigma^i = \bar{\nabla}_j\left[\frac{1}{2\bar{\alpha}}\bar{u}^{ij}\right] + \frac{2}{3}\bar{\nabla}^i K\psi^6 + 8\pi j^i\psi^{10}, \tag{3.29}$$

$$\bar{A}^{ij} = \frac{1}{2\bar{\alpha}}\Big[(\bar{\mathbb{L}}\beta)^{ij} - \bar{u}^{ij}\Big], \tag{3.30}$$

en este caso las variables son el factor conforme ψ y el vector de corrimiento β^i. Los parámetro libres son la métrica conforme $\bar{\gamma}_{ij}$, su derivada temporal $\bar{u}^{ij}$, la traza de la curvatura extrínseca K, la función de lapso densitizada $\bar{\alpha}$ y los términos de materia ρ y j^i. Las cantidades físicas se reconstruyen a partir de:

$$\gamma_{ij} = \psi^4\bar{\gamma}_{ij}, \tag{3.31}$$

$$K^{ij} = \psi^{-10}\bar{A}^{ij} + \frac{1}{3}\gamma^{ij}K, \tag{3.32}$$

$$\alpha = \gamma^{-\frac{1}{2}}\bar{\alpha}. \tag{3.33}$$

Hasta aquí se han presentado el tipo de ecuaciones a resolver para generar datos iniciales en las diversas descomposiciones y que son **validas para cualquier sistema físico**. Como se observa la constricción Hamiltoniana en todas las formulaciones tiene como forma general:

$$\nabla^2\psi + A\psi + B\psi^5 + \frac{C}{\psi^7} = S, \tag{3.34}$$

en donde A, B, C y S, son funciones dadas según la formulación (en los casos presentados $S = 0$, sin embargo se considera una forma más general de la ecuación). Si además, se escoge como métrica de fondo una métrica plana se obtiene que el operador Laplaciano es plano. Como se verá en el capítulo 4, (3.34) es una ecuación elíptica semi-lineal. Una de las aplicaciones de los métodos estudiados en esta tesis y del código Olliptic desarrollado para la tesis, es poder resolver la ecuación de constricción Hamiltoniana con la forma general dada en (3.34).

3.3. Aplicaciones

Como ejemplos de datos iniciales en relatividad numérica se presentarán dos casos en los cuales es necesario resolver la constricción Hamiltoniana y que requiere el uso de un resolvedor elíptico.

Ondas de Brill

El primer ejemplo, son las ondas de Brill [25]. Las ondas de Brill se construyen proponiendo una métrica axial simétrica con elemento de linea asociado:

$$\mathrm{d}s^2 = \psi^4\left[\mathrm{e}^{2q}(\mathrm{d}\rho^2 + \mathrm{d}z^2) + \rho^2\mathrm{d}\phi^2\right], \tag{3.35}$$

en donde q y ψ son funciones de ρ y z. Si se impone la condición de simetría temporal, la constricción de momento se satisface idénticamente. La constricción Hamiltoniana con la métrica de fondo propuesta toma la forma:

$$\bar{\nabla}^2\psi + \frac{1}{4}\Big(\partial_\rho^2 q + \partial_z^2 q\Big)\psi = 0, \tag{3.36}$$

en donde el Laplaciano en (3.36) es el operador usual en un espacio plano. La función q es en principio arbitraria, salvo por las condiciones de frontera que debe cumplir:

$$q\big|_{\rho=0} = 0, \tag{3.37}$$

$$\partial_\rho^n q\big|_{\rho=0} = 0, \tag{3.38}$$

$$q\big|_{r\to\infty} = \mathcal{O}(r^{-2}), \tag{3.39}$$

en donde n es un entero impar.

Datos iniciales para colisión de agujeros negros

Existen varios métodos para generar datos iniciales para colisión de agujeros negros, sin embargo se tomará solo un sistema que conduce a una ecuación elíptica semi-lineal y que genera datos para agujeros con momento lineal y angular. El sistema al que nos referimos fue desarrollado por Brandt y Brügmann [26]. Se escogen los siguientes valores para los parámetros libres:

$$K = 0, \tag{3.40}$$

$$\bar{\gamma}_{ij} = f_{ij}, \tag{3.41}$$

$$\psi|_{r\to\infty} = 1, \tag{3.42}$$

en donde f_{ij} representa una métrica plana en algún sistema de coordenadas. De esta forma los operadores involucrados son planos. En la descomposición transversa conforme, al escoger $K = 0$ y un sistema en vacío las constricciones de momento se desacoplan de la constricción Hamiltoniana. La constricción de momento bajo estas suposiciones toma la siguiente forma:

$$\bar{\nabla}^2 V^i + \frac{1}{3}\bar{\nabla}^i \bar{\nabla}_j V^j = 0, \tag{3.43}$$

una solución analítica a estas ecuaciones fue obtenida por York y Bowen [27] y es la siguiente:

$$V^i = -\frac{1}{4r}\Big[7P^i + n^i n_j P^j\Big] + \frac{1}{r^2}\epsilon^{ijk} n_j S_k, \tag{3.44}$$

en donde se puede mostrar que P^i es un parámetro vectorial que representa el momento lineal de un agujero negro sobre la hipersuperficie y S^i es su momento angular sobre la hipersuperficie. Así mismo, r es la coordenada radial, $n^i = x^i/r$ es un vector unitario normal a una esfera en el subespacio conformalmente plano y ϵ^{ijkl} es el seudo-tensor de Levi-Civita tridimensional.

Por otro lado, la constricción Hamiltoniana toma la forma:

$$\bar{\nabla}^2 \psi + \frac{1}{8}\bar{A}_{ij}\bar{A}^{ij}\psi^{-7} = 0. \tag{3.45}$$

Adicionalmente se puede suponer que el factor conforme tiene la siguiente forma:

$$\psi = \frac{1}{\chi} + u, \qquad \frac{1}{\chi} := \sum_{\sigma=1}^{N} \frac{\mu_\sigma}{2\|\vec{x} - \vec{c}_\sigma\|}, \tag{3.46}$$

en donde u es una función suave que cumple $u = 1 + \mathcal{O}(r^{-1})$. Sustituyendo (3.46) en la constricción Hamiltoniana (3.45) se obtiene:

$$\bar{\nabla}^2 u + \frac{\eta}{(1+\chi u)^7} = 0, \tag{3.47}$$

con

$$\eta := \frac{1}{8}\chi^7 \bar{A}_{ij}\bar{A}^{ij}, \tag{3.48}$$

la ecuación (3.47) es una ecuación elíptica en espacio plano que puede resolverse con el código Olliptic. La interpretación física de los parámetros de esta ecuación es la siguiente: La masa y posición de cada agujero negro esta parametrizada por μ_σ y $\vec{c}_\sigma$ respectivamente. El momento lineal y angular están parametrizados por P_σ y S_σ respectivamente.

Hasta aquí se han obtenido ya 2 ecuaciones elípticas que se pueden resolver con el código Olliptic en la etapa de datos iniciales, más adelante se mostrarán más aplicaciones del código, en particular se tratará el sistema Scrödinger-Poisson que es una aplicación del código Olliptic en la cual se resuelve la ecuación de Poisson a cada paso de tiempo.

Capítulo 4

Solución de ecuaciones elípticas

Como se mencionó en el primer capítulo el principal objetivo de esta tesis es desarrollar un código computacional para resolver ecuaciones elípticas necesarias para hacer simulaciones en relatividad numérica. Las ecuaciones que hay que resolver contienen fuentes complicadas y dependencia en la variable no lineal, lo cual complica los métodos para obtener soluciones.

Los métodos estudiados para resolver este tipo de ecuaciones serán numéricos, sin embargo, es necesario estudiar las propiedades de la ecuaciones diferenciales parciales elípticas para comprender los métodos utilizados. En la primer parte de este capítulo se estudiará de forma breve la teoría matemática relacionada a este tipo de ecuaciones, una discusión extensa del tema se puede encontrar en los libros que se dan como referencia [28, 29, 30, 31, 32]. En la segunda parte se mencionan algunos métodos analíticos y en la tercer parte los métodos numéricos para obtener soluciones de ecuaciones elípticas, una discusión más extensa sobre métodos numéricos se encuentra en [33, 34, 35, 36]. Así mismo, se presentan varios resultados numéricos y notación que se utilizará en el resto del trabajo, en particular se menciona el concepto de operador en diferencias que es muy útil para expresar la forma discretizada de las ecuaciones así como para obtener formulas para operadores a distintos ordenes de aproximación. Por último se mencionan los tipos de condiciones de frontera implementadas en el código y las formulas para hacer discretización de las mismas.

Como se mostrara más adelante hay métodos numéricos que destacan por la eficiencia, otros por la precisión y otros por la sencillez. Por eficiencia nos referimos a la capacidad de obtener la solución con cierta precisión empleando un menor trabajo de cálculo, precisión se refiere al orden de la aproximación y sencillez por la facilidad de implementarse en un código computacional, desafortunadamente estos tres elementos no son completamente compatibles, pero lo ideal es obtener un balance entre estas tres características.

El tipo de ecuaciones elípticas que se desea resolver, son ecuaciones de constricción para generar datos iniciales y ecuaciones elípticas necesarias en una evolución, que son por ejemplo, las mismas ecuaciones de constricción, ecuaciones para la función de lapso o el sistema Schrödinger-Poisson presentado más adelante.

4.1. Ecuaciones elípticas

De forma general una ecuación diferencial parcial de *segundo orden* definida en un dominio compacto[1] Ω de dimensión n con $(x^1, x^2, \ldots . x^n) \in \Omega$, es una relación de la forma:

$$F(x^i, u, u_{,i}, u_{ij}) = 0, \tag{4.1}$$

en donde $u : \Omega \to \mathbb{R}$ es una función[2] a determinar que cumple la relación. Una subclase de ecuaciones diferenciales parciales de segundo orden son aquellas en las que los términos con segundas derivadas de u se pueden separar del resto:

$$\mathcal{L}u = a^{ij} u_{,ij} + b(u) = 0, \tag{4.2}$$

para este tipo de ecuaciones se hace la siguiente clasificación:

- La ecuación se llama **cuasilineal** si:

$$\begin{aligned} a^{ij} &= a^{ij}(x^i, u, u_{,i}), \\ b(u) &= b(x^i, u, u_{,i}), \end{aligned}$$

  es decir, si cualquier término no lineal presente en la ecuación no involucra segundas derivadas de u.

- La ecuación es **semilineal** si:

$$\begin{aligned} a^{ij} &= a^{ij}(x^i), \\ b(u) &= b(x^i, u, u_{,i}), \end{aligned}$$

  es decir, si es lineal en las segundas derivadas de u.

- Por último la ecuación es **lineal** si:

$$\begin{aligned} a^{ij} &= a^{ij}(x^i), \\ b(u) &= b^i(x^j) u_{,i} + b(x^j) u + s(x^i), \end{aligned}$$

  en este caso el operador $\mathcal{L}$ es lineal en u, $u_{,i}$ y $u_{,ij}$ con coeficientes variables y un término de fuente $s(x^i)$.

A continuación se dan ejemplos de este tipo de ecuaciones en coordenadas cartesianas para dos dimensiones:

Tipo de ecuación	**Ejemplo**
No lineal	$(u_{,xx})^2 + (u_{,yy})^2 + xy = 0.$
Cuasilineal	$(u_{,y})^2 u_{,xx} + (u_{,x})^2 u_{,yy} + u^2 = 0.$
Semilineal	$xu_{,xx} + yu_{,yy} + u^2 + x^2 + y^2 = 0.$
Lineal	$xu_{,xx} + yu_{,yy} + xyu = 0.$
Lineal con coeficientes constantes	$u_{,xx} + u_{,yy} + u + \frac{1}{x} = 0.$

[1] Un dominio compacto en el sentido matemático es un subconjunto de $\mathbb{R}^n$ que es cerrado y acotado. En el sentido físico podemos pensar que es un espacio finito y que incluye las fronteras.

[2] En general la función u puede ir del dominio Ω a cualquier campo, sin embargo en este trabajo supondremos que el campo es el de los números reales.

Además de la clasificación de las ecuaciones dependiendo del grado de linealidad en sus coeficientes, como ya se mencionó en la introducción, existe una clasificación en término de las propiedades de las ecuaciones lineales. Considerando nuevamente los coeficientes de las derivadas de segundo orden a^{ij} y un vector arbitrario de n dimensiones $y = (y_1, y_2, \ldots, y_n)$ se construye la forma cuadrática:

$$Q_1(y) := a^{ij} y_i y_j, \tag{4.3}$$

si $u \in C^2(\Omega)$ las segundas derivadas de u conmutan, lo cual implica que $a^{ij} = a^{ji}$, es decir la matriz asociada $A = [a^{ij}]$ es una matriz $n \times n$ simétrica. Por álgebra lineal sabemos que es posible transformar $Q_1(y)$ a una forma en términos de los ejes principales mediante una transformación de $\{y^i\} \to \{z^i\}$ obteniendo la forma *canónica*:

$$Q_2(z) = \sum_{i=1}^{n} k^i z_i^2, \tag{4.4}$$

la ecuación expresada en esta forma se clasifica en tres tipos:

- **Elípticas** si todos los coeficientes k^i tienen el mismo signo.
- **Hiperbólicas** si existe un coeficiente k^i con signo distinto.
- **Parabólicas** si existe un coeficiente k^i nulo y el resto son del mismo signo.

Para ecuaciones no consideradas en estos tres tipos no existe una nomenclatura. La clasificación antes mencionada fue motivada por la clasificación análoga dada a las secciones cónicas en geometría analítica. Ejemplos de los tres tipos de ecuaciones se han dado ya en la introducción.

Adicionalmente hay que mencionar que existen tres tipos de problemas para ecuaciones diferenciales:

- **Problemas de valores iniciales.** Para ecuaciones diferenciales parciales de segundo orden en $n+1$ dimensiones consiste en dar el valor de la solución en la frontera de un subdominio de dimensión n y los valores de la solución y de su derivada respecto a la coordenada $n+1$ a un valor fijo de la misma en el subdominio (normalmente se interpreta a la coordenada $n+1$ como el tiempo, de modo que podemos pensar que se da el valor de la solución y su velocidad al tiempo $t = 0$). Se busca la solución en todo el dominio.
- **Problema de valores propios.** Este tipo de problemas es más común para ecuaciones ordinarias de la forma $\mathcal{L}u = \lambda u$ que para ecuaciones diferenciales parciales. Se da el valor de la función en las fronteras de un dominio n-dimensional y se busca el valor de una función u y de λ que cumplan la ecuación. Este tipo de problemas por lo general generan un espectro infinito (continuo o discreto), de soluciones y valores propios.
- **Problema de valores a la frontera.** Para ecuaciones diferenciales parciales de segundo orden en n dimensiones consiste en dar el valor de la solución en la frontera de un dominio de dimensión n. Se busca la forma de la solución en el dominio.

En lo que resta del trabajo se tratará de forma exclusiva ecuaciones elípticas lineales y semilineales en problemas de valores a la frontera; y en particular de la forma general:

$$\nabla^2 u + Au + Bu^5 + \frac{C}{(D + Eu)^7} = S, \tag{4.5}$$

en donde A, B, C, D, E y S son funciones de las coordenadas y ∇^2 es el operador de Laplace en un espacio plano[3], definida en un paralelepípedo como dominio.

Para ecuaciones elípticas existe un resultado importante que será de utilidad y que caracteriza el comportamiento de las soluciones de este tipo de ecuaciones. A continuación se enuncia dicho resultado:

Principio del máximo

Sea Ω un dominio acotado y abierto n-dimensional y en donde se cumple:

$$\nabla^2 u > 0, \tag{4.6}$$

con $u \in C(\bar{\Omega}) \cap C^2(\Omega)$. A las funciones que son solución de este tipo de ecuaciones se les conoce como funciones *superarmónicas*. El principio del máximo (versión fuerte) asegura que la solución alcanzará su valor máximo en la frontera $\partial\Omega$.

De cálculo sabemos que una función continua en un dominio acotado alcanza su máximo (y su mínimo) en la cerradura $\bar{\Omega}$. Si el máximo se alcanza en el interior, entonces en ese punto se cumple que $u_{,x^i} = 0$ y $u_{,x^i x^j} \leq 0$, lo cual genera una contradicción. Existe una forma débil de este principio en la cual no es necesaria la desigualdad estricta:

$$\nabla^2 u \geq 0, \tag{4.7}$$

en este caso el máximo se puede alcanzar en la frontera y también posiblemente en el interior, la forma usual de establecer este hecho es con la desigualdad:

$$\min_{\vec{x}\in\partial\Omega} u(\vec{x}) \leq u(\vec{x})\Big|_{\vec{x}\in\Omega} \leq \max_{\vec{x}\in\partial\Omega} u(\vec{x}), \tag{4.8}$$

este resultado garantiza que el valor de la función en el interior será un valor promedio respecto de los vecinos. De manera análoga se puede definir, para funciones que cumplen la desigualdad opuesta:

$$\nabla^2 u < 0, \tag{4.9}$$

un **principio del mínimo** en versión fuerte y débil. Las funciones que cumplen ese tipo de ecuación son *subarmónicas* y las que cumplen la ecuación de Laplace $\nabla^2 u = 0$ se llaman *armónicas*.

4.2. Métodos analíticos

Dado un problema de valores a la frontera, de valores iniciales o de valores propios existen tres preguntas de importancia teórica:

1. ¿Existe solución al problema?
2. En caso de que exista ¿es única la solución?
3. ¿La solución es estable? [4]

[3] En una segunda revisión del código Olliptic se incluirá una operador Laplaciano definido en un espacio curvo en general.

[4] A esta propiedad también se le conoce como dependencia continua respecto a los datos.

una problema para el cual se puede responder afirmativamente a estas tres preguntas se dice que esta *bien planteado*. Desde el punto de vista práctico surgen otras tres cuestiones importantes:

1. ¿Como se puede construir una solución? (exacta o aproximada)
2. ¿Que tan regular es la solución? (por ejemplo en términos de diferenciabilidad)
3. En caso de no poder construir una solución exacta ¿Que tan buena es la aproximación? (en caso de que exista)

lo que esperamos de una solución física es que cumpla afirmativamente las tres preguntas anteriores.

Para obtener soluciones analíticas los métodos más comunes son los siguientes:

Separación de variables

También conocido como método de Fourier o de solución por expansión en funciones propias, consiste en suponer que la solución se puede expresar como un producto de funciones o como una expansión en serie[5]. Dependiendo del operador diferencial y del dominio puede ser posible obtener la forma explicita de la solución o bien los coeficientes de la serie. Como se verá más adelante un procedimiento similar es utilizado en métodos espectrales para obtener soluciones aproximadas a la solución.

Como ejemplo de este método se resolverá la ecuación de calor en 1 dimensión, que además ilustra el hecho de que para este problema la solución evoluciona a una forma estacionaria, lo cual será de utilidad posteriormente para motivar los métodos iterativos. El problema es el siguiente:

$$\begin{array}{lll} \Theta(t,x)_{,t} - \Theta(t,x)_{,xx} = 0, & 0 < x < \pi, \quad t \geq 0, \\ \Theta(0,t) = \Theta(\pi,t) = 0, & t \geq 0, \\ \Theta(x,0) = \sin x, & 0 < x < \pi, \end{array} \tag{4.10}$$

en donde $\Theta(x,t)$ da la distribución de temperatura en una región unidimensional[6] de longitud π, con extremos a una temperatura 0 y una distribución de temperatura inicial dada por la función seno. El método de separación de variables consiste en suponer que la solución se expresa como el producto de dos funciones de una variable:

$$\Theta(x,t) = X(x)T(t), \tag{4.11}$$

de modo que la ecuación diferencial parcial a resolver se transforma en un sistema de ecuaciones diferenciales ordinarias:

$$\begin{array}{lll} X''(x) + \lambda^2 X(x) = 0, & 0 < x < \pi, & X(0) = X(\pi) = 0, \\ T'(t) + \lambda^2 T(t) = 0, & t \geq 0, & T(0) = \sin(x)/X(x), \end{array} \tag{4.12}$$

en donde el parámetro λ y la última condición acoplan al sistema. La solución más general a las ecuaciones es respectivamente:

$$\begin{array}{l} X(x) = A\sin(\lambda x) + B\cos(\lambda x), \\ T(t) = T_0 \mathrm{e}^{-\lambda^2 t}, \end{array} \tag{4.13}$$

[5] Es común utilizar series de Fourier para desarrollar la solución

[6] Por ejemplo una varilla con forma de semicirculo unitario

en donde A, B, T_0 y λ son constantes por determinar usando las condiciones de frontera. Las condiciones para X dan $B = 0$ y $\lambda = n$, con n entero. La condición para T es más fuerte y restinge aún más los valores: $T_0 = 1/A$ y $n = 1$. La solución completa al problema es:

$$\Theta(x,t) = \mathrm{e}^{-t}\sin x, \tag{4.14}$$

en donde la solución para $t \to \infty$ es una solución estacionaria $\Theta(x,t) = 0$. Físicamente el sistema tiende a equilibrio térmico: al no existir fuentes de calor el sistema toma la temperatura de los extremos de la barra. Existen por supuesto, problemas más complicados y aplicaciones más sofisticadas de este método, pero no es la intención de esta tesis explorar a fondo este tipo de técnicas.

Función de Green

A este método se le conoce también como solución por singularidades fundamentales. La idea básica es obtener para una ecuación:

$$\mathcal{L}u = f, \tag{4.15}$$

un operador inverso $\mathcal{L}^{-1}$ de modo que la solución se obtiene aplicandolo:

$$u = \mathcal{L}^{-1}f, \tag{4.16}$$

las soluciones por funciones de Green son la forma más teórica y elegante de obtener las soluciones. Al ser $\mathcal{L}$ un operador diferencial, el operador inverso debe ser un operador integral, la ecuación (4.16) tiene la forma:

$$u(\vec{x}) = \int_\Omega G(\vec{x},\vec{y})f(\vec{y})\mathrm{d}V_{\vec{y}}, \qquad \vec{x} \in \Omega, \tag{4.17}$$

en donde $G(\vec{x},\vec{y})$ se conoce como la función de Green del problema en cuestión. Para ejemplificar la técnica, se utilizará el siguiente problema:

$$\begin{array}{ll} u''(x) = -f(x), & 0 < x < \pi, \\ u(0) = u(\pi) = 0. & \end{array} \tag{4.18}$$

Para obtener una solución a la ecuación inhomogenea podemos usar el sistema auxiliar:

$$u'(x) := v(x), \tag{4.19}$$
$$v'(x) = -f(x). \tag{4.20}$$

De aquí podemos obtener una primer integral de estas ecuaciones:

$$u(x) = \int_0^x v(s)\mathrm{d}s, \tag{4.21}$$
$$v(x) = -\int_0^x f(s)\mathrm{d}s. \tag{4.22}$$

Ahora se multiplica (4.20) por la variable y se integra por partes:

$$\int_0^x sv'(s)\mathrm{d}s = -\int_0^x sf(s)\mathrm{d}s \qquad \Rightarrow$$
$$sv(s)\Big|_0^x - \int_0^x v(s)\mathrm{d}s = -\int_0^x sf(s)\mathrm{d}s,$$

en esta última ecuación usamos (4.21) y (4.22), evaluando y reordenando términos:

$$u(x) = \int_0^x (s - x) f(s) \mathrm{d}s, \tag{4.23}$$

la solución más general de la ecuación original esta dada por:

$$u(x) = c_1 + c_2 x + \int_0^x (s - x) f(s) \mathrm{d}s. \tag{4.24}$$

Imponiendo las condiciones de frontera obtenemos la solución del problema:

$$u(x) = \int_0^\pi G(x, s) f(s) \mathrm{d}s, \tag{4.25}$$

en donde:

$$G(s, x) = \begin{cases} s(\pi - x)/\pi, & s \leq x, \\ x(\pi - s)/\pi, & s \geq x. \end{cases} \tag{4.26}$$

Este ejemplo muestra algunas propiedades de las funciones de Green, por ejemplo la simetría y la dependencia en los valores de frontera. En la figura 4.1 se muestra la gráfica de la función de green 4.26.

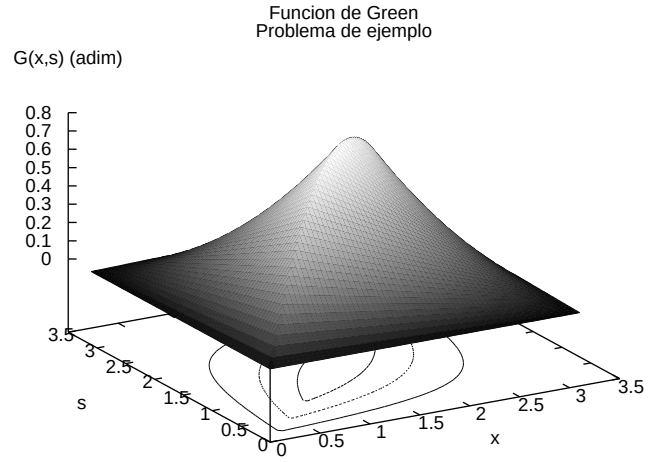

Figura 4.1: Gráfica de la función de Green para el problema de ejemplo. Se observan las propiedades de simetría de la función respecto a las variables x y s.

Métodos variacionales

La idea en este caso es aprovechar el **principio de Dirichlet** y aplicar métodos variacionales. Para el problema de valores a la frontera:

$$\nabla^2 u(\vec{x}) = 0, \qquad \vec{x} \in \Omega, \tag{4.27}$$

$$u(\vec{x}) = f(\vec{x}), \qquad \vec{x} \in \partial\Omega, \tag{4.28}$$

el principio de Dirichlet asegura que la solución $u \in \{v|v(\vec{x}) = f(\vec{x}),\ \vec{x} \in \partial\Omega\}$, es decir es una función que cumple la condición de frontera pero que además minimiza la integral:

$$E(v) = \int_\Omega \|\nabla v\|^2. \tag{4.29}$$

Para ejemplificar este método se usara el problema:

$$\begin{array}{ll} u''(x) = 0, & 0 < x < 1, \\ u(0) = 0, & u(1) = 1. \end{array} \tag{4.30}$$

Un conjunto de funciones que cumplen la condición de frontera es la sucesión $\{v_n = x^n\}$ con $n \in \mathbb{N}$, la integral de energía para esta sucesión es:

$$E(v_n) = \int_0^1 \left(\frac{\mathrm{d}(x^n)}{\mathrm{d}x}\right)^2 \mathrm{d}x = \frac{n^2}{2n-1}, \tag{4.31}$$

el valor de n que minimiza el valor de la integral es $n = 1$ de modo que la solución es $u(x) = v_1 = x$.

Hasta aquí se han presentado algunos ejemplos de métodos analíticos para solución de ecuaciones diferenciales, en las siguientes secciones se tratarán los métodos numéricos, sin embargo muchas de las ideas presentadas serán de utilidad para entender y motivar algunos de los métodos numéricos.

4.3. Métodos numéricos

El uso de métodos y soluciones numéricas en física tiene sus orígenes en trabajos previos a la invención de las primeras computadoras electrónicas, sin embargo con la ayuda de estos aparatos la simulaciones numéricas han adquirido un papel relevante en los trabajos de investigación en física y nos han permitido explorar soluciones a problemas que de lo contrario serían imposibles de resolver. En 1955 Fermi, Pasta y Ulam usaron una de las primeras computadoras de *Los Álamos National Laboratory*, la MANIAC (Mathematical Analyze, Numerical Integrator And Computer) con la cual realizaron la primer simulación numérica del comportamiento térmico de un cristal. El experimento numérico FPU [37] como se conoce actualmente, inauguró el uso de computadoras electrónicas para hacer investigación científica (no militar). El experimento FPU mostró características inesperadas ya que se observó que la no linealidad del sistema estudiado rompe con el principio de equipartición de la energía.

El uso de simulaciones numéricas es un arma de doble filo, si no se comprenden los métodos usados y en particular si no se conoce la magnitud de los errores de aproximación generados es posible llegar a conclusiones equivocadas. En esta sección se tratará principalmente los conceptos de convergencia, consistencia y estabilidad, que son características que debe cumplir un método numérico para garantizar que los resultados son validos.[7] Así mismo, se introducen algunos conceptos relacionados con la eficiencia y el trabajo computacional, los cuales son tópicos relacionados con el rendimiento de los códigos computacionales y los métodos numéricos y es un factor muy importante desde el punto de vista práctico.

Existen varios tipos de errores de aproximación involucrados en los métodos numéricos, el primero que se estudiará es el **error de truncado**.

[7] Desde el punto de vista matemático, para que sean válidos desde el punto de vista físico deben cumplir también condiciones de regularidad y unicidad.

4.3.1. Error de Truncado y orden de aproximación

Como se verá mas adelante los métodos en diferencias finitas se basan en utilizar expansiones en series de Taylor para obtener operadores en diferencias equivalentes a los operadores diferenciales. Pensando en un sistema unidimensional, por el momento, para simplificar la exposición se tiene que dada una función $u : \Omega \subset \mathbb{R} \to \mathbb{R}$ de clase C^∞ y una partición homogénea $P_h(\Omega)$ con norma h del dominio, podemos obtener la expansión de u en serie de Taylor:

$$u(x \pm h) = u(x) \pm \partial_x u(x)h + \frac{1}{2}\partial_x^2 u(x)h^2 \pm O(h^3), \tag{4.32}$$

en donde como notación en general se usara $u_{i\pm1} := u(x \pm h)$. Para h suficientemente pequeña la serie esta dominada por las primeras potencias de h, de modo que truncando la serie se pueden obtener aproximaciones a las derivadas. Para la serie mostrada en (4.32) el error de truncado corresponde a la función:

$$O(h^3) = \sum_{n=3}^{\infty} \frac{1}{n!}\partial_x^n u(x)h^n. \tag{4.33}$$

En general para dos funciones $f(h)$ y $g(h)$, se dice que f es de orden $g(h)$:

$$f(h) = O(g(h)), \qquad h \to 0, \tag{4.34}$$

si existen constantes $C > 0$ y $h_0 > 0$, tales que:

$$\left|\frac{f(h)}{g(h)}\right| < C, \qquad \forall\, h > h_0. \tag{4.35}$$

Para el ejemplo dado el **orden de aproximación** corresponde a una función polinomial cuadrática, o simplemente de orden cuadrático.

Error global y relativo

Suponiendo que u define la solución exacta a un problema diferencial en dos o más dimensiones:

$$\mathcal{L}u(\vec{x}) = f(\vec{x}), \tag{4.36}$$

y u^h la solución exacta para el operador discretizado $\mathcal{L}^h$:

$$\mathcal{L}^h u^h(\vec{x}) = f^h(\vec{x}), \tag{4.37}$$

en donde h representa un valor característico de la norma de la partición del dominio. Se define el **error global** como:

$$\epsilon^h := u - u^h, \tag{4.38}$$

así mismo se define el **error relativo**:

$$\epsilon_r^h := \frac{u - u^h}{u}, \tag{4.39}$$

así pues, el error global es una medida que indica qué tanto difiere la solución de la ecuación en diferencias respecto a la solución de la ecuación diferencial. El error relativo es útil cuando queremos conocer una medida en términos porcentuales del error global, puede ser que las magnitudes del error global sean muy grandes o extremadamente pequeñas, pero es la comparación con el valor de la función lo que determina realmente si la diferencia con la solución exacta es grande o pequeña.

Error de truncado local

El **error de truncado local** τ^h se define como el residuo de aplicar el operador discretizado $\mathcal{L}^h$ a la solución exacta:

$$\tau^h := \mathcal{L}^h u(\vec{x}) - f^h(\vec{x}), \tag{4.40}$$

usando en (4.40) la expresión para u dada en términos de (4.38), obtenemos simplemente:

$$\tau^h = \mathcal{L}^h \epsilon^h, \tag{4.41}$$

el error de truncado local es la aplicación del operador discretizado al error global. El error de truncado está directamente relacionado con el orden de aproximación de los operadores diferenciales, más adelante se determinaran formas de obtener operadores con diversos ordenes de aproximación.

4.3.2. Normas vectoriales y Matriciales

En esta sección se definirá el concepto de normas para vectores y matrices que será de utilidad en las próximas secciones. La norma de un vector $\vec{x}^T = \{x_1, x_2, .., x_n\}$ es un escalar que cumple:

- Es positiva definida: $\|\vec{x}\| \geq 0$.
- La norma es nula solo para el vector nulo: $\|\vec{x}\| = 0 \quad \Leftrightarrow \quad \vec{x} = 0$.
- Para $\alpha \in \mathbb{R}$ la norma factoriza escalares $\|\alpha\vec{x}\| = |\alpha|\|\vec{x}\|$.
- Se cumple la desigualdad del triángulo: $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$.

Existen muchas normas, las más comúnmente utilizadas son:

- La norma L_1: $\|\vec{x}\|_1 := \sum_{i=1}^n |x_i|$.
- La norma L_2: $\|\vec{x}\|_2 := \sqrt{\sum_{i=1}^n x_i^2}$.
- La norma L_∞: $\|\vec{x}\|_\infty := \text{máx}\{|x_i|\}$.
- En general la norma L_k: $\|\vec{x}\|_k := \left(\sum_{i=1}^n |x_i|^k\right)^{1/k}$.

otra norma muy utilizada es la raíz cuadrática media (conocida en ingles como rms - root mean square), que es la norma L_2 normalizada por el número de dimensiones:

$$\|\vec{x}\|_{\text{rms}} := \sqrt{\frac{\sum_{i=1}^n x_i^2}{n}},$$

este tipo de norma es muy utilizada en la implementación de métodos numéricos.

Definición 1 *L_p y L_q son equivalentes si y solo si para cualquier vector $\vec{x}$ existen números $a(p, q, n)$ y $b(p, q, n)$ para los cuales se cumple:*

$$\|\vec{x}\|_p \leq a\|\vec{x}\|_q,$$

$$\|\vec{x}\|_q \leq b\|\vec{x}\|_p,$$

por ejemplo, en un espacio vectorial de n dimensiones se cumple aplicando n veces la desigualdad del triangulo:

$$\begin{aligned} \|\vec{x}\|_2 &\leq \|\vec{x}\|_1, \\ \|\vec{x}\|_1 &\leq \sqrt{n}\|\vec{x}\|_2. \end{aligned}$$

Para espacios vectoriales sobre un campo $\mathbb{R}$ existe el siguiente:

Teorema 1 *Para $p \geq 1$, $q \geq 1$ y un espacio vectorial finito, las normas L_p y L_q son equivalentes.*

Una discusión más detallada se puede encontrar en [38]. Gracias al teorema anterior no es necesario especificar una norma particular en los resultados anteriores, aún cuando por ejemplo L_1 no es igual a la norma L_2 la equivalencia nos permite usar de forma indistinta una u otra, sin embargo para fijar ideas se utilizará en el resto del trabajo la norma L_1 por ser la mas simple de calcular.

Para las matrices, la norma se expresa en términos de la norma vectorial:

Definición 2 *La norma de la matriz $\mathbf{A}$ generada por la norma del vector $\|\vec{x}\|_p$ esta dada por:*

$$\|\mathbf{A}\|_p := \max_{\vec{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\cdot\vec{x}\|_p}{\|\vec{x}\|_p}.$$

Obtenemos las siguientes normas para matrices:

- La norma L_1: $\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$, el máximo de las sumas de las columnas.
- La norma L_2: $\|\mathbf{A}\|_2 = \sqrt{\max \lambda(\mathbf{A}\mathbf{A}^*)}$ en donde $\mathbf{A}^*$ es la matriz adjunta y $\lambda(\mathbf{A}\mathbf{A}^*)$ representa los valores propios de $\mathbf{A}\mathbf{A}^*$.
- La norma L_∞: $\|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$, es el máximo de las sumas de las filas.

4.3.3. Consistencia, convergencia y estabilidad.

Se dice que un método numérico es **consistente** con la ecuación diferencial si y solo si:

$$\lim_{h\to 0} \|\tau^h\| = 0, \tag{4.42}$$

lo cual significa que la versión discretizada del operador se reduce al operador diferencial en el límite cuando la norma de la partición tiende a cero. Típicamente $\tau^h \sim O(h^n)$ con n un entero positivo, lo cual garantiza que este tipo de métodos sean consistentes.

Un método numérico se dice que **converge** si y solo si:

$$\lim_{h\to 0} \|\epsilon^h\| = 0, \tag{4.43}$$

es decir, si la solución exacta del operador discretizado tiene como límite la solución exacta del operador diferencial al hacer tender la norma de la partición del dominio a cero. Hay que notar que la norma utilizada en (4.42) y (4.43) se refiere al vector construido con los valores escalares del error de truncado y del error global respectivamente tomados en cada punto del dominio.

Para operadores lineales, $\mathcal{L}^h$ tiene una matriz asociada, sea $\mathbf{A}$ dicha matriz, de modo que (4.41) se puede expresar en forma matricial:

$$\tau = \mathbf{A} \cdot \epsilon, \tag{4.44}$$

en donde τ y ϵ son los vectores asociados a τ^h y ϵ^h respectivamente. Supongamos que existe $\mathbf{A}^{-1}$, de modo que se puede obtener la solución al sistema anterior:

$$\epsilon = \mathbf{A}^{-1} \cdot \tau, \tag{4.45}$$

calculando ahora la norma del sistema:

$$\|\epsilon\| = \|\mathbf{A}^{-1} \cdot \tau\|, \tag{4.46}$$

usando la desigualdad de Cauchy-Schwarz obtenemos:

$$\|\epsilon\| \leq \|\mathbf{A}^{-1}\|\|\tau\|, \tag{4.47}$$

sabemos que $\tau \sim O(h^n)$ para que ϵ sea del mismo orden, la norma de $\mathbf{A}^{-1}$ debe ser acotada, a esta propiedad se le conoce como estabilidad del método y se enuncia formalmente a continuación.

Decimos que un operado lineal $\mathcal{L}^h$ con matriz asociada $\mathbf{A}$ es **estable** si y solo si, $\forall h$ existe $\mathbf{A}^{-1}$ y existen $C > 0$ (independiente de h) y h_0 tales que:

$$\|\mathbf{A}^{-1}\| \leq C, \qquad \forall\, h < h_0, \tag{4.48}$$

sustituyendo (4.48) y (4.42) en (4.47) obtenemos

$$\|\epsilon\| \leq C\|\tau\| \to 0, \qquad h \to 0, \tag{4.49}$$

lo cual implica (4.43). Para problemas de valores a la frontera se cumple que *estabilidad* y *consistencia* implica *convergencia*. Este resultado se conoce como *Teorema de equivalencia de Lax*. Para problemas de valores a la frontera tiene una forma trivial, pero para problemas de valores iniciales requiere una demostración más complicada.

Como veremos más adelante de la expresión (4.38) se pueden obtener relaciones útiles para hacer pruebas de convergencia para los códigos con lo cual se verifica que se están obteniendo soluciones aproximadas y que los fenómenos observados son representativos del sistema.

Las computadoras actuales no pueden trabajar con números reales (en el sentido matemático), debido a que es necesaria una cantidad infinita de memoria para almacenar la expansión decimal de un numero irracional, sin embargo las computadoras trabajan con aproximaciones conocidas como números de punto flotante, los cuales consisten de una aproximación al número real, hasta un cierto numero de cifras decimales. La diferencia entre el número real y el flotante que lo aproxima se conoce como **error de redondeo** de la maquina: $e^h = u^h - \tilde{u}^h$, en donde $\tilde{u}^h$ denota la aproximación flotante al valor exacto de u^h. El error de redondeo es la principal razón por la cual no es posible obtener la solución exacta u^h de (4.37), sin embargo, la precisión de los método utilizados normalmente es algunos ordenes de magnitud inferior al error de redondeo, por lo cual, en general el error de redondeo no será un factor importante en la aproximación. Podemos entonces, suponer que el valor obtenido por la maquina para el operador discretizado (4.37) es el exacto.

La ecuación (4.49) implica que el error global es del mismo orden que el error de truncado. Suponiendo que tenemos dos particiones del dominio con normas características h y H respectivamente, en donde por ejemplo se cumple que $H = qh$, entonces con un método de orden p se debe

cumplir:

$$\begin{aligned} \epsilon^H &= u - u^H \sim O(H^p) = O((qh)^p), \\ \epsilon^h &= u - u^h \sim O(h^p), \end{aligned}$$

de donde obtenemos:

$$\epsilon^H \sim q^p \epsilon^h, \tag{4.50}$$

entonces por ejemplo para un método de segundo orden ($p = 2$) y una relación entre los puntos de la partición de 2:1 el error global de la particion más burda debe ser 4 veces el error global correspondiente a una partición con la mitad de puntos.

Para utilizar la expresión (4.38) para hacer pruebas de convergencia es necesario conocer el valor de la solución exacta, lo cual en general no será posible (de lo contrario el código no tendría sentido), lo más usual en esos casos es utilizar 3 "resoluciones" o tamaños de partición, sean por ejemplo:

$$\begin{aligned} h' &= 2h, \\ h'' &= 4h, \end{aligned}$$

usando (4.38) obtenemos:

$$\begin{aligned} \epsilon^h &= u - u^h, \\ \epsilon^{h'} &= u - u^{h'}, \\ \epsilon^{h''} &= u - u^{h''}, \end{aligned}$$

de este sistema podemos eliminar u y obtener dos ecuaciones:

$$\begin{aligned} \epsilon^h - \epsilon^{h'} &= u^{h'} - u^h, \\ \epsilon^{h'} - \epsilon^{h''} &= u^{h''} - u^{h'}, \end{aligned}$$

del resultado anterior (4.50), observamos que para las relaciones de partición seleccionadas se cumple:

$$\begin{aligned} (1 - 2^p)\epsilon^h &\sim u^{h'} - u^h, \\ 2^p(1 - 2^p)\epsilon^h &\sim u^{h''} - u^{h'}, \end{aligned}$$

por ultimo obtenemos la relación que nos permite comprobar la convergencia de un código utilizando diferencias finitas:

$$u^{h'} - u^h \sim 2^p(u^{h''} - u^{h'}), \tag{4.51}$$

basta con calcular tres soluciones con distintas normas de partición del dominio (por simplicidad se toman relaciones 2:1) de modo que la diferencia entre las soluciones debe decrecer en la proporción dada por (4.51) en donde la potencia p depende del orden de la aproximación.

4.3.4. Discretización de operadores diferenciales

La mayoría de los métodos que se estudian en esta sección están basados en la discretización de operadores diferenciales usando diferencias finitas por lo que se dará una revisión sobre los procedimientos de discretización de operadores diferenciales. Para simplificar la exposición se tratará principalmente el caso de operadores unidimensionales.

De la expansión en serie de Taylor (4.32) de u podemos obtener al menos 3 aproximaciones de la primer derivada:

- **Lateral derecha (en ingles *forward*):**

$$\frac{\partial u_i}{\partial x} = \frac{u_{i+1} - u_i}{h} + \mathcal{O}(h). \tag{4.52}$$

- **Lateral izquierda (en ingles *backward*):**

$$\frac{\partial u_i}{\partial x} = \frac{u_i - u_{i-1}}{h} + \mathcal{O}(h). \tag{4.53}$$

- **Centrada (en ingles *central*):**

$$\frac{\partial u_i}{\partial x} = \frac{u_{i+1} - u_{i-1}}{2h} + \mathcal{O}(h^2). \tag{4.54}$$

la diferencia centrada se obtiene al restar la expansión en el punto $(i+1)$ de la expansión en $(i-1)$. Observamos que la aproximación centrada es de segundo orden $\mathcal{O}(h^2) \sim -1/6(\partial_{xxx}u)_i h^2$ mientras que las otras aproximaciones son de primer orden $\mathcal{O}(h) \sim \pm 1/2(\partial_{xx}u)_i h$. De forma análoga para la segunda derivada obtenemos una aproximación de segundo orden sumando la expansión en el punto $(i+1)$ y la correspondiente en el punto $(i-1)$:

$$\frac{\partial^2 u_i}{\partial x^2} = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \mathcal{O}(h^2). \tag{4.55}$$

Para obtener operadores con ordenes de aproximación superior se puede utilizar el **Método de Coeficientes Indeterminados**. Para ilustrar el método se calculara el operador de primera derivada lateral a segundo orden de aproximación. Dicho operador tiene la forma genérica:

$$\frac{\partial u_i}{\partial x} = \frac{au_i + bu_{i-1} + cu_{i-2}}{h} + \mathcal{O}(h^2), \tag{4.56}$$

utilizando la expansión en serie de Taylor obtenemos:

$$u_{i-2} = u_i - 2h\partial_x u_i + 2h^2\partial_x^2 u_i - \frac{(2h)^3}{6}\partial_x^3 u_i + \cdots, \tag{4.57}$$

$$u_{i-1} = u_i - h\partial_x u_i + \frac{h^2}{2}\partial_x^2 u_i - \frac{(h)^3}{6}\partial_x^3 u_i + \cdots, \tag{4.58}$$

sustituyendo en (4.56) obtenemos:

$$\frac{\partial u_i}{\partial x} = \frac{1}{h}(a+b+c)u_i - (b+2c)\partial_x u_i + \frac{h}{2}(b+4c)\partial_x^2 u_i + \mathcal{O}(h^3), \tag{4.59}$$

para que (4.56) sea una identidad se debe cumplir:

$$\begin{aligned} a+b+c &= 0, && (4.60)\\ b+2c &= -1, && (4.61)\\ b+4c &= 0, && (4.62) \end{aligned}$$

utilizando la solución del sistema obtenemos:

$$\frac{\partial u_i}{\partial x} = \frac{3u_i - 4u_{i-1} + u_{i-2}}{2h} + \mathcal{O}(h^2), \tag{4.63}$$

esta es una aproximación a segundo orden "lateral izquierda" de la primer derivada. De forma análoga se pueden obtener formulas para derivadas de orden superior y en dominios de mayor dimensión. Por ejemplo para obtener una segunda derivada en 3 dimensiones a segundo orden de aproximación usamos la relación:

$$\frac{\partial^2 u_{ijk}}{\partial x^2} = \frac{au_{ijk} + bu_{i-1,jk} + cu_{i-2,jk}}{h^2} + \mathcal{O}(h^2), \tag{4.64}$$

el sistema de ecuaciones resultante en este caso es:

$$\begin{aligned} a+b+c &= 0, && (4.65)\\ b+2c &= 0, && (4.66)\\ b+4c &= 2, && (4.67) \end{aligned}$$

de donde obtenemos la segunda derivada de segundo orden respecto a x lateral izquierda:

$$\frac{\partial^2 u_{ijk}}{\partial x} = \frac{u_{ijk} - 2u_{i-1,jk} + u_{i-2,jk}}{h^2} + \mathcal{O}(h^2). \tag{4.68}$$

Para obtener fácilmente formulas para operadores diferenciales se utiliza el concepto de **operadores en diferencias**:

- **Desplazamiento:** $E^n u_i := u_{i+n}$.
- **Incremento:** $\delta^+ u_i := u_{i+1} - u_i = (E-1)u_i$.
- **Decremento:** $\delta^- u_i := u_i - u_{i-1} = (1-E^{-1})u_i$.
- **Semi-central:** $\delta u_i := u_{i+1/2} - u_{i-1/2} = (E^{1/2} - E^{-1/2})u_i$.
- **Central:** $\bar{\delta} u_i := \frac{1}{2}(u_{i+1} - u_{i-1}) = \frac{1}{2}(E - E^{-1})u_i$.
- **Promedio:** $\mu u_i := \frac{1}{2}(u_{i+1/2} + u_{i-1/2}) = \frac{1}{2}(E^{1/2} + E^{-1/2})u_i$.
- **Derivada:** $Du := \frac{\partial u}{\partial x}$.

Con estos operadores la expansión en serie de Taylor (4.32) toma la forma:

$$Eu_i = \left(1 + hD + \frac{(hD)^2}{2!} + \frac{(hD)^3}{3!} + \cdots\right) u_i, \tag{4.69}$$

o bien

$$Eu_i = e^{hD}u_i, \tag{4.70}$$

de esta identidad obtenemos:

$$hD = \ln E, \tag{4.71}$$

esta última expresión es muy útil para obtener aproximaciones de las derivadas con precisión arbitraria.

Regresando a la derivada lateral izquierda, obtenemos:

$$hD = \ln E = -\ln[1 - \delta^-], \tag{4.72}$$

o bien

$$hD = \delta^- + \frac{(\delta^-)^2}{2} + \frac{(\delta^-)^3}{3} + \frac{(\delta^-)^4}{4} + \cdots \tag{4.73}$$

usando la identidad:

$$(\delta^-)^2 := \delta^-\delta^- = (1 - E^{-1})(1 - E^{-1}) = 1 - 2E^{-1} + E^{-2}, \tag{4.74}$$

y truncando (4.73) después de los primeros dos términos, obtenemos:

$$hDu_i = \delta^- u_i + \frac{1}{2}(1 - 2E^{-1} + E^{-2})u_i, \tag{4.75}$$

aplicando los operadores y simplificando obtenemos nuevamente la expresión (4.63). De forma análoga se puede obtener la expresión para la primer derivada lateral derecha a segundo orden y manteniendo mas términos en las expansiones se puede mejorar el orden de aproximación.

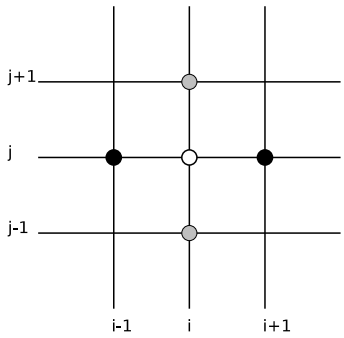

Figura 4.2: Molécula computacional 2 dimensional. Es una representación de un espacio discretizado en donde se represetan los puntos en el dominio que se utilizan en un operador en diferencias finitas.

Para extender los operadores a más dimensiones se procede de forma análoga al caso unidimensional, definiendo operadores que actúan solo en ciertas direcciones, por ejemplo para dos dimensiones, podemos definir primeras y segundas derivadas con puntos en la dirección de la derivada que deseamos calcular, como se muestra en la figura 4.2, con puntos en negro obtenemos la primer derivada centrada en la dirección x:

$$\partial_x u_{i,j} = \frac{u_{i+1,j} - u_{i+1,j}}{2h_x} + \mathcal{O}(h_x^2), \tag{4.76}$$

con puntos en gris se calcula la primer derivada centrada en la dirección y:

$$\partial_y u_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2h_y} + \mathcal{O}(h_y^2), \tag{4.77}$$

de esta forma podemos definir operadores discretos para cada dirección:

$$\delta_x^2 u_{i,j} := u_{i-1,j} - 2u_{i,j} + u_{i+1,j}, \tag{4.78}$$
$$\delta_y^2 u_{i,j} := u_{i,j-1} - 2u_{i,j} + u_{i,j+1}, \tag{4.79}$$

y construir operadores más complicados como el Laplaciano en dos dimensiones:

$$\nabla_+^2 := \frac{\delta_x^2}{h_x^2} + \frac{\delta_y^2}{h_y^2}, \tag{4.80}$$

el cual opera sobre $u_{i,j}$ de la siguiente forma:

$$\nabla_+^2 u_{i,j} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_y^2}, \tag{4.81}$$

se usó la notación con un signo + para denotar que se usan puntos verticales y horizontales, sin embargo también podemos definir un operador Laplaciano usando puntos sobre diagonales:

$$\nabla_\times^2 := \frac{(\mu_y \delta_x)^2}{h_x^2} + \frac{(\mu_x \delta_y)^2}{h_y^2}, \tag{4.82}$$

que en termino de los operadores:

$$(\mu_y \delta_x)^2 = (E_y + 2 + E_y^{-1})(E_x - 2 + E_x^{-1}), \tag{4.83}$$
$$(\mu_x \delta_y)^2 = (E_x + 2 + E_x^{-1})(E_y - 2 + E_y^{-1}), \tag{4.84}$$

y suponiendo por simplicidad que tomamos una partición homogénea del dominio $h_x = h_y$, el operador Laplaciano diagonal toma la forma:

$$\nabla_\times^2 = \frac{E_x E_y + E_x E_y^{-1} - 4 + E_x^{-1} E_y + E_x^{-1} E_y^{-1}}{2h^2}, \tag{4.85}$$

en la figura 4.3 se presenta un esquema de la molécula computacional para el operador de Laplace definido sobre puntos diagonales. Se observa que el operador no actúa de forma integra sobre la malla lo cual supone una dificultad ya que la propagación numérica de la información hace inestable los métodos usados con éste operador. Más adelante se tratará en detalle el problema de la propagación numérica de la información.

Un operador Laplaciano mas general se define tomando una combinación lineal de los operadores sobre diagonales y sobre puntos en cruz, sin embargo en la mayoría de las aplicaciones se utiliza el operador ∇_+.

4.3.5. Diferencias finitas: Método directo

El método directo en diferencias finitas para obtener la solución numérica de ecuaciones elípticas consiste en discretizar el problema de valores a la frontera:

$$\mathcal{L}u(\vec{x}) = f(\vec{x}), \qquad \vec{x} \in \Omega, \tag{4.86}$$

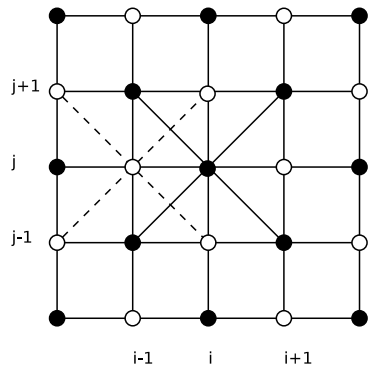

Figura 4.3: Molécula computacional para el operador de Laplace usando puntos sobre la diagonal: El operador así definido se evalúa en dos clases de puntos sobre la malla (puntos negros y blancos) de modo que la propagación numérica se da en dos sub-mallas distintas.

en donde $\mathcal{L}$ es un operador elíptico como se definió en la sección 4.1 y en donde además se tiene la condición de frontera

$$\mathcal{B}u = B(\vec{x}), \qquad \vec{x} \in \partial\Omega, \tag{4.87}$$

en donde, en este caso $\mathcal{B}$ es un operador que define una condición de frontera como las tratadas más adelante en la sección 4.4.

Con los metodos tratados en la sección anterior obtenemos la versión discretizada del problema:

$$\mathcal{L}^h u^h = f^h(\vec{x}), \qquad \vec{x} \in \Omega^h, \tag{4.88}$$

en donde como en casos anteriores se etiqueta con h a los operadores y funciones para enfatizar que se trata de un espacio discretizado con norma de partición h.

Para simplificar podemos suponer que el operador elíptico es lineal (tipo Laplace) y en tres dimensiones, de modo que podemos expresarlo en términos de operadores en diferencias centradas obteniendo:

$$\mathcal{L}^h u^h := \nabla_+ u^h + \left(b_x(\vec{x})^h \frac{\bar{\delta}_x}{h_x} + b_y(\vec{x})^h \frac{\bar{\delta}_y}{h_y} + b_z(\vec{x})^h \frac{\bar{\delta}_z}{h_z} \right) u^h, \tag{4.89}$$

en donde $\left\{ b_x^h(\vec{x}), b_y^h(\vec{x}), b_z^h(\vec{x}) \right\}$ son los coeficientes de las primeras derivadas. Se obtiene un sistema que se puede expresar en términos matriciales.

El método directo consiste en resolver el problema algebraico de invertir la matriz que define al operador para obtener la solución. Para ejemplificar se usará la ecuación de Poisson en 3 dimensiones en coordenadas cartesianas, en un dominio cúbico.

$$\Omega = \{(x, y, z) \mid x_1 \leq x \leq x_{N-1}, \quad y_1 \leq y \leq y_{N-1}, \quad z_1 \leq z \leq z_{N-1}\}, \tag{4.90}$$

el problema esta dado por:

$$\partial_x^2 u + \partial_y^2 u + \partial_z^2 u = \rho(x, y, z), \qquad (x, y, z) \in \Omega, \tag{4.91}$$

con condiciones tipo Dirichlet[8]:

$$u(x, y, z) = B(x, y, z), \qquad (x, y, z) \in \partial\Omega. \tag{4.92}$$

[8]Es decir dando el valor de la solución en la frontera (ver sección 4.4)

Utilizando un dominio con partición homogénea:

$$\Omega^h = \{(x_1 + h_x(i-1), y_1 + h_y(j-1), z_1 + h_z(k-1)) \mid i,j,k \in \{1,\ldots,N\}\}, \tag{4.93}$$

en donde:

$$\begin{aligned} h_x &= \frac{x_{N-1} - x_1}{N-1}, \\ h_y &= \frac{y_{N-1} - y_1}{N-1}, \\ h_z &= \frac{z_{N-1} - z_1}{N-1}. \end{aligned}$$

Las ecuaciones validas en el interior de Ω^h son

$$\left(\frac{E_x - 2 + E_x^{-1}}{h_x^2} + \frac{E_y - 2 + E_y^{-1}}{h_y^2} + \frac{E_z - 2 + E_z^{-1}}{h_z^2} \right) u_{ijk} = \rho_{ijk}, \tag{4.94}$$

con $\rho_{ijk} := \rho(x_1 + h_x(i-1), y_1 + h_y(j-1), z_1 + h_z(k-1))$, además las ecuaciones de frontera son simplemente:

$$\begin{aligned} u_{1,j,k} &= B_{1,j,k}, \\ u_{N-1,j,k} &= B_{N-1,j,k}, \\ u_{i,1,k} &= B_{i,1,k}, \\ u_{i,N-1,k} &= B_{i,N-1,k}, \\ u_{i,j,1} &= B_{i,j,1}, \\ u_{i,j,N-1} &= B_{i,j,N-1}, \end{aligned}$$

en donde $B_{i,j,k} := B(x_1 + h_x(i-1), y_1 + h_y(j-1), z_1 + h_z(k-1))$, en esta representación se puede hacer el mapeo de índices:

$$I \longrightarrow N(N(i-1) + j - 1) + k, \tag{4.95}$$

con esta transformación el sistema (4.94) toma la forma:

$$\begin{aligned} &\frac{1}{h_x^2} u_{I-N^2} + \frac{1}{h_y^2} u_{I-N} + \frac{1}{h_z^2} u_{I-1} \\ &\qquad - \left(\frac{2}{h_x^2} + \frac{2}{h_y^2} + \frac{2}{h_z^2} \right) u_I \\ &+ \frac{2}{h_z^2} u_{I+1} + \frac{1}{h_y^2} u_{I+N} + \frac{1}{h_x^2} u_{I+N^2} = \rho_I, \end{aligned} \tag{4.96}$$

en donde $I \in \{N^2, \ldots, N^3 - N^2\}$, en esta expresión se observa claramente la estructura matricial del sistema, utilizando además las expresiones correspondientes a las condiciones de frontera se obtiene un sistema de N^3 ecuaciones con N^3 incógnitas. Obtener la solución de este sistema numéricamente requiere invertir la matriz asociada. Para un operado general la matriz resultante no es simétrica, para invertir este tipo de matrices se utiliza el método de eliminación Gaussiana, el cual consiste en hacer una descomposición LU (lower-upper) factorizando la matriz original en su parte triangular superior y la parte triangular inferior:

$$\mathbf{M} = \mathbf{L} \cdot \mathbf{U}, \tag{4.97}$$

de esa forma el sistema original:

$$\mathbf{M}\vec{x} = \vec{b}, \tag{4.98}$$

se puede descomponer en dos sub-sistemas:

$$\mathbf{L}\vec{y} = \vec{b}, \tag{4.99}$$

$$\mathbf{U}\vec{x} = \vec{y}, \tag{4.100}$$

como se muestra en [36] y [35] los algoritmos directos más eficientes para obtener la solución de esta clase de sistemas requieren un trabajo computacional para una matriz $M \times M$ dado por:

$$W_{LU} \simeq \frac{2}{3}M^3, \tag{4.101}$$

de modo que para un problema 3 dimensional discretizado usando una particion homogénea de N puntos en cada dirección obtenemos una matriz con $M = N^3$ puntos por lo que el trabajo de calculo es de alrededor de $W_{LU} \sim N^9$ lo cual en tiempo de maquina y para altas "resoluciones" se traduce en varias horas de cálculo para sistemas relativamente simples. Como se verá más adelante existen mejores opciones para resolver ecuaciones elípticas de forma numérica.

4.3.6. Métodos iterativos

La idea principal de los métodos iterativos es aprovechar las propiedades estudiadas en la sección 4.1, en particular el principio del Máximo (Mínimo) de las ecuaciones súper-armónicas (sub-armónicas). Así pues, se puede pensar en que cada punto del espacio discretizado y sus primeros vecinos como un subproblema de valores a la frontera en donde solo hay un punto interior, sabemos pues, que si $\nabla^2\phi > 0$ ($\nabla^2\phi < 0$) el máximo (mínimo) será alcanzado en la frontera, en este caso en los puntos vecinos y el punto interior tendrá un valor más o menos promedio al de los vecinos. Para obtener un método iterativo usaremos como motivación la ecuación de difusión.

Sea Ω una región acotada con frontera $\partial\Omega$ y $u(\vec{x}, t) : \Omega \times \mathbb{R} \longrightarrow \mathbb{R}$ solución de la ecuación diferencial:

$$\frac{\partial u(\vec{x}, t)}{\partial t} = \nabla^2 u(\vec{x}, t) + \rho(\vec{x}), \tag{4.102}$$

en donde se ha supuesto de forma explicita que la fuente no depende del tiempo. La solución esta dada por la serie:

$$\{u^0, u^1, u^2, \cdots, u^n, \cdots, u^\infty\}, \tag{4.103}$$

en donde $u^\infty(\vec{x}) : \Omega \longrightarrow \mathbb{R}$ se conoce como estado estacionario y es solución del problema correspondiente a la ecuación de Poisson:

$$\nabla^2 u(\vec{x}) + \rho(\vec{x}) = 0. \tag{4.104}$$

La versión discretizada tridimensional usando diferencias centradas de (4.102) esta dada por:[9]

$$\frac{u^{n+1}_{ijk} - u^n_{ijk}}{\Delta t} - \frac{1}{\Delta x^2}(E_x + E_y + E_z - 6 + E_z^{-1} + E_y^{-1} + E_x^{-1})u^n_{ijk} = \rho_{ijk}, \tag{4.105}$$

la solución de esta ecuación se puede expresar en términos de la solución estacionaria:

$$u^n_{ijk} = u^\infty_{ijk} + \Delta u^n_{ijk}, \tag{4.106}$$

[9] En este caso Δx denota la norma característica de la partición del dominio espacial.

en esta expresión Δu^n_{ijk} representa la desviación respecto al solución estacionaria. Dado que la solución estacionaria cumple (4.104), la desviación cumple la ecuación discretizada:

$$\Delta u^{n+1}_{ijk} = \Delta u^n_{ijk} + D(E_x + E_y + E_z - 6 + E_z^{-1} + E_y^{-1} + E_x^{-1})\Delta u^n_{ijk}, \tag{4.107}$$

con $D = \Delta t/\Delta x^2$. Suponiendo que la solución tiene una perturbación armónica:

$$\Delta u^{n+1}_{ijk} = e^{\mathrm{i}i\phi_x} e^{\mathrm{i}j\phi_y} e^{\mathrm{i}k\phi_z} \Delta \hat{u}^{n+1}_{ijk}, \tag{4.108}$$

en donde $\phi_x = \pi/N_x$, $\phi_y = \pi/N_y$, $\phi_z = \pi/N_z$ y $\Delta\hat{u}^{n+1}$ cumple:

$$\Delta\hat{u}^{n+1} = \mathbf{G}(\phi_\mathbf{x}, \phi_\mathbf{y}, \phi_\mathbf{z})\Delta\hat{u}^n = [\mathbf{G}(\phi_\mathbf{x}, \phi_\mathbf{y}, \phi_\mathbf{z})]^n \Delta\hat{u}^1, \tag{4.109}$$

el factor $\mathbf{G}(\phi_\mathbf{x}, \phi_\mathbf{y}, \phi_\mathbf{z})$ se conoce como *matriz de amplificación* y para un sistema estable se debe cumplir:

$$\rho(\mathbf{G}) = \max_k |\lambda_k| \leq 1, \tag{4.110}$$

en donde $\rho(\mathbf{G})$ se conoce como *radio espectral* de la matriz $\mathbf{G}$. Sustituyendo (4.108) y (4.109) en (4.107) obtenemos:

$$\begin{aligned} e^{\mathrm{i}i\phi_x} e^{\mathrm{i}j\phi_y} e^{\mathrm{i}k\phi_z} [\mathbf{G}]^{n+1} =& e^{\mathrm{i}i\phi_x} e^{\mathrm{i}j\phi_y} e^{\mathrm{i}k\phi_z} [\mathbf{G}]^n \\ &+ D(E_x + E_y + E_z - 6 + E_z^{-1} + E_y^{-1} + E_x^{-1}) e^{\mathrm{i}i\phi} e^{\mathrm{i}j\phi_y} e^{\mathrm{i}k\phi_z} [\mathbf{G}]^n \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad G(\phi_x, \phi_y, \phi_z) &= 1 + D(e^{\mathrm{i}\phi_x} + 2 + e^{-\mathrm{i}\phi_x} + e^{\mathrm{i}\phi_y} - 2 + e^{-\mathrm{i}\phi_y} + e^{\mathrm{i}\phi_z} - 2 + e^{-\mathrm{i}\phi_z}) \\ &= 1 - 4D(\sin^2\frac{\phi_x}{2} + \sin^2\frac{\phi_y}{2} + \sin^2\frac{\phi_z}{2}), \end{aligned}$$

para que se cumpla (4.110), $D \leq \frac{1}{6}$, en particular tomando $D = \frac{1}{6}$ se obtiene la convergencia más rápida. sustituyendo el valor para D en (4.107) obtenemos:

$$u^{n+1}_{ijk} = \frac{1}{6}\left[(E_x + E_y + E_z + E_z^{-1} + E_y^{-1} + E_x^{-1})u^n_{ijk} - \Delta x^2 \rho_{ijk}\right], \tag{4.111}$$

la relación iterativa (4.111) define el **método de Jacobi**. Hay que notar que para este método se requiere conocer todos los valores de la iteración anterior. Una forma alternativa y más general, pero menos física de obtener el método de Jacobi es definir una matriz error en cada punto:

$$r^h := \mathcal{L}^h u^h - f^h, \tag{4.112}$$

recorriendo cada punto de la malla podemos obtener una u^h que satisface puntualmente la ecuación, al modificar los valores de u^h en cada punto la solución global convergerá iterativamente a la solución. Por ejemplo para la ecuación anterior:

$$r_{ijk} := \left(\frac{E_x + E_y + E_z - 6 + E_z^{-1} + E_y^{-1} + E_x^{-1}}{\Delta x^2}\right) u^n_{ijk} - \rho_{ijk}, \tag{4.113}$$

el valor de u_{ijk} que satisface que $r_{ijk} = 0$ es:

$$u^{n+1}_{ijk} = \frac{u^n_{i+1,j,k} + u^n_{i-1,j,k} + u^n_{i,j+1,k} + u^n_{i,j-1,k} + u^n_{i,j,k+1} + u^n_{i,j,k-1}}{6} - \frac{\Delta x^2}{6}\rho_{ijk},$$

obtenemos la misma ecuación (4.111) solo que con los operadores aplicados de forma explicita. De esta expresión se observa que el valor en el punto u_{ijk} es el promedio de los vecinos menos un promedio respecto al valor de la fuente. Si pensamos por ejemplo que la ecuación describe un problema eléctrico con una densidad de carga ρ, entonces el término $h^2\rho_{ijk}$ es aproximadamente el valor de la carga total en un cuadrado de lado h alrededor del punto (ih, jh, kh).

La solución para este caso es muy sencilla ya que el operador que estamos tratando como ejemplo es lineal, sin embargo, para ecuaciones no lineales, encontrar la solución para u_{ijk} que satisface que el error es cero puntualmente puede llevar a ecuaciones tracendentes, sin embargo se puede usar el método de Newton para obtener las raíces de la ecuación.

El método de Newton para obtener las raíces de una función $f : \mathbb{R} \longrightarrow \mathbb{R}$ es un método iterativo que consiste en aproximarse a la raíz utilizando las abscisas de las rectas tangentes, un diagrama del procedimiento se muestra en la figura 4.4

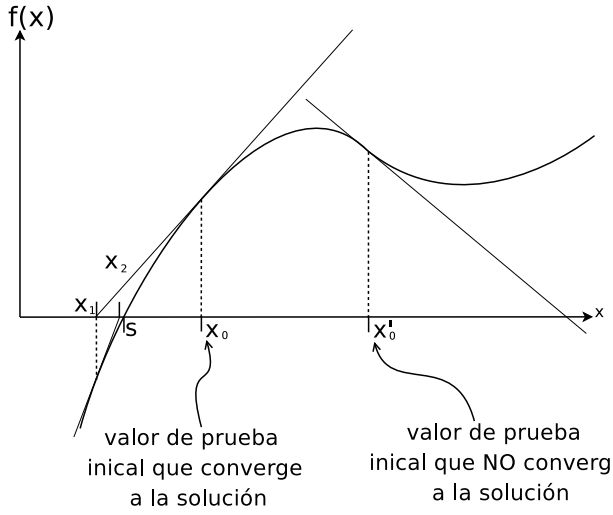

Figura 4.4: Diagrama que ejemplifica geométricamente el método de Newton para encontrar raíces de funciones $f : \mathbb{R} \to \mathbb{R}$. Para funciones con raíces complejas es posible dar un valor inicial que no converge a la solución, sin embargo, si el valor inicial es cercano a la raíz, el método converge rápidamente.

Supongamos que queremos obtener la raíz s de una función f, tal que podemos expresarla de la siguiente forma:

$$f(x) = x - g(x), \tag{4.114}$$

en donde g es una función que se determinará posteriormente. Sea h la distancia entre s y un punto arbitrario (un valor de prueba inicial) x_0, expandiendo alrededor de s en serie de Taylor obtenemos:

$$f(s) = 0 = f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \cdots \tag{4.115}$$

para $f'(x_0) \neq 0$, truncando la serie y usando sólo el termino lineal en h, obtenemos

$$h_0 := h = -\frac{f(x_0)}{f'(x_0)}, \tag{4.116}$$

ponemos entonces usar $x_1 = x_0 + h_0$ como un nuevo punto inicial el cual en general se encontrará más cerca de s.[10] Continuando de esta manera obtenemos un método iterativo:

$$\begin{aligned} x_{n+1} &= x_0 + h_0 + \cdots + h_n, \\ h_n &= -\frac{f(x_n)}{f'(x_n)}, \end{aligned}$$

y de forma más compacta se pueden condenzar estas dos ecuaciones en una sola:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{4.117}$$

de aquí obtenemos que:

$$g(x_n) = \frac{f(x_n)}{f'(x_n)}, \tag{4.118}$$

la ecuación (4.117) es la formula usual del método de Newton y aplicándola al caso del método de Jacobi obtenemos para un operador no lineal:

$$u_{ijk}^{n+1} = u_{ijk} - r_{ijk}^n \left[\frac{\partial r_{ijk}^n}{\partial u_{ijk}}\right]^{-1}_{u_{ijk}=u_{ijk}^n}. \tag{4.119}$$

Se puede derivar el método de Jacobi para un sistema lineal arbitrario (no necesariamente un sistema de ecuaciones en diferencias):

$$\mathbf{A}\vec{x} = \vec{b}, \tag{4.120}$$

la idea básica es hacer una descomposición LDU de A, de la siguiente forma:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}, \tag{4.121}$$

en donde $\mathbf{L}$ es la parte triangular estrictamente inferior, $\mathbf{D}$ es la parte diagonal y $\mathbf{U}$ es la parte triangular estrictamente superior. Reescribimos (4.120) en la forma:

$$\mathbf{D}\vec{x}^{\,n+1} = \vec{b} - (\mathbf{L} + \mathbf{U})\vec{x}^{\,n}, \tag{4.122}$$

en donde n denota el número de iteración. Este tipo de descomposición iterativa converge a la solución siempre que $\mathbf{A}$ sea una matriz con diagonal dominantes, es decir que cumpla con la condición:

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{N} |a_{ij}|, \qquad i \in \{1, 2, \ldots, N\}, \tag{4.123}$$

en donde se da la desigualdad estricta al menos para una valor de i.

El sistema en esta forma es trivial de invertir, para $\mathbf{D}$ no singular:

$$\vec{x}^{\,n+1} = \mathbf{D^{-1}}\vec{b} - \mathbf{D^{-1}}(\mathbf{L} + \mathbf{U})\vec{x}^{\,n}. \tag{4.124}$$

De esta forma se pueden obtener otros métodos iterativos como se mostrará en las siguientes secciones.

[10] Es posible que x_1 no se encuentre más cerca de la solución, esto ocurre cuando la función tiene raíces complejas o múltiples. En la figura 4.4 por ejemplo, si se selecciona x_0' como valor inicial no se convergerá a ninguna solución.

Algoritmo de Gauss-Seidel

Para motivar el método de Gauss-Seidel regresamos a la descomposición LDU del sistema lineal, pero usando el hecho de que los métodos implícitos usualmente convergen más rápido, escribimos (4.120) en la forma:

$$(\mathbf{L}+\mathbf{D})\vec{x}^{\,n+1} = \vec{b} - \mathbf{U}\vec{x}^{\,n}, \tag{4.125}$$

para la ecuación de Poisson obtenemos:

$$\frac{1}{\Delta x^2}(E_x^{-1}+E_y^{-1}+E_z^{-1}-6)u_{ijk}^{n+1} = -\rho_{ijk} - \frac{1}{\Delta x^2}(E_x+E_y+E_z)u_{ijk}^{n}, \tag{4.126}$$

reordenando términos:

$$u_{ijk}^{n+1} = \frac{1}{4}\big[(E_x+E_y+E_z)u_{ijk}^{n} + (E_x^{-1}+E_y^{-1}+E_z^{-1})u_{ijk}^{n+1} + \Delta x^2\rho_{ijk}\big], \tag{4.127}$$

para aplicar el método debemos conocer los valores de $u_{i-1,j,k}$, $u_{i,j-1,k}$ y $u_{i,j,k-1}$ en la iteración $n+1$, lo cual es posible si recorremos los arreglos en el orden numérico de los índices. Como se verá en el análisis de convergencia el método de Gauss-Seidel converge usando la mitad de iteraciones que el método de Jacobi, así mismo no es necesario almacenar los valores de la iteración anterior ya que se pueden actualizar los valores del mismo arreglo.

Otros métodos iterativos

Se puede generalizar los métodos iterativos usando las ideas de las secciones anteriores. Regresando al sistema lineal (4.120) en donde $\mathbf{A}$ es la matriz de coeficientes de algún operador discretizado (por ejemplo el operador de Poisson como en los casos anteriores) podemos definir el vector de error global:

$$\vec{\epsilon}^{\,n} := \vec{x}^{\,n} - \vec{x}^{\infty}, \tag{4.128}$$

en donde $\vec{x}^{\infty}$ es la solución estacionaria o exacta con:

$$\lim_{n\to\infty} \vec{\epsilon}^{\,n} = 0. \tag{4.129}$$

Supongamos que $\mathbf{A}$ se puede expresar como la suma de dos matrices:

$$\mathbf{A} = \mathbf{M} + \mathbf{N}, \tag{4.130}$$

en donde se obtiene la solución iterando:

$$\mathbf{M}\vec{x}^{\,n+1} = \vec{b} - \mathbf{N}\vec{x}^{\,n}. \tag{4.131}$$

Definiendo ahora un vector residuo dado por:

$$\vec{\tau}^{\,n} := \mathbf{A}\vec{x} - \vec{b}, \tag{4.132}$$

obtenemos la relación:

$$\vec{\tau}^{\,n} = \mathbf{A}\vec{\epsilon}^{\,n}, \tag{4.133}$$

recuperamos nuevamente la expresión (4.44).

Para los casos ya estudiados:

- Jacobi:

$$\begin{aligned} \mathbf{M} &= \mathbf{D}, \\ \mathbf{N} &= \mathbf{L} + \mathbf{U}. \end{aligned}$$

- Gauss-Seidel:

$$\begin{aligned} \mathbf{M} &= \mathbf{D} + \mathbf{L}, \\ \mathbf{N} &= \mathbf{U}. \end{aligned}$$

además existen al menos dos métodos más. El método de Sobre-relajación sucesiva (SOR por sus siglas en ingles - Successive over-relaxation) y el método simétrico de sobre-relajación sucesiva (SSOR - Symmetric successive over-relaxation):

- SOR:

$$\begin{aligned} \mathbf{M} &= \omega^{-1}\mathbf{D} + \mathbf{L}, \\ \mathbf{N} &= (1-\omega^{-1})\mathbf{D} + \mathbf{U}. \end{aligned}$$

- SSOR:

$$\begin{aligned} \mathbf{M} &= \frac{(\mathbf{D}+\omega\mathbf{U})\mathbf{D}^{-1}(\mathbf{D}+\omega\mathbf{L})}{\omega(2-\omega)}, \\ \mathbf{N} &= -\frac{[(1-\omega)\mathbf{D}-\omega\mathbf{L}]\mathbf{D}^{-1}[(1-\omega)\mathbf{D}-\omega\mathbf{U}]}{\omega(2-\omega)}. \end{aligned}$$

En donde $1 < \omega < 2$ es un parámetro que depende del método y del sistema. (Para SOR y la ecuación de Poisson el valor óptimo es $\omega_{opt} = 2/(1+\sqrt{1-z_J^2})$, en donde z_J es el radio de convergencia del método de Jacobi que se define en la siguiente sección). La forma explicita del método SOR esta dada por:

$$u_{ijk}^{n+1} = u_{ijk}^{n} + \omega d_{ijk}^{n}, \tag{4.134}$$

$$d_{ijk}^{n} := \tilde{u}_{ijk}^{n} - u_{ijk}^{n}, \tag{4.135}$$

en donde $\tilde{u}_{ijk}^{n}$ esta dada por el método de Gauss-Seidel.

Regresando al caso general, de las expresiones (4.131) y de:

$$(\mathbf{M}+\mathbf{N})\vec{x}^{\infty} = \vec{b}, \tag{4.136}$$

obtenemos en términos de $\vec{\epsilon}^{\,n}$:

$$\mathbf{M}\vec{\epsilon}^{\,n+1} = -\mathbf{N}\vec{\epsilon}^{\,n}, \tag{4.137}$$

para $\mathbf{M}$ no singular:

$$\begin{aligned} \vec{\epsilon}^{\,n+1} &= -\mathbf{M}^{-1}\cdot\mathbf{N}\vec{\epsilon}^{\,n}, \\ &= -(\mathbf{M}^{-1}\mathbf{N})^{n+1}\vec{\epsilon}^{\,0}, \end{aligned}$$

nuevamente obtenemos una expresión para la matriz de amplificación:

$$\mathbf{G} = -\mathbf{M}^{-1}\mathbf{N} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}, \tag{4.138}$$

la cual debe cumplir para un sistema estable (4.110).

Análisis de estabilidad de von Newmann y radio de convergencia

El análisis de estabilidad de John von Newmann desarrollado en la década de los 40's en el National Laboratory of Los Álamos-EUA, es una de las técnicas más populares para analizar el comportamiento y estabilidad de ecuaciones en diferencias. La idea básica del método es expandir la solución de la ecuación en diferencias en series de Fourier, por ejemplo para un sistema tridimensional:

$$u_{ijk}^n = \sum_{a=1}^{N_x-1}\sum_{b=1}^{N_y-1}\sum_{c=1}^{N_z-1} v_{abc}^n \sin(ax_i)\sin(by_j)\sin(cz_k), \tag{4.139}$$

con $u_{0,j,k}^n = u_{N_x,j,k}^n = u_{i,0,k}^n = u_{i,N_y,k}^n = u_{i,j,0}^n = u_{i,j,N_z}^n = 0$ y $x_i = i\pi\Delta x$, $y_j = j\pi\Delta y$ y $z_k = k\pi\Delta z$ en donde v_{abc}^n son los coeficientes de la serie sinusoidal de Fourier. De esta forma usando la identidad:

$$\sin(\pi a(i-1)\Delta x) - 2\sin(\pi ai\Delta x) + \sin(\pi a(i+1)\Delta x) = 2(\cos(\pi a\Delta x) - 1), \tag{4.140}$$

en la relación

$$(E_x - 2 + E_x^{-1})u_{ijk}^n = \sum_{a=1}^{N_x-1}\sum_{b=1}^{N_y-1}\sum_{c=1}^{N_z-1}\Big(v_{abc}^n \sin(by_j)\sin(cz_k)\big[\sin(\pi a(i-1)\Delta x) - 2\sin(\pi ai\Delta x) + \sin(\pi a(i+1)\Delta x)\big]\Big),$$

obtenemos:

$$(E_x - 2 + E_x^{-1})u_{ijk} = -\sum_{a=1}^{N_x-1}\sum_{b=1}^{N_y-1}\sum_{c=1}^{N_z-1} \alpha_a v_{abc}^n \sin(ax_i)\sin(by_j)\sin(cz_k), \tag{4.141}$$

en donde se ha definido $\alpha_a := 2[1 - \cos(\pi a\Delta x)]$ de forma análoga obtenemos una relación para la coordenada y:

$$(E_y - 2 + E_y^{-1})u_{ijk} = -\sum_{a=1}^{N_x-1}\sum_{b=1}^{N_y-1}\sum_{c=1}^{N_z-1} \beta_b v_{abc}^n \sin(ax_i)\sin(by_j)\sin(cz_k), \tag{4.142}$$

con $\beta_b := 2[1 - \cos(\pi b\Delta y)]$ y para z:

$$(E_z - 2 + E_z^{-1})u_{ijk} = -\sum_{a=1}^{N_x-1}\sum_{b=1}^{N_y-1}\sum_{c=1}^{N_z-1} \gamma_c v_{abc}^n \sin(ax_i)\sin(by_j)\sin(cz_k), \tag{4.143}$$

con $\gamma_c := 2[1 - \cos(\pi c\Delta z)]$, es claro que $0 \le \alpha_a \le 4$, $0 \le \beta_b \le 4$ y $0 \le \gamma_c \le 4$. Para la ecuación de Poisson tridimensional y el método iterativo de Jacobi (4.111) usando (4.139) con las identidades (4.141), (4.142) y (4.143) obtenemos:

$$v_{abc}^{n+1} = v_{abc}^n[1 - \frac{1}{6}(\alpha_a + \beta_b + \gamma_c)], \tag{4.144}$$

para garantizar estabilidad absoluta:

$$|v_{abc}^{n+1}| \le |v_{abc}^n|, \tag{4.145}$$

de modo que:

$$|1 - \frac{1}{6}(\alpha_a + \beta_b + \gamma_c)| \leq 1, \tag{4.146}$$

entonces $\omega_{abc} := 1 - \frac{1}{6}(\alpha_a + \beta_b + \gamma_c)$ alcanza su máximo para $\alpha_a = \beta_b = \gamma_c = 4$ entonces $\omega_{abc} = -1$. Por lo tanto $\forall a, b, c$ el método de Jacobi es estable.

Es posible obtener con estos resultados una expresión para el **radio de convergencia**, definido por:

$$z := \frac{\Delta u^{n+1}}{\Delta u^n}, \tag{4.147}$$

en donde Δu^n es la desviación respecto al estado estacionario o solución exacta del operador en diferencias finitas (4.106). Hay que notar que Δu^n cumple (4.111) por lo que repitiendo el análisis de estabilidad obtenemos una relación análoga a (4.144), de modo que sustituyendo en (4.147) obtenemos:

$$z_J = 1 - \frac{1}{6}(\alpha_a + \beta_b + \gamma_c), \tag{4.148}$$

los errores de mayor longitud de onda se dan para $a = b = c = 1$ tomando $\Delta x = \Delta y = \Delta z$, $\alpha_1 = \beta_1 = \gamma_1$ de modo que el radio de convergencia para los errores más persistentes esta dado por:

$$\begin{aligned} z_J^{max} &= \cos(\pi \Delta x), \\ &\approx 1 - \frac{\pi^2 \Delta x^2}{2}. \end{aligned}$$

Hay que notar que de (4.147) obtenemos que aplicando n veces:

$$\Delta u^n \simeq (z_J)^n \Delta u^0, \tag{4.149}$$

por otra parte, para que $\vec{\epsilon}^n$ disminuya en orden de magnitud 10^{-d} se debe cumplir:

$$\Delta u^n = 10^{-d} \Delta u^0, \tag{4.150}$$

entonces comparando (4.149) y (4.150) obtenemos:

$$\begin{aligned} (z_J)^n = &\quad 10^{-d} &\Rightarrow \\ n \log_{10} z_J = &\quad -d &\Rightarrow \\ -n\frac{\pi^2 \Delta x^2}{2} \approx &\quad -d, \end{aligned}$$

en donde se uso que $z_J \approx 1 - \frac{\pi^2 \Delta x^2}{2}$ y la identidad $\log_{10}(1 - x) \approx -x$. Despejando n de la última expresión obtenemos que se requieren $n \approx \frac{2d}{\pi^2 \Delta x^2}$ iteraciones para disminuir el error en un orden de magnitud 10^{-d}. Para un problema típico $\Delta x \sim 1/N$ con N el número de puntos en una dirección del dominio (estamos suponiendo que la partición es igual y homogénea en cada dirección) y d es aproximadamente el número de dígitos en la corrección, de modo que $n \sim O(N^2)$. El correspondiente trabajo computacional esta dado por:

$$W_J := z_J O(N^2) \approx O(N^4). \tag{4.151}$$

4.3.7. Métodos Multigrid

El método Multigrid es una forma eficiente de aprovechar el hecho de que los método iterativos convergen rápido para modos con longitudes de onda corta respecto al tamaño de la discretización. La idea básica es utilizar varias "resoluciones" de la malla: se utilizan mallas burdas para obtener la parte de la solución de longitud de onda más larga (la forma de la solución vista a gran escala) que respecto a particiones de baja resolución lucen como longitudes cortas y se usan mallas finas para obtener la parte de la solución de longitud de onda más corta (los detalles de la solución).

Para ilustrar el fenómeno de convergencia pensemos en el siguiente problema en una dimensión:

$$u''(x) = 0, \qquad x \in (-10,-1) \cup (1,10)\,, \tag{4.152}$$

$$u(x) = 1, \qquad x \in [-1,1]\,, \tag{4.153}$$

$$u(-10) = 0, \tag{4.154}$$

$$u(10) = 0, \tag{4.155}$$

la solución esta dada por la siguiente función definida en intervalos:

$$u(x) = \begin{cases} \frac{x+10}{9}, & x \in (-10,-1), \\ 1, & x \in [-1,1], \\ -\frac{x-10}{9}, & x \in (1,10). \end{cases} \tag{4.156}$$

Podemos hacer un experimento numérico usando el método de Gauss-Seidel usando dos mallas, la más fina con $n = 97$ puntos en la partición y una más burda con $N = 17$ puntos en la partición y como forma inicial de la solución $u^0(x) = 0$. En la figura 4.5 y 4.6 se muestran gráficas de las primeras iteraciones para cada resolución y se comparan con la solución analítica a la que deben converger.

En la resolución más baja no se puede representar correctamente la solución debido a que la parte con $x \in [-1,1]$, en donde la solución es constante, se tiene una longitud más pequeña que la distancia entre puntos en la malla en esa zona, sin embargo se observa que despues de pocas iteraciones la forma de la solución numérica no cambia. Se obtiene con esa resolución una forma global aproximada pero que en los detalles no coincide con la solución analítica. También es posible observar la propagación de la información: como forma inicial se colocan todos los valores en cero, en la primer iteración se observa que solo los puntos con $x > -1$ han sufrido una modificación, esto es debido a que, como ya se mencionó, en los métodos iterativos se realiza un promedio recorriendo la malla numérica en un cierto orden, en este caso de izquierda a derecha. La información se propaga poco a poco de derecha a izquierda en el semieje negativo a cada iteración.

Por otro lado, para la resolución más alta se observa que es posible definir bien la forma de la solución ya que hay puntos que cubren la región $[-1,1]$, sin embargo la información se propaga de forma más lenta, después de un número igual de iteraciones la solución numérica se sigue aproximando a la forma analítica.

De la descomposición en modos normales del error global es posible obtener más información sobre la convergencia de los métodos iterativos. Regresando a la forma vectorial de un operador elíptico lineal (4.120), el error global en la iteración n-esima se relaciona con la distribución inicial de error ϵ_j^0 por:

$$\vec{\epsilon}^{\,n} = \mathbf{G}^n \vec{\epsilon}^{\,0}, \tag{4.157}$$

en donde $\mathbf{G}$ es la matriz de amplificación dada por (4.138), si suponemos que $\mathbf{G}$ tiene un conjunto completo de valores propios λ_j y vectores propios $\vec{\Phi}_j$:

$$\mathbf{G}\vec{\Phi}_j = \lambda_j \vec{\Phi}_j, \tag{4.158}$$

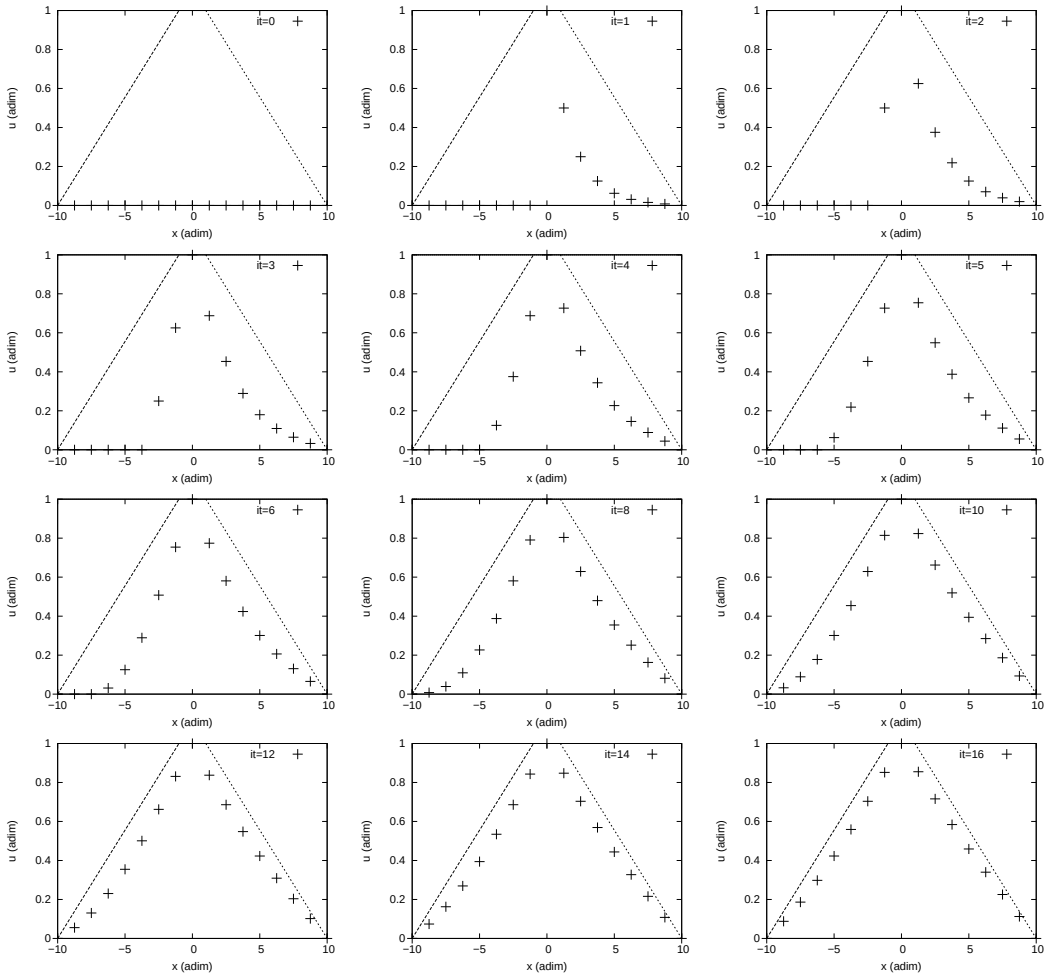

Figura 4.5: Evolución del método iterativo para una resolución baja. En este caso la representación de la solución no es precisa pero rápidamente se llega a un "estado estacionario", es decir, se obtiene la forma general de la solución pero con poca precisión.

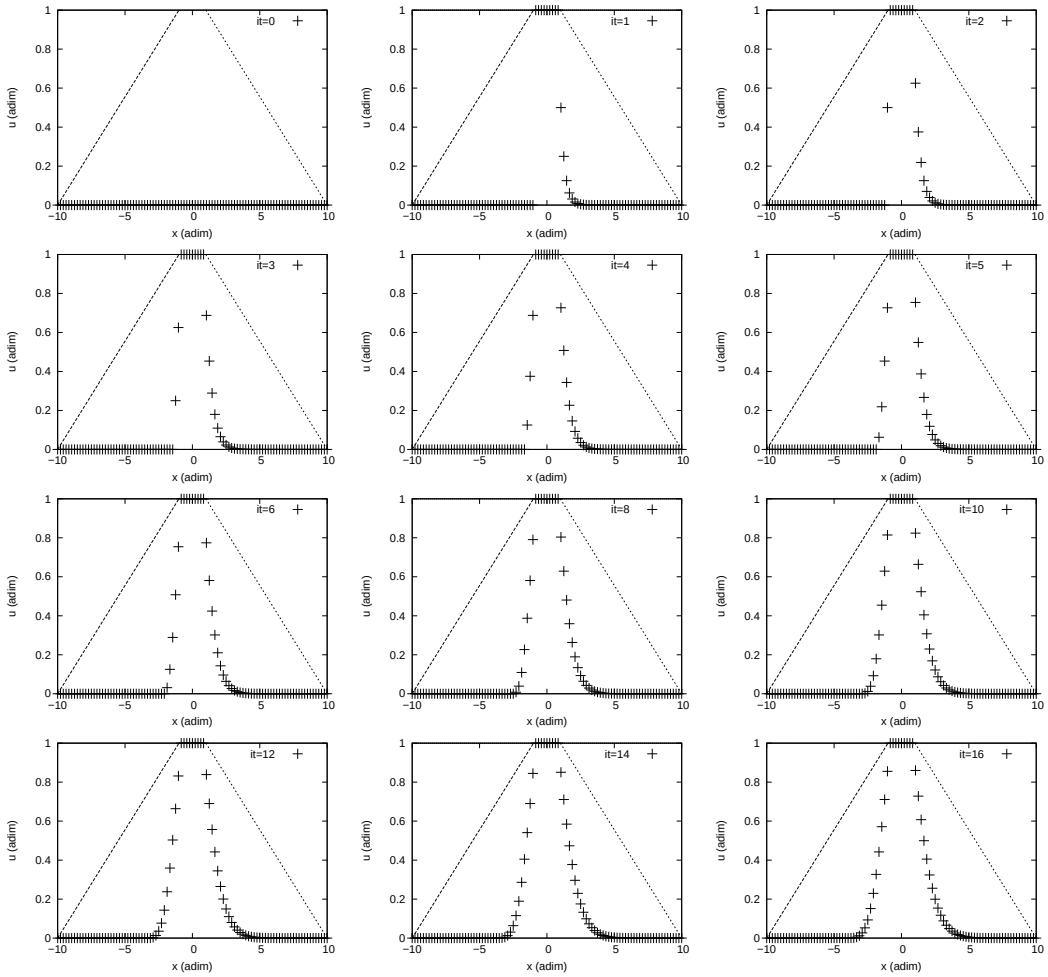

Figura 4.6: Evolución del método iterativo para una resolución alta. Para una resolución alta los detalles de la solución se pueden representar bien, sin embargo, la convergencia es muy lenta. Para el mismo número de iteraciones que en el caso de la figura 4.5 se observa que no se ha llegado a un "estado estacionario".

podemos expandir la distribución del error en término de esos vectores:

$$\vec{\epsilon}^{\,0} = \sum_j \epsilon_j^0 \vec{\Phi}_j, \tag{4.159}$$

en donde $\epsilon_j^0 = \vec{\Phi}_j \cdot \vec{\epsilon}^{\,0}$, son los coeficientes correspondientes en la expansión. Sustituyendo (4.159) en (4.157) y aplicando n veces la relación (4.158) obtenemos:

$$\vec{\epsilon}^{\,n} = \sum_j \lambda_j^n \epsilon_j^0 \vec{\Phi}_j, \tag{4.160}$$

recordando que si el método es convergente $\lambda_j < 1$ para cualquier j. En la figura 4.7 se muestra la gráfica de la sucesión de funciones λ_j^n en donde es claro que al aumentar n los valores de λ_j más grandes convergen más lento a 0.

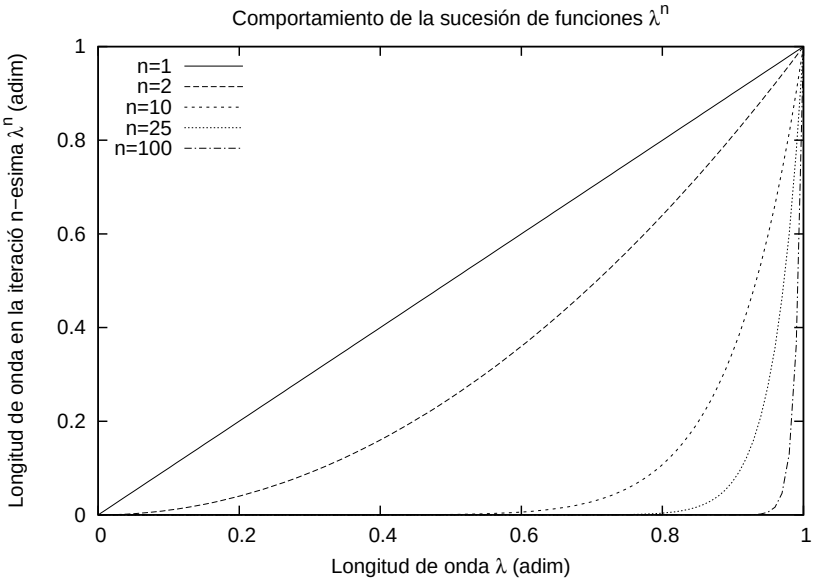

Figura 4.7: Gráfica de la sucesión de funciones λ^n. Se observa que las longitudes de onda más pequeñas decaen más rápido al aumentar n.

Regresando a la expresión (4.133), obtenemos que la descomposición del residuo esta dada por:

$$\vec{\tau}^{\,n} = \sum_j \lambda_j^n \epsilon_j^0 \mathbf{A} \vec{\Phi}_j, \tag{4.161}$$

en algunos casos $\mathbf{A}$ y $\mathbf{G}$ tienen los mismos vectores propios (por ejemplo para el método de Jacobi con condiciones de frontera periódicas). Si ese es el caso, obtenemos:

$$\vec{\tau}^{\,n} = \sum_j \lambda_j^n(\mathbf{G}) \epsilon_j^0 \lambda_j(\mathbf{A}) \vec{\Phi}_j, \tag{4.162}$$

en donde $\lambda_j^n(\mathbf{G})$ y $\lambda_j(\mathbf{A})$ son los valores propios asociados a $\mathbf{G}$ y $\mathbf{A}$ respectivamente. Si $\left\{\vec{\Phi}_j\right\}$ son vectores propios ortonormales, la norma L_2 de $\vec{\tau}^n$ esta dada por:

$$\|\vec{\tau}^{\,n}\| = \left[\sum_j \left(\lambda_j^n(\mathbf{G})\epsilon_j^0\lambda_j(\mathbf{A})\right)^2\right]^{\frac{1}{2}}, \tag{4.163}$$

la norma del residuo se utiliza comúnmente para medir la convergencia del método. En la figura 4.8 se muestra una gráfica con el comportamiento típico del residuo como función del número de iteraciones. De la gráfica de la sucesión de longitudes de onda (figura 4.7), sabemos que los modos con longitud de onda más grande tardan más en ser eliminados, por lo que las componentes del residuo que se eliminan primero son las de longitud de onda corta y las que tardan más (en la figura 4.8 la cola exponencial) son de longitud de onda larga.

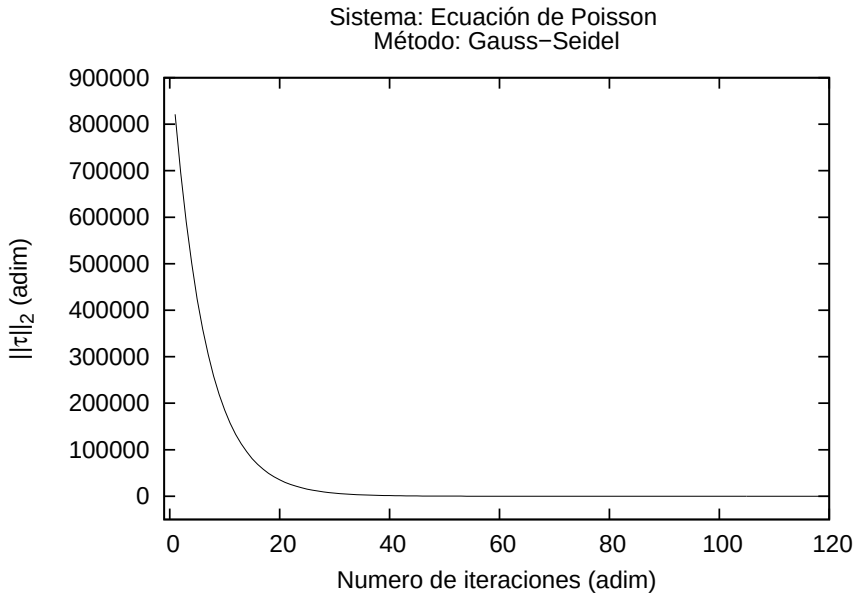

Figura 4.8: Gráfica de la norma L_2 del residuo como función del número de iteraciones.

Existen al menos dos estrategias para implementar el método Multigrid:

Transferencia de la solución

La idea es resolver el problema en la malla más burda para obtener una forma inicial global, transferir la solución a la malla más fina usándola como forma inicial y resolver en la malla más fina para obtener los detalles de la solución, este es el algoritmo que se ha implementado en Olliptic. El algoritmo esta dado por el siguiente procedimiento:

1. Relajar $\mathcal{L}^H u^H = \rho^H$ en la malla burda para obtener una solución inicial.
2. Transferir la solución u^H a la malla fina para tener una condición inicial u_0^h.

3. Relajar $\mathcal{L}^h u^h = \rho^h$ en la malla fina usando la condición inicial u_0^h.

4. Probar la convergencia, si es necesario regresar al primer paso.

Transferencia del residuo

En este caso el procedimiento consiste en transferir el residuo para obtener una corrección a la solución, este algoritmo es un poco más complicado, a continuación se describen los pasos a realizar:

1. En la malla más fina relajar $\mathcal{L}^h u^h = \rho^h$.

2. Transferir el residuo $r^h := \mathcal{L}^h u^h - \rho^h$ a la malla burda para obtener r^H.

3. Obtener la corrección Δu^H resolviendo $\mathcal{L}^H(\Delta u^H) = r^H$ a un bajo costo computacional.

4. Transferir la corrección a la malla más fina $u^h \rightarrow u^h + \Delta u^h$.

5. Probar la convergencia, si es necesario regresar al primer paso.

Es común realizar "ciclos V" los cuales consisten en iniciar con una malla fina M^p, transferir la solución o el residuo a mallas mas burdas hasta un nivel M^0, resolver el problema con una tolerancia en el residuo inferior a la precisión del orden del método, transferir la solución a las mallas más finas M^q y resolver en la malla más fina el problema. Una variante es utilizar ciclos "W" que consisten en realizar varios ciclos V posiblemente con diferentes niveles de mallas.

Para implementar cualquiera de los algoritmos es necesario definir el operador de restricción $\mathbf{R}_h^H$ que transfiere la solución o el residuo de una malla de resolución más grande a una más burda y el operador de prolongación $\mathbf{P}_H^h$ que transfiere la solución o el residuo de una malla burda a una más fina.

El operador de restricción en el código Olliptic que se ha implementado esta dado por:

$$\mathbf{R}_h^H u_{i,j,k}^h \longrightarrow u_{2i,2j,2k}^H, \tag{4.164}$$

en donde $u_{i,j,k}^h$ denota una función en la malla fina y $u_{I,J,K}^H$ denota una función en la malla burda. Este tipo de operador se conoce como operador de inyección. Así mismo, el operador de prolongación definido en Olliptic realiza una interpolación lineal de la malla más burda a la más fina. Cada transferencia entre mallas genera errores que se conocen en el área de procesamiento de señales como aliasing de la solución. Estos errores son de longitud de onda muy corta y son aniquilados muy rápidamente. Aún cuando se utiliza interpolación lineal para la transferencia entre malla, la solución final preserva el orden del método ya que la forma generada por la transferencia únicamente es utilizada como forma inicial para realizar el siguiente proceso de iteración en una malla más fina. Utilizar una interpolación de mayor orden puede ser necesario por ejemplo para métodos de mallas adaptativas, en particular si existen zonas con cambios bruscos de resolución. En la figura 4.9 se muestra la comparación entre el comportamiento de la norma L_2 del residuo entre el método Gauss-Seidel y el Multigrid. En la gráfica de la izquierda se observa que al transferir la solución de una malla a otra se generan modos de longitud de onda corta, que en la gráfica aparecen como picos que se desvanecen luego de algunas iteraciones. Por otro lado en la gráfica de la derecha se observa, con un rango más estrecho de valores, que la solución transferida a la malla más fina converge más rápido.

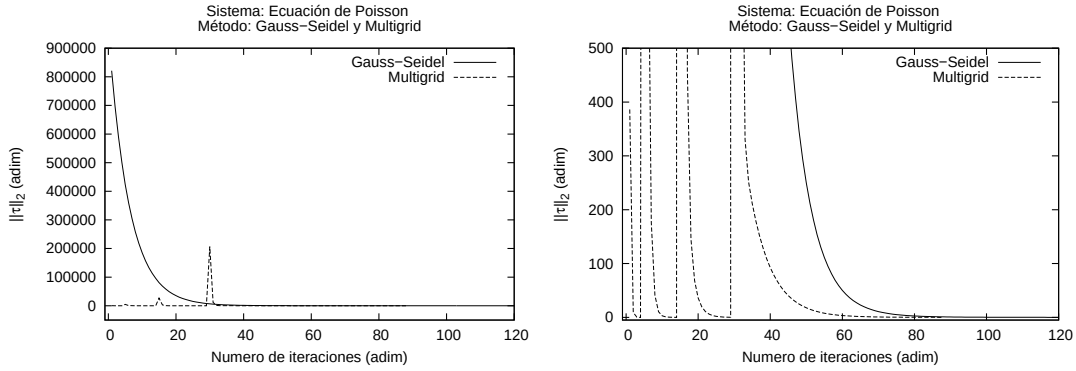

Figura 4.9: Se presenta la gráfica de la norma L_2 del residuo, como función del número de iteraciones comparando el método de Gauss-Seidel y el método Multigrid para el mismo sistema. En el método Multigrid se presentan 4 niveles de mallas las cuales se distinguen debido a que en cada transferencia se generan modos de longitud de onda corta que son rápidamente aniquilados y que tienen una amplitud que depende de la resolución. En la segunda gráfica se muestra con un rango más restringido, como se llega al mismo nivel de convergencia más rápido.

Por último hay que mencionar que para la implementación del método la jerarquía de submallas más simple es tomando relaciones 2:1 en el tamaño de la partición lo cual implica que el número de puntos en cada submalla cumple:

$$N^{n+1} = 2N^n + 1, \tag{4.165}$$

en donde N^n denota el numero de puntos en la partición en la submalla n. Dada una malla base n^0, se pueden generar con la formula (4.165) una familia de mallas con las cuales se puede implementar el método Multigrid. Para el código Olliptic se han usado como base los siguientes valores de n^0 : 3, 4, 6, 8, 10, 12 y 14.

La principal ventaja del método Multigrid es la eficiencia, para estimar el trabajo computacional W^m que requiere el método Multigrid en un ciclo V, podemos suponer que empleamos m sub-mallas y que el ciclo V inicia y termina en la misma malla (la más fina), de modo que se vista cada malla dos veces, si suponemos que se realizan σ ciclos de relajación por malla (usualmente son entre 4 y 10 ciclos), con submallas con relación 2:1 entre cada nivel, a un costo computacional δw^l el trabajo total será:

$$W^m \approx 2\Big(\Big(\frac{\sigma}{2}\Big)^d + W^{m-1}\Big)\delta w^m, \tag{4.166}$$

procediendo de forma recursiva:

$$W^m \approx 2\left[1 + \Big(\frac{\sigma}{2}\Big)^d + \Big(\frac{\sigma}{2}\Big)^{2d} + \cdots + \Big(\frac{\sigma}{2}\Big)^{(m-1)d}\right]\delta w^m = \frac{1 - \big(\frac{\sigma}{2}\big)^{md}}{1 - \big(\frac{\sigma}{2}\big)^d}\delta w^m, \tag{4.167}$$

en donde d denota la dimensión del espacio (en este caso 3). El costo computacional de relajar cada malla es proporcional al número de puntos en la partición, es decir $\delta w^m \sim \mathcal{O}(N)$ de modo que el trabajo computacional será del orden de puntos en la partición. Como hemos visto la

Método	Directo / Iterativo	Trabajo comp.	Almacenamiento
Multigrid	Iterativo	$\mathcal{O}(N)$	$\mathcal{O}(N)$
SSOR (Chebyshev)	Iterativo	$\mathcal{O}(N^{5/4})$	$\mathcal{O}(N)$
SOR	Iterativo	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N)$
Gradiente Conjugado	Iterativo	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N)$
Gauss-Seidel	Iterativo	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$
Jacobi	Iterativo	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$
Eliminación Gauss (MS)	Directo	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N \log N)$
Eliminación Gauss (MD)	Directo	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$

Tabla 4.1: Características de algunos métodos de solución de sistemas algebraicos. El almacenamiento se refiere al tamaño de los arreglos utilizados para implementar el método. La estimación del método Multigrid se hace suponiendo que se utiliza como método de relajación el método de Gauss-Seidel, si se usa un método SOR puede aumentar aún más la eficiencia. MS se refiere a un sistema con matriz simétrica y MD se refiere a un sistema con matriz densa.

solución de ecuaciones elípticas usando métodos en diferencias finitas se reduce a resolver sistemas de ecuaciones algebraicas.

En la tabla 4.1 se hace un resumen de las características de los principales métodos para solución de sistemas de ecuaciones, algunos de ellos no se han mencionado aquí pero los detalles se pueden encontrar en las referencias [35, 36]. Como se puede observar el método Multigrid es el más eficiente y utilizando métodos más rápidos pare efectuar la relajación se puede aumentar aún más la eficiencia, en el caso del método implementado en Olliptic se ha usado como método de relajación el de Gauss-Seidel pero utilizando por ejemplo un método SOR se puede llega a tener eficiencia logarítmica. De modo que resolviendo los sistemas de ecuaciones algebraicos con el método Multigrid es posible obtener de forma muy eficiente soluciones a ecuaciones diferenciales parciales de tipo elíptico.

4.3.8. Métodos espectrales

Dentro de la gama de métodos utilizados para obtener soluciones numéricas aproximadas a ecuaciones diferenciales, existen al menos dos tipos de métodos, que se diferencian por el tipo de discretización en el cual se basan. Los métodos en diferencias finitas que hemos estudiado en los capítulos anteriores, se basan en la discretización de los operadores. Los métodos espectrales que se estudiarán en esta sección se basan en la discretización de la solución.

Los método espectrales consiste en tomar una función de prueba u_N la cual es una serie truncada con una base dada y con coeficientes desconocidos:

$$u(\vec{x}) \approx u_n(\vec{x}) := \sum_{i=0}^{n} a_i \phi_i(\vec{x}), \tag{4.168}$$

en donde tenemos la libertad de tomar una base *conveniente* $\phi_i(\vec{x})$ así como el número n de elementos en la serie. Con esta forma de la solución obtenemos una función error:

$$R(\vec{x}; a_0, \ldots, a_n) := \mathcal{L}u_n - f(\vec{x}), \tag{4.169}$$

en donde $\mathcal{L}$ es un operador diferencial que define a nuestra ecuación. Deseamos escoger funciones de prueba w_i tales que minimicen a la función error:

$$(w_i, R(\vec{x}; a_0, \ldots, a_n)) := \int_\Omega \omega(\vec{x}) w_i(\vec{x}) R(\vec{x}; a_0, \ldots, a_n) \mathrm{d}V = 0, \tag{4.170}$$

en donde ω es una función de peso para el producto. Dependiendo de la función de prueba utilizada, el método se clasifica de la siguiente forma:

- Método seudoespectral: $w_i(\vec{x}) := \delta(\vec{x} - \vec{x}_i)$, en donde δ es la distribución delta de Dirac y $\vec{x}_i$ son puntos de colocación para hacer interpolación (ver más adelante). En este caso se obtiene un sistema de ecuaciones para R:

$$R(\vec{x}_i; a_0, \ldots, a_n) = 0, \qquad i \in 1, \cdots, N, \tag{4.171}$$

  este es el método más simple y será el que se tratará en el resto de la sección.

- Método de momentos: $w_i(\vec{x}) := \|\vec{x}\|^i$, este método solo es práctico para problemas con pocos términos en la expansión.

- Método de mínimos cuadrados: $w_i(\vec{x}) := \mathcal{L}\phi_i(\vec{x})$, este método solo se aplica a operadores lineales, para operadores no lineales el sistema de ecuaciones a resolver es muy complicado.

- Método de Galerkin: $w_i(\vec{x}) := \phi_i(\vec{x})$, si se usan funciones base lineales a trozos para hacer la expansión de la solución se obtiene el método de elementos finitos.

Tomando como ejemplo el método más simple, es decir el seudo-espectral, debemos tomar $n+1$ puntos *convenientes* $\{x_0, \ldots, x_n\}$ en nuestro dominio, obtenemos un sistema de ecuaciones. La solución de este sistema nos da la mejor aproximación a la solución con una expansión en serie de orden n con la base seleccionada.

La base más *conveniente* depende del problema: Para problemas con condiciones de frontera periódicas la mejor opción son series de Fourier, si el dominio es acotado entonces los polinomios de Chebyshev son la elección natural, y si el dominio es todo el espacio, entonces las funciones de Chebyshev son la mejor selección.

Los puntos más convenientes en el dominio son aquellos en los que las derivadas de la base se anulan, estos puntos se conocen como **puntos de interpolación**.

El objetivo del método es trasformar la ecuación diferencial en un problema algebraico con lo cual solo necesitamos un resolvedor de ecuaciones lineales para obtener la solución.

Como ejemplo se tomará un sistema electromagnético bidimensional consistente en una región cuadrada interior a un potencial 1 y una región cuadrada externa a un potencial 0 (ver figura 4.10).

La ecuación bidimensional de Poisson:

$$\partial_{xx} u(x, y) + \partial_{yy} u(x, y) = \rho(x, y), \tag{4.172}$$

describe la solución al problema y como se tienen condiciones de frontera en un dominio finito se deben usar polinomios de Chebyshev. En 2 dimensiones la función de prueba toma la forma:

$$u_n(x, y) := \sum_{i=0}^{n} \sum_{j=0}^{n} a_{ij} T_i(x) T_j(y), \tag{4.173}$$

donde $T_i(\cos\theta) := \cos(i\theta)$. Este sistema tiene dos tipos de ecuaciones en el dominio Ω:

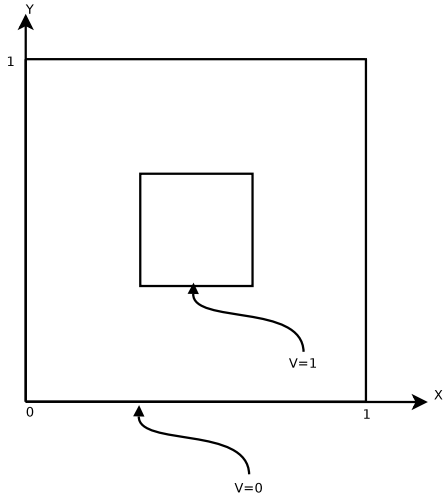

Figura 4.10: Problema electrodinámico de ejemplo. Se presenta el diagrama de una placa cargada con un potencial 1 en su interior y con un marco exterior a un potencial 0

1. Ecuaciones de frontera. Estas ecuaciones son la función prueba evaluada en los puntos de interpolación e igualada a 1 o 0.

   Condición de frontera externa:
   $$u_n(x_i, y_j) = 0, \tag{4.174}$$
   en donde $(x_i, y_j) \in \partial\Omega_{ext}$, Condición de frontera interna:
   $$u_n(x_i, y_j) = 1, \tag{4.175}$$
   donde $(x_i, y_j) \in \partial\Omega_{int}$

2. Ecuaciones internas. Estas ecuaciones son la función de error evaluada en los puntos de interpolación e igualada a 0
   $$R(x_i, y_j, a_{00}, a_{01}, \ldots, a_{nn}) = 0. \tag{4.176}$$

Nuevamente el problema se reduce a obtener la solución de un sistema de n^m ecuaciones con n^m incógnitas y aunque se obtiene una mayor precisión aún es poco eficiente tratar de obtener la solución del problema de esta forma. Sin embargo, utilizando una combinación entre métodos seudo-espectrales para formular el sistema de ecuaciones a resolver y métodos iterativos para resolver el problema algebraico es posible obtener un método eficiente y muy preciso.

4.4. Condiciones de frontera

Como se ha mencionado, las ecuaciones elípticas conforman un problema de valores a la frontera por lo que hay que especificar condiciones de frontera adecuadas al problema que deseamos describir. Para ecuaciones diferenciales parciales de segundo orden existen 3 tipos de condiciones de frontera:

- Condición tipo **Dirichlet**: En este tipo de condición de frontera se especifica el valor de la solución en la frontera.

$$u(\vec{x}) = B_D(\vec{x}), \qquad \vec{x} \in \partial\Omega, \tag{4.177}$$

  en donde $B_D : \partial\Omega \to \mathbb{R}$ es una función suave y Ω es el dominio de la solución.[11]

- Condición tipo **Newmann**: Similar al caso anterior pero se especifica el valor de la primer derivada de la solución:

$$\frac{\partial u(\vec{x})}{\partial x^i} = B_N(\vec{x}), \qquad \vec{x} \in \partial\Omega, \tag{4.178}$$

  de igual manera que el caso anterior $B_N : \partial\Omega \to \mathbb{R}$ es una función arbitraria pero suave.

- Condición tipo **Robin**: Se da la condición usando una combinación lineal de las anteriores:

$$a(\vec{x})u(\vec{x}) + b(\vec{x})\frac{\partial u(\vec{x})}{\partial x^i} = B_R(\vec{x}), \qquad \vec{x} \in \partial\Omega, \tag{4.179}$$

  en este caso $a : \partial\Omega \to \mathbb{R}$, $b : \partial\Omega \to \mathbb{R}$ y $B_I : \partial\Omega \to \mathbb{R}$ son funciones arbitrarias suaves.

En el código se han implementado las condiciones que se requieren para los problemas y ejemplos, sin embargo como se mostrará más adelantes es fácil añadir nuevas condiciones de frontera. Para los problemas de prueba de las ecuaciones se han implementado condiciones tipo Dirichlet con una función constante: $B_D(\vec{x}) = B_0$. Para las pruebas físicas se han implementado dos tipo de condiciones de frontera. Tipo Dirichlet para el sistema Scrödinger-Poisson:

$$B_D(\vec{x}) = -\frac{M}{r}, \qquad r = \|\vec{x}\|, \tag{4.180}$$

en donde M es la masa asociada al campo escalar (ver sección 6.2). Este tipo de condición de frontera representa el término monopolar para el campo gravitacional clásico.

Para ondas de Brill se implementó el siguiente tipo de condición de frontera:

$$u(\vec{x}) = A + \frac{B}{r^n}, \qquad r = \|\vec{x}\|, \tag{4.181}$$

en donde típicamente se toma: $A = 1$ y $n = 1$. Esta condición es tipo Dirichlet y en general el coeficiente B es desconocido. Sin embargo, se puede transformar esta condición en una tipo Robin que no depende de B con el siguiente procedimiento:

- Se obtiene la derivada de (4.181) y se despeja B:

$$B = -\frac{r^{n+1}}{n}\frac{\partial u(\vec{x})}{\partial r}. \tag{4.182}$$

- Sustituyendo (4.182) en (4.181) obtenemos una condición tipo Robin que no contiene B:

$$u(\vec{x}) + \frac{r}{n}\frac{\partial u(\vec{x})}{\partial r} = A, \tag{4.183}$$

  en donde $a(\vec{x}) = 1$, $b(\vec{x}) = \frac{r}{n}$ y $B_R(\vec{x}) = A$.

[11] Como antes $\partial\Omega$ denota la frontera de Ω

se ha observado experimentalmente que la forma más conveniente de implementar numéricamente esta condición en un dominio cúbico es tomando derivadas perpendiculares en las caras del dominio. Por ejemplo para la coordenada cartesiana x obtenemos de la identidad:

$$\begin{aligned}\frac{\partial u}{\partial x} &= \frac{\partial r}{\partial x}\frac{\partial u}{\partial r}, \\ &= \frac{x}{r}\frac{\partial u}{\partial r},\end{aligned}$$

en donde se han despreciado los términos angulares, despejando la derivada de (4.183) y sustituyendo en la expresión anterior:

$$\frac{\partial u}{\partial x} = \frac{nx}{r^2}(A-u), \tag{4.184}$$

para discretizar esta ecuación podemos usar por ejemplo, para la cara derecha (ver figura 4.11) el operador (4.73), obtenemos:

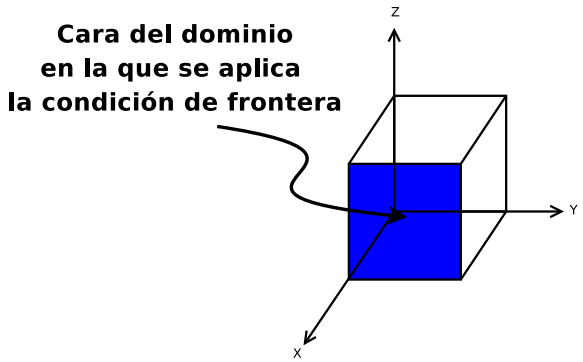

Figura 4.11: Para un dominio cubico se aplica una condición tipo Robin en la cara mostrada usando la formula (4.187)

$$\frac{1}{\Delta x}\left(\delta_x^- + \frac{(\delta_x^-)^2}{2} + \frac{(\delta_x^-)^3}{3} + \frac{(\delta_x^-)^4}{4} + \cdots\right) u_{ijk} = \frac{nx}{r^2}(A - u_{ijk}), \tag{4.185}$$

en donde la serie se trunca dependiendo del orden de aproximación del método utilizado. Por ejemplo para los métodos de segundo orden obtenemos de forma explícita:

$$\frac{1}{2\Delta x}\left(3u_{ijk} - 4u_{i-1,jk} + u_{i-2,jk}\right) = \frac{nx}{r^2}(A - u_{ijk}), \tag{4.186}$$

de aquí se despeja el valor incógnita u_{ijk}:

$$u_{ijk} = \left[\frac{nx}{r^2}A + \left(\frac{4E_x^{-1} - E_x^{-2}}{2\Delta x}\right)u_{ijk}\right]\left[\frac{nx}{r^2} + \frac{3}{2\Delta x}\right]^{-1}, \tag{4.187}$$

esta última expresión es la que se ha codificado en el código. Para las aristas y esquinas del dominio se procede de forma análoga, definiendo derivadas direccionales. Por ejemplo para la arista $X-Y$

se define $\rho := \sqrt{x^2+y^2}$ y se calcula:

$$\begin{aligned}\frac{\partial u}{\partial \rho} &= \frac{\partial r}{\partial \rho}\frac{\partial u}{\partial r},\\ &= \frac{\rho}{r}\frac{\partial u}{\partial r},\end{aligned}$$

y usando que la identidad:

$$\begin{aligned}\frac{\partial u}{\partial \rho} &= \frac{\partial x}{\partial \rho}\frac{\partial u}{\partial x}+\frac{\partial y}{\partial \rho}\frac{\partial u}{\partial y},\\ &= -\frac{\rho}{x}\frac{\partial u}{\partial x}-\frac{\rho}{y}\frac{\partial u}{\partial y},\end{aligned}$$

obtenemos la expresión:

$$\frac{1}{x}\frac{\partial u}{\partial x}+\frac{1}{y}\frac{\partial u}{\partial y}=\frac{n(A-u)}{r^2}, \tag{4.188}$$

en esta expresión sustituimos los operadores de derivada adecuados, por ejemplo para la arista $X-Y$ mostrada en la figura 4.12 obtenemos la formula:

$$u_{ijk}=\left[\frac{n}{r^2}A+\left(\frac{4E_x^{-1}-E_x^{-2}}{2x\Delta x}+\frac{4E_y^{-1}-E_y^{-2}}{2y\Delta y}\right)u_{ijk}\right]\left[\frac{n}{r^2}+\frac{3}{2x\Delta x}+\frac{3}{2y\Delta y}\right]^{-1}, \tag{4.189}$$

este tipo de condición de frontera es la implementada en el código. Por último para las esquinas se usa la identidad:

$$\begin{aligned}\frac{\partial u}{\partial r} &= \frac{\partial x}{\partial r}\frac{\partial u}{\partial x}+\frac{\partial y}{\partial r}\frac{\partial u}{\partial y}+\frac{\partial z}{\partial r}\frac{\partial u}{\partial z},\\ &= \frac{r}{x}\frac{\partial u}{\partial x}+\frac{r}{y}\frac{\partial u}{\partial y}+\frac{r}{z}\frac{\partial u}{\partial z},\end{aligned}$$

obtenemos la relación:

$$\frac{1}{x}\frac{\partial u}{\partial x}+\frac{1}{y}\frac{\partial u}{\partial y}+\frac{1}{z}\frac{\partial u}{\partial z}=\frac{n}{r^2}(A-u). \tag{4.190}$$

Como ejemplo aplicamos los operadores correspondientes a la esquina mostrada en la figura 4.13, obtenemos la formula:

$$u_{ijk}=\frac{\frac{n}{r^2}A+\left(\frac{4E_x^{-1}-E_x^{-2}}{2x\Delta x}+\frac{4E_y^{-1}-E_y^{-2}}{2y\Delta y}+\frac{4E_z^{-1}-E_z^{-2}}{2z\Delta z}\right)u_{ijk}}{\frac{n}{r^2}+\frac{3}{2x\Delta x}+\frac{3}{2y\Delta y}+\frac{3}{2z\Delta z}}, \tag{4.191}$$

como se observa, aplicar la condición de frontera requiere de dividir la misma en tres tipo de regiones: Caras (6), aristas (12) y esquinas (8), en el código se implementa la aplicación de cada condición de frontera en *métodos* separados.[12]

Las mallas utilizadas por CACTUS se definen para una partición fija, sin embargo, el método Multigrid como ya se ha visto, requiere de mallas con distintas resoluciones. Debido a esto el código Olliptic se ha paralelizado directamente usando la biblioteca MPI. Al compilar el código en paralelo, es necesario hacer una partición del dominio de tal forma que cada procesador resuelve un sub-problema del original. Es por ello que se deben definir condiciones de frontera entre los dominios asignados a los procesadores. Este tipo de condiciones de frontera no son físicas, por lo que los detalles al respecto se encuentran en el Apéndice A

[12]Para detalles consultar la sección 5.2

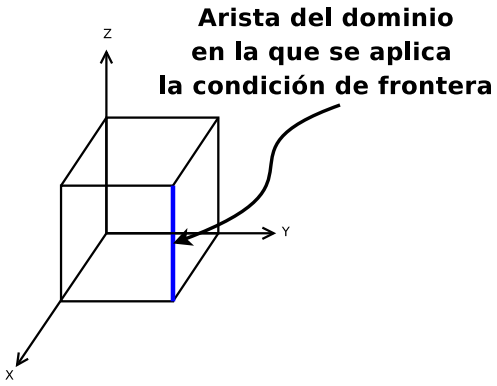

Figura 4.12: Para un dominio cubico se aplica una condición tipo Robin en la arista indicada en la figura usando la formula (4.189)

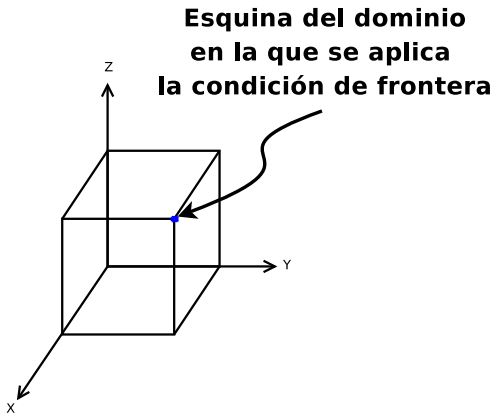

Figura 4.13: Para un dominio cubico se aplica una condición tipo Robin en la arista indicada en la figura usando la formula (4.191)

Capítulo 5

Código Olliptic

El código desarrollado esta diseñado para ser fácilmente modificado y poder incorporar mejoras a los métodos existentes (como por ejemplo aumentar el orden de aproximación usado en los métodos o el tipo de ecuaciones a resolver), así como nuevos métodos de solución. Para la primer versión del código se implementaron los métodos más eficientes y sencillos, pero en esta tesis se deja abierta la posibilidad de incorporar los más precisos o algunos híbridos.

El mayor trabajo al momento de implementar una nueva funcionalidad en un código computacional es la parte de depuración del código que consiste en asegurarse que el código realice las tareas para las cuales fue diseñado y que contenga el mínimo de errores. Por la complejidad de los sistemas es imposible tener un código 100 % libre de errores, sin embargo como ya se mencionó para un código numérico hay pruebas básicas que garantizan la obtención de una solución aproximada al problema planteado.

En este capítulo se da una revisión de las características del código, se plantean dos sistemas de prueba con solución analítica y se presentan los resultados correspondientes. Se observa que el código funciona bien para los casos de prueba obteniendo convergencia a segundo orden. El rendimiento de Olliptic es el esperado para los dos métodos implementados, el tiempo de convergencia escala de forma lineal con el número total de puntos en la partición para el método Multigrid y de forma cuadrática para el método de Gauss-Seidel. Además se observa una buena implementación de las rutinas de paralelización.

5.1. Código CACTUS

En esta sección se darán algunos conceptos necesarios para comprender el funcionamiento del código Olliptic, los detalles sobre el código CACTUS se pueden encontrar en [1, 2]. CACTUS fue desarrollado como un código de relatividad numérica y aún cuando tiene aplicaciones en otras áreas la discusión se centrará sobre las aplicaciones en Relatividad Numérica. La idea básica del código CACTUS es proveer de una infraestructura para que físicos, puedan desarrollar módulos que desempeñan alguna función especifica dentro de una simulación numérica. Las simulaciones en físicas son experimentos numéricos, si hacemos una analogía con experimentos físicos reales, CACTUS es un "dispositivo" al que se le pueden "conectar" detectores, generadores de datos, controladores y muchos otros dispositivos. Todos esos dispositivos son por supuesto códigos computacionales, programas que realizan una función especifica y para los cuales CACTUS gestiona la intercomunicación. A los módulos de CACTUS se les conoce como espinas, que a su vez están agrupadas en

arreglos. Una de las ventajas de CACTUS es que se pueden programar las espinas en diferentes lenguajes de programación como: C, C++ y Fortran y se obtiene un binario consistente.

Olliptic requiere para su funcionamiento las espinas de los siguientes arreglos:

- `CactusBase`. Provee funciones básicas como gestión de coordenadas, condiciones de frontera y de entrada y salida de datos.
- `CactusPUGH`. Provee de un controlador en paralelo de mallas homogéneas, es decir se encarga de hacer particiones del dominio a través de los procesadores (en caso de usar varios) y además se encarga de la partición de los subdominios en mallas homogéneas y de gestionar la comunicación entre los procesadores.
- `CactusUtils`. Provee de dispositivos para monitorear que no se presenten divergencias en los datos y "cronómetros".

El código Olliptic por el momento tiene pocas funciones implementadas y no puede interactuar completamente con otras espinas de CACTUS, sin embargo en futuras versiones se realizarán mejoras para aumentar su versatilidad, en particular se implementará una interfase para la espina EllBase de CACTUS. EllBase permite utilizar diversos resolvedores elípticos dentro de las espinas de CACTUS y de esa forma abstraer la solución de las ecuaciones en una interfase común.

5.2. Detalles del código

El código Olliptic como ya se mencionó, se programó usando el lenguaje C++ y un paradigma de programación conocido como programación orientada a objetos (en adelante POO). La diferencia básica entre la programación estructurada y la POO es que en el primer caso en los programas se definen tipos de datos (enteros, de punto flotante o caracteres), y funciones que actúan sobre los datos. Por el contrario en la POO se definen clases de objetos que consisten en encapsular un conjunto de datos y funciones en un solo ente. Las funciones se conocen como **metodos** y se encargan de definir los valores que adquieren los datos miembro del objeto y también permiten interactuar con otros miembros. La ventaja que tiene la POO es que una vez definido un objeto es posible abstraerse de la implementación del mismo y con solo conocer los métodos es posible utilizar sus funciones.

El código Olliptic implementa solo 2 clases de objetos: `GridOBJ` que se encarga de definir funciones en submallas a partir de la jerarquía de mallas hecha por CACTUS y `EllOBJ` que se encarga de definir el operador diferencial elíptico.

La clase `GridOBJ` contiene los siguientes métodos:

- `GridObj(const cGH *GH)` el constructor por defecto el cual toma como argumento un apuntador a la jerarquía de mallas de CACTUS `cctkGH` y del cual hereda toda la información sobre el dominio.
- `CCTK_REAL dd[xyz][fcb]2(const int i, const int j, const int k)` la segunda derivada a 2do orden en donde [xyz] denota alguna de las componentes y [fcb] denota la deriva lateral izquierda, centrada o lateral derecha respectivamente. A este método se le proporcionan los índices de la partición.
- `int Get_GN[xyz]()` el número de puntos en la partición global definida por CACTUS.

- `int Get_LN[xyz]()` el número de puntos en la partición local a cada procesador definida por CACTUS.
- `int Get_n[xyz]()` el número de puntos en la partición de la submalla definida para el método Multigrid.
- `int Get_H[xyz]()` el tamaño de la partición global definida por CACTUS.
- `int Get_h[xyz]()` el tamaño de la partición de la submalla definida para el método Multigrid.

Además se definen los siguientes operadores, para dos objetos de la clase `GridObj A` y `B`:

- `A=B`. Copia todas las propiedades del objeto `B` al objeto `A`
- `A-=B`. Para un objeto `B` con N puntos en la partición, define un objeto `A` sobre una submalla con $n = (N + 1)/2$ puntos y transfiere los valores de `B`.
- `A+=B`. Para un objeto `B` con n puntos en la partición, define un objeto `A` sobre una malla más grande con $N = 2n - 1$ puntos y se hace una interpolación lineal los valores de `B`.
- `A+B`. Suma los valores de dos objetos de la clase `GridObj` definidos sobre particiones iguales.
- `A-B`. Resta los valores de dos objetos de la clase `GridObj` definidos sobre particiones iguales.
- `A(i,j,k)`. define los valores de `A`.

En la práctica para resolver una ecuación ya implementada solo es necesario utilizar el constructor.

Los métodos de `EllObj` son los siguientes:

- `EllObj(const cGH *GH)` el constructor por defecto el cual, al igual que la clase `GridOBJ`, toma como argumento un apuntador a la jerarquía de mallas de CACTUS `cctkGH` y del cual hereda toda la información sobre el dominio.
- `void Boundary(const int b = DIRICHLET, const double coeff = 0.0)` define el tipo de condición de frontera, b puede tomar los valores: `DIRICHLET` (que es el valor por defecto), `DIRICHLET-POISSON` y `ROBIN`, según sea el caso y `coeff` un valor flotante de doble precisión, el valor por defecto es 0.0.
- `void Define_Method(const int m = MULTIGRID, const int s = 1, double e = 0.001, double e0 = 0.001,double si = 0.001)` en donde se define el método a utilizar, m puede tomar los valores `MULTIGRID` y `GAUSS_SEIDEL`; `s` define el número de submallas usadas en el método Multigrid, el valor por defecto es 1; `e` define la tolerancia de la solución final en donde se toma $\|r^h\|_2 < e$ con r^h el residuo de aplicar el operador elíptico a la solución; `e0` es la tolerancia para el nivel más burdo de las submallas y se define de manera análoga; `si` es la medida de relajación dada por $\|r^{h'}\|_\infty < si$ y se aplica solo para mallas con resolución intermedia ente el nivel más fino y el más burdo.
- `void Define_Equation(const int t = POISSON)` define la ecuación a resolver, t puede tomar los valores `POISSON` para $\nabla^2 u = S$, `LINEAR` para $\nabla^2 u + Au = S$ y `SEMILINEAR` para $\nabla^2 u + Au + Bu^5 + C/(D + Eu)^7 = S$

- `CCTK_REAL & Define_[ABCDE](const int i,const int j,const int k)` son métodos que definen los valores de los coeficientes para las ecuaciones, hay que tomar en cuenta que por ejemplo si se define la ecuación como tipo Poisson, todos los valores de los coeficientes serán ignorados en los calculos.

La clase `EllObj` tiene definido un solo operador, para L un elemento de la clase `EllObj`, S y U elementos de la case `GridObj`:

- $U = S/L$: En este caso es una forma simbólica de expresar la solución de la ecuación $LU = S$, en donde L es un operador elíptico definido con los métodos antes descritos, S es la fuente y U es la solución.

Muchas veces un ejemplo aclara conceptos abstractos, en el apéndice B, se muestra el código que implementa en una espina ficticia la solución a un problema de ejemplo y en donde se clarifican muchas de las ideas aquí presentadas.

5.3. Casos de prueba

Es posible probar un código para resolver ecuaciones diferenciales generando fuentes a partir de una función que cumple las condiciones de frontera. La solución propuesta se sustituye en la ecuación y la función resultante se usa como fuente para la ecuación. Con la solución analítica se pueden hacer pruebas de convergencia del método. Como se mencionó anteriormente la diferencia entre la solución numérica y el valor de la solución exacta define el error global de aproximación:

$$\epsilon^h := u - u^h. \tag{5.1}$$

Para un método de segundo orden, al aumentar el número de puntos en la partición al doble, el error debe disminuir 4 veces, si el método es de cuarto orden el error debe disminuir 16 veces al refinar la partición. Para las dos tipos de ecuaciones utilizadas se realizó una prueba de convergencia usando al menos tres resoluciones y comparando la solución numérica con la solución exacta se calculó el error global. Los resultados se presentan en la sección de convergencia para cada ecuación.

Para probar la eficiencia de los métodos implementados (por el momento el método de relajación Gauss-Seidel y el método Multigrid) se hicieron dos pruebas. La primera es una comparación entre métodos en donde se muestra el tiempo que toma obtener la solución con respecto al número de puntos en la partición para cada uno de los métodos. En la comparación entre métodos, se hace un análisis para corroborar la eficiencia teóricas. La segunda prueba es para conocer la eficiencia del código trabajando en varios procesadores, para este tipo de prueba se hicieron corridas a resolución fija y variando el número de procesadores utilizados. Los resultados de estas pruebas se presentan en la sección de eficiencia.

Por último en cada caso se muestran cortes de las gráficas de la solución. Se muestran la superficies solución para cortes en tres direcciones. En todos los casos se obtiene el resultado esperado ya que la fuente se construye explicitamente para obtener una solución dada. Junto con las gráficas de la solución se presentan las fuentes utilizadas y los coeficientes de la ecuación.

5.3.1. Ecuación de Poisson

La ecuación de Poisson será el primer tipo de ecuación elíptica que se estudiará. Diversos problemas en física se describen utilizando ecuaciones de Poisson, el campo electromagnético y

el gravitacional en la física clásica son dos ejemplos típicos. Relatividad general es una teoría en donde también se requiere obtener soluciones a la ecuación de Poisson en muchos casos con fuentes complicadas. Existen muchos métodos para obtener soluciones de la ecuación de Poisson, sin embargo hay casos en los que debido a la forma de las fronteras o bien debido a que la fuente es complicada no es posible obtener una solución analítica, en esos casos las soluciones numéricas son indispensables.

La ecuación de Poisson en coordenadas cartesianas y en un espacio plano esta dada por:

$$\partial_x^2 u(x,y,z) + \partial_y^2 u(x,y,z) + \partial_z^2 u(x,y,z) = \rho(x,y,z), \tag{5.2}$$

para probar Olliptic para esta ecuación se propuso como solución una función que cumple las condiciones de frontera tipo Dirichlet $u(x,y,z) = 0$ para $(x,y,z) \in \partial\Omega$ y con

$$\Omega = \left\{(x,y,z) \in \mathbb{R}^3 | -L_x \leq x \leq L_x, -L_y \leq y \leq L_y, -L_z \leq z \leq L_z\right\} \tag{5.3}$$

un cubo con $L_x = L_y = L_z = 10$. La función propuesta es:

$$u(x,y,z) = -\cos(c_x x)\cos(c_y y)\cos(c_z z), \tag{5.4}$$

en donde $c_x = (1+2n_x)\pi/L_x$, $c_y = (1+2n_y)\pi/L_y$ y $c_z = (1+2n_z)\pi/L_z$; $2n_x$, $2n_y$ y $2n_z$ son valores enteros que determinan el numero de nodos que tiene la solución en cada dirección. Se utilizó esta clase de solución porque es posible probar la convergencia para tres longitudes de onda al mismo tiempo, en particular para las pruebas se utilizó $n_x = 0$, $n_y = 2$ y $n_z = 4$. Sustituyendo la solución dada (5.4) en (5.2) obtenemos la fuente:

$$\rho(x,y,z) = -(c_x^2 + c_y^2 + c_z^2)u(x,y,z). \tag{5.5}$$

Utilizando esta fuente el código debe recuperar la solución dada.

Convergencia

El método Multigrid utiliza como método de relajación el de Gauss-Seidel, lo cual implica que si el sistema converge usando el método Multigrid, entonces converge con el método Gauss-Seidel. Por esta razón las pruebas de convergencia se hicieron para el método Multigrid. Se utilizaron tres particiones homogéneas del dominio con: $N = 33$, $N = 65$ y $N = 129$ puntos en cada dirección y que en adelante se denotarán por $4h = 0.625$, $2h = 0.3125$ y $h = 0.15625$ respectivamente.

Para mostrar la convergencia se analizaron cortes que pasan por el origen en tres direcciones obteniendo el error en cada eje. En la figura 5.1, se muestran cortes en los tres ejes de la solución y se comparan las tres resoluciones. El perfil que se obtiene es el esperado, se observan funciones cosenoidales negativas, con 0 nodos en el eje X, 4 nodos en el eje Y y 8 nodos en el eje Z. Se observa que las gráficas se superponen bien, solo la función de menor resolución se distingue del resto.

Debido a que que la fuente se construyó explicitamente para obtener la solución propuesta, se cuenta con una forma directa de calcular el error global. El código genera una tabla con los valores de la función en los puntos del dominio discretizado. Esta tabla se procesa, tomando la diferencia entre el valor de la tabla y el correspondiente valor analítico.[1] Se obtienen de esta forma tablas con los errores en cada punto del dominio discretizado.

En la figura 5.2 se presentan cortes en los tres ejes en donde se gráfica el error global. Se observan funciones que también oscilan con amplitudes menores a 0.06, por lo que obtenemos un

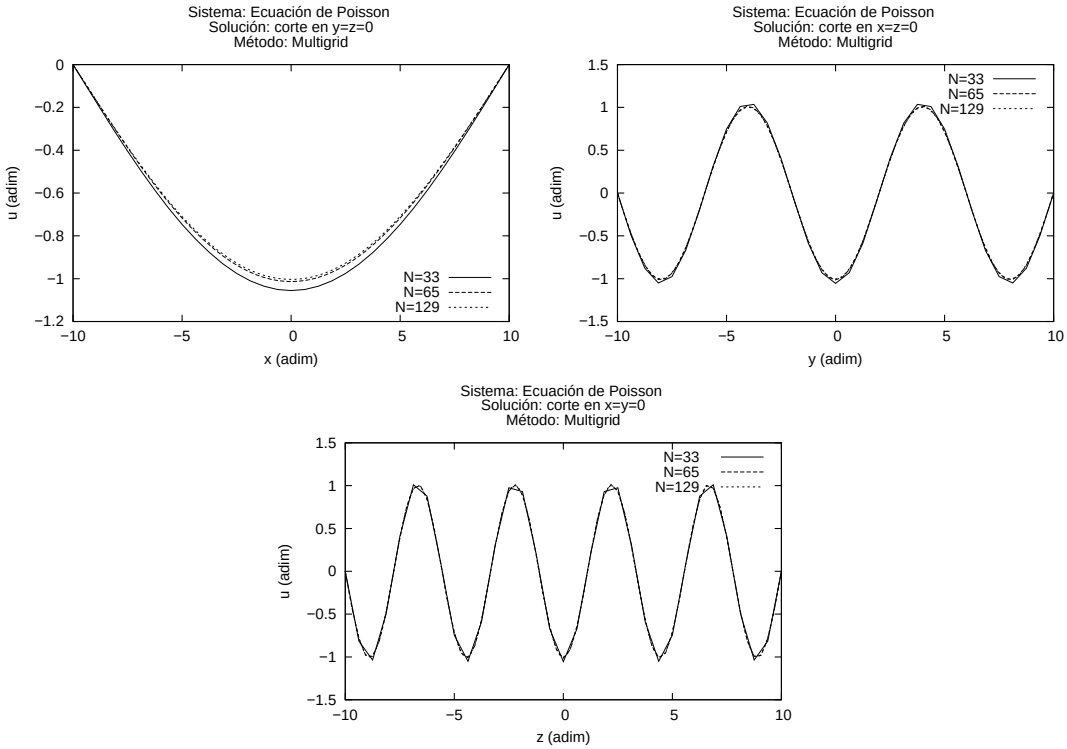

Figura 5.1: Gráfica de la solución a la ecuación de Poisson para tres resoluciones. Se obtienen formas muy similares en los tres casos lo cual se espera para una solución suave.

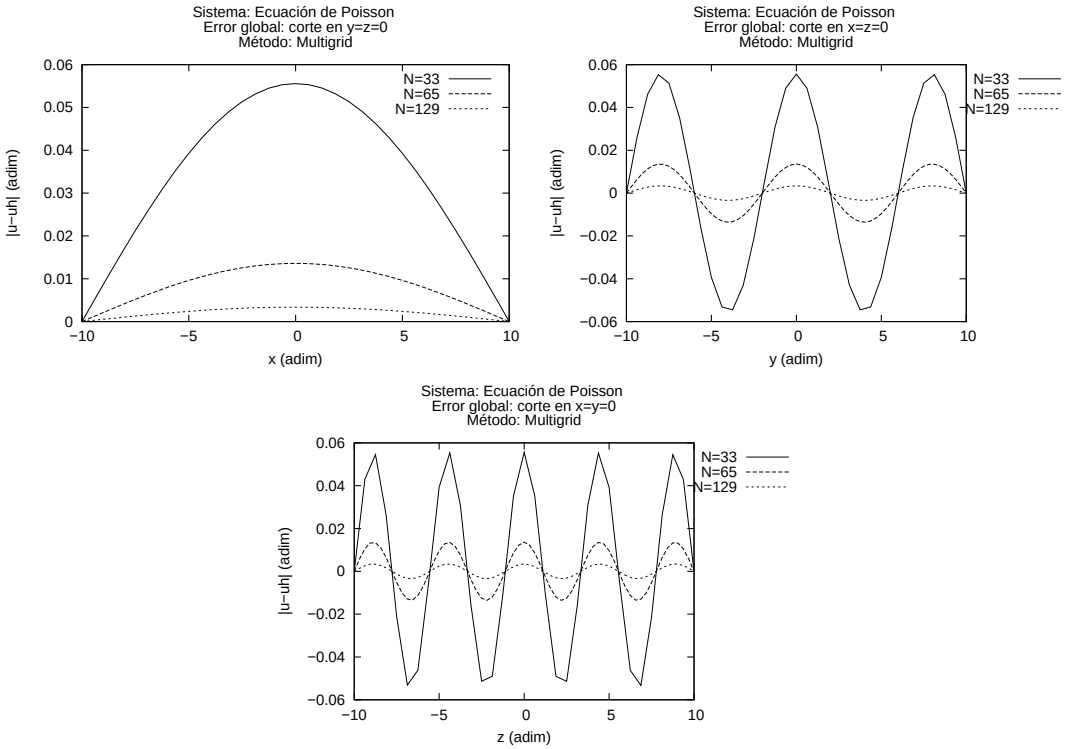

Figura 5.2: Gráfica del error global de la solución de la ecuación de Poisson para tres resoluciones. Se observa que el error tiende a cero al aumentar la resolución lo cual indica que la solución converge.

error relativo del mismo orden (ya que el valor máximo de la solución es 1.0). Así mismo, se observa que el error disminuye al aumentar el número de puntos en el dominio.

Por último se presenta la prueba de convergencia, al trabajar con un método de segundo orden y mallas con proporciones de puntos de 2:1, se debe presentar una convergencia cuadrática. En la figura 5.3 se grafica nuevamente el error global pero en este caso para la malla con partición $2h$ es multiplicado por un factor 4, y el error de la malla con partición h es multiplicado por un factor 16, que son las proporciones esperadas en convergencia cuadrática. En las gráficas se observa una muy buena correspondencia entre las tres resoluciones.

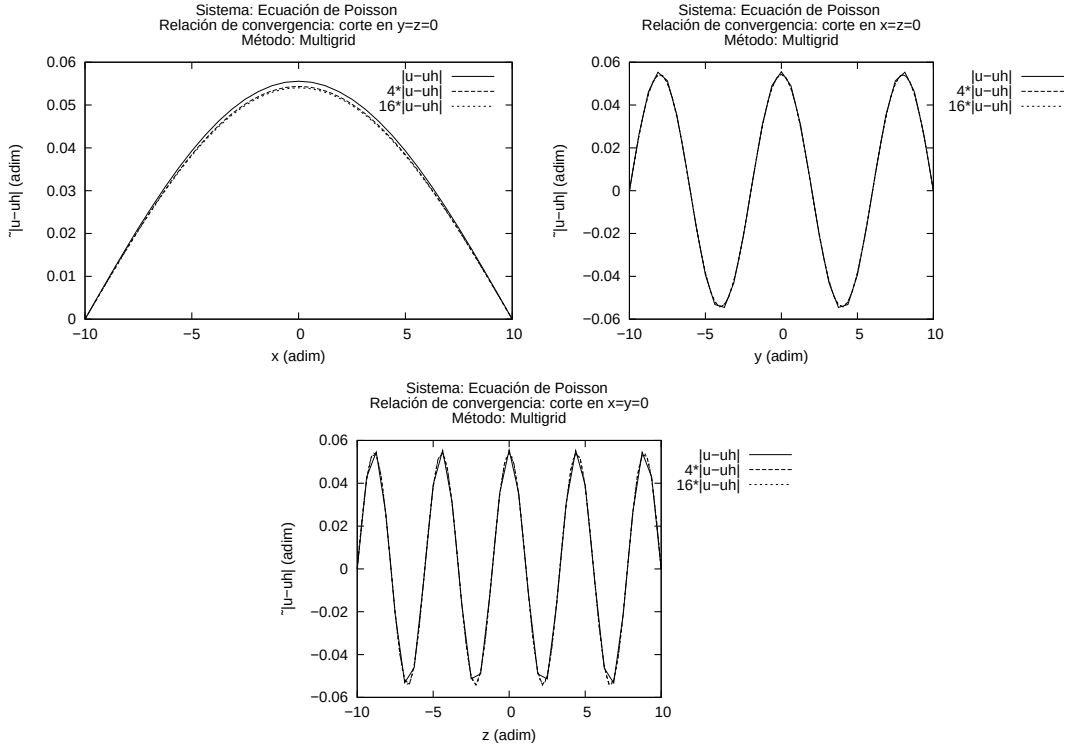

Figura 5.3: Prueba de convergencia para la ecuación de Poisson. Se observa que las gráficas de los errores se superponen al multiplicarlas por el factor adecuado, para la malla $2h$ por un factor 4 y para la malla de resolución h por un factor 16, con estas pruebas se muestra convergencia a segundo orden.

Eficiencia de los métodos en serie

Para la comparación entre el método Multigrid y el Gauss-Seidel, se obtuvo la solución del problema en 1 procesador usando 8 resoluciones.[2] Se hicieron 6 corridas para medir el tiempo y poder hacer una estadística. De estas muestras se tomó el promedio como valor representativo

[1] En términos prácticos el valor utilizado como analítico es también aproximado al introducirlo en la máquina, pero aún así, es una aproximación del mismo orden que el error de truncado, que para Olliptic es de doble precisión.

[2] En una maquina de escritorio con procesador AMD-Sempron a 1.8 GHz y 1 GB de RAM.

y la desviación estándar como medida de la incertidumbre. Se utilizó una tolerancia de la norma $L_1 = 0.0005$, lo cual garantiza que la suma de los valores absolutos de las componentes de la matriz residuo es menor a 0.0005.

El sistema calcula el tiempo utilizado por el procesador y el tiempo real que tarda en ejecutar el programa. El primer tiempo se denota como tiempo de *cpu* y al tiempo real también se le conoce como tiempo del *reloj de pared*. Es importante destacar que el tiempo real incluye el tiempo utilizado en cargar los datos en la memoria, en escribir datos al disco duro y en procesos internos del sistema operativo. El tiempo real esta influenciado por la carga de trabajo de la maquina utilizada por los usuarios o por otros procesos. El tiempo real es el que más puede variar de una ejecución a otra.

Para el método Multigrid se muestran los resultados en la Tabla 5.1. En esa tabla se presenta el número de sub-mallas utilizadas, la tolerancia L_1^H del residuo en la malla de menor resolución, la partición de la malla, el número de puntos totales, el tiempo (real y de cpu) promedio de las 6 corridas y las desviación estándar como medida de incertidumbre para los tiempos.

Parámetros Multigrid		Puntos por eje (adim)				Tiempo (seg)	
Sub-mallas	L_1^H	Nx	Ny	Nz	N_{total}	Real	cpu
1	0.0001	33	33	33	35937	0.091 ± 0.0015	0.089 ± 0.0022
2	0.00007	65	65	65	274625	1.75 ± 0.07	1.62 ± 0.078
2	0.00007	81	81	81	531441	4.8 ± 0.39	4.48 ± 0.059
2	0.00005	97	97	97	912673	10.6 ± 0.99	9.3 ± 0.43
2	0.00005	105	105	105	1157625	14.6 ± 0.74	14.5 ± 0.78
3	0.00005	113	113	113	1442897	20 ± 1.1	18.4 ± 0.23
3	0.00005	129	129	129	2146689	33.9 ± 0.17	33.7 ± 0.16
3	0.00001	145	145	145	3048625	52 ± 1.5	50.8 ± 0.45

Tabla 5.1: Rendimiento del código Olliptic (método Multigrid). Se utilizan particiones homogéneas, procurando que el número de puntos totales abarquen un dominio uniforme.

Para el método de Gauss-Seidel, se utilizó un procedimiento similar. Para obtener condiciones equivalentes, se ejecutaron las corridas alternando las correspondientes al método Multigrid con el Gauss-Seidel.

En la tabla 5.2 se presentan los resultados del método Gauss-Seidel. La diferencia entre el método Multigrid y el Gauss-Seidel es notoria al comparar las tablas. Para resoluciones bajas los tiempos son similares, sin embargo, al aumentar el número de puntos en la partición los tiempos del método Gauss-Seidel aumentan a un mayor ritmo que para el método Multigrid.

En la figura 5.4 se muestran las gráficas de rendimiento. La comparación en las gráficas muestra que el método Multigrid es más eficiente que el método de Gauss-Seidel para el tiempo real y de cpu. El método Multigrid presenta un comportamiento más errático debido a la cantidad de parámetros que se pueden ajustar: tolerancia de la norma en la malla de reolución más baja, tolerancia en la relajación de las mallas intermedias y número de mallas intermedias. En la comparación se observa que en todos los casos el método Multigrid resultó más eficiente, así mismo, se observa que el ritmo de crecimiento en los tiempos del método Gauss-Seidel es mucho mayor. La última gráfica que se presenta en la figura 5.4 es la correlación entre el tiempo real y de cpu. La desviación respecto de la identidad de la correlación es una medida de la saturación del sistema, se observa que al aumentar la resolución la desviación aumenta. Al saturar la memoria del sistema se requiere mayor uso del

Puntos por eje (adim)				Tiempo (seg)	
Nx	Ny	Nz	N_{total}	Real	cpu
33	33	33	35937	0.123 ± 0.0058	0.117 ± 0.0086
65	65	65	274625	3.2 ± 0.19	2.94 ± 0.078
81	81	81	531441	9.5 ± 0.23	8.3 ± 0.15
97	97	97	912673	21.2 ± 0.98	19.7 ± 0.47
105	105	105	1157625	30 ± 1.8	28.6 ± 0.84
113	113	113	1442897	43 ± 2.2	40.8 ± 0.81
129	129	129	2146689	80 ± 4.1	79 ± 1.5
145	145	145	3048625	146 ± 6.8	138 ± 1.3

Tabla 5.2: Rendimiento del código Olliptic (método Gauss-Seidel). Es notorio que el tiempo aumenta a un ritmo mayor que en el caso Multigrid

disco duro lo cual aumenta el tiempo de ejecución.

De acuerdo con la tabla 4.1 el método de Gauss-Seidel requiere un trabajo computacional de orden cuadrático como función del número total de puntos en la partición para obtener la solución, mientras que el método Multigrid es de orden lineal. En las gráficas de rendimiento de la figura 5.4 se observa una concordancia con el análisis de convergencia de esos métodos. Para corroborar la relación de forma más sistemática se ha aplicado un análisis de regresión lineal a los datos de las tablas, en el caso del método Gauss-Seidel tomando las raíces cuadradas de los tiempos. En la tabla 5.3 se muestran los parámetros obtenidos. Valores de R^2 cercanos a 1 indican una correlación lineal entre la distribución de datos. Si se analizan los datos de la tabla 5.2 para el método de Gauss-Seidel sin tomar las raíces de los tiempo se obtiene que $R^2 \simeq 0.95$ lo cual indica claramente que no tiene un comportamiento lineal.

Con los valores obtenidos se han graficado las funciones de ajuste $ax+b$ para el método Multigrid y $(ax+b)^2$ para el método Gauss-Seidel que se muestran en la figura 5.4 y que muestran una muy buena concordancia con tiempos de convergencia.

	Gauss-Seidel		Multigrid	
Parámetro	real	cpu	real	cpu
Correlación R^2	0.993	0.993	0.989	0.985
Pendiente a	0.0000038	0.0000037	0.000018	0.000017
Ordenada b	0.83	0.77	-3.79	-4.04

Tabla 5.3: Parámetros del análisis de regresión lineal aplicado a los datos de las tablas 5.1 y 5.2 tomando las raíces cuadradas de los tiempos en el caso del método Gauss-Seidel. Los valores del coeficiente R^2 cercanos a 1 indican una correlación lineal.

Eficiencia de los métodos en paralelo, método Gauss-Seidel

Para probar la eficiencia del método Gauss-Seidel en paralelo, se utilizaron tres particiones de 150, 200 y 250 puntos en cada dirección, en un dominio cúbico de lado 20. Para ejecutar las

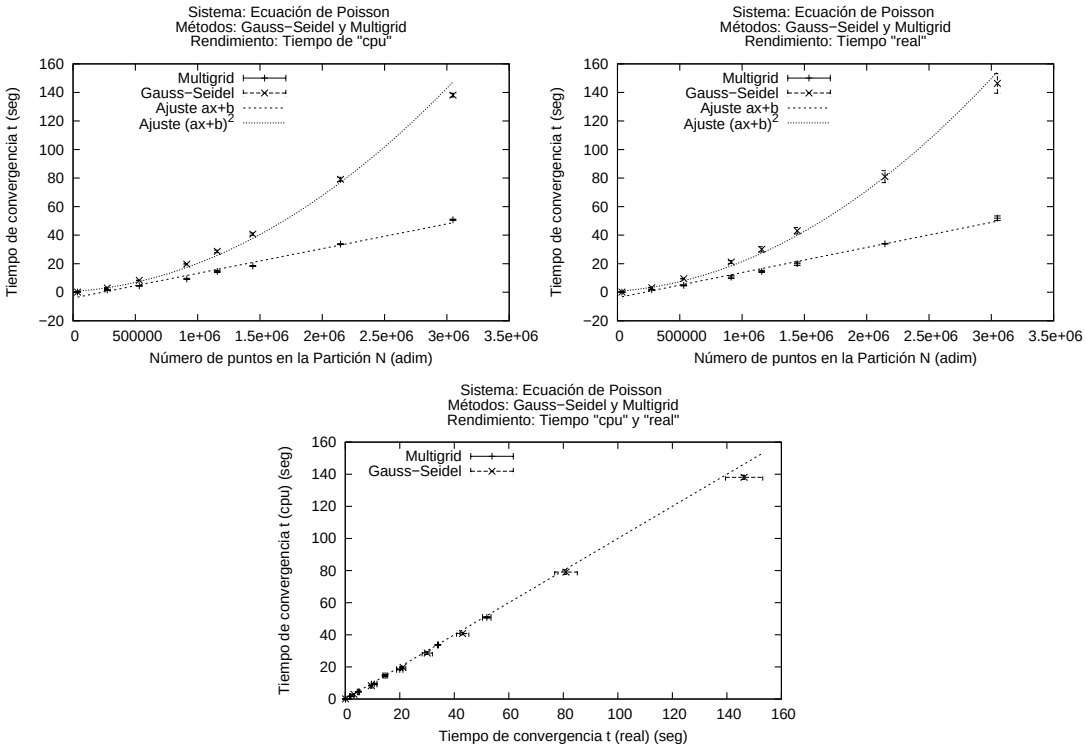

Figura 5.4: Prueba de rendimiento. Se muestran las gráficas comparativas entre el método Multigrid y el de Gauss-Seidel. En la parte superior izquierda se gráfica el tiempo de procesador como función del número total de puntos en la partición, en la parte superior derecha se encuentra la gráfica del tiempo real como función del número total de puntos para los dos métodos, en las dos gráficas se incluye también la función de ajuste correspondiente según los parámetros de la tabla 5.3. Por último en la parte inferior derecha la correlación entre el tiempo real y de cpu para los dos métodos. La desviación respecto a la identidad muestra una saturación del sistema.

pruebas se utilizó el cluster Ek-Bek del Laboratorio de Supercómputo Astrofísico [39], perteneciente al grupo Ollin. Se realizaron 5 corridas en 1, 2, 4 , 8 y 12 procesadores para medir el tiempo y hacer una estimación de la incertidumbre, usando una toleración del residuo para la solución final de $L_1 = 0.005$. En la tabla 5.4 se muestran los resultados midiendo el tiempo de cómputo total. Este tiempo considera el tiempo de cálculo, el tiempo que tarda el sistema en transmitir los datos entre procesadores y en guardar la información. Los detalles sobre la paralelización se pueden encontrar en el apéndice A, pero en los resultados se observa que el tiempo disminuye al aumentar el número de procesadores utilizados, hasta llegar a un mínimo que depende del tamaño de la partición. Para las dos resoluciones más bajas el tiempo mínimo se da en 8 procesadores, para la de mayor resolución se da en 12 procesadores.

Puntos en la partición:	3375000	8000000	15625000
Procesadores	Tiempo real (seg)		
1	49.49 ± 0.01	202.00 ± 0.02	605.20 ± 0.010
2	27.73 ± 0.12	109.67 ± 0.19	321.23 ± 0.68
4	15.16 ± 0.10	58.20 ± 0.11	167.64 ± 0.03
8	8.35 ± 0.07	31.97 ± 0.51	90.38 ± 0.06
12	11.85 ± 0.31	39.34 ± 1.38	83.30 ± 1.13

Tabla 5.4: Eficiencia del método Gauss-Seidel en varios procesadores (tiempo real). Se observa que el tiempo total de computo para la solución, el cual incluye el tiempo que tarda el sistema en trasmitir los datos entre procesadores, disminuye al aumentar el número de procesadores, llega a un mínimo que depende de la resolución usada y aumenta nuevamente.

En la tabla 5.5 se muestran los resultados correspondientes al tiempo de cpu, en este caso se observa que el tiempo de cálculo siempre disminuye al aumentar el número de procesadores. A pesar de que el uso del cpu disminuye rápidamente al aumentar el número de procesadores, por limitaciones en la implementación y en el hardware,[3] esta reducción en tiempo no se ve reflejada en el tiempo total de ejecución.

Puntos en la partición:	3375000	8000000	15625000
Procesadores	Tiempo cpu (seg)		
1	49.04 ± 0.02	200.98 ± 0.05	603.29 ± 0.04
2	25.88 ± 0.02	104.82 ± 0.05	310.15 ± 0.07
4	13.27 ± 0.02	53.05 ± 0.09	156.65 ± 0.10
8	6.76 ± 0.02	27.07 ± 0.02	79.74 ± 0.07
12	4.63 ± 0.03	18.45 ± 0.03	54.18 ± 0.01

Tabla 5.5: Eficiencia del método Gauss-Seidel en varios procesadores (tiempo de cpu). Se observa que el tiempo total de cpu para la solución disminuye al aumentar el número de procesadores y en este caso no presenta un aumento

Con la intención de obtener una función para describir el rendimiento del código en paralelo se ha hecho un análisis de regresión lineal a los datos de las tablas 5.4 y 5.5 tomando inversos de los

[3]Como se concluye más adelante la red que conecta los nodos del cluster Ek-Bek no es suficientemente rápida como para realizar el trabajo de forma eficiente en más procesadores.

tiempos para tratar de ajustar una función $1/(aNp+b)$. Los parámetros obtenidos se presentan en la tabla 5.6. El comportamiento de los tiempos de cpu se ajusta muy bien a esta relación, en cambio para hacer el análisis de los tiempos reales se requirió eliminar del análisis el último valor de la tabla 5.4, se probaron otras formas funcionales para el ajuste, sin embargo no se pudo determinar una mejor relación funcional.

	N=3375000		N=8000000		N=15625000	
Parámetro	real	cpu	real	cpu	real	cpu
Correlación R^2	0.99889	0.99974	0.99852	0.99979	0.98439	0.99984
Pendiente a	0.01416	0.01781	0.00374	0.00448	0.00098	0.00153
Ordenada b	0.00742	0.00342	0.00159	0.00069	0.00145	0.00020

Tabla 5.6: Parámetros del análisis de regresión lineal aplicado a los datos de las tablas 5.4 y 5.5 tomando los inversos de los tiempos. Se observa que el coeficiente R^2 en todos los casos tiene valores cercanos a 1, sin embargo fue necesario quitar del análisis los últimos valores del tiempo real.

En la figura 5.5 se muestran las gráficas correspondientes a los valores de las tablas 5.4, 5.5 y las funciones de ajuste. Se observa que al aumentar el número de puntos en la partición es más notoria la diferencia entre las corridas en paralelo y en serie mostrando un decrecimiento $1/Np$ hasta llegar a un mínimo.

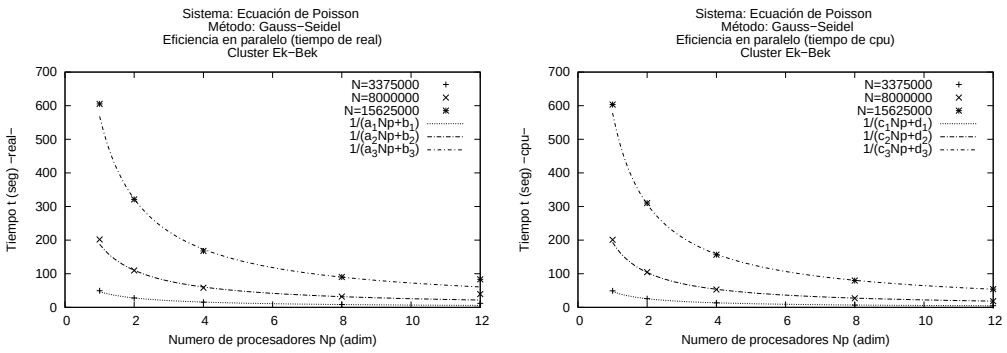

Figura 5.5: Gráficas de la eficiencia de método Gauss-Seidel en paralelo. En la gráfica del lado izquierdo se presenta el tiempo total de computo como función del número de procesadores, se observa que la implementación presenta su mayor eficiencia alrededor de 8 procesadores. En la gráfica del lado derecho se presenta el tiempo de uso del cpu como función del número de procesadores, en este caso el tiempo siempre disminuye al aumentar el número de procesadores. Se observa que el tiempo de cpu depende del número de procesadores Np como una función $1/(aNp+b)$, mientras que el ajuste no es valido para el tiempo real en el último valor.

En la figura 5.6 se muestra la correlación entre el tiempo total y de cpu para las tres resoluciones. La desviación respecto a la identidad es una buena medida de que tan eficiente es la implementación en paralelo del método. Se muestran las gráficas en escala logarítmica para apreciar mejor las diferencias. Para la resolución más baja (gráfica superior izquierda) y la intermedia (gráfica superior derecha), se aprecia que 8 procesadores tiene la mejor relación entre tiempo real y de procesador

ya que es el punto más cercano al origen y se encuentra cerca de la identidad. Al aumentar la resolución el tiempo de procesamiento se equipara al de transmisión de datos entre procesadores, para la resolución mas alta (gráfica inferior) se observa que la mejor relación entre tiempo de procesador y de cpu se da en 12 procesadores.

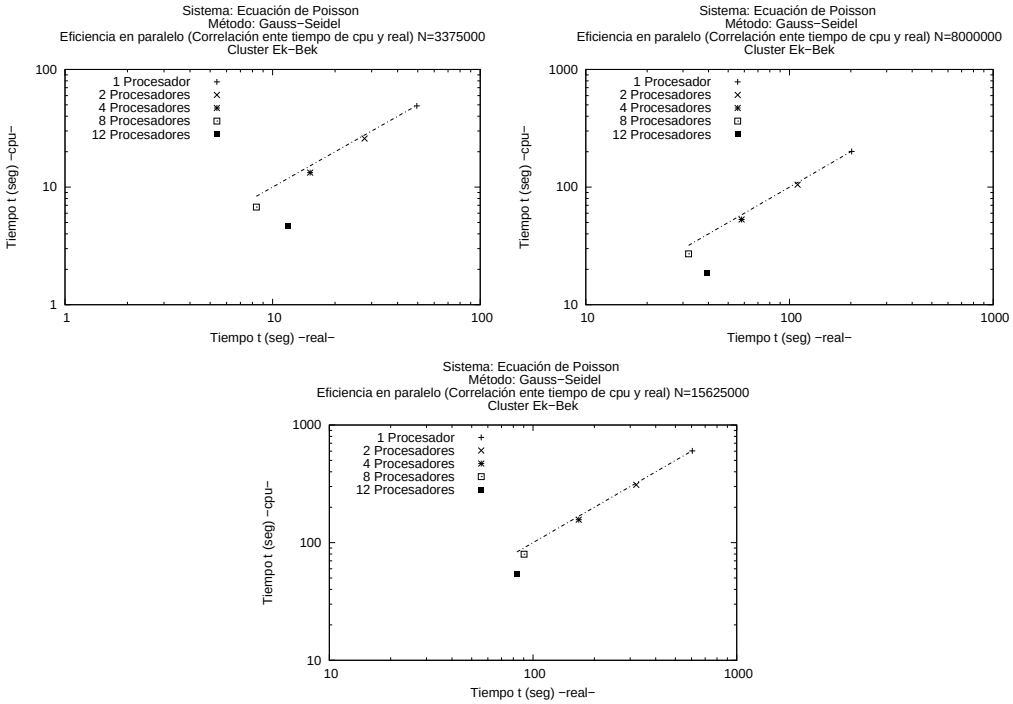

Figura 5.6: Gráficas de la correlación entre los tiempos para tres resoluciones usando el método Gauss-Seidel en paralelo (escala logarítmica). Se observa que la correlación entre el tiempo de cpu y real depende de el número total de puntos en la malla y el número total de procesadores en el que se ejecuta el proceso.

Al tener acceso a KanBalam, la supercomputadora de la UNAM, se realizaron pruebas de eficiencia en ese sistema, el cual cuenta con una mejor infraestructura de Hardware que el cluster del Laboratorio de Supercómputo Astrofísico.

En este caso se utilizaron los mismos parámetros que en los casos de prueba en serie, con una tolerancia de la norma más pequeña $L_1 = 0.00001$ de la solución final y una partición con $N = 15625000$ puntos. Se realizaron corridas en 1, 2, 4, 8, 16 y 32 procesadores observando muy buen rendimiento en este Hardware.

En la tabla 5.7 se presentan los resultados de las corridas, tanto el tiempo de cpu como el tiempo total disminuye al aumentar el número de procesadores utilizados.

En la figura 5.7 se muestran las gráficas correspondientes a los datos de la tabla 5.7. Para este sistema se ha obtenido un muy buen rendimiento en paralelo, lo cual se observa en las curvas de ajuste que describen un comportamiento inversamente proporcional en el tiempo respecto al número de procesadores utilizado. Este comportamiento tiene un límite que depende del Hardware del sistema, del tamaño de la partición utilizada y de la implementación del método de paralelización.

Puntos en la partición:	15625000	
Procesadores	Tiempo -real- (seg)	Tiempo -cpu- (seg)
1	1764.54 ± 0.04	1759.12 ± 0.02
2	825.28 ± 0.05	821.97 ± 0.04
4	438.02 ± 0.07	425.40 ± 0.02
8	233.08 ± 0.02	218.73 ± 0.05
16	152.05 ± 0.03	140.67 ± 0.09
32	76.44 ± 0.01	72.07 ± 0.09

Tabla 5.7: Eficiencia del método Gauss-Seidel en varios procesadores corriendo en la súper computadora Kan-Balam de la UNAM. Se observa que el tiempo para la solución disminuye al aumentar el número de procesadores observando un buen rendimiento de paralelización

En la gráfica de correlación se observa que hasta 32 procesadores el sistema se comporta muy bien.

Nuevamente se realiza un análisis de regresión lineal a los datos de la tabla 5.7 para tratar de determinar la forma funcional que describe mejor el comportamiento de los datos. En este caso se utilizó el inverso de los tiempos para hacer el análisis y se encontró que una relación $1/(aNp + b)$ se ajusta muy bien a los datos. En la tabla 5.8 se presentan los valores de los parámetros utilizados en las curvas de ajuste que se presentan en las gráficas.

	N=15625000	
Parámetro	real	cpu
Correlación R^2	0.99451	0.99452
Pendiente a	0.00039	0.00042
Ordenada b	0.00054	0.00055

Tabla 5.8: Parámetros del análisis de regresión lineal aplicado a los datos de las tablas 5.7 tomando los inversos de los tiempos. Se observa que el coeficiente R^2 en todos los casos tiene valores cercanos a 1, lo cual indica que el ajuste es bueno y la curva propuesta $1/(aNp + b)$ describe bien el comportamiento de la eficiencia en paralelo.

Eficiencia de los métodos en paralelo, método Multigrid

Para la prueba de paralelización del método Multigrid existen restricciones sobre el número de puntos en las particiones dadas por la ecuación (4.165). Cada subdominio asignado a un procesador debe cumplir la relación en el número de puntos de la partición local. Esta restricción hace que el número de puntos en la partición global no permanezca constante, sin embargo se pueden escoger particiones que dejan casi constante el número de puntos globales.

En la tabla 5.9 se presenta la forma en que se han distribuido los procesadores en el dominio y los puntos en la partición. Las mallas utilizadas tienen un promedio de 24520564 puntos totales con una desviación estándar de 227611 lo cual da una variación relativa de 0.93 %. Al tener una variación relativamente pequeña podemos considerar que los resultados de las corridas son más o menos equivalentes y podemos comparar los tiempos de convergencia obtenidos como función del número de procesadores utilizados.

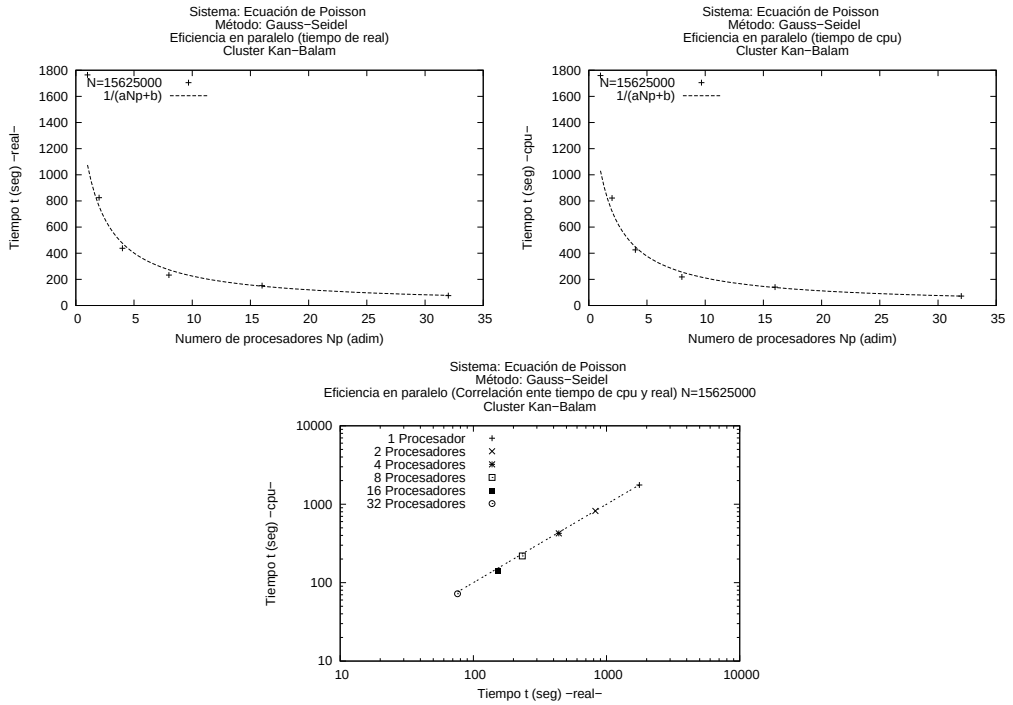

Figura 5.7: Gráficas de la eficiencia de método Gauss-Seidel en paralelo usando la supercomputadora Kan-Balam. En la gráfica del lado izquierdo se presenta el tiempo total de computo como función del número de procesadores, se observa que el tiempo disminuye al aumentar el número de procesadores utilizados. En la gráfica del lado derecho se presenta el tiempo de uso del cpu como función del número de procesadores. Se observa una dependencia funcional $t(Np) = 1/(aNp + b)$ se ajusta bien a los datos, los parámetros obtenidos para a y b en cada caso se presentan en la tabla 5.8. En la gráfica de la parte inferior se muestra la correlación entre tiempo real y de cpu, se observa que todos los valores se mantienen siempre cerca de la diagonal lo cual indica que no se ha llegado al límite de las comunicaciones o memoria del sistema.

Puntos Locales			Procesadores por eje			Puntos Globales				
NxL	NyL	NzL	Eje x	Eje y	Eje z	NxG	NxG	NyG	N_{total}	N_{Proc}
289	289	289	1	1	1	289	289	289	24137569	1
289	145	145	1	2	2	289	290	290	24304900	4
145	145	145	2	2	2	290	290	290	24389000	8
145	145	97	2	2	3	290	290	291	24473100	12
145	145	73	2	2	4	290	290	292	24557200	16
145	97	97	2	3	3	290	291	291	24557490	18
97	97	97	3	3	3	291	291	291	24642171	27
145	73	73	2	4	4	290	292	292	24726560	32
73	73	73	4	4	4	292	292	292	24897088	64
								Pormedio	24520564	

Tabla 5.9: Partición del dominio por procesadores. Se escoge un número de puntos en la partición local válido para aplicar el método Multigrid y que además mantenga el número de puntos totales más o menos constante.

Puntos en la partición:		∼24520564	
Sub-mallas	Procesadores	Tiempo -real- (seg)	Tiempo -cpu- (seg)
5	1	667.15 ± 0.32	661.80 ± 0.10
4	4	773.13 ± 0.45	761.53 ± 0.02
4	8	410.20 ± 0.26	398.44 ± 0.11
4	12	111.54 ± 0.89	104.49 ± 0.09
3	16	243.53 ± 0.48	225.59 ± 0.03
4	18	88.65 ± 0.23	77.36 ± 0.10
5	27	63.89 ± 0.56	53.33 ± 0.12
3	32	137.09 ± 0.12	122.18 ± 0.02
3	64	67.45 ± 0.46	52.93 ± 0.19

Tabla 5.10: Eficiencia del método Multigrid en varios procesadores corriendo en Kan-Balam. Se observa un comportamiento errático en donde el tiempo de convergencia no decrece monotonamente, se puede observar que existen particiones preferentes en donde se paraleliza mejor el sistema.

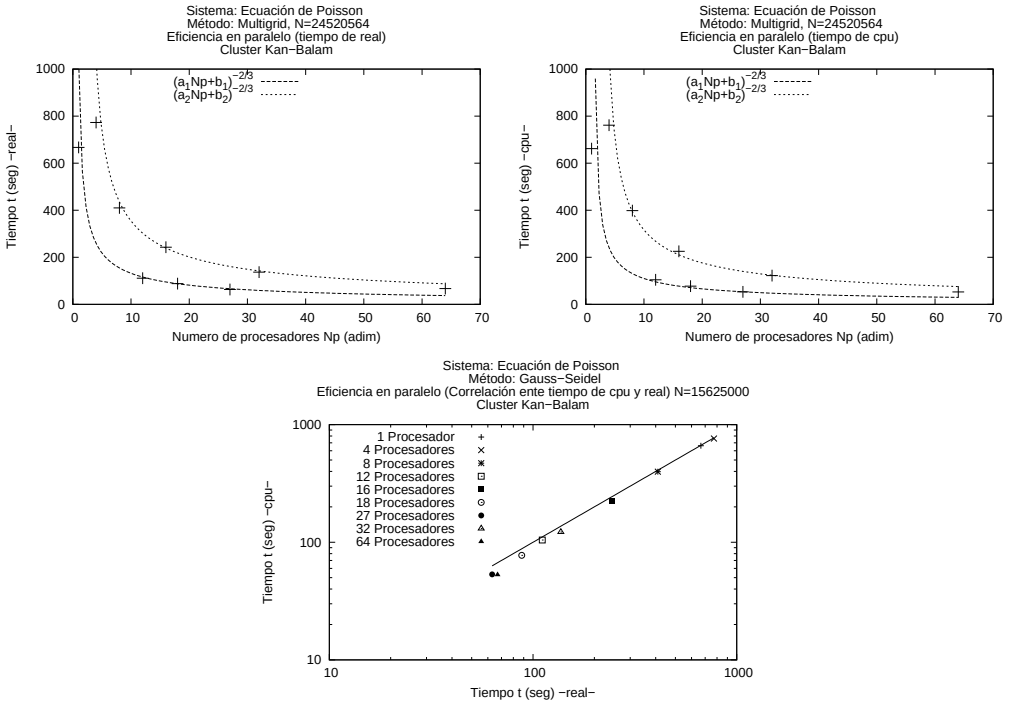

Figura 5.8: Gráficas de la eficiencia en paralelo del método Multigrid. Se presenta el tiempo como función del número de procesadores, en la gráica superior para el tiempo real y en la gráfica intermedia para el tiempo de cpu. Se observa que el comportamiento se distingue en dos grupos según la curva de ajuste, posteriormente se observa que esa distinción se debe al número máximo de sub-mallas que se utilzan en el método Multigrid. En la gráfica inferior se presenta correlación entre el tiempo real y de cpu, se observa que no hay saturación del sistema.

Se procedió ha realizar la prueba en el cluster Kan-Balam con los mismos parámetros que en el caso de Gauss-Seidel aunque con las particiones dadas en la tabla 5.9. Los resultados se presentan en la tabla 5.10 y a primera vista parecen tener un comportamiento errático. En la figura 5.8 se presentan las gráficas de los datos de la tabla 5.10, en las figuras se observa que los datos se ajustan bien a dos curvas distintas obteniendo dos subconjuntos de números de procesadores con la misma forma funcional pero con diferentes parámetros de ajuste.

	Grupo 1 de procesadores		Grupo 2 de procesadores	
Parámetro	real	cpu	real	cpu
Correlación R^2	0.99547	0.98685	0.99699	0.99437
Pendiente a	0.00007	0.00010	0.00002	0.00002
Ordenada b	-0.00004	-0.00013	-0.00005	-0.00007

Tabla 5.11: Parámetros del análisis de regresión lineal aplicado a los datos de las tablas 5.10 tomando potencias $t^{-3/2}$ de los tiempos. Se observa que el coeficiente R^2 en todos los casos tiene valores cercanos a 1, lo cual indica que el ajuste es bueno y la curva propuesta $1/(aNp + b)^{2/3}$ describe bien el comportamiento de los dos subconjuntos de datos.

Para determinar las curvas de ajuste se separaron los datos en dos grupos, el primero con 1,12 18 y 27 procesadores y el segundo con 4, 8, 16 y 32. Para estos grupos se calculó la potencia $t^{-3/2}$ de los tiempos que fue la que dio el mejor ajuste en ambos grupos de datos y se hizo el analisis de regresión lineal como en los casos anteriores. Los parámetros de ajuste se presentan en la tabla 5.11.

El comportamiento errático se puede entender si regresamos a la restricción que genera el método Multigrid y a la implementación del mismo. Como se observa en la tabla 5.10, el número de sub-mallas esta acotado por la partición con menor número de puntos en una dirección dada. Los puntos utilizados pertenecen a dos secuencias de mallas, una de ellas es 10, 19, 37, **73**, **145** y **289**, la otra es 4, 7, 13, 25, 49 y **97**. Esto permite por ejemplo al usar 27 procesadores, tener 5 sub-mallas para implementar el método multigrid. Lo que podemos concluir es que la paralelización en el método Multigrid depende del conjunto de mallas a utilizar de modo que al utilizar el código en un problema específico se deberá tener en cuenta esta relación entre procesadores y submallas.

Se ha determinado que la implementación de la paralelización del método Gauss-Seidel en un cluster con buen Hardware tiene una eficiencia muy buena, lo cual nos da confianza sobre el rendimiento que presentará el código para su uso en problemas de investigación. Así mismo, la implementación del código con el método Multigrid en paralelo presenta una mejora de rendimiento al utilizar de forma adecuada las submallas. Será hasta los problemas físicos que se presentarán resultados de corridas en paralelo, el resto de las pruebas se realizaron en 1 solo procesador.

Solución

En la parte de convergencia se mostraron gráficas de las soluciones para las tres resoluciones. En la figura 5.9 se muestran las gráficas de tres cortes de la solución pero en este caso mostrando superficies. Se tomaron los cortes XY con $z = 0$, XZ con $y = 0$ y YZ con $x = 0$. De las gráficas se pueden identificar en la solución tres longitudes características, correpondientes a la separación de los nodos y que en términos del análisis de convergencia que se hizo en capítulos anteriores, corresponde a 3 longitudes de propagación del error.

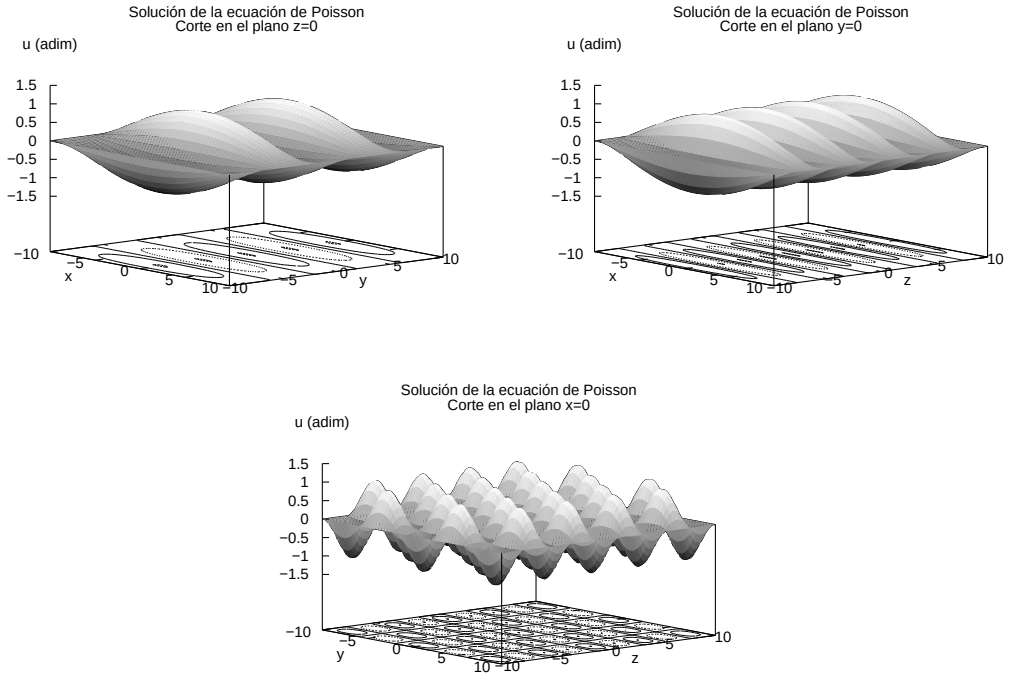

Figura 5.9: Cortes de la solución en superficies para la ecuación de Poisson. Se muestran tres cortes, los planos Z=0, Y=0 y X=0, se observa que la solución obtenida tiene la forma esperada, con 3 longitudes de onda características.

Así mismo, en la figura 5.10 se presentan los correspondientes cortes de la función que se utilizó como fuente, ecuación (5.5). Como se observa en las ecuaciones (5.4) y (5.5) y en las figuras, la solución y la fuente difieren en signo y por un factor que modula la fuente.

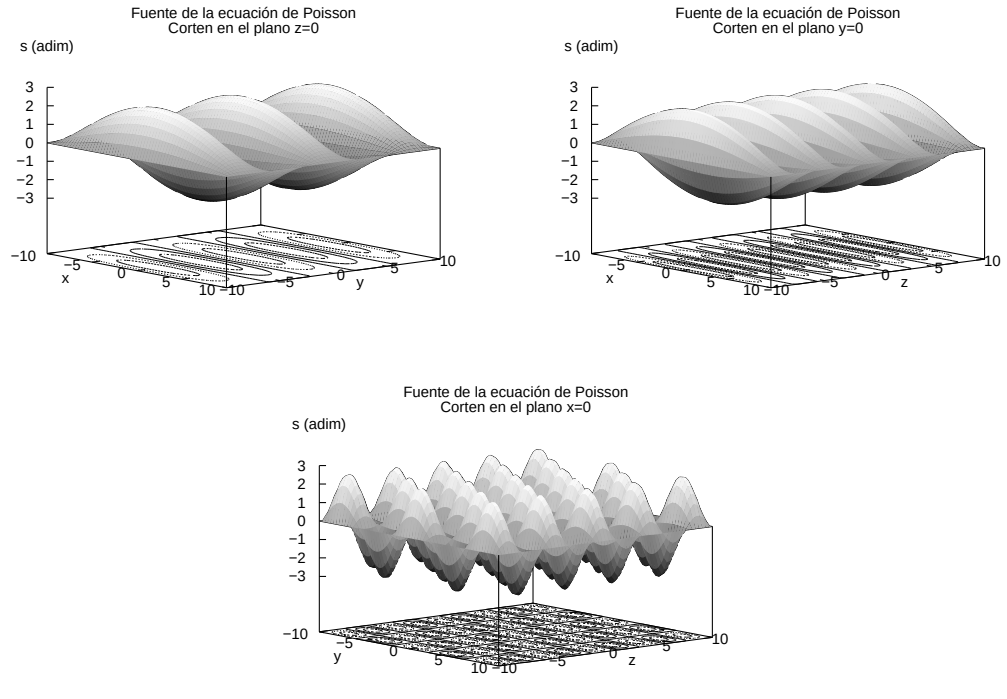

Figura 5.10: Cortes de la fuente en superficies para la ecuación de Poisson. Nuevamente se muestran tres cortes, los mismos planos Z=0, Y=0 y X=0 que se graficaron para la solución. La función fuente fue construida especificamente para obtener la solución mostrada en la figura 5.9.

Como ya se mencionó se seleccionó esta solución para poder estudiar el comportamiento de tres longitudes de propagación del error al mismo tiempo, es posible con esta fuente estudiar el comportamiento del código al variar el número de nodos en distintas proporciones. La fuente dada no representa una fuente generada por algún problema físico en particular, sin embargo es posible tener una configuración eléctrica de cargas positivas y negativas dentro de una caja cubica cuya densidad tenga la forma de la fuente (5.5) y tal que las paredes de la caja se encuentren a potencial 0 (conectadas a tierra). La solución de la ecuación de Poisson para ese problema representa el potencial de esa distribución y como sabemos, el gradiente de ese potencial es el campo eléctrico.

5.3.2. Ecuación elíptica lineal

Un análisis idéntico al anterior se realizó para la ecuación lineal:

$$\nabla^2 u(x,y,z) + A(x,y,z)u(x,y,z) = \rho(x,y,z), \tag{5.6}$$

en donde A es una función de las coordenadas suave y que cumple las condiciones de frontera. Se utilizó la misma solución que en el caso anterior:

$$u(x, y, z) = -\cos(c_x x)\cos(c_y y)\cos(c_z z), \tag{5.7}$$

en donde $c_x = (1 + 2n_x)\pi/L_x$, $c_y = (1 + 2n_y)\pi/L_y$ y $c_z = (1 + 2n_z)\pi/L_z$. Nuevamente $2n_x$, $2n_y$ y $2n_z$ son valores enteros que determinan el numero de nodos que tiene la solución en cada dirección. Con la intención de tener una forma diferente de la solución, en este caso se usaron los valores $n_x = 0$, $n_y = 1$ y $n_z = 2$ y como factor de la parte lineal se utilizó la función:

$$A(x, y, z) = (x^2 - L_x^2)(y^2 - L_y^2)(y^2 - L_y^2)\sin\left(\frac{x^2}{L_x} + \frac{y^2}{L_y} + \frac{z^2}{L_z}\right). \tag{5.8}$$

en donde se tomo el mismo dominio que para la ecuación de Poisson , con $L_x = L_y = L_z = 10$. El coeficiente A es un paraboloide modulado por una función senoidal, obteniendo una función suave que tiene oscilaciones y que cumple la condición de frontera. Inicialmeten se escogió esta función de forma arbitraria, pero resultó ser un coeficiente interesante en términos de la convergencia.

Sustituyendo la solución dada (5.7) en (5.6) obtenemos la fuente:

$$\rho(x, y, z) = -\left(c_x^2 + c_y^2 + c_z^2 - A(x, y, z)\right) u(x, y, z). \tag{5.9}$$

Como se vera más adelante esta fuente no tiene una forma trivial y no es tan simétrica, sin embargo se obtiene la solución esperada.

Convergencia

Para las pruebas de convergencia se utilizaron particiones homogéneas del dominio con: $N = 33$, $N = 65$, $N = 129$ y $N = 257$ puntos en cada dirección que nuevamente denotaremos como $8h$, $4h$, $2h$ y h. Dado que la fuente se construye a partir de la solución una vez más es posible obtener el error global directamente.

Iniciaremos el análisis mostrando cortes de la solución en donde se superponen las cuatro resoluciones (ver figura 5.11). Se observa una mejor correspondencia entre las cuatro resoluciones, en este caso no se aprecian diferencias. Se obtienen las soluciones cosenoidales esperadas, con 0 nodos en el eje X, 2 nodos en el eje Y y 4 nodos en el eje Z. Una observación importante es que aún cuando la fuente y el término lineal complican la ecuación que se desea resolver, la solución obtenida es mejor.

En la figura 5.12 se muestra la gráfica del error global para cortes en tres direcciones. Se observa que los errores son muy pequeños y que en el centro crecen un poco, a diferencia de la ecuación de Poisson en donde el error oscilaba, en este caso no se distinguen oscilaciones debido a que hay una gran diferencia entre el valor promedio del error y el valor del error en el centro, sin embargo el máximo error es mucho menor que el obtenido en el caso de la ecuación de Poisson.

En la figura 5.13 se muestra la prueba de convergencia. Inicialmente se hicieron las pruebas usando tres resoluciones, sin embargo se observó que el sistema no convergía como se esperaba, por lo cual se realizaron pruebas con mayor resolución, la resolución $2h$ y h muestran una mejor correspondencia ya que las gráficas se sobreponen como se espera para sistemas de segundo orden, las demás resoluciones sin embargo parecen converger más rápido en esa zona.

El hecho de que no se obtenga una convergencia adecuada hace sospechar de la solución, aún cuando la convergencia es más rápida. Como existe una diferencia muy grande en el valor central se procedió a examinar en detalle las gráficas. En la figura 5.14 se muestra un rango de valores más

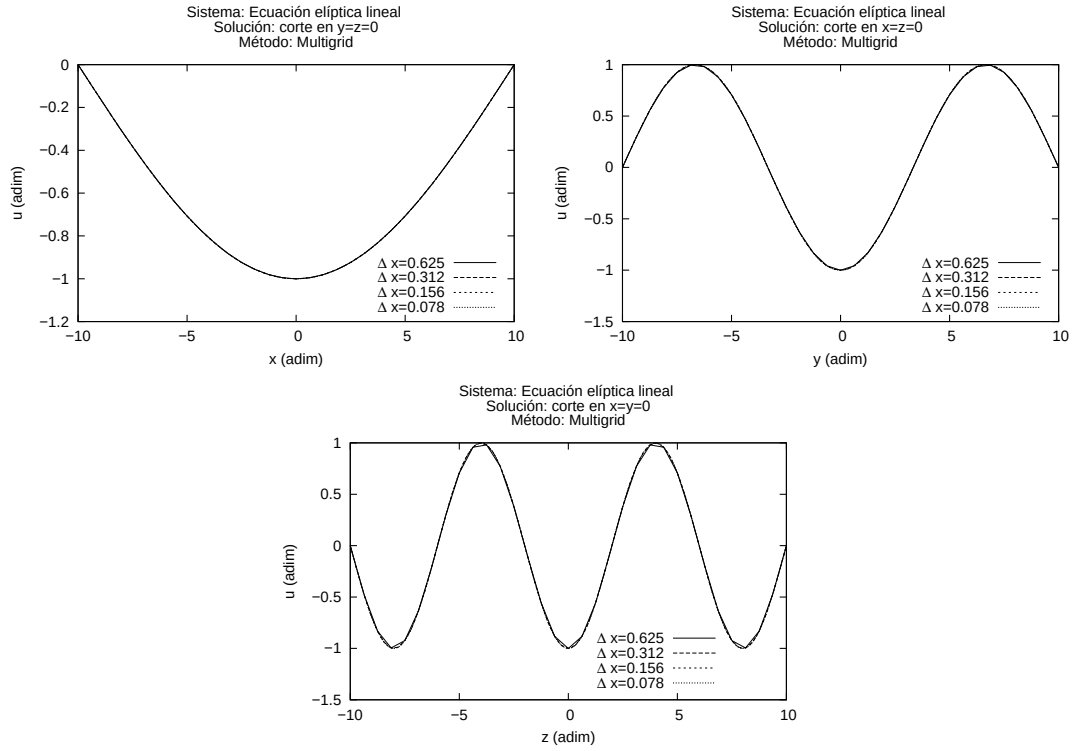

Figura 5.11: Gráficas de la solución para la ecuación elíptica lineal. Se observa una muy buena coincidencia de las cuatro resoluciones, a diferencia del caso de la ecuación de Poisson, aquí no se distingue diferencia a simple vista.

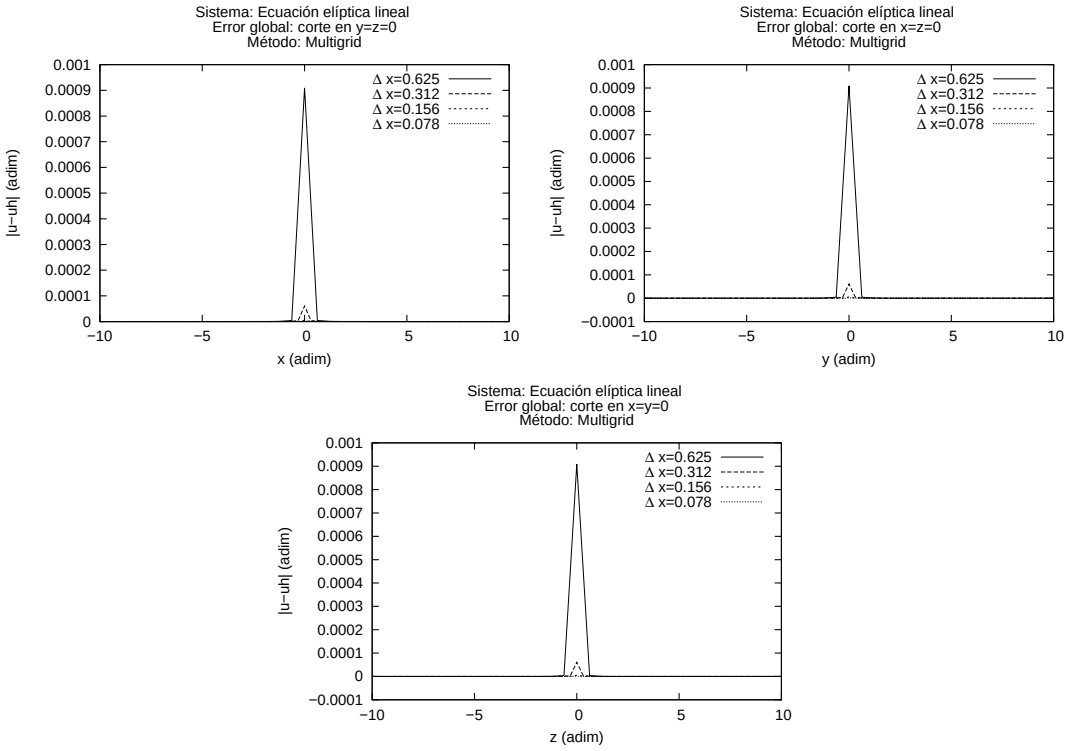

Figura 5.12: Gráfica del error global para la ecuación elíptica lineal para cuatro resoluciones. Se observa que el error tiende muy rápido a cero al aumentar la resolución, la zona de mayor error se da en el centro y tiene un valor relativo de 0.009 para la resolución más baja.

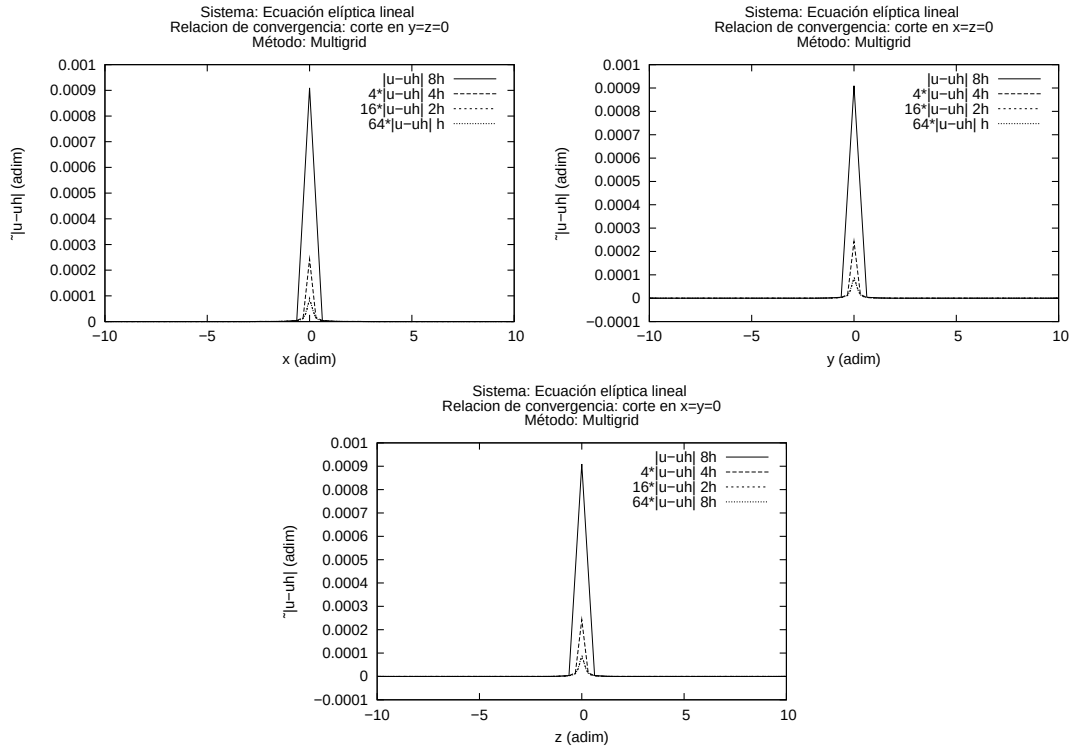

Figura 5.13: Prueba de convergencia para la ecuación elíptica lineal. Los valores de la resolución $2h$ se multiplican por 4, los de $4h$ por 16 y los de $8h$ por 64. Se deberían sobreponer las gráficas, sin embargo se observa que la convergencia en esa zona es más rápida de lo esperado.

estrecho que en las correspondientes gráficas de la figura 5.12. En este caso se obtiene el resultado esperado: se observan oscilaciones y diferencias entre las distintas resoluciones, así como valores en la parte central que crecen mucho respecto del resto y que eran los únicos que destacaban en la figura anterior.

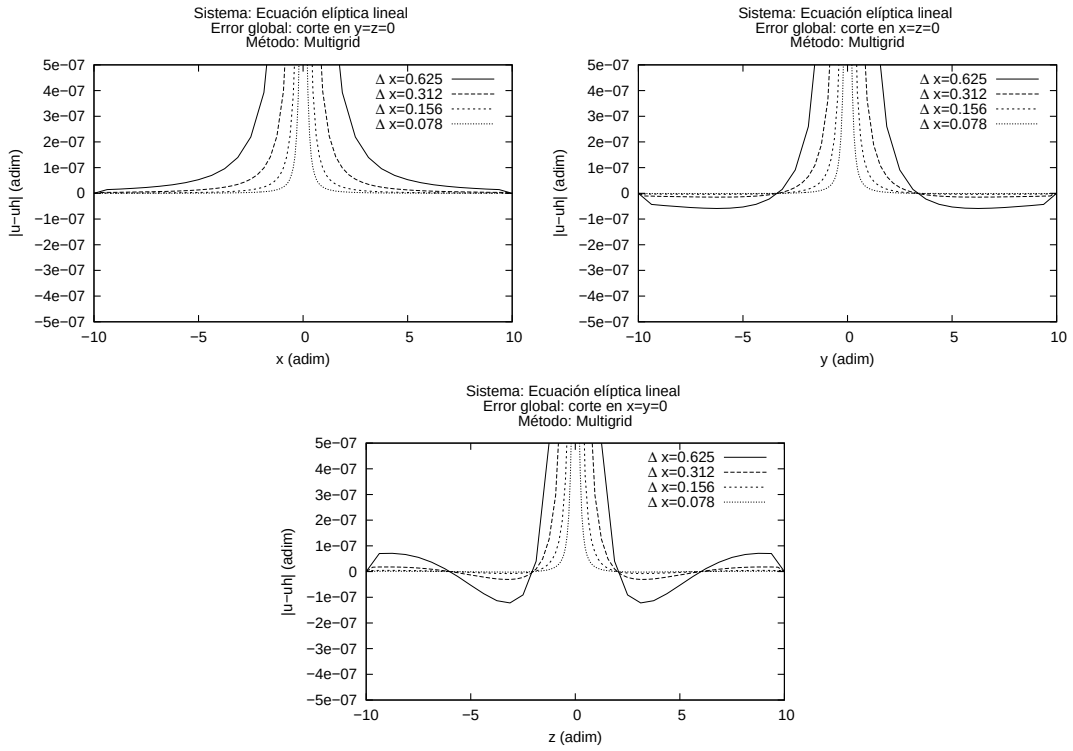

Figura 5.14: Gráfica del error global para la ecuación elíptica lineal y cuatro resoluciones. Observando en detalle las gráficas del error, con un rango más restringido, se obtiene el comportamiento esperado. La amplitud de los errores es muy pequeña para esta solución.

Así mismo en la figura 5.15 se muestran las gráficas en detalle que corresponden a la figura 5.13. En estas gráficas se muestra claramente la convergencia a segundo orden, con excepción de valores cercanos al origen y a resoluciones bajas, se observa que los valores se superponen al ser multiplicados por el factor adecuado lo cual confirma que se cumple (4.50).

Eficiencia de los métodos en serie

Nuevamente se hizo una comparación entre el método Multigrid y el método de Gauss-Seidel. Para el sistema utilizado, la convergencia ha sido mucho más rápida que en el caso de la ecuación de Poisson. La diferencia entre los dos métodos es menos notoria para este sistema. En la tabla 5.12 se muestra el resultado para el método Multigrid, nuevamente se muestra el tamaño de la partición en cada eje y el número de puntos total. Así mismo, se muestra el tiempo de ejecución real y de cpu. Como los tiempos de convergencia en este caso son pequeños sería posible explorar resoluciones más altas, sin embargo las limitaciones en memoria de la maquina utilizada no permitió explorar

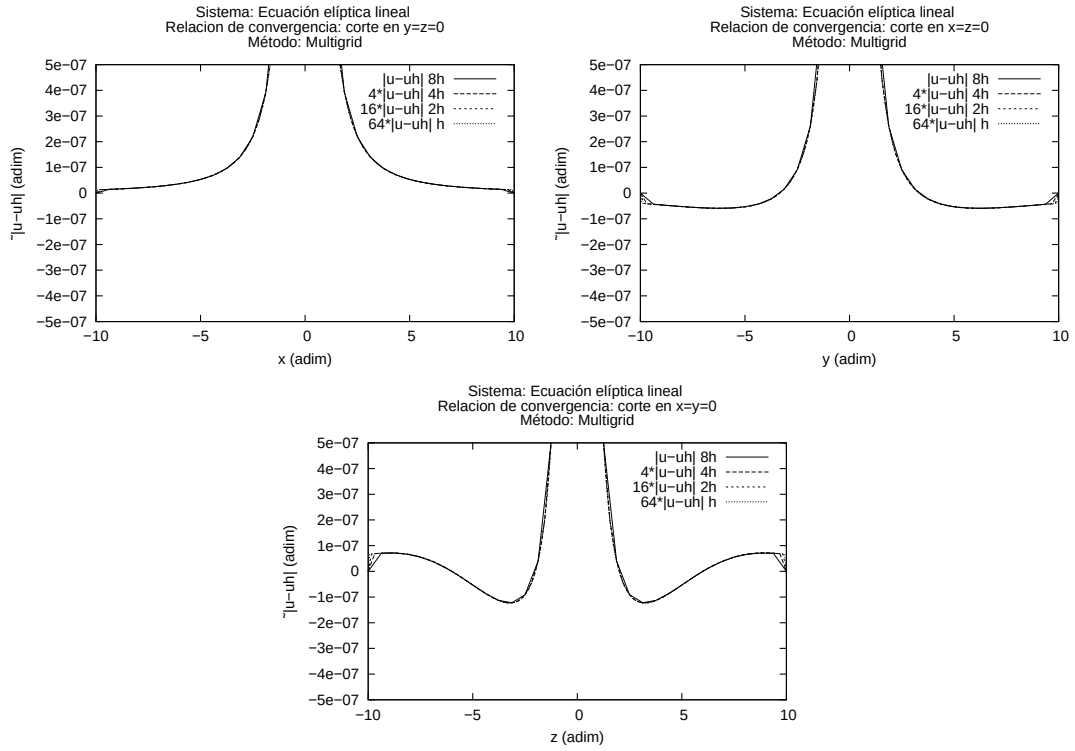

Figura 5.15: Prueba de convergencia para la ecuación elíptica lineal con cuatro resoluciones. En las gráficas de valores más restringidos se observa claramente que fuera de la zona central, los datos se corresponden en la proporción adecuada, lo cual muestra convergencia a segundo orden.

particiones con más de 161^3 puntos..

Parámetros Multigrid		Puntos por eje (adim)				Tiempo (seg)	
Sub-mallas	L_1^H	Nx	Ny	Nz	N_{total}	Real	cpu
1	0.0001	33	33	33	35937	0.06 ± 0.003	0.06 ± 0.0042
2	0.0007	65	65	65	274625	0.48 ± 0.017	0.48 ± 0.012
2	0.0006	81	81	81	531441	1.07 ± 0.038	1.07 ± 0.039
2	0.0005	97	97	97	912673	2.10 ± 0.066	2.12 ± 0.076
2	0.0005	105	105	105	1157625	2.58 ± 0.14	2.58 ± 0.078
3	0.0005	113	113	113	1442897	3.57 ± 0.19	3.57 ± 0.22
3	0.0005	129	129	129	2146689	5.80 ± 0.25	5.31 ± 0.042
3	0.0005	145	145	145	3048625	8.40 ± 0.57	7.90 ± 0.14
3	0.0005	161	161	161	4173281	14.27 ± 0.90	11.28 ± 0.16

Tabla 5.12: Rendimiento del código Olliptic (método Multigrid). Se ha tomado un espectro más amplio de resoluciones, sin embargo, los tiempos de ejecución son muy pequeños comparados con el caso de la ecuación de Poisson. Para el método Multigrid, el tiempo más grande de la ecuación lineal se compara con la cuarta resolución de la ecuación de Poisson que es 4 veces más pequeña.

Puntos por eje (adim)				Tiempo (seg)	
Nx	Ny	Nz	N_{total}	Real	cpu
33	33	33	35937	0.05 ± 0.004	0.05 ± 0.004
65	65	65	274625	0.49 ± 0.014	0.48 ± 0.010
81	81	81	531441	1.07 ± 0.074	1.11 ± 0.112
97	97	97	912673	2.32 ± 0.178	2.33 ± 0.137
105	105	105	1157625	2.95 ± 0.076	2.99 ± 0.071
113	113	113	1442897	4.34 ± 0.534	4.14 ± 0.168
129	129	129	2146689	7.35 ± 0.389	6.58 ± 0.062
145	145	145	3048625	11.84 ± 0.414	11.05 ± 0.107
161	161	161	4173281	21.94 ± 3.970	16.81 ± 0.155

Tabla 5.13: Rendimiento del código Olliptic (método Gauss-Seidel). Para este sistema no se observaron grandes diferencias con el método Multigrid, sin embargo si se observa que el método Multigrid es más eficiente.

En al figura 5.16 se muestran las gráficas correspondientes a las tablas presentadas. Se observa que nuevamente el método Multigrid es más eficiente en todos los casos. En la gráfica de correlación entre el tiempo real y de cpu se observa que para las resoluciones altas la diferencia entre los dos tiempos crece mucho, esto es debido a que a esas resoluciones, se esta muy cerca del límite de memoria de la maquina (el tiempo de cálculo es mucho menor que el tiempo que tarda el sistema en administrar la memoria).

Se ha visto que el rendimiento de los métodos para este caso es muy similar para los dos métodos. Esto se puede explicar debido a que el análisis de convergencia de donde se obtienen los ordenes

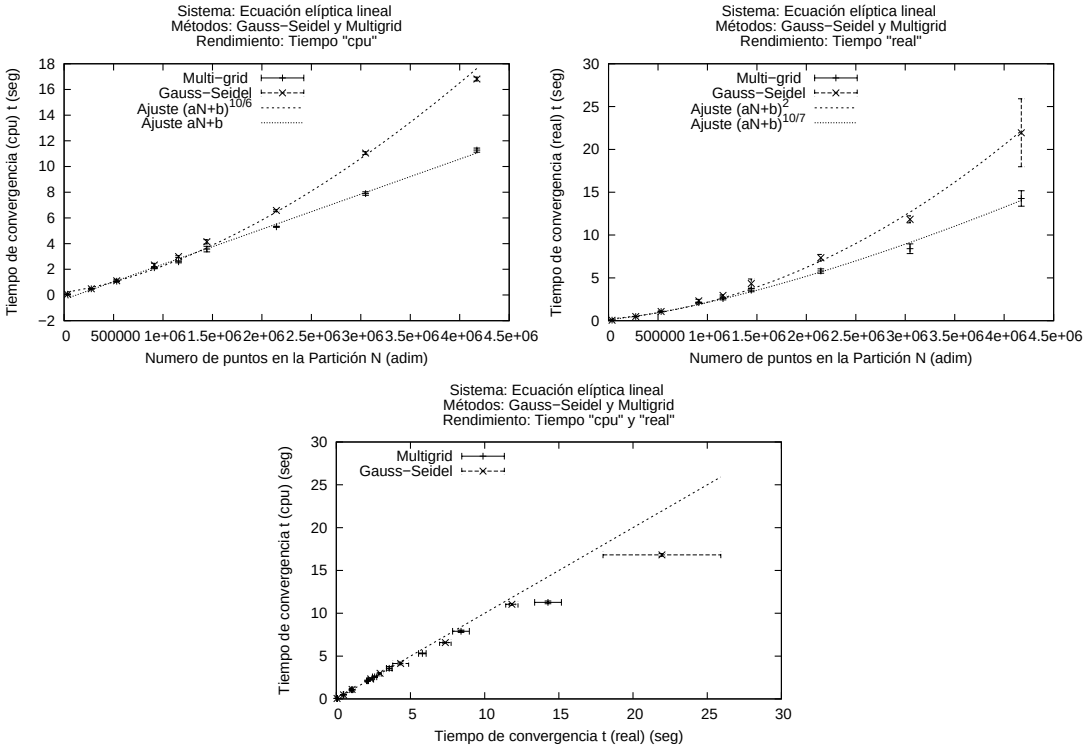

Figura 5.16: Prueba de rendimiento. Se muestran las gráficas comparativas entre el método Multigrid y el de Gauss-Seidel. En la parte superior izquierda la comparación del tiempo de procesador como función del número total de puntos en la partición. Por último en la parte inferior derecha la correlación entre el tiempo real y de cpu para los dos métodos. Se grafican también las funciones de ajuste correspondientes.

de eficiencia se hacen para casos muy simples como la ecuación de Poisson. En este caso es posible que el coeficiente de la parte lineal modifique el comportamiento de los métodos ya que presenta valores muy grandes respecto al valor de la solución de modo que domina el comportamiento de la ecuación. Del análisis de regresión lineal de las tablas 5.12 y 5.13 se ha obtenido que las funciones que mejor se ajustan a los datos no corresponden a las dadas en la tabla 4.1. En este caso para el método multigrid la función que mejor se ajusta a los tiempos de cpu si es lineal $aN + b$, pero para el tiempo real es una potencia $(aN + b)^{10/7}$, para el método Gauss-Seidel los tiempos de cpu se ajustan a $(aN + b)^{10/6}$ y para el tiempo real $(aN + b)^2$, los valores de a y b para cada caso se encuentran en la tabla 5.14. Con esos valores se han graficado las curvas de ajuste en las gráficas correspondientes.

	Multigrid		Gauss-Seidel	
Parámetro	real	cpu	real	cpu
Correlación R^2	0.996	0.998	0.991	0.993
Pendiente a	0.0000015	0.0000027	0.000010	0.000012
Ordenada b	0.22	-0.34	0.45	0.38
Potencia	10/7	1	2	10/6

Tabla 5.14: Parámetros del análisis de regresión lineal aplicado a los datos de las tablas 5.12 y 5.13 tomando las raíces de la potencia respectiva para la columna de datos correspondiente.

Solución

La forma de la solución se muestra en la figura 5.17. Se grafican cortes de la solución, en donde se observa que en efecto se obtiene la solución construida. Las superficies en este caso presentan menos oscilaciones debido a que se construyó la solución con menos nodos. se observa también que la solución presenta ciertas simetrías, que como se verá más a delante, en este caso no se intuyen de la fuente. En el caso de la ecuación de Poisson, a cada pozo le correspondía una protuberancia y al contrario, que es un comportamiento típico de la ecuación.

La forma del coeficiente lineal utilizado se presenta en la figura 5.18, como se observa se utilizó una función suave y simétrica. Se muestran los tres cortes, aún cuando los tres son idénticos. Se observa que en muchas regiones el coeficiente tiene valores muy grandes (en valor absoluto) lo cual puede contribuir a una rápida convergencia. Para concluir eso hay que recordar que el método de relajación requiere obtener la solución a una ecuación polinomial utilizando el método de Newton, de esa forma se satisface el operador puntualmente.

Por último en la figura 5.19 se muestran los cortes de la fuente utilizada, se observa que en este caso la función es mucho más complicada que en caso de la ecuación de Poisson. Hay que notar que la función es muy plana cerca del origen, y los valores en esa zona son también pequeños.

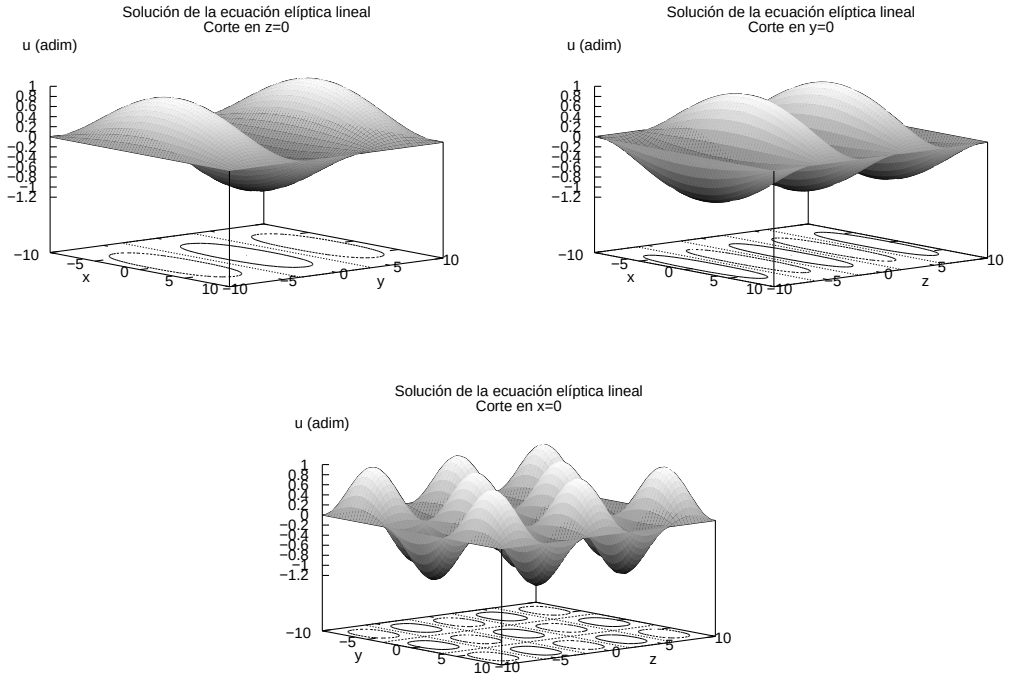

Figura 5.17: Cortes de la solución para la ecuación elíptica lineal. Se muestran tres cortes, los planos Z=0, Y=0 y X=0. Se observa que la solución es una función suave y con la forma esperada.

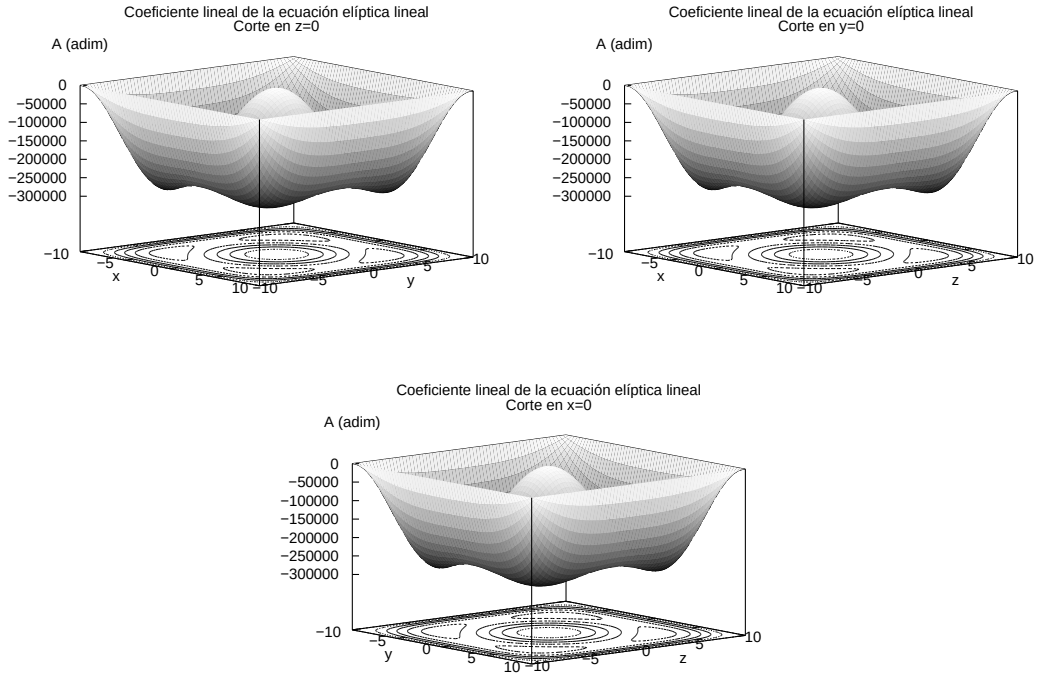

Figura 5.18: Cortes del coeficiente lineal $A(x, y, z)$ para la ecuación elíptica lineal. Se muestran tres cortes, los planos Z=0, Y=0 y X=0. Por construcción la función utilizada es simétrica por lo que todos los cortes son iguales.

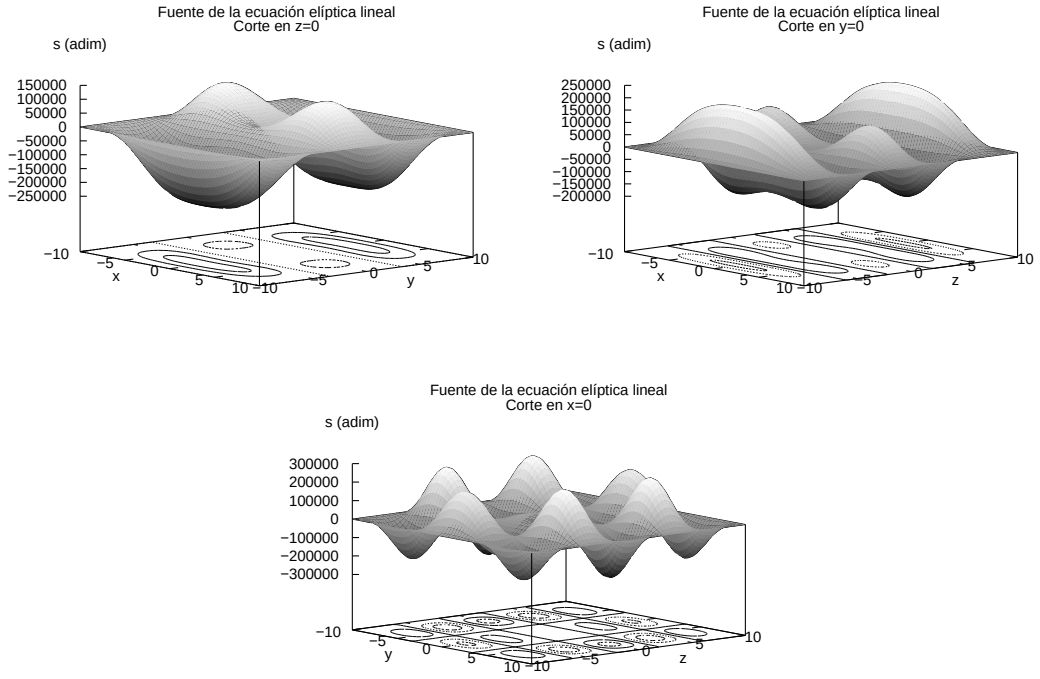

Figura 5.19: Cortes de la fuente usada para probar la ecuación elíptica lineal. Se muestran los mismos cortes: los planos Z=0, Y=0 y X=0. Se obtiene una fuente más o menos complicada, pero como ya se mostró, se obtiene la solución esperada.

Capítulo 6

Casos físicos

Una vez que se puede garantizar que el código genera soluciones correctas se procede a estudiar casos físicos de interés. Para ello se plantearon dos casos físicos

El primer problema es generar datos iniciales para ondas de Brill. Como se mostró en la sección 3.3, la ecuación resultante para este problema es una ecuación lineal sin fuente. Se escogió este problema debido a que es posible comparar los datos generados por otras espinas de CACTUS con los datos generados por Olliptic. Además las ecuaciones de constricción para el momento se satisfacen idénticamente y solo resta integrar la ecuación de constricción Hamiltoniana en un espacio plano.

El segundo problema a estudiar fue el sistema Schrödinger-Poisson. Para este problema se requiere obtener la solución de la ecuación de Poisson tomando como fuente la densidad de probabilidad de la ecuación de Schrödinger para un campo escalar. Este sistema es de interés desde el punto de vista de investigación y los resultados que se obtienen con el código Olliptic pueden dar lugar a estudios que no han sido publicados. Trabajos recientes se han realizado aprovechando simetrías esférica y axial [40, 13, 14, 15], para las pruebas se han utilizado esos resultados, en particular se han reproducido la configuración de equilibrio para el caso esféricamente simétrico y se ha comparado con los resultados presentados en esos trabajos.

6.1. Ondas de Brill

Las ondas de Brill son una solución axial simétrica de las ecuaciones de Einstein, para las cuales en algunos casos pueden aparecer horizontes aparentes [41], así mismo, han sido utilizadas como casos de prueba para códigos que generan datos iniciales. Como se mostró en la sección 3.3, para generar datos iniciales para ondas de Brill se debe resolver la constricción Hamiltoniana que toma la forma (3.36) que se presenta nuevamente aquí:

$$\bar{\nabla}^2\psi + \frac{1}{4}\Big(\partial_\rho^2 q + \partial_z^2 q\Big)\psi = 0, \tag{6.1}$$

en donde ψ es un factor conforme a una métrica plana, ρ y z son las coordenadas cilíndricas. Además la función q debe cumplir:

$$q\big|_{\rho=0} = 0, \tag{6.2}$$

$$\partial_\rho^n q\big|_{\rho=0} = 0, \tag{6.3}$$

$$q\big|_{r\to\infty} = \mathcal{O}(r^{-2}), \tag{6.4}$$

en donde n es un entero impar y $r = \sqrt{\rho^2 + z^2}$ es la coordenada radial. En trabajos previos de miembros del grupo [42, 43] se han utilizado dos tipos de funciones q para realizar pruebas, una de ellas fue inicialmente considerada por Eppley [44]:

$$q = a\frac{\rho^2}{1 + \left(\frac{r}{\lambda}\right)^n}, \tag{6.5}$$

en donde a y λ son parámetros constantes y $n \geq 4$. Otra función es la que consideró Holz [45]:

$$q = a\rho^2 \mathrm{e}^{-r^2}, \tag{6.6}$$

en donde nuevamente a es un parámetro constante, las ecuaciones para q dadas por (6.5) y (6.6) han sido implementadas en el código, sin embargo para las pruebas solo se ha utilizado la última, con una amplitud $a = 1$.

Se realizaron pruebas para en un dominio cúbico de lado 12.8 usando en total 12 resoluciones con particiones homogéneas: $\Delta x = 0.4$, $\Delta x = 0.25$, $\Delta x = 0.2$, $\Delta x = 0.18$, $\Delta x = 0.16$, $\Delta x = 0.13$, $\Delta x = 0.12$, $\Delta x = 0.11$, $\Delta x = 0.1$, $\Delta x = 0.09$, $\Delta x = 0.08$ y $\Delta x = 0.06$. Además en este caso se han usado condiciones de frontera tipo Robin (ver sección 4.4), que garantizan que el factor conforme es de orden $\mathcal{O}(r^{-2})$ en infinito.

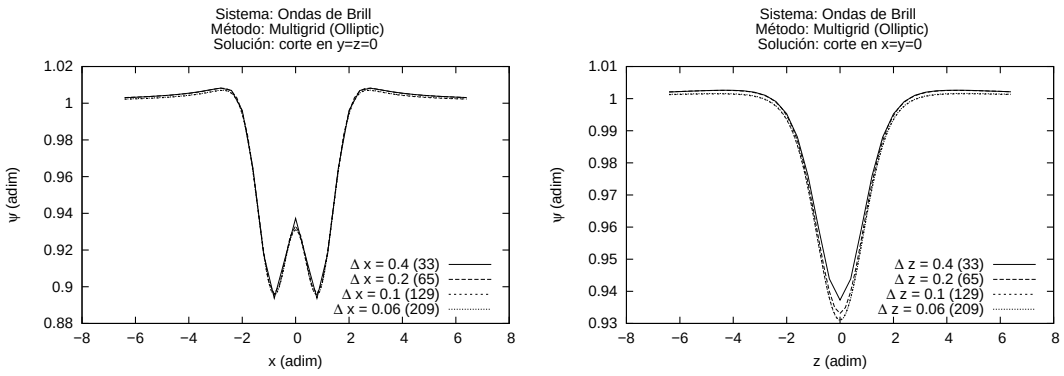

Figura 6.1: Gráfica de la solución para el factor conforme $\psi(\rho, z)$ de las ondas de Brill. Se utilizó para este caso la función de Holz con una amplitud $a = 1$. Como el sistema presenta simetría axial solo se presentan dos cortes.

Nuevamente se han hecho las pruebas correspondientes de convergencia, en la figura 6.1 se muestran gráficas del factor conforme obtenido en dos direcciones que pasan por el origen, es decir son las gráficas de $\psi(\rho, 0)$ y $\psi(0, z)$. Se observa una muy buena correspondencia con excepción

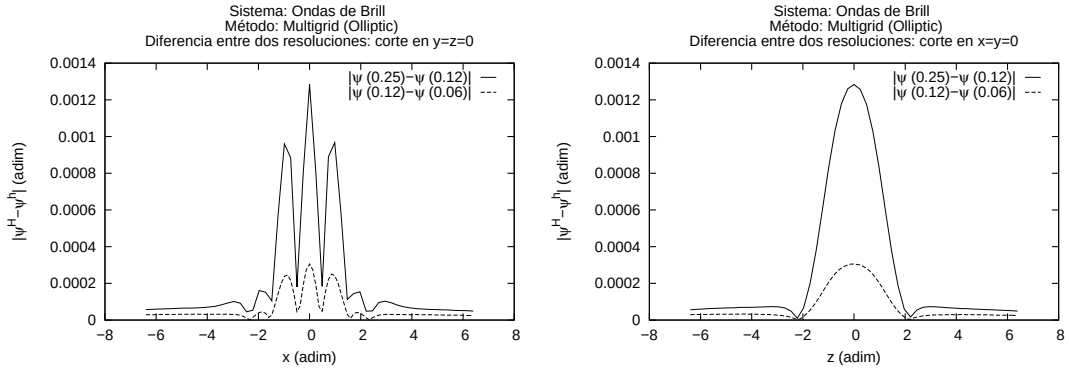

Figura 6.2: Gráfica del error global del factor conforme para ondas de Brill usando cuatro resoluciones. Se observa que el error disminuye al aumentar el número de puntos en la partición.

de la resolución más baja que se distingue de las demás. Cabe recalcar que para esta solución se utilizaron fronteras tipo Robin.

Con las soluciones obtenidas para 3 resoluciones se han hecho las pruebas de convergencia usando proporciones 2:1, se han seleccionado las soluciones con resolución $\Delta x = 0.25$, $\Delta x = 0.12$ y $\Delta x = 0.06$ que son las que mayor rango abarcan. En la figura 6.2 se muestra la gráfica de la diferencia entre las soluciones, es decir $\psi^h - \psi^{2h}$ y $\psi^{2h} - \psi^{4h}$. En este caso se observa que la diferencia ya no es una función suave como en los casos de prueba, sin embargo si decrece al aumentar la resolución lo cual supone convergencia.

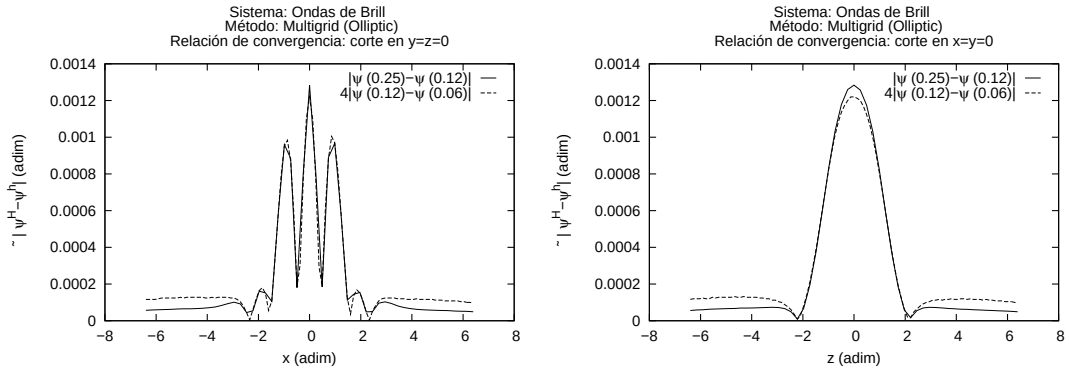

Figura 6.3: Prueba de convergencia del factor conforme para ondas de Brill usando tres resoluciones. Se observa convergencia a segundo orden ya que las gráficas, en gran parte se superponen al ser multiplicados sus valores por factores adecuados al orden de convergencia (en este caso por ser de segundo orden se multiplica por 4) y solo cerca de la frontera se observa que se pierde un poco de precisión.

En la figura 6.3 se presenta la prueba de convergencia. Se observa que al multiplicar las diferencias entre soluciones por el factor adecuado las gráficas se superponen, excepto en las fronteras, en donde si bien se observa convergencia, no parece que el error disminuya al mismo ritmo. La

implementación correcta de condiciones de frontera numéricamente es un problema complicado. Para condiciones tipo Dirichlet en donde se da el valor de la solución en la frontera, no existen muchos problemas ya que aún cuando se definan, por ejemplo, condiciones que no son suaves (pero son continuas) o en dominios complicados (en nuestro caso un cubo, en donde aristas y esquinas pueden generar soluciones que no son suaves), la solución en el interior del dominio toma los valores dados por el operador en diferencias. Para condiciones de frontera tipo Robin o Newmann en donde esta involucrada la derivada de la función no se puede garantizar que las fronteras no introducirán errores adicionales. Esto debido a que la condición se da sobre la derivada que es en sí misma una aproximación.

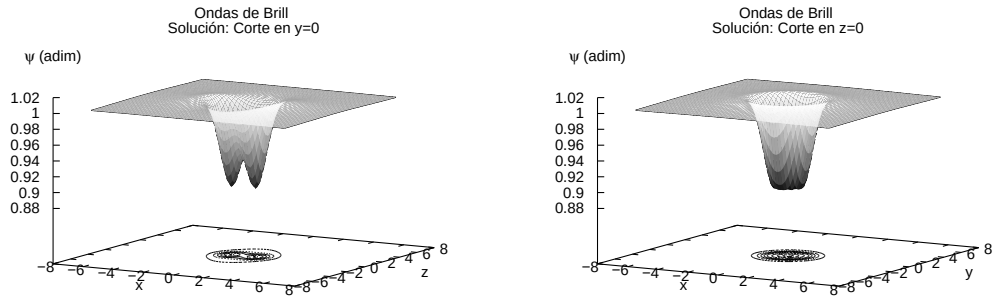

Figura 6.4: Cortes del factor conforme para ondas de Brill. Se muestran cortes con los planos Y=0 y Z=0.

Cortes de la solución en los planos $x = 0$ y $z = 0$ se presentan en la figura 6.4. La característica de simetría que presenta el factor conforme dan gráficas practicamente idénticas para los cortes en $x = 0$ y $y = 0$, por lo que solo se presenta la gráfica $x = 0$, cualquier otra gráfica en torno a z presentará la misma forma. En este tipo de gráficas se aprecia mejor el decaimiento de la solución en las fronteras.

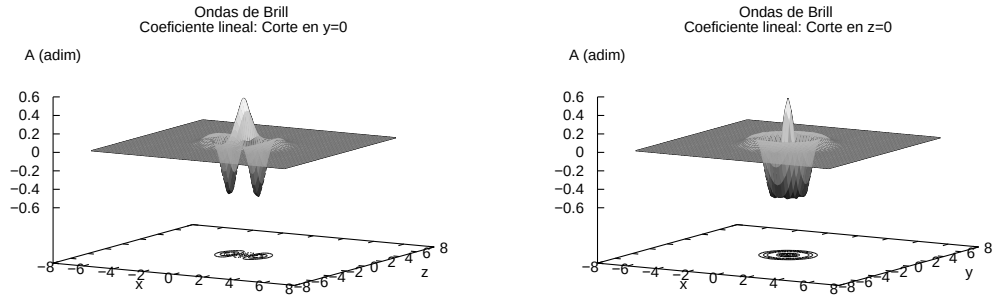

Figura 6.5: Cortes del coeficiente lineal para ondas de Brill. Es el factor lineal el que genera una solución no trivial debido a que para este caso no hay fuentes.

Por último en la figura 6.5 se muestran cortes del coeficiente lineal de la ecuación:

$$A = \frac{1}{4}\Big(\partial_\rho^2 q + \partial_z^2 q\Big). \tag{6.7}$$

Para construir este factor se han calculado las derivadas analíticamente y se han implementado en el código. Dado que en la ecuación no existen fuentes (términos independientes de ψ) el factor lineal, junto con las condiciones de frontera, determinan completamente la forma de la solución. Esa es otra característica que las ondas de Brill permiten probar para el código Olliptic ya que en las pruebas para la ecuación lineal se había construido de forma explicita una fuente para obtener la solución.

Generación de datos iniciales para ondas de Brill es una de las pruebas básicas de los resolvedores elípticos en CACTUS, como ya se mencionó CACTUS tiene un resolvedor elíptico Multigrid llamado BAM_Elliptic. Olliptic es un código que tiene como objetivo generalizar y mejorar las funciones de BAM_Elliptic por lo que una comparación directa entre BAM_Elliptic y Olliptic parece ser una prueba necesaria, sin embargo al tratar de utilizar BAM_Elliptic para generar datos iniciales para ondas de Brill no fue posible obtener una convergencia adecuada, además de que se presentaron problemas al variar el tamaño del dominio. Como BAM_Elliptic es una espina que no ha sido actualizada desde hace tiempo, se procedió a comparar Olliptic con otra espina de acceso público que resuelve datos iniciales en CACTUS. La espina a la que me refiero es EllSOR, la cual implementa un método de sobre relajación sucesiva (ver sección 4.3) y con la cual sí fue posible obtener datos que presentaron convergencia.

En la figura 6.6 se muestran las gráficas de la prueba de convergencia para la generación de datos iniciales para ondas de Brill con los mismos parámetros que los datos generado con Olliptic, pero usando la espina EllSOR. Como se puede observar, se cumple bien la relación de convergencia a segundo orden. En la figura 6.7 se presentan las soluciones generadas por Olliptic y por EllSOR para tres resoluciones altas, se observa que las mayores diferencias se presentan en la frontera. En la figura 6.8 se presenta la gráfica de la diferencia entre los datos generados por EllSOR y Olliptic, es decir $(\psi_{\mathrm{Oll}} - \psi_{\mathrm{Ell}})$, sabemos que los datos pasaron las pruebas de convergencia, pero además, esta gráfica indica que los datos convergen entre si.

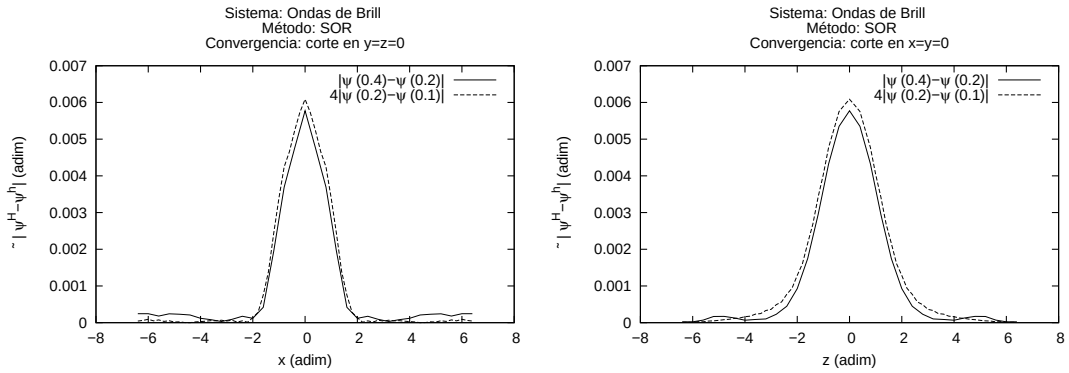

Figura 6.6: Prueba de convergencia para el factor conforme de datos iniciales para ondas de Brill generados por la espina EllSOR.

Con las pruebas antes realizadas podemos suponer que los datos generados por los códigos son comparables, por lo que se procedió a medir la eficiencia relativa, es decir se comparan los

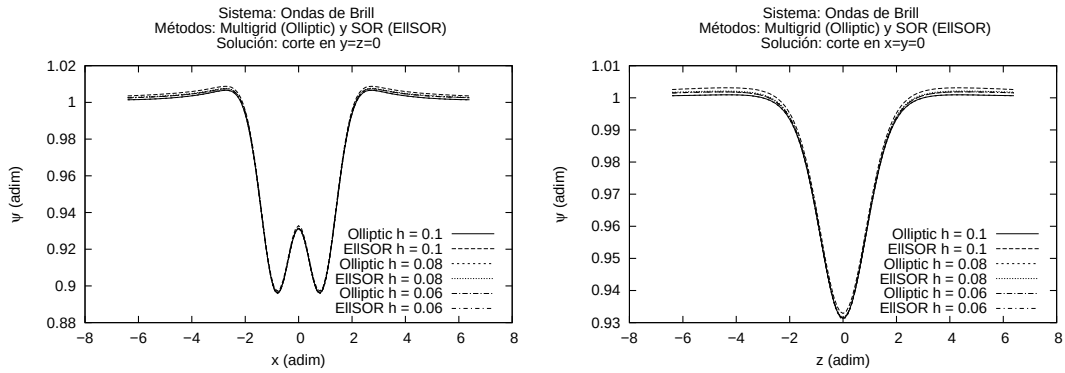

Figura 6.7: Soluciones generadas por la espina EllSor y Olliptic para tres resoluciones. Se observa que la diferencia entre los datos disminuye al aumentar la resolución lo cual indica convergencia entre los datos generados por los dos códigos.

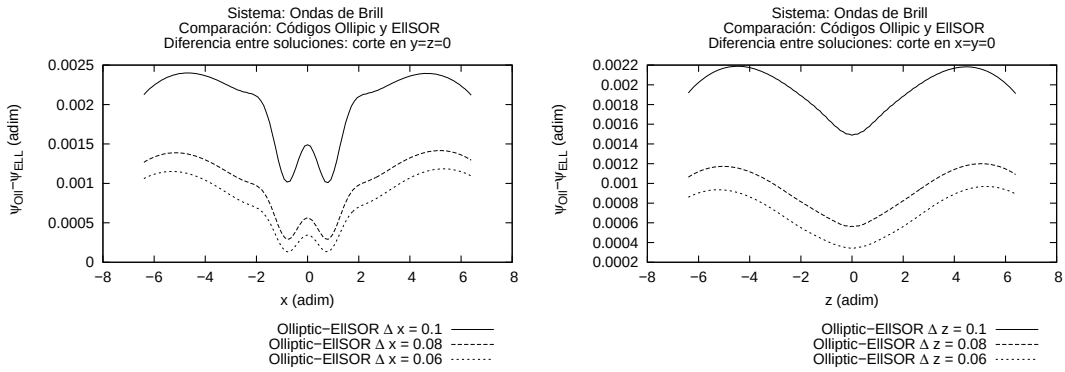

Figura 6.8: Diferencia entre los datos generados por la espina EllSor y los generados por Olliptic para tres resoluciones. Se observa que la diferencia entre los datos disminuye al aumentar la resolución lo cual indica convergencia entre los datos generados por los dos códigos. La zona que decrece más lento es la frontera.

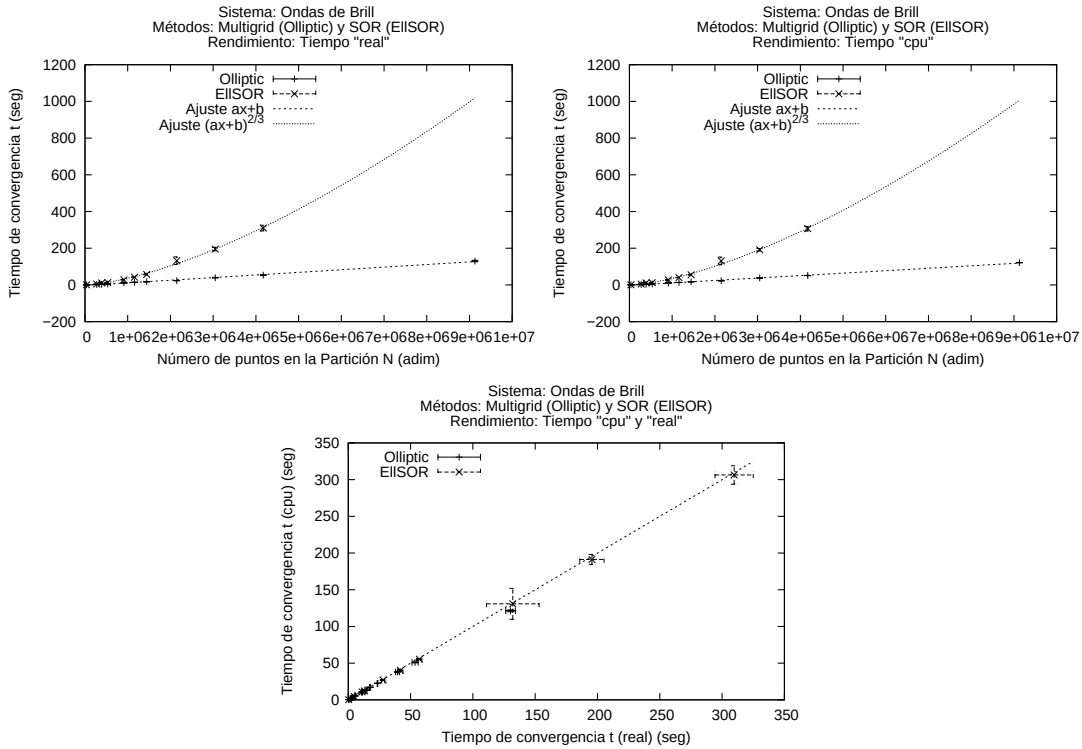

Figura 6.9: Gráfica del tiempo de ejecución real y de cpu para la espina EllSOR y Olliptic, para varias resoluciones resolviendo el mismo sistema para generación de datos iniciales para ondas de Brill. En la gráfica se comparan los tiempos real y de cpu como función del número de puntos en la partición y se comparan con las funciones de ajuste.

		Tiempo (seg) EllSOR		Tiempo (seg) Olliptic	
$\Delta[xyz]$	N_{Total}	real	cpu	real	cpu
0.40	35937	0.359 ± 0.0025	0.354 ± 0.0042	0.375 ± 0.0022	0.372 ± 0.0044
0.20	274625	4.825 ± 0.070	4.57 ± 0.042	2.118 ± 0.062	2.06 ± 0.033
0.18	389017	11.1 ± 0.20	10.8 ± 0.11	3.504 ± 0.21	3.19 ± 0.057
0.16	531441	13.1 ± 0.70	12.2 ± 0.29	5.437 ± 0.032	5.38 ± 0.022
0.13	912673	28 ± 1.32	27.0 ± 0.25	10.992 ± 0.40	10.6 ± 0.11
0.12	1157625	41 ± 1.92	40.1 ± 0.67	14.201 ± 0.82	13.4 ± 0.26
0.11	1442897	57 ± 1.99	55.3 ± 0.67	17.528 ± 0.45	17.1 ± 0.14
0.10	2146689	132 ± 21	130 ± 21	23.460 ± 0.30	22.54 ± 0.044
0.09	3048625	195 ± 9.7	191 ± 6.7	39.544 ± 1.5	38.1 ± 0.67
0.08	4173281	309 ± 15	306 ± 12	53.616 ± 2.4	50.9 ± 0.57
0.06	9129329	-	-	130 ± 3.8	121 ± 1.2

Tabla 6.1: Comparación entre el tiempo real y de cpu que toma a las espinas EllSOR y Olliptic resolver la ecuación para generar datos iniciales para ondas de Brill en 1 procesador. Se observa que a resoluciones bajas, el rendimiento es similar, al aumentar la resolución el código Olliptic mejora mucho respecto a EllSOR. Para EllSOR no se obtuvo el último valor debido a que hacer las seis corridas llevaría demasiado tiempo.

tiempos necesarios para alcanzar datos con la misma tolerancia en la norma del residuo. En la tabla 6.1 se presentan los resultados de los tiempos medidos para 11 resoluciones con 6 corridas para cada resolución y una resolución más alta para el caso de Olliptic que no fue posible obtener para EllSOR. Nuevamente se presenta como medida de incertidumbre la desviación estándar de los tiempos medidos y como valor representativo el promedio de las corridas. En la figura 6.9 se presenta la gráfica correspondiente a la tabla 6.1; en las gráficas se observa claramente la diferencia en rendimiento de los dos códigos resolviendo el mismo problema y nuevamente se muestra que Olliptic puede resolver la ecuación de forma más eficiente, además se presentan las funciones de ajuste.

En este caso obtenemos de la tabla 4.1 que el método SOR tiene una eficiencia de orden 3/2 como función del número total de puntos en la partición, mientras que como ya se ha mencionado el método Multigrid es de orden lineal. En las gráficas de rendimiento de la figura 6.9 se observa una concordancia con el análisis de convergencia de esos métodos. Nuevamente se corroboró la relación aplicado un análisis de regresión lineal a los datos de las tablas, en el caso del método SOR tomando potencias 2/3 de los tiempos. En la tabla 6.2 se muestran los parámetros obtenidos.

En las gráficas se usan funciones de ajuste $ax + b$ para el método Multigrid y $(ax + b)^{3/2}$ para el método SOR que muestran una muy buena concordancia.

Hasta aquí se han resuelto problemas con parámetros construidos especificamente para probar el código, en forma individual y comparando con EllSOR. En todos los casos se han mostrado las ventajas en rendimiento que presenta Olliptic, en la siguiente sección se presenta un problema que requiere un mayor trabajo de cálculo debido a que se debe resolver la ecuación de Poisson en tiempo de evolución, es decir a cada paso de tiempo. Así mismo el sistema Schrödinger-Poisson representa un problema con aplicaciones actuales en investigación.

	SOR (EllSOR)		Multigrid (Olliptic)	
Parámetro	real	cpu	real	cpu
Correlación R^2	0.996	0.995	0.996	0.997
Pendiente a	0.000011	0.000011	.000014	0.000013
Ordenada b	-0.13	-0.25	-2.81	-2.23

Tabla 6.2: Parámetros del análisis de regresión lineal aplicado a los datos de la tabla 6.1 tomando potencias 2/3 de los tiempos en el caso del método SOR. Hay que recordar que los valores del coeficiente R^2 cercanos a 1 indican una correlación lineal.

6.2. Sistema Schrödinger-Poisson

Uno de los problemas fundamentales de la cosmología actual y en el cual existe un intenso trabajo de investigación es determinar la naturaleza de la materia obscura. El modelo cosmológico más exitoso se conoce como "Λ Cold Dark Matter" o modelo de materia obscura fría con constante cosmológica Λ. Las partículas supersimétricas son los candidatos más aceptados para representar materia obscura, sin embargo existen otros modelos que se han propuesto como candidatos para conformar la materia obscura, en particular se han propuesto partículas asociadas a un campo escalar [11, 46]. Hasta el momento no se ha podido determinar cual es el modelo que mejor describe el fenómeno físico, o bien detectar experimentalmente ese tipo de partículas. En esta sección de la tesis se explorarán algunas aplicaciones del código Olliptic para modelar el comportamiento de cúmulos de campo escalar sujetos a campos gravitacionales.

Existen trabajos teóricos y experimentos numéricos enfocados a determinar las características de la materia obscura tipo campo escalar. Una de las propiedades más importantes de este modelo es que los campos escalares pueden formar condensados de Bose, que a escalas astrofísicas generan estructuras [47, 48, 49, 50, 46]; numéricamente se ha estudiado esta propiedad bajo el régimen relativista para campos escalares complejos y reales [51, 52, 53, 54]. Otra propiedad importante del modelo de materia obscura tipo campo escalar es que las configuraciones de equilibrio mantienen un perfil de densidad suave en el origen, lo cual es una ventaja respecto a los modelos de partículas que han mostrado que generan perfiles de densidad con cúspides en el origen.

En esta sección se estudia la evolución de un campo escalar bajo la influencia de un campo gravitacional débil. La descripción de este sistema requiere resolver un sistema acoplado conformado por la ecuación de Schrödinger bajo la influencia del potencial gravitacional y la ecuación de Poisson con la densidad de la función de onda como fuente. El sistema de ecuaciones a resolver conforma una prueba para el resolvedor elíptico ya que se requiere resolver una ecuación elíptica a cada paso de tiempo de forma eficiente. Para métodos simples como los de relajación se convierte en una tarea muy lenta si se desea evolucionar sistemas en tres dimensiones y sin simetrías ya que los tiempos de convergencia de la solución son demasiado grandes en sistemas con buena resolución y considerando que se debe obtener la solución a cada paso de tiempo la evolución toma demasiado tiempo de cómputo.

Para probar el código se utilizan algunos resultados presentados en [40], en ese artículo se realizan varios experimentos numéricos para configuraciones con simetría esférica y configuraciones de equilibrio. Una vez que se demuestra que el código reproduce los resultados realizados con los códigos que aprovechan la simetría esférica de las configuraciones de equilibrio, se procede a hacer una evolución para dos distribuciones que colisionan en órbita. El trabajo realizado para esta sección tiene como antecedentes los trabajos dados en las referencias [40, 13, 14, 15]. El código

utilizado para resolver la ecuación de Schrödinger fue escrito por Francisco Siddhartha Guzmán de la Universidad Michoacana de San Nicolás de Hidalgo, con algunas modificaciones que hice bajo su supervisión.

6.2.1. Límite de campo débil del sistema Einstein-Klein-Gordon

El tensor de energía-momento para un campo escalar ϕ esta dado por:

$$T_{ab} = \frac{1}{2}\Big[\phi_{,a}\phi^*_{,b} + \phi^*_{,a}\phi_{,b} - g_{ab}(\phi^{,c}\phi^*_{,c} + m^2|\phi|^2)\Big], \tag{6.8}$$

en donde $|\phi|^2 = \phi\phi^*$, m denota la masa de las partículas asociadas al campo escalar y el asterisco denota el operador de conjugación compleja. De la conservación del tensor de energía-momento ${T^{ab}}_{;b} = 0$, se obtienen la ecuaciones de Klein-Gordon:[1]

$$\Box\phi - m^2\phi = 0. \tag{6.9}$$

Aquí $\Box := (1/\sqrt{-g})\partial_a[\sqrt{-g}g^{ab}\partial_b]$ es el operador d´Alambertiano covariante. Estas ecuaciones junto con las ecuaciones de Einstein:

$$G_{ab} = 8\pi T_{ab}, \tag{6.10}$$

dan la evolución del campo escalar en un campo gravitacional.

Para un campo gravitacional débil, utilizando un desarrollo post-Newtoniano, la ecuación de Klein-Gordon tiene como límite la ecuación de Schrödinger:

$$i\frac{\partial\psi}{\partial t} = -\nabla^2\psi + U\psi + \Lambda|\psi|^2\psi, \tag{6.11}$$

en donde λ es un parámetro de acoplamiento. Las ecuaciones de Einstein tienen como límite la ecuación de Poisson:

$$\nabla^2 U = \rho, \tag{6.12}$$

en donde $\psi = \psi(x, y, z, t)$ y $U = U(x, y, z, t)$ son la función de onda asociada al campo escalar y el potencial gravitacional generado por la distribución respectivamente y $\rho = \psi\psi^*$ es la densidad de campo escalar. En este régimen es posible calcular valores esperados para magnitudes físicas, de forma análoga a como se hace en mecánica cuántica. Se definen la siguientes cantidades:

$$K = -\frac{1}{2}\int \psi^*\nabla^2\psi \, \mathrm{d}^3x, \tag{6.13}$$

$$W = \frac{1}{2}\int \psi^* U\psi \, \mathrm{d}^3x, \tag{6.14}$$

$$I = \int \Lambda|\psi|^4 \mathrm{d}^3x, \tag{6.15}$$

$$M = \int \rho \, \mathrm{d}^3x, \tag{6.16}$$

que corresponden a la energía cinética, la energía potencial y la masa total del sistema respectivamente. Con estas cantidades podemos obtener importantes propiedades del sistema, por ejemplo la energía total $E := K + W$ y la masa total M determinan si el sistema es cerrado (en tal caso esas cantidades permanecen en un valor constante) o bien se trata de un sistema abierto. Otra relación importante es la de virialización $2K + W + 3I$ [55], que para sistemas estacionarios tiende a cero.

[1] En la ecuación anterior se denota la derivada covariante con ; y en lo que sigue se usarán unidades en donde la constante de Planck $\hbar$, la velocidad de la luz c y la constante de gravitación universal G son cantidades unitarias.

6.2.2. Configuración de equilibrio esféricamente simétrica

Para probar el código se usará como base una configuración esféricamente simétrica de equilibrio que ha sido bien estudiada en [40]. El sistema Schrödinger-Poisson con simetría esférica y $\Lambda = 0$ tiene la forma:

$$i\frac{\partial\psi}{\partial t} + \frac{1}{2r}\frac{\partial^2}{\partial r^2}(r\psi) = U\psi, \tag{6.17}$$

$$\frac{1}{r}\frac{\partial^2}{\partial r^2}(r\psi) = \rho, \tag{6.18}$$

en donde $r = \sqrt{x^2+y^2+z^2}$ es la coordenada radial. Si suponemos que la solución se descompone como el producto de dos funciones (ver sección 4.2), $\psi = \varphi(r)\mathrm{e}^{-i\omega t}$, imponiendo condiciones de regularidad para φ en el origen, $\varphi(0) < \infty$ y $\partial_r\varphi(0) = 0$, y suponiendo también que el campo se anula en infinito, $\varphi(r \to \infty) = 0$, obtenemos el sistema de valores propios:

$$\frac{\mathrm{d}^2}{\mathrm{d}r^2}(r\varphi) = 2r(U-\omega), \tag{6.19}$$

$$\frac{\mathrm{d}^2}{\mathrm{d}r^2}(rU) = r\varphi^2. \tag{6.20}$$

Para un valor central $\varphi(0) = 1$ existe un solo valor de ω que cumple las condiciones de frontera. El sistema se resuelve numéricamente con un "método de disparo" (*shooting method* en ingles) [35]. Se obtiene una familia de soluciones que representan diferentes funciones de onda las cuales contienen un espectro discreto de nodos. En el artículo citado se muestra que la configuración con 0 nodos es un estado base, es decir, cualquier configuración con 1 o más nodos es una configuración de equilibrio inestable y decae a la configuración con 0 nodos emitiendo materia escalar.

En la figura 6.10 se muestra la gráfica de la función obtenida al resolver el sistema (6.19-6.20), hay que notar que $\psi(\tau = 0, r) = \varphi(\tau = 0, r)$. Se utiliza ese perfil como forma inicial para la función de onda, y se obtiene según [40] los valores mostrados en la tabla 6.3.

Condiciones de frontera

Las condiciones de frontera son una de las modificaciones que se hicieron al código que resuelve la ecuación de Schrödinger debido a que no se habían implementado condiciones de frontera adecuadas. Se implementaron condiciones de frontera usando una "esponja" que básicamente consiste en agregar un potencial con parte imaginaria dada por la función:

$$V_{im} = -i\frac{1}{2}V_0\left[2 + \tanh\left(\frac{r-r_c}{\delta}\right) - \tanh\left(\frac{r_c}{\delta}\right)\right], \tag{6.21}$$

que es una versión suave de una función escalón con amplitud V_0, ancho r_c y longitud de la transición δ. En la figura 6.11 se muestra un corte para $Z = 0$, con $V_0 = 1$, $r_c = 0.8$ y $\delta = 0.1$. El efecto que tiene este tipo de potencial es el de atrapar partículas de campo escalar en una región imaginaria.

Para la ecuación de Poisson se toma como condición de frontera el término monopolar (ver sección 4.4):

$$U(\vec{x}) = -\frac{M}{\|\vec{x}\|}, \qquad \vec{x} \in \partial\Omega, \tag{6.22}$$

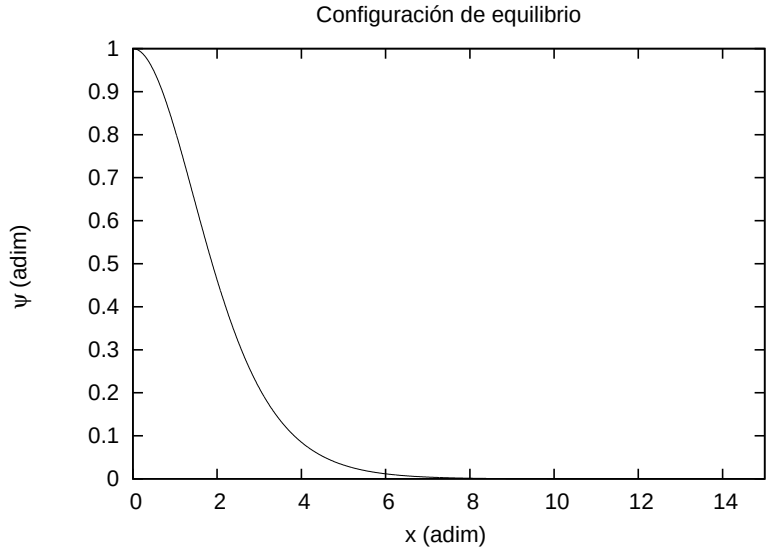

Figura 6.10: Perfil de la función de onda para una configuración de equilibrio.

ω	M	$U(x=0)$	K	W
-0.69223	2.0622	-1.3418	0.47585	-0.95169

Tabla 6.3: Valores esperados de la configuración de equilibrio sin nodos para el sistema Schrödinger-Poisson, publicados en [40].

en donde M es la masa total del condensado de campo escalar. Esta condición supone que toda la masa se encuentra confinada en el interior del dominio Ω, lo cual implica que para la función de onda se debe cumplir $|\psi(\tau, \vec{x})| \to 0$ para $\vec{x} \in \partial\Omega$. Por construcción la configuración de equilibrio presentada anteriormente cumple esta condición y como se mostrará más adelante, la evolución de esta configuración preserva la condición de frontera para la función de onda.

Evolución de una configuración de equilibrio

En esta sección se presentan los resultados de evolucionar la configuración de equilibrio mostrada en la figura 6.10. Se realizaron tres evoluciones en un dominio cúbico de lado 60. Se escogió este dominio, relativamente grande, debido a que será en este mismo en el que se realizarán las pruebas de evolución para dos densidades de campo escalar colisionando en orbita. La evolución se hizo hasta un tiempo $\tau = 30$, y se uso el potencial imaginario (6.21) con $V_0 = 1$, $r_c = 28$ y $\delta = \Delta x$.

Las resoluciones utilizadas son particiones homogéneas con 64, 129 y 257 puntos, las dos primeras se hicieron en 1 procesador y la última se hizo en 8 procesadores.[2] En particular la resolución más baja contiene muy pocos puntos para el tamaño de partición y como se verá más adelante no describe correctamente el estado estacionario, sin embargo, es útil para observar la convergencia.

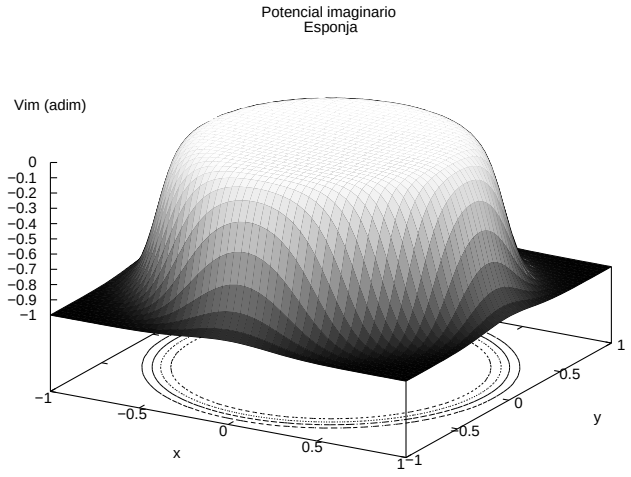

Figura 6.11: Potencial imaginario utilizado como esponja. El potencial es una versión suave de una función escalón radial con profundidad V_0, ancho r_c y longitud de la transición δ

Los resultados obtenidos se muestra a continuación. En la figura 6.12 se muestra la gráfica para las tres resoluciones de la energía total E y de la relación de virialización $2K + W$. La energía en los tres casos se observa que tiende a un valor constante, de aproximadamente -0.5. el valor negativo indica que se trata de un sistema confinado, y solo la resolución más baja presenta oscilaciones evidentes, lo cual se espera debido a que para el tamaño del dominio se ha tomado una partición con muy pocos puntos. La relación de virialización en los casos de las resoluciones más altas tienden rápidamente a 0 lo cual debe ocurrir para una configuración de equilibrio. Para la resolución más

[2]Todas ellas en el cluster de computadoras Ek-Bek del grupo de relatividad numérica del ICN-CINVESTAV

baja se observan oscilaciones muy amplias, que decrecen lentamente en amplitud. Es claro que la partición con menos puntos no describe correctamente el problema ya que no reproduce el comportamiento esperado.

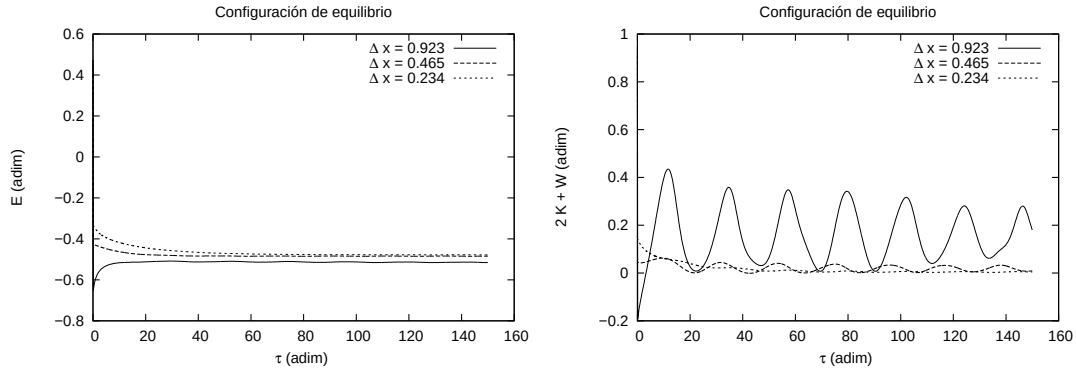

Figura 6.12: Gráfica de $E = K + W$ y $2K + W$ como función del tiempo τ para una configuración de equilibrio y tres resoluciones. En todos los casos la energía llega a un estado más o menos estacionario el cual se mantiene constante al aumentar la resolución. Por otro lado, la relación de virialización presenta un comportamiento de convergencia más notorio al tender a 0 al aumentar la resolución.

En la figura 6.13 se muestran los valores de la energía cinética K, la energía potencial W y la masa total M de la configuración. Los valores se comparan con los resultados de la tabla 6.3, obtenidos con un código que evoluciona las ecuaciones suponiendo simetría esférica. Nuevamente se observa que los valores para las resoluciones altas tienen oscilaciones que decrecen tendiendo al valor de referencia. El valor de la masa en los tres casos se mantiene casi constante, lo cual indica que no hay perdida de masa (las variaciones son del orden de 0.1 % en el peor de los casos).

Otros valores que se pueden comparar con el caso esférico es el valor mínimo del potencial que se da en la tabla 6.3 y el máximo de la densidad, que para una configuración de equilibrio debe permanecer en 1. Estrictamente la densidad no debe variar en absoluto y solo la función de onda ψ debe presentar oscilaciones, sin embargo, por tratarse de una aproximación numérica siempre existen pequeñas variaciones. En la figura 6.14 se muestran las gráficas comparativas entre los valores de referencia y los resultados obtenidos para las tres resoluciones de prueba. Se observa nuevamente que los valores oscilan con amplitudes que decrecen y que convergen al valor de referencia al aumentar la resolución.

De los resultados mostrados se puede concluir que el código Olliptic es útil para resolver ecuaciones elípticas durante una evolución y que de los resultados que se pueden obtener de él son útiles para un trabajo de investigación en el área de la relatividad numérica.

6.2.3. Evolución de dos distribuciones en colisión orbital

Una aplicación importante en el caso del sistema Schrödinger-Poisson es la evolución de dos distribuciones a las cuales se les proporciona un momento angular respecto del centro de masa de las mismas, de forma que orbitan y posteriormente colisionan debido a su propia interacción gravitacional.

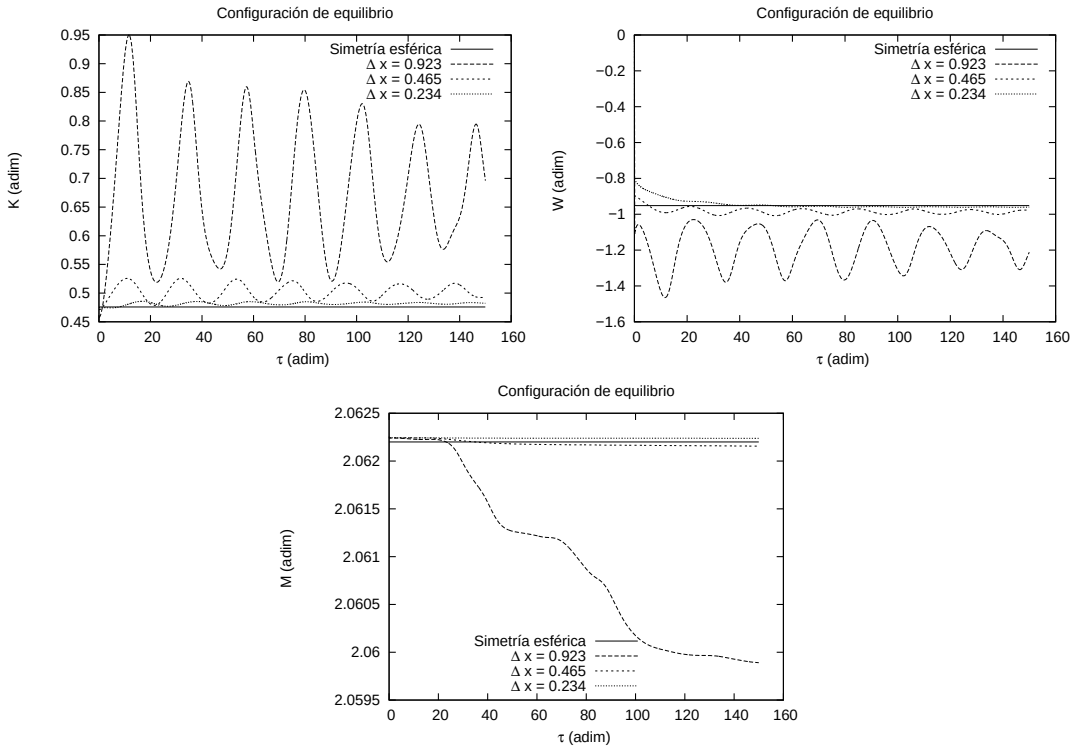

Figura 6.13: Gráficas de la energía cinética K, potencial W y cantidad de masa M para una configuración de equilibrio y tres resoluciones. Se comparan los valores obtenidos con el código Olliptic con los obtenidos con un código con simetría esférica, se observa que al aumentar la resolución los resultados convergen a los del caso esférico.

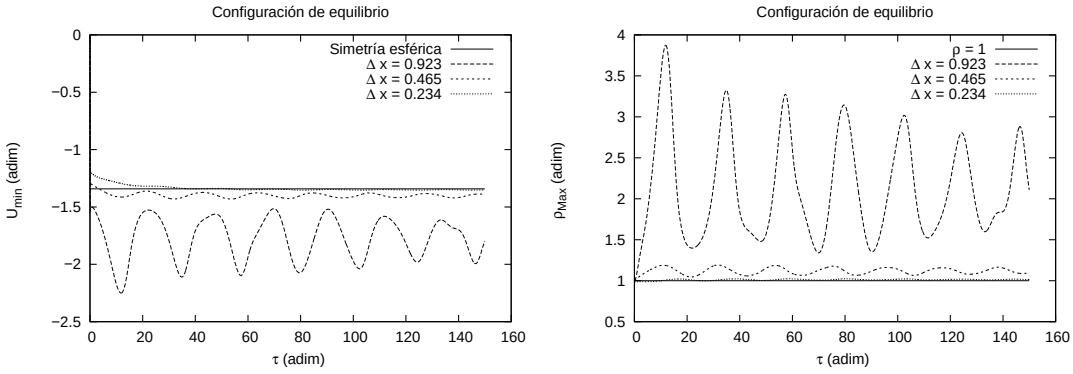

Figura 6.14: Gráficas del mínimo del potencial U y del valor máximo de ρ para una configuración de equilibrio y tres resoluciones. Para el mínimo del potencial también se tiene un valor de referencia del caso esférico y en el caso del máximo de la densidad por tratarse de una configuración estacionaria debe permanecer en el valor inicial es decir en $\rho_{max} = 1$

Para evolucionar este tipo de sistema se construye una distribución inicial que es la superposición de dos configuraciones de equilibrio suficientemente alejadas de tal forma que interfieran poco. La función de onda total debida a dos configuraciones de equilibrio ψ_1 y ψ_2 es $\psi = \psi_1 + \psi_2$ y su densidad $|\psi_1 + \psi_2|^2$. Para medir la interferencia se calcula $< (\psi_1, \psi_2) >$ y se escoge la distancia para que el valor de esta magnitud sea menor que el orden de aproximación de los calculos.

Para dotar de momento a las distribuciones se utiliza un operador de momento cuántico: $\psi_1 \rightarrow e^{i\vec{p}_1 \cdot \vec{r}} \psi_1$ y $\psi_2 \rightarrow e^{i\vec{p}_2 \cdot \vec{r}} \psi_2$. El efecto es una variación en los valores esperados del momento en la dirección de los vectores $\vec{p}_1$ y $\vec{p}_2$.

En la espina Schrödinger se ha implementado la posibilidad de construir dos distribuciones de la función de onda separadas por una distancia que para la evolución realizada fue de 26 unidades en el eje Z y que además están moduladas por la aplicación de un operador de momento. Para la evolución analizada se dio un momento lineal en la dirección Y para una de las distribuciones +0.1 unidades y para la otra -0.1 unidades.

Se utilizó un dominio cúbico de lado 60, una resolución con $\Delta x = 0.469$ evolucionando con 8 procesadores en el cluster Ek-Bek del Laboratorio de Supercómputo Astrofísico y hasta un tiempo $\tau = 1000$.

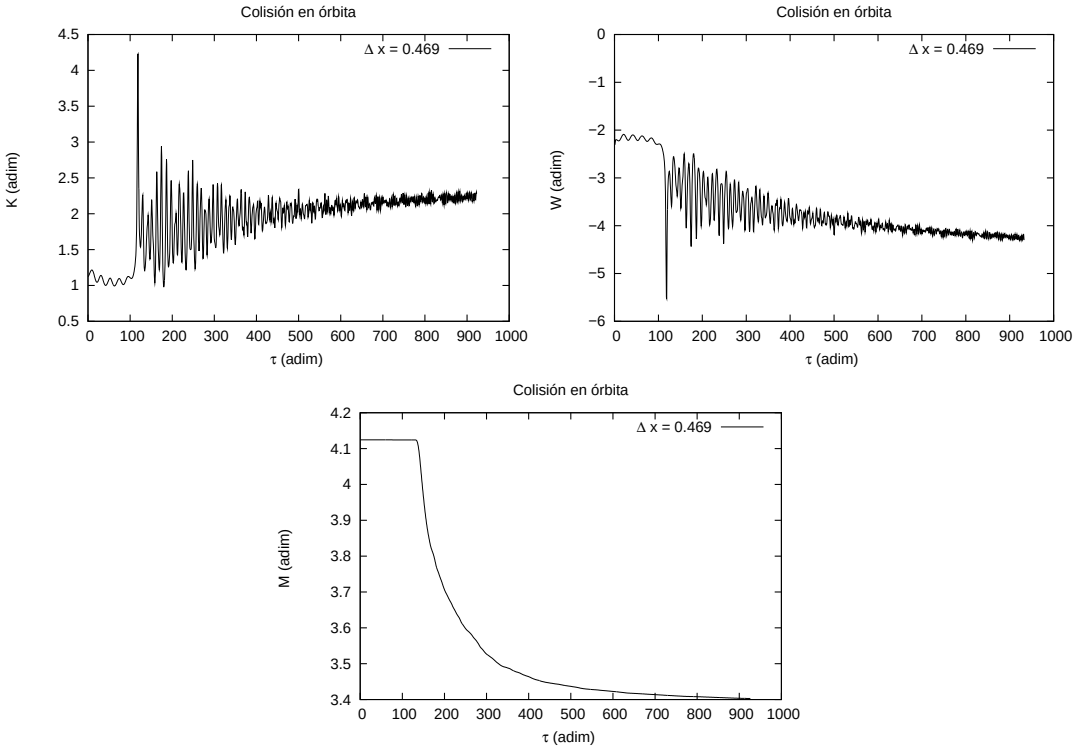

Figura 6.15: Gráficas de la energía cinética K, potencial W y cantidad de masa M para una configuración orbitante que colisiona. Se observa una simetría entre la energía potencial y cinética. Así mismo, se da un variación muy rápida con oscilaciones de gran amplitud después de la colisión (que se da más o menos en $\tau = 130$). En la gráfica de la masa se observa claramente el instante de la colisión ya que hay una expulsión de materia que luego de un tiempo parece que se estabiliza.

Para analizar los resultados se procederá de forma similar al caso esférico, analizando las variables observables del sistema cuántico. En la figura 6.15 se presentan las gráficas de la energía cinética, potencial y la masa total. En las tres gráficas se observa la misma característica, para un tiempo $\tau < 120$ las energías oscilan con amplitudes pequeñas y la masa permanece constante. Para $\tau \simeq 120$ se da un cambio de comportamiento, las energías oscilan con una gran amplitud que se amortigua al evoluciona el sistema y la masa decrece rápidamente y parece tender a un valor constante. Al analizar la evolución de las densidades se obtiene que el valor $\tau \simeq 120$ corresponde al momento en que la distribuciones colisionan (ver figura 6.20), la disminución en la masa corresponde a la expulsión de materia escalar y las variaciones en la energía a un proceso de excitación del campo que posteriormente parece estabilizarse. El hecho de que la masa disminuya, sugiere que el potencial imaginario en efecto absorbe la materia que sale expulsada.

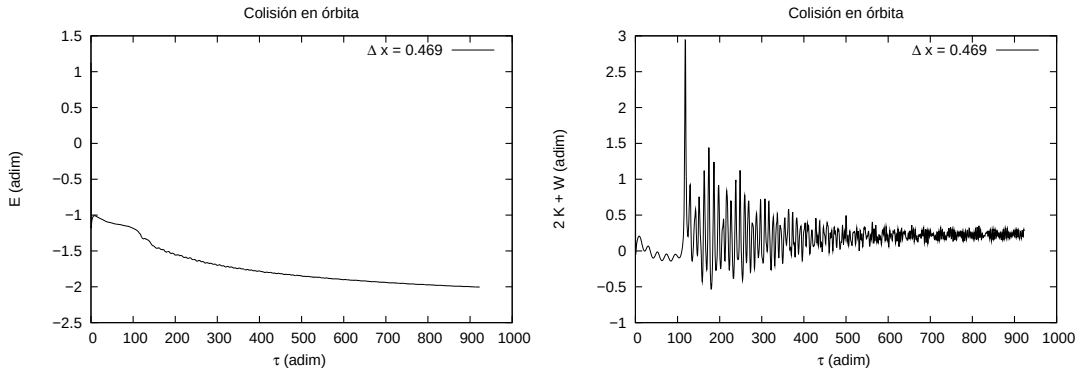

Figura 6.16: Gráfica de $E = K + W$ y $2K + W$ como función del tiempo τ para una configuración de dos distribuciones de campo escalar que orbitan y colisionan. Se observa que la energía no permaneces constante, lo cual indica una perdida de masa o de energía. Se observa que hasta el tiempo que se evolucionó el sistema no se virializó, la relación de virialización se mantiene diferente de cero durante toda la evolución.

La energía total y la relación de virialización mostradas en la figura 6.16 presentan un comportamiento similar, la energía decrece con un cambio brusco en el momento de la colisión y parece tender a estabilizarse, la relación de virialización es un poco más representativa, los tiempos de evolución no suficientes para que el sistema se relaje por completo y llegue aun estado estacionario, pero al igual que en los otros casos se observa una excitación al momento de la colisión de estos parámetros.

Por último podemos analizar el máximo de la densidad y el mínimo del potencial, en la figura 6.17 se muestra la gráfica de esta variables como función de τ. Se observa que las distribuciones de densidad evolucionan con un máximo de densidad constante que se mantienen en 1 hasta el momento en que colisionan, es entonces cuando se generan grandes oscilaciones y el máximo de densidad llega hasta cerca de 14 unidades, posteriormente las oscilaciones se amortiguan sin embargo el tiempo de evolución no es suficiente para observar si se estabiliza en algún valor. El mínimo de potencial presenta un comportamiento muy similar, en este caso teniendo una valor inicial casi constante en -1.5 y llegando hasta -4.5 al momento de la colisión. Es necesario hacer más evoluciones para poder caracterizar completamente el sistema, con una mejor resolución y por más tiempo, sin embargo para este trabajo lo que se desea mostrar es que el código Olliptic es útil para resolver este tipo de problemas.

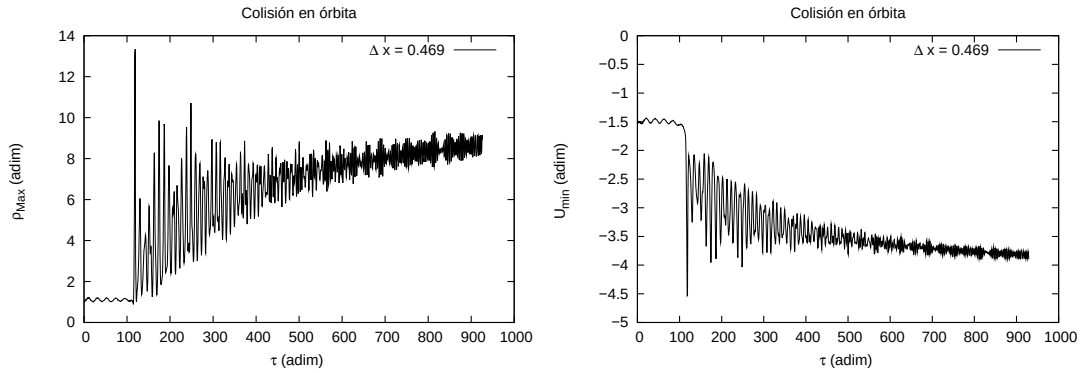

Figura 6.17: Gráficas del máximo de la densidad y mínimo del potencial como función de τ. El máximo de la densidad se mantiene constante en aproximadamente 1 hasta antes de la colisión, posteriormente el condensado resultante oscila violentamente y tiende a estabilizarse. Un comportamiento similar presenta el mínimo del potencial.

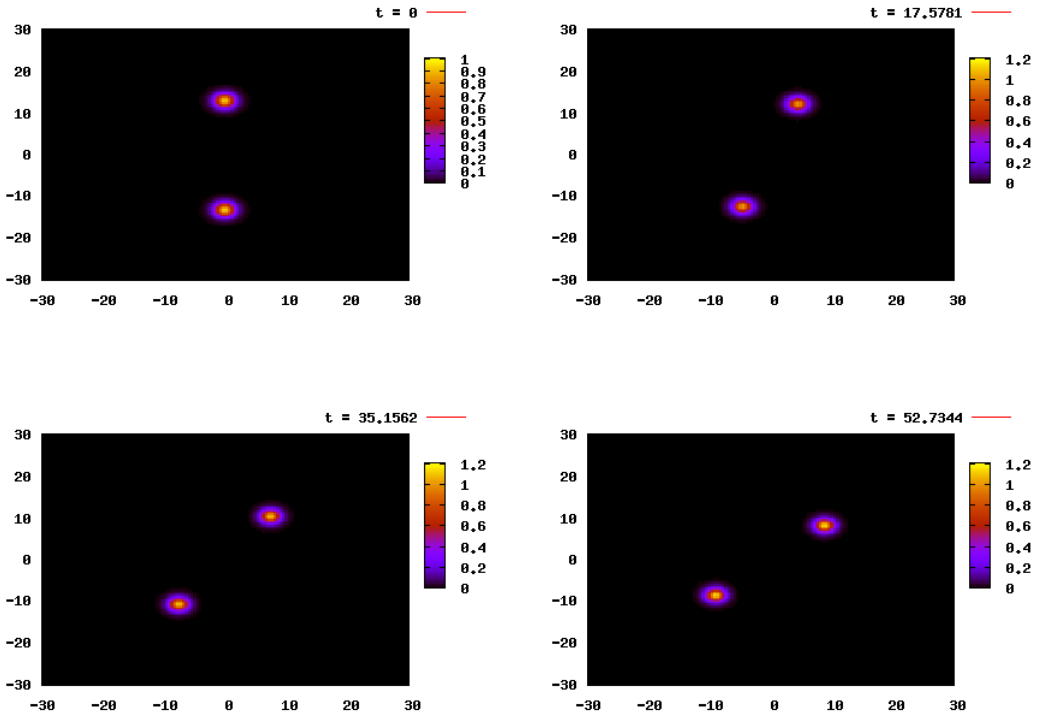

Figura 6.18: Gráficas de la densidad para los primeros pasos de tiempo, en un corte en el plano $x = 0$. Se observa la evolución de la densidad que orbita y se aproxima.

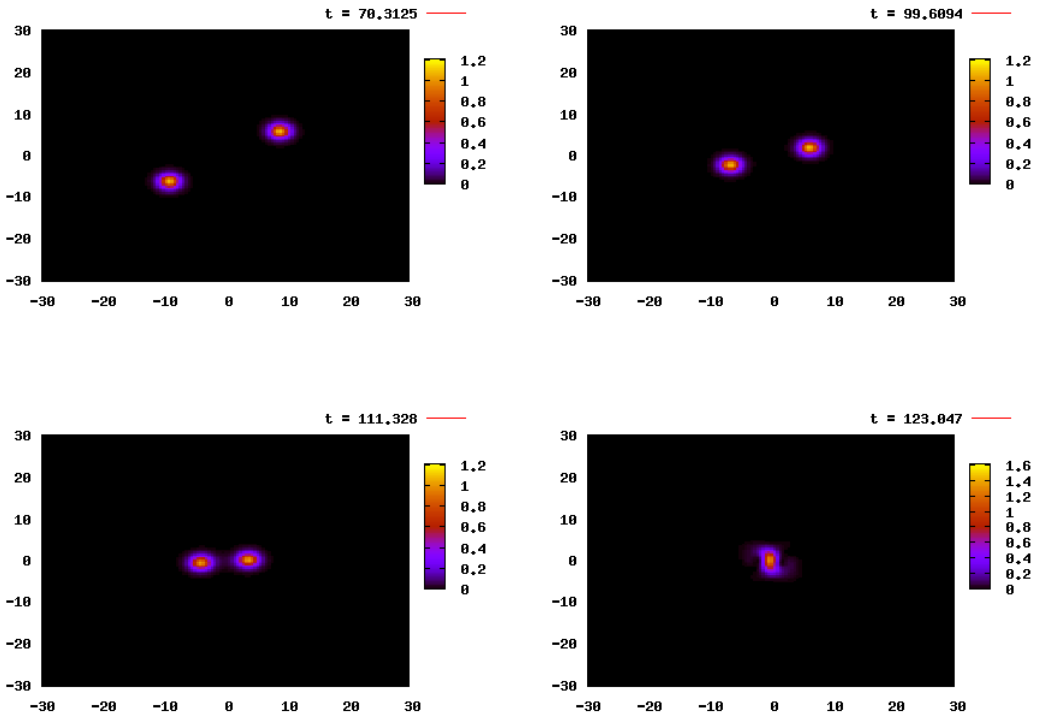

Figura 6.19: Gráficas de la densidad para pasos de tiempo cercanos al evento de colisión (corte en el plano $x = 0$). Se observa la evolución de la densidad que da 1/4 de orbita y colisiona. La distribución resultante al momento posterior a la colisión no tiene forma esférica.

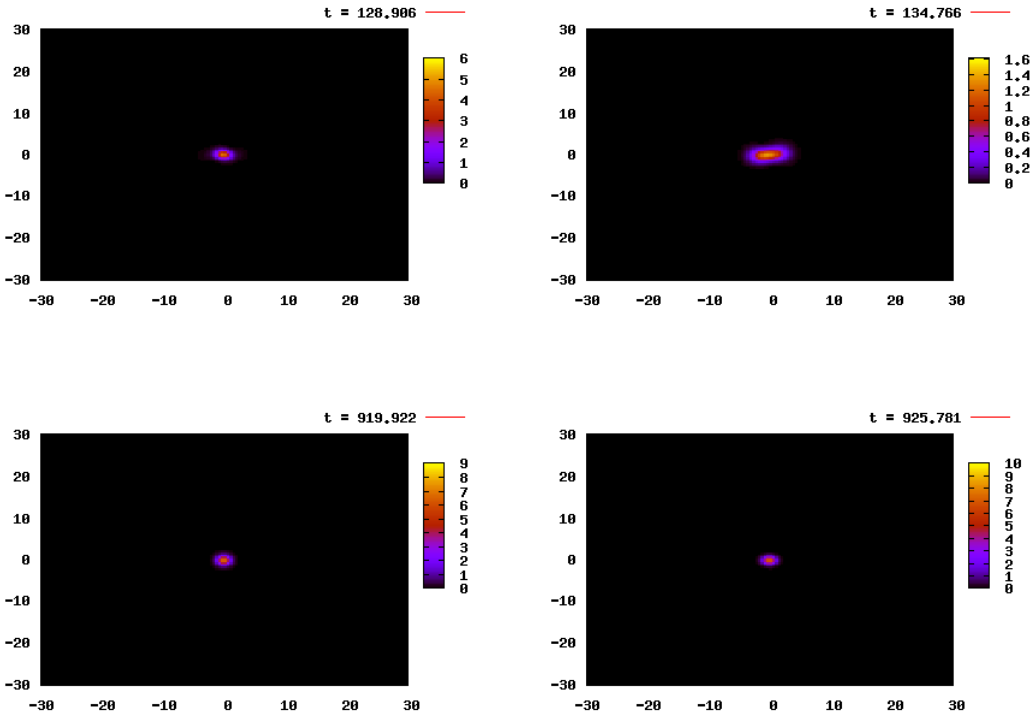

Figura 6.20: Gráficas de la densidad para pasos de tiempo posteriores a la colisión (corte en el plano $x = 0$). La distribución resultante recupera una forma esférica luego de varias iteraciones.

Capítulo 7

Conclusiones

Se ha construido y caracterizado el código computacional Olliptic que implementa los métodos numéricos Multigrid y Gauss-Seidel para resolver ecuaciones elípticas cuya principal aplicación esta orientada a la relatividad numérica pero que también se puede utilizar para resolver ecuaciones elípticas en otras áreas de la física. El código se programó en lenguaje C++ y utilizando programación orientada a objetos lo cual representa una ventaja en términos de diseño que permitirá realizar modificaciones y extensiones del mismo de forma más fácil. El código se ha programado como parte de las bibliotecas del código CACTUS, es decir es una "espina" que es utilizada en conjunto con otras aplicaciones ya implementadas en CACTUS.

El código Olliptic se ha probado con ecuaciones y fuentes construidas de forma explicita para poder comparar con soluciones analíticas, de estas pruebas se ha observado convergencia a segundo orden y una gran eficiencia al utilizar el método Multigrid que, en el caso de la ecuación de Poisson, se corresponde muy bien con las relaciones de eficiencia esperada para cada uno de los métodos. Se ha observado que las relaciones de eficiencia dependen de la ecuación y de las fuentes, de modo que es posible que la naturaleza de la ecuación, la forma funcional de los coeficientes y de la fuente influyan en el rendimiento del método. Para la ecuación lineal probada, el coeficiente del término lineal aceleró la convergencia y presentó un error global muy pequeño, las diferencias en rendimiento entre el método Multigrid y Gauss-Seidel fueron pequeñas, esto se explica debido a que el coeficiente lineal utilizado presenta valores mucho más grandes que la solución y la fuente de modo que ese término domina la ecuación. Hay que notar que las relaciones de eficiencia suponen ecuaciones elípticas muy simples por lo que no se espera que sigan siendo válidas para ecuaciones más complicadas.

Las ecuaciones de prueba sirvieron también para determinar el rendimiento del código en paralelo, se encontró que los algoritmos de paralelización corriendo en Kan-Balam la supercomputadora de la UNAM, se comportan bien hasta 64 procesadores que fue el número máximo de procesadores utilizados, mostrando que el código es eficiente en paralelo utilizando un cluster con buen hardware. Por otro lado, las pruebas de paralelización del método Gauss-Seidel en el cluster Ek-Bek del Laboratorio de Supercómputo Astrofísico, muestran que disminuye el tiempo de computo del cpu pero que el tiempo total llega a un mínimo que depende de la resolución global utilizada y que posteriormente aumenta, esto sugiere que el sistema satura las comunicaciones muy rápidamente. Las pruebas en Ek-Bek sirvieron para mostrar que es necesario tener buena infraestructura de comunicación en el cluster que se utilice.

El código Olliptic se aplicó a dos problemas físicos. La generación de datos iniciales para ondas

de Brill fue el primer problema a resolver. Se compararon los resultados con otra espina de CACTUS que resuelve ecuaciones elípticas. La comparación natural con Olliptic es la espina BAM_Elliptic que también implementa un método Multigrid, sin embargo esto no fue posible debido a que no se pudo obtener convergencia utilizado la espina BAM_Elliptic. Se procedió entonces a comparar con la espina EllSOR. Se encontró que Olliptic usando el método Multigrid presenta una eficiencia mucho mayor que EllSOR, en ambos casos el rendimiento es el esperado para los métodos según el análisis de rendimiento. Se encontró que la implementación de las condiciones de frontera para Olliptic se deben mejorar ya que presentan una convergencia más lenta que la esperada. No obstante, el sistema presentó un mejor comportamiento al variar el tamaño del dominio, en el caso de EllSOR el código falla al tratar de resolver la ecuación de datos iniciales para ondas de Brill en dominios mayores a los utilizados mientras que Olliptic no presenta ese problema.

El segundo problema físico en donde se aplicó Olliptic fue la evolución del sistema Schrödinger-Poisson que es utilizado en el contexto del modelo de materia obscura tipo campo escalar. Este sistema representa un reto para el código Olliptic debido a que hay que resolver la ecuación de Poisson a cada paso de tiempo con una fuente que varía, que en este caso es la densidad de campo escalar, que evoluciona según la ecuación de Schrödinger. Debido a que Olliptic solo resuelve la ecuación de Poisson fue necesario utilizar la espina Schrödinger que se encarga de resolver la correspondiente ecuación y que se acopla en CACTUS a Olliptic intercalándose en la ejecución a cada paso de tiempo resolviendo entre las dos espinas el sistema acoplado. Como antecedente de este problema se han estudiado casos que presentan simetrías, en particular simetría esférica y axial. Los resultados del caso esféricamente simétrico se utilizaron como prueba de los códigos reproduciendo satisfactoriamente los resultados. Por último se probó el código en un caso que no presenta simetrías obteniendo resultados que solo con métodos eficientes como los implementados en Olliptic es práctico evolucionar debido que con otros resolvedores llevaría demasiado tiempo de cálculo. El estudio del sistema Schrödinger-Poisson en el caso no simétrico puede dar por si mismo lugar a un trabajo extenso y queda fuera de los objetivos de la tesis el análisis exhaustivo de los resultados obtenidos, sin embargo es un buen punto de partida para continuar explorando las implicaciones de este sistema.

Se puede concluir de forma general que se ha construido un código que resuelve ecuaciones elípticas de forma eficiente, que por el momento tiene implementadas pocas funcionalidades pero que debido al diseño es posible con un poco más de desarrollo obtener un código muy versátil y eficiente. Por último se pueden señalar algunas posibles extensiones y mejoras. Dentro de las mejoras que se pueden implementar fácilmente se encuentra el aumentar el orden de discretización de los operadores, por ejemplo usando aproximaciones de cuarto orden, en particular es posible mejorar el comportamiento de las condiciones de frontera tipo Robin utilizando operadores de mayor orden. Con procedimientos un poco más complicados, se puede implementar un método híbrido Multigrid-Seudoespectral en el cual la matriz de coeficientes se invierta con un sistema iterativo Multigrid aumentando al mismo tiempo la eficiencia y el orden de aproximación. Para aplicaciones en Relatividad Numérica es indispensable implementar operadores en espacios curvos. En cuestión de rendimiento, se puede implementar un método de relajación más eficiente para el método Multigrid como por ejemplo SOR, obteniendo convergencias logarítmicas, lo cual representa la máxima eficiencia posible. Por último el mayor reto en cuanto a desarrollo del código es implementar soluciones para sistemas con mallas inhomogeneas por ejemplo las utilizadas por el driver CARPET. Hay que mencionar que todas esta extensiones del código serán implementadas en Olliptic muy pronto y que por el momento el grupo de relatividad numérica Ollin ya cuenta con una base de desarrollo y con un código útil para resolver ecuaciones elípticas. Se ha continuado explorando y generando datos para el sistema Schrödinger-Poisson que pronto serán analizados.

Apéndice A

Programación en paralelo

El código CACTUS se encarga directamente de la paralelización de los códigos de evolución, sin embargo el controlador de mallas PUGH solo genera mallas homogéneas y de un cierto tamaño de partición en cada evolución, por ello fue necesario implementar la paralelización de las submallas de Olliptic directamente usando la biblioteca MPI.

La primera dificultad que presenta paralelizar un código usando directamente las bibliotecas MPI dentro de CACTUS es la forma en la que el controlador PUGH hace la distribución del dominio entre los procesadores asignados para hacer la evolución; es posible indicar a CACTUS una distribución particular, sin embargo siempre se asignan bloques del dominio. Sin embargo para paralelizar el método de Gauss-Seidel es necesario asignar a cada procesador una distribución de puntos tipo "tablero de ajedrez" en la que se alterna la asignación de procesadores de tal forma que cada procesador trabaja con puntos cuyos vecinos pertenecen a otro procesador. En la figura A.1 se muestra la forma en que CACTUS discretiza el dominio y la forma en que se debe discretizar para aplicar el método de Gauss-Seidel en paralelo.

Las dos distribuciones son incompatibles, en el primer caso cada procesador trabaja con arreglos reducidos con el número de puntos asignados y transmite la información entre procesadores por medio de fronteras "no físicas", en el segundo caso cada procesador trabaja con arreglos completos con el número de puntos del dominio global pero solo hace cálculos con un número restringido de ellos, la sincronización de la información en este caso se hace al actualizar la información de cada arreglo con los elementos modificados de todos los vecinos.

Para la primer versión del código Olliptic por simplicidad se optó por aplicar un método similar al usado por cactus en la evolución, heredando la partición que genera CACTUS y usando fronteras interprocesadores para actualizar la información. Es decir, se aplica un subproblema elíptico en cada procesador con dos tipos de frontera, las físicas de las cuales ya se habló anteriormente y las fronteras entre procesadores que sincronizan la información.

CACTUS define un arreglo de una dimension y seis entradas (una entrada para cada cara del subdominio cúbico) llamado `cctk_bbox[i]`. Este arreglo se define para cada procesador y se asigna el valor 0 si la frontera es "no física" y 1 si es física. En Olliptic se define un arreglo equivalente de dos dimensiones `bound_box[i][j]` con `i` que va de 0 a 2 y `j` que vale 0 o 1 y que define cada cara del dominio usando macros. Los macros usados son `#define Xi 0][0`, `#define Xf 0][1`, `#define Yi 1][0`, `#define Yf 1][1`, `#define Zi 2][0` y `#define Zf 2][1` de modo que se puede hacer referenca a la cara X_i del dominio simplemente con `bound_box[Xi]`. El arreglo `bound_box[Xi]` guarda más infomación que la contraparte de CACTUS ya que se asignan

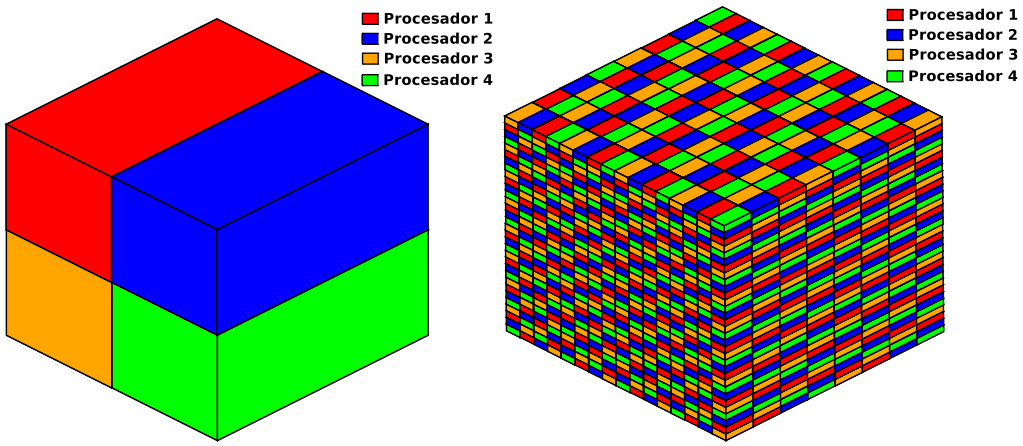

Figura A.1: En este diagrama se presentan dos formas de dividir el dominio entre procesadores. En el lado izquierdo se muestra la manera en la que lo hace CACTUS para un ejemplo usando 4 procesadores, el dominio se divide en 4 regiones para cada procesador que se representan con un color. En la parte derecha se muestra el ordenamiento tipo tablero de ajedrez, en este caso el dominio se divide en varias regiones disconexas, en el segundo caso la idea es usar un refinamiento en el cual cada sub-región contenga solo un punto de la malla.

valores negativos a las fronteras físicas (que identifican la cara a la que perteneces tomando valores de -1 a -6 que corresponden al orden dado en los macros $X_i, X_f, \cdots, Z_f$) y si la frontera no es física se asigna un valor positivo que corresponde al número de procesador asignado al vecino según la numeración hecha por MPI.

Imponer condiciones de frontera entre procesadores para el método Multigrid genera una dificultad ya que pictóricamente podemos pensar que para generar las sub-mallas se extraen puntos del dominio original. Por ejemplo pensando en 1 dimensión, si la malla más fina consta de 65 puntos, quitando puntos interiores de forma alternada se obtiene una malla mas burda de 33 puntos. Con estas dos mallas es posible aplicar el método Multigrid. Si ahora se hace una asignación de los puntos entre dos procesadores se genera una interfase en el dominio, CACTUS utiliza en esa región subdominios que se traslapan y a los puntos que se encuentran en la intersección se les conoce como puntos fantasmas. Los puntos fantasmas se utilizan para actualizar la información en la frontera entre dos procesadores.

En la figura A.2 se muestra un esquema del procedimiento a seguir para sincronizar la información entre procesadores para el método Multigrid. En ese diagrama se observa que para la malla más burda el punto fantasma del cual se debe transmitir la información se ha extraído del dominio, por lo que es necesario extrapolar la información de los puntos interiores. El orden de la extrapolación genera una restricción sobre el número mínimo de puntos interiores en cada subdominio. En la primer versión de Olliptic se ha usado una extrapolación cúbica por lo que se requieren tres puntos interiores para transmitir la información del punto faltante. Hay que mencionar que este tipo de descomposición del dominio presenta la ventaja de que el número de puntos a transmitir entre procesadores es una dimensión menor que en el caso del ordenamiento tipo tablero de ajedrez, ya que solo se debe transmitir el plano en el que se encuentran los puntos fantasmas. Así mismo presenta la desventaja de que la transmisión de la información se da localmente en los subdominios y solo pasa por la frontera de modo que el orden de la extrapolació afecta directamente la convergencia.

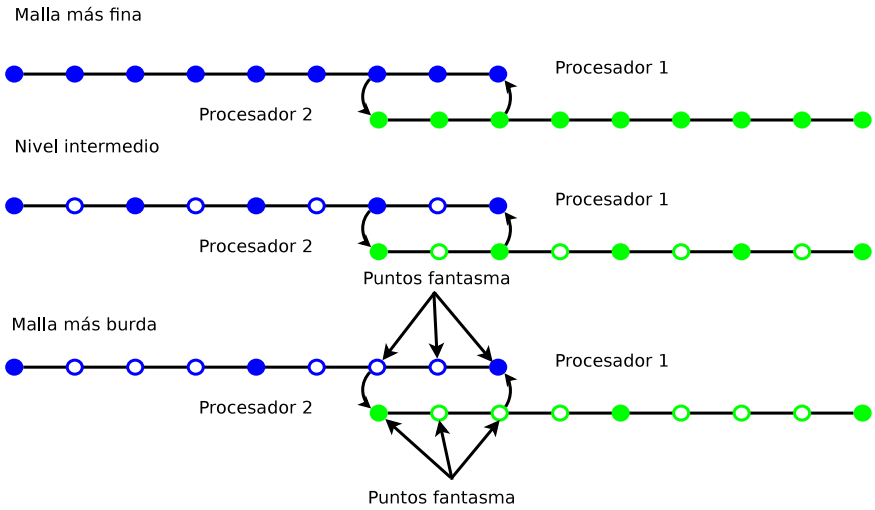

Figura A.2: En este diagrama se muestra el proceso de refinamiento de mallas del método Multigrid para una partición de dos procesadores y tres puntos fantasma. En el nivel con la mayor resolución el valor de la frontera en el dominio del procesador 1 se copia del segundo punto interior del dominio asignado al procesador 2. Al generar una submalla de menor resolución se asigna el valor del primer punto interior del procesador 2 a la frontera del procesador 1 (los puntos sin color de relleno son los que han sido extraidos de la malla). En la malla más burda de este ejemplo, se ha quitado el punto del cual se debe transmitir la información a la frontera, de modo que se debe extrapolar de los puntos interiores el valor que le corresponde.

Detalles técnicos sobre programación en paralelo usando la biblioteca MPI se pueden consultar en [36, 56], en el código fuente de Olliptic están comentados algunos detalles técnicos sobre la paralelización y la transmisión de arreglos de información entre los procesadores.

Apéndice B

Código de ejemplo

En esta sección se darán fragmentos de código para generar una espina que resuelve un problema elíptico usando Olliptic. La espina debe incluir el código Olliptic por medio de la definición de las clases `EllObj.h` y `GridObj.h`. El código CACTUS se encarga de toda la interface con el usuario y solo es necesario poder compilar CACTUS siguiendo las guias dadas en la documentación del código.

Supongamos que queremos resolver la ecuación de Poisson, con una densidad de carga $\rho = 1$ en una región $r < 1$ y cero en el exterior. La solución será definida un paralelepípedo generado por CACTUS, en la espina Olliptic solo debemos poner las instrucciones que se muestran a continuación:

```
#include "EllObj.h"     //La definicion de las clases EllObj y GridObj

#include "GridObj.h"    //se incluyen en estas instrucciones.

#include "cctk.h" //El resto de archivos incluidos corresponde a CACTUS

#include "cctk_Parameters.h"

#include "cctk_Arguments.h"


//Se inicia el codigo como cualquier espina de C++ en CACTUS (ver manuales)

extern "C" void
Olliptic_Poisson (CCTK_ARGUMENTS)
{


    // Se declaran los argumentos y parametros de CACTUS

    DECLARE_CCTK_ARGUMENTS;

    DECLARE_CCTK_PARAMETERS;
```

```
//Toma el numero de puntos en la malla
//definido por CACTUS

int nx = cctk_lsh[0];

int ny = cctk_lsh[1];

int nz = cctk_lsh[2];



GridObj U;    //Objeto grid para almacenar la solucion, no reqiere parametros
              //ya que se construye una malla base vacia.

GridObj S(cctkGH); //Objeto grid para definir la fuente (hereda
                   //la jerarqu'ia de mallas de CACTUS)

EllObj L(cctkGH);   //Objeto el'iptico que define la ecuaci'on


//Inicia la definici'on del operador, fronteras tipo
//Dirichlet iguales a 0.0

L.Boundary(DIRICHLET, 0.0);


//define el m'etodo utilizando las variables que proporcina
//la espina Olliptic para leer los valores del archivo de parametros

L.Define_Method(MULTIGRID, nVcycles, epsilon, epsilon0, sigma);


//define la ecuaci'on, en este caso Poisson
L.Define_Equation(POISSON);


 //define la fuente

for (int i = 0; i < nx; i++)
    for (int j = 0; j < ny; j++)
        for (int k = 0; k < nz; k++){

            //recupera el indice virtual de CACTUS

            int vindex = CCTK_GFINDEX3D(cctkGH, i, j, k);
```

```
                //define la coordenada radial
                double r = sqrt( x[vindex] * x[vindex] +
                                 y[vindex] * y[vindex] +
                                 z[vindex] * z[vindex] );


                if (r > 1.0)

//s minuscula corresponde a una funcion de CACTUS, la S mayuscula a un objeto de
//Olliptic.

                S(i,j,k) = s[vindex] = 0.0;  //Densidad cero fuera de una esfera
                                              //de radio 1

                else

                S(i,j,k) = s[vindex] = 1.0; //Densidad unitaria dentro de una esfera
                                            //de radio 1
            }

    //ya se definio el operador y la fuente, la solucion se obtiene
    //aplicando el operador inverso:

    U = S / L;

    //La solucion se puede regresar a una funci'on de CACTUS:

    for (int i = 0; i < nx; i++)
        for (int j = 0; j < ny; j++)
            for (int k = 0; k < nz; k++){

                int vindex = CCTK_GFINDEX3D(cctkGH, i, j, k);

                u[vindex] = U(i,j,k);

            }
    //Termina la espina de ejemplo

}
```

Bibliografía

[1] G. Allen, T. Dramlitsch, I. Foster, N. Karonis, M. Ripeanu, E. Seidel, and B. Toonen. Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus. In *Proceedings of Supercomputing 2001*, Denver, USA, 2001. http://www.cactuscode.org.

[2] T. Goodale, G. Allen, G. Lanfermann, J. Masso, T. Radke, E. Seidel, and J. Shalf. The Cactus Framework and Toolkit: Design and Applications . In *Vector and Parallel Processing - VECPAR '2002, 5th International Conference*. Springer, 2003. http://www.cactuscode.org.

[3] R. Arnowitt, S. Deser, and C.W. Misner. The dynamics of general relativity. In L. Witten, editor, *Gravitation, An Introduction to Current Research*. John Wiley & Sons, 1962.

[4] L.L. Smarr. *The Structure og General Relativity with a Numerical Illustration: The Collision of Two Black Holes*. PhD thesis, The University of Texas at Austin, 1975.

[5] K. Eppley. *The Numerical Evolution of the Collision of Two Black Holes*. PhD thesis, Princeton University, 1975.

[6] Frans Pretorius. Evolution of Binary Black-Hole Spacetimes. *Phys. Rev. Lett.*, 95:121101, 2005.

[7] John G. Baker, Joan Centrella, Dae-Il Choi, Michael Koppitz, and James van Meter. Gravitational wave extraction from an inspiraling configuration of merging black holes. *Phys. Rev. Lett.*, 96:111102, 2006.

[8] M. Campanelli, C. O. Lousto, P. Marronetti, and Y. Zlochower. Accurate Evolutions of Orbiting Black-Hole Binaries Without Excision. *Phys. Rev. Lett.*, 96:111101, 2006.

[9] http://www.carpetcode.org/.

[10] http://root.cern.ch/.

[11] V. Sahni and L. Wang. New cosmological model of quintessence and dark matter. *Phys. Rev. D*, 58:023503, 1998.

[12] Tonatiuh Matos and L. Arturo Ureña-López. On the nature of dark matter. *Int. J. Mod. Phys. D*, 13:2287–2291, 2004.

[13] F. Siddhartha Guzmán and L. Arturo Ureña-López. Gravitational Cooling of Self-gravitating Bose Condensates. *The Astrophysical Journal*, 645(2):814–819, 2006.

[14] Argelia Bernal and F. Siddhartha Guzmán. Scalar field dark matter: Nonspherical collapse and late-time behavior. *Phys.Rev. D*, 74:063504, 2006.

[15] Argelia Bernal and F. Siddhartha Guzmán. Scalar field dark matter: Head-on interaction between two structures. *Phys.Rev. D*, 74:103002, 2006.

[16] Jeffrey Winicour. Characteristic Evolution and Matching. In *Living Reviews in Relativity*. http://relativity.livingreviews.org.

[17] H. Friedrich. Conformal Einstein evolution. In *Lecture Notes in Physics*, volume 604. Springer Berlin / Heidelberg, 2002.

[18] Jr. York, J. W. Kinematics and dynamics of general relativity. In *Sources of gravitational radiation*, pages 83–126, 1979.

[19] M. Alcubierre. *Introduction to* $3+1$ *Numerical Relativity*. Oxford Univ. Press, Oxford, **in preparation**, 2007.

[20] T. Nakamura, K. Oohara and Y. Kojima. General relativistic collapse to black holes and gravitational waves from black holes. *Progress of Theoretical Physics Supplement*, 90:1–218, 1987.

[21] M. Shibata and T. Nakamura. Evolution of three-dimensional gravitational waves: Harmonic slicing case. *Phys. Rev. D*, 52:5428, 1995.

[22] Thomas W. Baumgarte and Stuart L. Shapiro. Numerical integration of Einstein's field equations. *Phys. Rev. D*, 59(2):024007, Dec 1998.

[23] Gregory B. Cook. Initial Data for Numerical Relativity. In *Living Reviews in Relativity*. http://relativity.livingreviews.org.

[24] James W. York. Conformal "Thin-Sandwich" Data for the Initial-Value Problem of General Relativity. *Phys. Rev. Lett.*, 82(7):1350–1353, 1999.

[25] D. S. Brill. On the positive definite mass of the Bondi-Weber-Wheeler time-symmetric gravitational waves. *Ann. Phys.*, 7:466, 1959.

[26] S. Brandt and B. Brügmann. A Simple Construction of Initial Data for Multiple Black Holes. *Phys. Rev. Lett.*, 78(19):3606–3609, 1997.

[27] J. M. Bowen and J. W. York. Time-asymmetric initial data for black holes and black-hole collisions. *Phys. Rev. D*, 21(8):2047–2056, 1980.

[28] Michael E. Taylor. *Partial Differential Equations I*, volume 115 of *Applied Mathematical Science*. Springer, 1996.

[29] Michael E. Taylor. *Partial Differential Equations II*, volume 116 of *Applied Mathematical Science*. Springer, 1996.

[30] Michael E. Taylor. *Partial Differential Equations III*, volume 117 of *Applied Mathematical Science*. Springer, 1996.

[31] Karl E. Gustafson. *Introduction to Partial Differential Equations and Hilbert Space Methods*. Dover, 1999.

[32] H. F. Weinberger. *A First Course in Partial Differential Equations: with Complex Variables and Transform Methods*. Dover, 1965.

[33] Peter Knabner and Lutz Angermann. *Numerical Methods for Elliptic and Parabolic Partial Differential Equations*. Springer, 2003.

[34] Aslak Tveito and Ragnar Winther. *Introduction to Partial Differential Equations: A Computational Approach* . Texts in Applied Mathematics. Springer, 2004.

[35] William T. Vetterling, Brian P. Flannery, William H. Press, and Saul A. Teukolsky. *Numerical Recipes in C++*. Cambridge University Press, 1992.

[36] George Em Karniadakis and Roberet M. Kirby II. *Parallel Scientific Computing in C++ and MPI*. Cambridge University Press, 2003.

[37] E. Fermi, J. Pasta, and S. Ulam. Technical report, Los Alamos National Laboratory, 1955.

[38] Gene Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[39] http://pelusa.fis.cinvestav.mx/LaSumA/.

[40] F. Siddhartha Guzmán and L. Arturo Ureña-López. Evolution of the Schrödinger-Newton system for a self-gravitating scalar field. *Phys. Rev. D*, 69(124033), 2004.

[41] R. Beig and N. Ó Murchadha. Trapped surfaces due to concentration of gravitational radiation. *Phys. Rev. Lett.*, 66(19):2421–2424, May 1991.

[42] M. Alcubierre, S. Brandt, B. Bruegmann, C. Gundlach, J. Masso, E. Seidel, and P. Walker. Test-beds and applications for apparent horizon finders in numerical relativity. *Class. Quantum Grav.*, 17:2159–2190, 2000.

[43] Milton Ruiz, Miguel Alcubierre, and Dario Nunez. Regularization of spherical and axisymmetric evolution codes in numerical relativity. arXiv:0706.0923v1.

[44] Kenneth Eppley. Evolution of time-symmetric gravitational waves: Initial data and apparent horizons. *Phys. Rev. D*, 16(6):1609–1614, Sep 1977.

[45] D. E. Holz, W. A. Miller, M. Wakano, and J. A. Wheeler. Coalescence of primal gravity waves to make cosmological mass without matter. In B. L. Hu and T. A. Jacobson, editors, *Directions in General Relativity: Papers in Honor of Dieter Brill, Volume 2*, pages 339–+, 1993.

[46] Matos, Tonatiuh and Arturo Ureña-López, L. . Further analysis of a cosmological model with quintessence and scalar dark matter. *Phys. Rev. D*, 63(6):063506, Feb 2001.

[47] Wayne Hu, Rennan Barkana, and Andrei Gruzinov. Fuzzy Cold Dark Matter: The Wave Properties of Ultralight Particles. *Phys. Rev. Lett.*, 85(6):1158–1161, Aug 2000.

[48] Alexandre Arbey, Julien Lesgourgues, and Pierre Salati. Quintessential halos around galaxies. *Phys. Rev. D*, 64(12):123528, Nov 2001.

[49] Alexandre Arbey, Julien Lesgourgues, and Pierre Salati. Cosmological constraints on quintessential halos. *Phys. Rev. D*, 65(8):083514, Apr 2002.

[50] Alexandre Arbey, Julien Lesgourgues, and Pierre Salati. Galactic halos of fluid dark matter. *Phys. Rev. D*, 68(2):023511, Jul 2003.

[51] Edward Seidel and Wai-Mo Suen. Formation of solitonic stars through gravitational cooling. *Phys. Rev. Lett.*, 72(16):2516–2519, Apr 1994.

[52] Jayashree Balakrishna, Edward Seidel, and Wai-Mo Suen. Dynamical evolution of boson stars. II. Excited states and self-interacting fields. *Phys. Rev. D*, 58(10):104004, Sep 1998.

[53] M. Alcubierre, R. Becerril, F. S. Guzman, T. Matos, D. Nuñez and and L. A. Urena-López. Numerical studies of ϕ^2-oscillatons. *Class. Quantum Grav.*, 20(2883), 2003.

[54] M. Alcubierre, F. S. Guzman, T. Matos, D. Nuñez, L. A. Ureña-López and P. Wiederhold. Galactic collapse of scalar field dark matter. *Class. Quantum Grav.*, 19(5017), 2002.

[55] Xian Zhi Wang. Cold Bose stars: Self-gravitating Bose-Einstein condensates. *Phys. Rev. D*, 64(12):124009, Nov 2001.

[56] http://webct.ncsa.uiuc.edu:8900/webct/public/show_courses.pl/.