ASPECTOS COMPUTACIONALES DEL MÉTODO "LOCAL DISCONTINUOUS GALERKIN" PARA MALLAS NO ESTRUCTURADAS EN 3D

Por

Filánder de los Ángeles Sequeira Chavarría

Tesis sometida en cumplimiento parcial de los requerimientos para el grado de

MAESTRÍA EN CIENCIAS

en

MATEMÁTICA APLICADA

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ

Noviembre, 2010

Aprobada por:

Lev Steinberg, Ph.D Miembro, Comité Graduado	Fecha
Pedro Vásquez Urbano, Dr Miembro, Comité Graduado	Fecha
Paul Castillo, Ph.D Presidente, Comité Graduado	Fecha
Miguel Vélez Reyes, Ph.D Representante de Estudios Graduados	Fecha
Silvestre Colón, M.Sc Director del Departamento	Fecha

Resumen de Disertación Presentado a Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

ASPECTOS COMPUTACIONALES DEL MÉTODO "LOCAL DISCONTINUOUS GALERKIN" PARA MALLAS NO ESTRUCTURADAS EN 3D

Por

Filánder de los Ángeles Sequeira Chavarría

Noviembre 2010

Consejero: Paul Castillo, Ph.D
Departamento: Ciencias Matemáticas

Muchos de los problemas físicos y de ingeniería son modelados a través de ecuaciones diferenciales e integrales, los cuales en la mayoría de los casos, no tienen una solución analítica. Por esta razón, se recurre a aproximar la solución por medio de métodos numéricos.

En las últimas décadas, se han propuesto una serie de métodos basados en aproximaciones discontinuas, con el fin de preservar propiedades físicas del problema, conocidos como métodos discontinuos de Galerkin (DG por sus síglas en ingles). En particular, el método "Local Discontinuous Galerkin" (LDG), ha sido ampliamente estudiado en los últimos años. Varios artículos se han dedicado al análisis de la estabilidad y la convergencia del método aplicado a los problemas de difusión lineal y no lineal y a operadores diferenciales de alto orden. En esta tesis se proponen varios aspectos acerca de una implementación eficiente del método LDG aplicado a problemas lineales elípticos de segundo orden para mallas no estructuradas en 3D.

Todos los trabajos e implementaciones, conocidas hasta la fecha, para el método LDG con mallas no estructuradas son para dominios en $\mathbb{R}^2$. Por tal razón, se desarrolló un marco computacional abstracto para implementar varios métodos DG. Los operadores más relevantes del LDG se describen de manera simplificada utilizando el producto de Kronecker. Se propone un algoritmo rápido para ensamblar el complemento de Schur de manera explícita, el cual reduce almacenamiento. Además, el código descrito muestra una gran flexibilidad debido a que esta basado en el diseño de estructuras de datos abstractas. Se presentan una serie de experimentos numéricos para validar el código y para ilustrar el funcionamiento del método para mallas no estructuradas en 3D.

En artículos previos se ha probado que el condicionamiento espectral de la matriz de rigidez presenta un comportamiento asintótico de $\mathcal{O}(h^{-2})$ en mallas estructuradas y no estructuradas donde h es el tamaño de la malla. Lo que hace necesario precondicionadores eficientes. Se presentan precondicionadores multiniveles semi-algebraicos para aproximaciones lineales que utilizan la base de interpolación de Lagrange. Estos fueron implementados y estudiados para la matriz del sistema generado de la discretización del LDG. Se muestra numéricamente que su rendimiento no se degrada o por lo menos se incrementa muy levemente según aumenta el número de incógnitas. Los precondicionadores son probados en problemas con grandes saltos en los coeficientes.

Abstract of Dissertation Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Sciences

COMPUTATIONAL ASPECTS OF THE "LOCAL DISCONTINUOUS GALERKIN" METHOD FOR UNSTRUCTURED MESHES IN 3D

By

Filánder de los Ángeles Sequeira Chavarría

November 2010

Chair: Paul Castillo, Ph.D
Major Department: Mathematical Sciences

Many physical and engineering problems are modeled through differential and integral equations, which in most cases, do not have an analytical solution. For this reason, we approximate the solution by numerical methods.

In recent decades, in order to preserve physical properties of the problem, a series of methods based in discontinuous approximations was proposed, known as Discontinuous Galerkin methods (DG). In particular, the Local Discontinuous Galerkin method (LDG), has been widely studied in recent years. Several papers have been devoted to the analysis of the stability and convergence of the method applied to linear and nonlinear diffusion problems and high order differential operators. In this thesis, we propose several aspects of the LDG method applied to linear second order elliptic problems on unstructured meshes in 3D.

All work and implementations known to date, for the LDG method with unstructured meshes, were for domains in $\mathbb{R}^2$. For this reason, we developed a computational abstract framework to implement several DG methods. The most relevant operators for LDG are described, in simplified way, using Kronecker product. We

propose a fast algorithm to assemble the Schur complement explicitly, which reduces the storage. In addition, the described code shows great flexibility because it is based on the design of abstract data structures. We present a series of numerical experiments to validate the code and to illustrate the performance of the method on unstructured meshes in 3D.

In previous papers it has been proven that the spectral condition number of the stiffness matrix exhibits an asymptotic behavior of $\mathcal{O}(h^{-2})$ on structured and unstructured meshes where h is the mesh size. Thus efficient preconditioners are mandatory. We present semi-algebraic multilevel preconditioners for linear approximations that use Lagrange type interpolatory basis. These were tested and studied for the matrix of the system generated from LDG discretization. We show numerically that their performance does not degrade or at least increases very slowly as the number of unknowns augment. Preconditioners are tested on problems with high jumps in the coefficients.

A Dios y a mis padres, German y Marlene, gracias por la fortaleza, y el ánimo para seguir hacia adelante.

AGRADECIMIENTOS

A mi consejero, Paul Castillo, quien me guió en mis estudios, me abrió las puertas a mi desarrollo profesional y me brindó su amistad.

Al profesor Mario Marín y a Luis Rodríguez, por sus buenos consejos, el gran apoyo que siempre me ha ayudado y la amistad sincera.

Al personal admistrativo y la facultad del departamento de matemáticas de la Universidad de Puerto Rico, por brindarme la oportunidad de realizar mis estudios de maestría.

A mi familia, mis nuevos amigos en Puerto Rico, y mis viejos amigos en Costa Rica, por su apoyo constante y buenos deseos.

TABLA DE CONTENIDO

página

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

DG	Discontinuous Galerkin.
DGM	Discontinuous Galerkin Method.
LDG	Local Discontinuous Galerkin.
AMG	Algebraic Multigrid.
ASM	Algebraic Symmetric Multigrid.
OOP	Object-Oriented Programming.
CSR	Compressed Sparse Row.
CG	Conjugate Gradient.
PCG	Preconditioned Conjugate Gradient.

LISTA DE SÍMBOLOS

$\mathbb{R}$	Conjunto de los números reales.
$\partial\Omega$	Frontera del dominio Ω.
$\vec{\boldsymbol{n}}$	Vector normal.
f	Fuente de la ecuación diferencial.
$g_{\mathcal{D}}$	Data para la condición de borde de Dirichlet.
$g_{\mathcal{R}}$	Data para la condición de borde de Robin.
$\mathcal{T}_h$	Partición del dominio por tetrahedros.
$L_2(\Omega)$	Espacio de funciones continuas Lebesgue integrables con la norma 2.
$\mathcal{P}_k(T)$	Conjunto de polinomio de grado menor o igual a k en la celda T.
u_h	Aproximación de la solución. Solución discreta.
$u_{h,T}$	Aproximación de la solución en la celda T.
$\widehat{u_{h\{\mathcal{s},T\}}}$	Flujo numérico de u_h en la cara $\mathcal{s}$ de la celda T.
$\widehat{\boldsymbol{q}_{h\{\mathcal{s},T\}}}$	Flujo numérico de $\boldsymbol{q}_h$ en la cara $\mathcal{s}$ de la celda T.
$\alpha_{\mathcal{s}}(u_h)$	Término de estabilidad.
$v\|_T$	La función v restringida a la celda T.
$\operatorname{diam}(T)$	Diámetro de la celda T.
$\kappa(A)$	Condicionamiento espectral de la matriz A.
$\mathcal{J}_k$	Jacobiano en la celda T_k.
φ_i^k	Elemento i de la base local de la celda T_k.
$\dim S$	Dimensión del espacio S.
$[IF]_{\mathcal{s}}$	Operador interior a la cara $\mathcal{s}$.
$[EF]_{\mathcal{s}}$	Operador exterior a la cara $\mathcal{s}$.
$\mathcal{N}_k$	Vecindario de la celda T_k.
μ	Pasos de un método de relajación.

CAPÍTULO 1
INTRODUCCIÓN

En las últimas décadas se han propuesto métodos de alto orden que combinan las ideas de las técnicas de elemento finito y de volumen finito. Con el fin de preservar propiedades físicas del problema, se han diseñado una serie de métodos llamados "Discontinuous Galerkin Methods" (DGM por sus siglas en inglés) o métodos de Galerkin discontinuos. En [1] se desarrolló un marco teórico abstracto bajo el cual se presentó un análisis de convergencia y de estabilidad de los métodos más importantes hasta el 2002. En [2] se realizó una comparación numérica de varios métodos DG para un problema elíptico modelo en 2D, en el cual se analizó el orden de convergencia, estabilidad, almacenamiento de memoria requerido, el rendimiento al resolver el sistema lineal y el comportamiento del condicionamiento espectral de la matriz de rigidez. En esta tesis se trabajará con el método "Local Discontinuous Galerkin" (LDG), el cual fué propuesto por B. Cockburn y C.W. Shu en [3] para problemas transitorios de convección-difusión, y analizado por P. Castillo, B. Cockburn, I. Perugia y D. Schötzau [4], para un problema modelo estacionario de difusión lineal. En [5] se realizó un análisis *hp* de convergencia para problemas lineales. El caso de difusión no lineal se discutió en [6].

Las implementaciones del método LDG conocidas hasta la fecha se restringen a mallas no estructuradas para dominios de $\mathbb{R}^2$. Para nuestro conocimiento existe solamente una librería que implementa el método LDG en 3D: *Deal II* desarrollada por W. Bangerth, R. Hartmann y G. Kanschat en [7]. Sin embargo, esta librería utiliza únicamente mallas Cartesianas y para algunos problemas donde el dominio es

irregular, como en medios porosos, es necesario el uso de mallas no estructuradas, ya que modelan mejor la geometría. En este trabajo se presenta una implementación del método LDG para dominios en 3D la cual utiliza mallas no estructuradas y polinomios de alto orden.

El paradigma de programación orientada a objetos provee de manera natural los mecanismos necesarios para una implementación desde un nivel abstracto. Como resultado se obtiene un código modular, fácil de mantener y que puede ser extendido por el usuario. El código se desarrolló utilizando el lenguaje de programación C++ y consiste de un conjunto de módulos independientes los cuales proveen una interfaz abstracta. De esta forma el código no está sujeto a clases concretas específicas como por ejemplo el uso de cuadraturas con una distribución de nodos simétrica o de bases de polinomios particulares.

En [2] se demostró que el condicionamiento espectral de la matriz de rigidez posee un orden asintótico de $\mathcal{O}(h^{-2})$ donde h es el tamaño de la malla. Por esta razón es importante la búsqueda de precondicionadores eficientes, los cuales reducen el condicionamiento de la matriz del sistema lineal, agilizando la resolución del mismo. En [8] se presenta un precondicionador semi-algebraico para el método LDG para mallas no estructuradas en 2D. En este trabajo se extienden las ideas presentadas en [8] para dominios en 3D, además de introducir algunas nuevas variantes de las propuestas al precondicionador semi-algebraico, como el uso de un método de relajación a nivel de bloques y otras ideas para resolver el problema auxiliar.

La tesis está organizada como sigue: en el Capítulo 2 se presenta brevemente la formulación del método LDG aplicado a un problema modelo; en el Capítulo 3 se discuten los operadores más importantes del método LDG. En el Capítulo 4 se mencionan algunas de las estructuras utilizadas para la implementación del método LDG, y en el Capítulo 5 se presentan algunos experimentos numéricos con el propósito de validar el código. En el Capítulo 6 se discuten los precondicionadores

multiniveles algebraicos y semi-algebraicos, se muestran experimentos numéricos para ilustrar lo eficiente y robustas que son las técnicas propuestas y finalmente, se presentan algunas conclusiones en la Capítulo 7.

CAPÍTULO 2
EL MÉTODO "LOCAL DISCONTINUOUS GALERKIN"

A continuación se presenta una breve descripción del método LDG, para aproximar la solución de problemas elípticos en 3D y condiciones de frontera de tipo Dirichlet y Robin:

$$\begin{aligned} -\nabla \cdot \mathcal{D}\nabla u &= f, \quad \text{en } \Omega, \\ u &= g_{\mathcal{D}}, \text{ sobre } \partial\Omega_{\mathcal{D}}, \\ au + \mathcal{D}\nabla u \cdot \vec{\boldsymbol{n}} &= g_{\mathcal{R}}, \text{ sobre } \partial\Omega_{\mathcal{R}}, \end{aligned}$$

donde $\mathcal{D}$ es un tensor a trozos simétrico positivo definido, Ω un dominio convexo y acotado de $\mathbb{R}^3$, y $\partial\Omega = \partial\Omega_{\mathcal{D}} \cup \partial\Omega_{\mathcal{R}}$.

Se introduce una nueva variable $\boldsymbol{q} = \mathcal{D}\nabla u$, la cual en algunos casos, como en medios porosos, representa el opuesto de la velocidad del fluido, o mejor conocida como la velocidad de Darcy, y la variable u representa la presión hidrostática [9]. Otra interpretación física para estas variables, se presenta en problemas de electromagnetismo, donde u es el potencial eléctrico, y $\boldsymbol{q}$ representa el opuesto del campo eléctrico [10]. En este sentido, la nueva variable también podría ser definida como $\boldsymbol{q} = -\mathcal{D}\nabla u$, la cual modela directamente una cantidad física de interés. Al realizar el cambio de variable indicado, obtenemos las ecuaciones de primer orden:

$$\begin{aligned} \mathcal{D}^{-1}\boldsymbol{q} &= \nabla u, \text{ en } \Omega, && (2.1)\\ -\nabla\cdot\boldsymbol{q} &= f, \quad \text{en } \Omega, && (2.2)\\ u &= g_D, \text{ sobre } \partial\Omega_{\mathcal{D}},\\ au+\boldsymbol{q}\cdot\vec{\boldsymbol{n}} &= g_{\mathcal{R}}, \text{ sobre } \partial\Omega_{\mathcal{R}}. \end{aligned}$$

Sea $\mathcal{T}_h$ una partición del dominio Ω, la cual consiste de un conjunto de tetrahedros. La formulación débil del problema se obtiene multiplicando la ecuación (2.1) y (2.2) por funciones de prueba $\boldsymbol{r}$ y v respectivamente, donde $\boldsymbol{r}$ es una función vectorial y v una función escalar. A diferencia del elemento finito clásico integramos por partes en cada elemento $T\in\mathcal{T}_h$, de donde se obtiene la siguiente formulación débil:

$$\begin{aligned} \int_T \mathcal{D}^{-1}\boldsymbol{q}\cdot\boldsymbol{r} &= \oint_{\partial T} u\boldsymbol{r}\cdot\vec{\boldsymbol{n}}_T - \int_T u\nabla\cdot\boldsymbol{r}, && (2.3)\\ \int_T \boldsymbol{q}\cdot\nabla v &= \oint_{\partial T} v\boldsymbol{q}\cdot\vec{\boldsymbol{n}}_T + \int_T fv, && (2.4) \end{aligned}$$

donde $\vec{\boldsymbol{n}}_T$ es un vector normal unitario exterior al elemento T. La solución exacta $(u,\boldsymbol{q})$ se aproxima por el par $(u_h,\boldsymbol{q}_h)$ que se encuentran en $\mathcal{V}_h\times\mathcal{M}_h$, donde

$$\begin{aligned} \mathcal{V}_h &= \left\{u\in L_2(\Omega) \;:\; u|_T\in\mathcal{P}_{k_T}(T), \forall\, T\in\mathcal{T}_h\right\},\\ \mathcal{M}_h &= \left\{\boldsymbol{q}\in\left(L_2(\Omega)\right)^3 \;:\; \boldsymbol{q}|_T\in\mathcal{P}_{k_T}(T)^3, \forall\, T\in\mathcal{T}_h\right\}. \end{aligned}$$

El espacio $\mathcal{P}_{k_T}(T)$ es el conjunto de polinomios de grado k_T en la celda T, donde $k_T\leq k$, para k fijo. La solución discreta $(u_h,\boldsymbol{q}_h)$ es definida usando la formulación débil anterior para cada $T\in\mathcal{T}_h$ y $(v,\boldsymbol{r})\in\mathcal{V}_h(T)\times\mathcal{M}_h(T)$ de la siguiente manera:

$$\int_T \mathcal{D}^{-1}\boldsymbol{q}_h \cdot \boldsymbol{r} - \oint_{\partial T} \widehat{u_{h\{\mathcal{S},T\}}}\boldsymbol{r}\cdot\vec{\boldsymbol{n}}_T + \int_T u_h \nabla\cdot\boldsymbol{r} = 0, \tag{2.5}$$

$$\int_T \boldsymbol{q}_h \cdot \nabla v - \oint_{\partial T} v\widehat{\boldsymbol{q}_{h\{\mathcal{S},T\}}}\cdot\vec{\boldsymbol{n}}_T + \oint_{\partial T} v\alpha_{\mathcal{S}}(u_h)\cdot\vec{\boldsymbol{n}}_T = \int_T fv, \tag{2.6}$$

donde $\widehat{u_{h\{\mathcal{S},T\}}}$ y $\widehat{\boldsymbol{q}_{h\{\mathcal{S},T\}}}$ son llamados *flujos numéricos*, los cuales aproximan las trazas de las funciones u y $\boldsymbol{q}$, respectivamente, en la cara $\mathcal{S}$ del elemento T, $\mathcal{S}\in\partial T$. La formulación anterior no se restringe a mallas conformes como las que se utilizan en elemento finito clásico. Las definiciones de los espacios permiten que los grados de los polinomios puedan ser diferentes en cada elemento $T\in\mathcal{T}_h$, por lo que es posible tener refinamiento *hp* para el método LDG (Figura 2–1). El término $\alpha_{\mathcal{S}}(u_h)$ juega el papel de parámetro de estabilidad, el cual garantiza la existencia de la solución del problema discreto bajo ciertas condiciones.

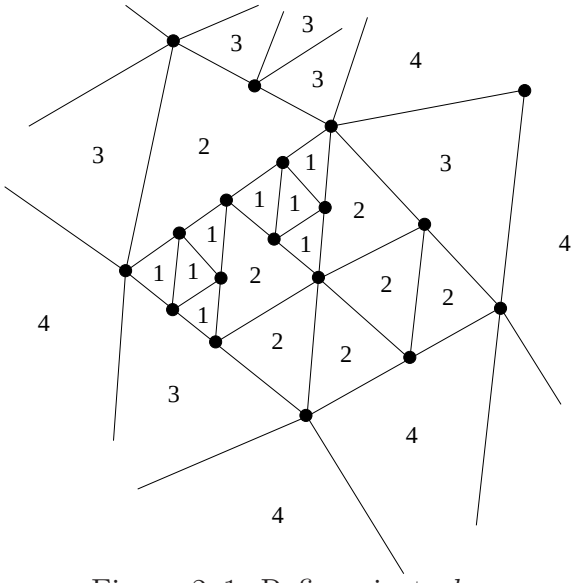


Figura 2–1: Refinamiento *hp*

Sea $\mathcal{s}$ una cara interior, la cual es compartida por los elementos T y K. Los flujos numéricos para el LDG vienen dados de la siguiente manera:

$$\begin{aligned}\widehat{u_h}_{\{\mathcal{s},T\}} &= \{u_h\} + \boldsymbol{\beta}_{\mathcal{s}} \cdot [\![u_h]\!], \\ \widehat{\boldsymbol{q}_h}_{\{\mathcal{s},T\}} &= \{\boldsymbol{q}_h\} - \boldsymbol{\beta}_{\mathcal{s}} \cdot [\![\boldsymbol{q}_h]\!], \\ \alpha_{\mathcal{s}}(u_h) &= \eta_{\mathcal{s}} [\![u_h]\!],\end{aligned}$$

donde $[\![v]\!]$ y $\{v\}$ denotan el *salto* y el *promedio* (respectivamente) de la función v, es decir:

$$[\![v]\!] = v|_K \vec{\boldsymbol{n}}_K + v|_T \vec{\boldsymbol{n}}_T \quad \text{y} \quad \{v\} = \frac{1}{2}(v|_K + v|_T),$$

los parámetros $\boldsymbol{\beta}_{\mathcal{s}}$ y $\eta_{\mathcal{s}}$ dependen de la cara. Donde $\boldsymbol{\beta}_{\mathcal{s}}$ puede ser escogido de manera albitraria, mientras que $\eta_{\mathcal{s}}$ es una función que depende del diámetro de la cara $\mathcal{s}$ ($h_{\mathcal{s}}$), definida como:

$$\eta_{\mathcal{s}}(h_{\mathcal{s}}) = \frac{\lambda_{\mathcal{s}}}{h_{\mathcal{s}}}.$$

La convergencia y la estabilidad del método LDG fueron probadas en [4] para $\lambda_{\mathcal{s}} > 0$. Vemos que según la definión de $\widehat{u_h}_{\{\mathcal{s},T\}}$, podemos escribir el flujo como una combinación convexa, de u_h en la celda T y en la celda K:

$$\widehat{u_h}_{\{\mathcal{s},T\}} = \gamma_{\mathcal{s}} u_{h,T} + \zeta_{\mathcal{s}} u_{h,K}, \tag{2.7}$$

con $\gamma_{\mathcal{s}} + \zeta_{\mathcal{s}} = 1 \quad \text{y} \quad \gamma_{\mathcal{s}}, \zeta_{\mathcal{s}} \geq 0$, lo que garantiza consistencia en el método. La definición de los flujos numéricos para una cara de frontera $\mathcal{s}$, se muestra en la Tabla 2–1. Donde las condiciones de Neumann se obtienen de las condiciones de Robin tomando $a = 0$, es decir, $\mathcal{D}\nabla u \cdot \vec{\boldsymbol{n}} = g_{\mathcal{N}}$.

Tabla 2–1: Definición de los flujos numéricos en la frontera

Flujo	Dirichlet	Neumann	Robin
$\widehat{u_{h\{\mathcal{S},T\}}}$	$g_{\mathcal{D}}$	u_h	u_h
$\widehat{\boldsymbol{q}_{h\{\mathcal{S},T\}}} \cdot \vec{\boldsymbol{n}}_T$	$\boldsymbol{q}_h \cdot \vec{\boldsymbol{n}}_T$	$g_{\mathcal{N}}$	$g_{\mathcal{R}} - a u_h$
$\alpha_{\mathcal{S}}(u_h)$	$\eta_{\mathcal{S}} \left(u_h - g_{\mathcal{D}}\right) \vec{\boldsymbol{n}}_T$	0	0

La idea general para implementar cualquier condición de borde consiste en definir el flujo numérico en esa cara, como el valor de la condición de borde, si esta se conoce, en caso contrario, se repesenta en términos de la variable asociada u_h o $\boldsymbol{q}_h$. Por ejemplo, para condiciones de tipo Dirichlet se tiene $u = g_{\mathcal{D}}$, es decir, se conoce el valor de la variable u en esa cara, entonces se considera $\widehat{u_{h\{\mathcal{S},T\}}} = g_{\mathcal{D}}$. En el caso de la variable $\boldsymbol{q}$, no se tiene valor, entonces se toma $\widehat{\boldsymbol{q}_{h\{\mathcal{S},T\}}} = \boldsymbol{q}_h$.

CAPÍTULO 3
SISTEMA LINEAL EN SU FORMA MIXTA

El sistema lineal generado por el método LDG tiene la estructura de bloques:

$$\begin{pmatrix} D & B \\ -B^T & S \end{pmatrix} \begin{pmatrix} \boldsymbol{q}_h \\ u_h \end{pmatrix} = \begin{pmatrix} b_q \\ b_u \end{pmatrix}, \tag{3.1}$$

donde las matrices D, B, $-B^T$ y S son las versiones discretas de la difusión, gradiente, divergencia y estabilidad, respectivamente.

En este capítulo describimos en detalle el cálculo de las matrices D, B y S tanto para mallas estructuradas como no estructuradas. Por simplicidad, se asume que el grado de aproximación es el mismo en todas las celdas. En el caso de grado variable, se consideran listas de bases y cuadraturas para cada celda, lo que conduce a tener diferentes operadores asociados a una celda. El uso de una base jerárquica puede ayudar a simplificar la implementación. Una forma de construir una base jerárquica en el tetrahedro, es tomar la base canónica y ordenar los elementos de la misma, de menor a mayor grado. Por ejemplo, para grado $p = 1$ se tienen los elementos $\{1, x, y, z\}$, mientras que para grado $p = 2$ los elementos son $\{1, x, y, z, x^2, y^2, z^2, xy, xz, yz\}$. Otra forma de construir una base jerárquica es a través del proceso de ortogonalización de Gram-Schmidt, primero ordenando los elementos de la base que se va a ortogonalizar, de menor a mayor grado. En [11] se mencionan otras formas de construir bases jerárquicas. En la Tabla 3–1 se presentan las matrices de masa de la base canónica para grado $p = 1$ y $p = 2$. Se puede apreciar que la matriz de masa de una base jerárquica para grado p, contiene las matrices de

masa para los grados menores que p, lo que permite en la implementación almacenar solamente la matriz de masa con grado mayor, y así, de ella extraer las matrices de los demás grados. Se consideran únicamente aquellas mallas cuyas celdas son equivalentes mediante una transformación lineal o afín a una celda fija llamada celda de referencia. En otras palabras, se denota por $\hat{T}$ la celda de referencia entonces para cada celda $T_k \in \mathcal{T}_h$ existe un mapeo afín $\phi_k(\hat{x}) = \mathcal{J}_k\hat{x} + b$ tal que $\phi_k(\hat{T}) = T_k$, donde el Jacobiano $\mathcal{J}_k$ satisface $|\mathcal{J}_k| = \det\mathcal{J}_k > 0$.

Tabla 3–1: Matriz de masa de la base canónica para $p = 1, 2$

$p = 1$

$$\begin{pmatrix} \frac{1}{6} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} \\ \frac{1}{24} & \frac{1}{60} & \frac{1}{120} & \frac{1}{120} \\ \frac{1}{24} & \frac{1}{120} & \frac{1}{60} & \frac{1}{120} \\ \frac{1}{24} & \frac{1}{120} & \frac{1}{120} & \frac{1}{60} \end{pmatrix}$$

$p = 2$

$$\begin{pmatrix} \frac{1}{6} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{60} & \frac{1}{60} & \frac{1}{60} & \frac{1}{120} & \frac{1}{120} & \frac{1}{120} \\ \frac{1}{24} & \frac{1}{60} & \frac{1}{120} & \frac{1}{120} & \frac{1}{120} & \frac{1}{360} & \frac{1}{360} & \frac{1}{360} & \frac{1}{360} & \frac{1}{720} \\ \frac{1}{24} & \frac{1}{120} & \frac{1}{60} & \frac{1}{120} & \frac{1}{360} & \frac{1}{120} & \frac{1}{360} & \frac{1}{360} & \frac{1}{720} & \frac{1}{360} \\ \frac{1}{24} & \frac{1}{120} & \frac{1}{120} & \frac{1}{60} & \frac{1}{360} & \frac{1}{360} & \frac{1}{120} & \frac{1}{720} & \frac{1}{360} & \frac{1}{360} \\ \frac{1}{60} & \frac{1}{120} & \frac{1}{360} & \frac{1}{360} & \frac{1}{210} & \frac{1}{1260} & \frac{1}{1260} & \frac{1}{840} & \frac{1}{840} & \frac{1}{2520} \\ \frac{1}{60} & \frac{1}{360} & \frac{1}{120} & \frac{1}{360} & \frac{1}{1260} & \frac{1}{210} & \frac{1}{1260} & \frac{1}{840} & \frac{1}{2520} & \frac{1}{840} \\ \frac{1}{60} & \frac{1}{360} & \frac{1}{360} & \frac{1}{120} & \frac{1}{1260} & \frac{1}{1260} & \frac{1}{210} & \frac{1}{2520} & \frac{1}{840} & \frac{1}{840} \\ \frac{1}{120} & \frac{1}{360} & \frac{1}{360} & \frac{1}{720} & \frac{1}{840} & \frac{1}{840} & \frac{1}{2520} & \frac{1}{1260} & \frac{1}{2520} & \frac{1}{2520} \\ \frac{1}{120} & \frac{1}{360} & \frac{1}{720} & \frac{1}{360} & \frac{1}{840} & \frac{1}{2520} & \frac{1}{840} & \frac{1}{2520} & \frac{1}{1260} & \frac{1}{2520} \\ \frac{1}{120} & \frac{1}{720} & \frac{1}{360} & \frac{1}{360} & \frac{1}{2520} & \frac{1}{840} & \frac{1}{840} & \frac{1}{2520} & \frac{1}{2520} & \frac{1}{1260} \end{pmatrix}$$

En la Figura 3–1 se muestra la celda de referencia $\hat{T}$ seleccionada en esta tesis. La mayor parte de los cálculos, principalmente los que involucran integración, se realizan en $\hat{T}$ y luego, por medio del Jacobiano $\mathcal{J}_k$ de la función ϕ_k se mapean a la celda T_k.

Para cada $T_k \in \mathcal{T}_h$ denotamos por $\Phi = \{\varphi_i^k\}$ una base para el espacio local de polinomios $\mathcal{V}_h(T_k)$ (variable primaria o potencial) y por $\mathbf{\Psi} = \Phi \times \Phi \times \Phi$ una base para el espacio local de polinomios $\mathcal{M}_h(T_k)$ (variable secundaria o gradiente). Obsérvese que estas bases pueden ser construidas a partir de bases en el elemento

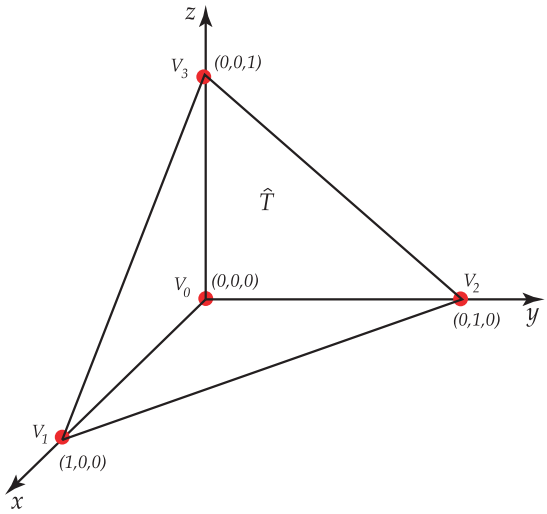


Figura 3–1: Celda de referencia $\hat{T}$

de referencia ($\varphi_i^k = \hat{\varphi}_i \circ \phi_k^{-1}$) como se muestra en el diagrama conmutativo de la Figura 3–2.

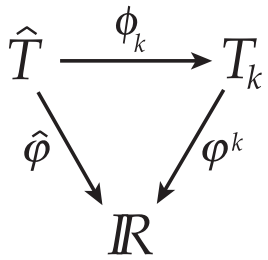


Figura 3–2: Construcción de una base en T_k a partir de una en $\hat{T}$

Por regla de la cadena se cumple que:

$$\varphi_i^k \;=\; \hat{\varphi}_i \circ \phi_k^{-1},$$

de donde

$$[D\varphi_i^k] \;=\; [D\hat{\varphi}_i] \cdot [D\phi_k^{-1}],$$

donde $[D\phi_k^{-1}] = \mathcal{J}_k^{-1}$, y

$$\mathcal{J}_k^{-1} \;=\; \begin{bmatrix} \partial_x\hat{x} & \partial_y\hat{x} & \partial_z\hat{x} \\ \partial_x\hat{y} & \partial_y\hat{y} & \partial_z\hat{y} \\ \partial_x\hat{z} & \partial_y\hat{z} & \partial_z\hat{z} \end{bmatrix}.$$

Finalmente, se denota por N el número total de celdas en $\mathcal{T}_h$.

3.1 Operador de difusión

Sea $\mathcal{D}_k$ el tensor de difusión de la celda T_k, el cual se asume constante en cada celda y representa las propiedades del material en esa celda, la representación matricial del operador discreto de difusión relativo a la base $\mathbf{\Psi}$ es una matriz diagonal de $N \times N$ bloques. Además, como $\mathbf{\Psi}$ es un producto de bases escalares, cada bloque de la diagonal puede ser factorizado mediante el producto Kronecker de dos matrices como se muestra a continuación

$$D_{kk} = \left[\int_{T_k} \boldsymbol{\psi}_i^k \cdot \mathcal{D}_k^{-1} \boldsymbol{\psi}_j^k \right] = |\mathcal{J}_k| \mathcal{D}_k^{-1} \otimes M,$$

donde $M = \left[\int_{\hat{T}} \hat{\varphi}_i \hat{\varphi}_j \right]$ es la matriz de masa asociada a la celda de referencia $\hat{T}$. Obsérvese que siendo D una matriz diagonal por bloques esta puede ser invertida fácilmente. Además la inversa de cada bloque de la diagonal es también un producto Kronecker:

$$D_{kk}^{-1} = \left[\int_{T_k} \boldsymbol{\psi}_i^k \cdot \mathcal{D}_k^{-1} \boldsymbol{\psi}_j^k \right]^{-1} = |\mathcal{J}_k|^{-1} \mathcal{D}_k \otimes M^{-1}.$$

Nótese que las dimensiones globales de las matrices D y D^{-1} son

$$N\text{dim } \mathcal{M}_h(\hat{T}) \times N\text{dim } \mathcal{M}_h(\hat{T}) \;=\; 3N\text{dim } \mathcal{V}_h(\hat{T}) \times 3N\text{dim } \mathcal{V}_h(\hat{T}).$$

Gracias al producto Kronecker y a que las celdas son linealmente equivalentes no es necesario almacenar explícitamente ninguna de estas matrices. Todo se reduce al cálculo de la matriz M^{-1}.

3.2 Operador de gradiente

La representación discreta del gradiente está dada por la matriz B, la cual tiene una estructura esparcida de $N \times N$ bloques rectangulares cuyas dimensiones son dim $\mathcal{M}_h(T_k) \times$ dim $\mathcal{V}_h(T_k)$. El operador gradiente asociado a la celda T_k se obtiene a partir de la siguiente expresión:

$$\int_{T_k} u_h \nabla \cdot \boldsymbol{\psi}_m^k - \oint_{\partial T_k} \widehat{u_{h\{s,T_k\}}} \boldsymbol{\psi}_m^k \cdot \vec{\boldsymbol{n}}_k. \tag{3.2}$$

Como todas las celdas son linealmente equivalentes a la celda de referencia $\hat{T}$, la contribución al bloque B_{kk} por el término que involucra la integral de volumen de la ecuación (3.2) puede ser calculado como sigue:

$$\left[\int_{T_k} \varphi_n^k \nabla \cdot \boldsymbol{\psi}_m^k\right] = |\mathcal{J}_k| \left(\begin{bmatrix} \partial_x \hat{x} \\ \partial_y \hat{x} \\ \partial_z \hat{x} \end{bmatrix} \otimes DX + \begin{bmatrix} \partial_x \hat{y} \\ \partial_y \hat{y} \\ \partial_z \hat{y} \end{bmatrix} \otimes DY + \begin{bmatrix} \partial_x \hat{z} \\ \partial_y \hat{z} \\ \partial_z \hat{z} \end{bmatrix} \otimes DZ \right),$$

donde

$$DX = \left[\int_{\hat{T}} \hat{\varphi}_n \partial_{\hat{x}} \hat{\varphi}_m\right], \quad DY = \left[\int_{\hat{T}} \hat{\varphi}_n \partial_{\hat{y}} \hat{\varphi}_m\right] \quad \text{y} \quad DZ = \left[\int_{\hat{T}} \hat{\varphi}_n \partial_{\hat{z}} \hat{\varphi}_m\right].$$

Las matrices DX, DY y DZ se calculan solamente para la celda de referencia. Los demás coeficientes pueden ser obtenidos de la información geométrica de la celda T_k, mediante $\mathcal{J}_k^{-1}$.

Ahora, para la integral de superficie de la ecuación (3.2), consideramos s una cara interior, compartida por las celdas T_k y T_i, como se muestra en la Figura 3–3.

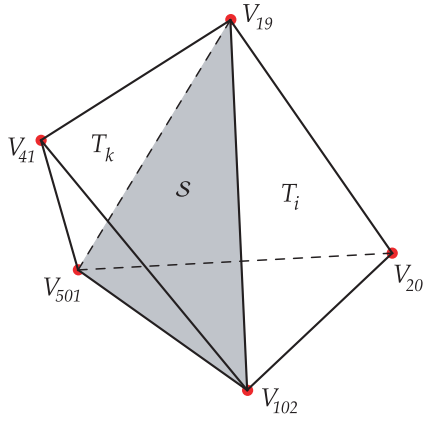


Figura 3–3: Cara interior

Sean u_h^k y u_h^i las aproximaciones de la solución en las celdas T_k y T_i respectivamente. El flujo numérico $\widehat{u_h}_{\{\mathcal{S},T_k\}}$ se define como una combinación convexa (ver 2.7) de estas cantidades de la siguiente manera:

$$\widehat{u_h}_{\{\mathcal{S},T_k\}} \;=\; \gamma_i u_h^k + \zeta_i u_h^i,$$

donde $(\gamma_i,\zeta_i) = (1/2,1/2)$, $(\gamma_i,\zeta_i) = (1,0)$ o $(\gamma_i,\zeta_i) = (0,1)$. Las últimas dos opciones para la selección de estos parámetros fueron utilizadas en [12] en el diseño de heurísticas para la reducción del número de bloques no nulos en el complemento de Schur. Para una cara interior $\mathcal{S}$ se tiene

$$\oint_{\mathcal{S}} \widehat{u_h}_{\{\mathcal{S},T_k\}} \boldsymbol{\psi}_m^k \cdot \vec{\boldsymbol{n}}_k \;=\; \gamma_i \oint_{\mathcal{S}} u_h^k \boldsymbol{\psi}_m^k \cdot \vec{\boldsymbol{n}}_k + \zeta_i \oint_{\mathcal{S}} u_h^i \boldsymbol{\psi}_m^k \cdot \vec{\boldsymbol{n}}_k.$$

La primera integral involucra únicamente las trazas de u_h dentro de la celda T_k y contribuye al bloque B_{kk}. La segunda integral involucra las trazas por ambos lados de la cara y contribuye exclusivamente al bloque B_{ki} que se encuentra fuera de la diagonal.

La representación matricial correspondiente es:

$$\left[\gamma_i \oint_{\mathcal{S}} u_h^k \boldsymbol{\psi}_m^k \cdot \vec{\boldsymbol{n}}_k\right] \;=\; \gamma_i \vec{\boldsymbol{n}}_k \otimes [IF]_{\mathcal{S}} \quad \text{y} \quad \left[\zeta_i \oint_{\mathcal{S}} u_h^i \boldsymbol{\psi}_m^k \cdot \vec{\boldsymbol{n}}_k\right] \;=\; \zeta_i \vec{\boldsymbol{n}}_k \otimes [EF]_{\mathcal{S}},$$

donde $[IF]_{\mathcal{S}}$, lo llamaremos operador interior de cara, y $[EF]_{\mathcal{S}}$, operador exterior de cara. Estas son matrices de tamaño dim $\mathcal{V}_h(T_k) \times$ dim $\mathcal{V}_h(T_k)$ y dim $\mathcal{V}_h(T_k) \times$ dim $\mathcal{V}_h(T_i)$ respectivamente y, están definidas de la siguiente manera

$$[IF]_{\mathcal{S}} = \left[\oint_{\mathcal{S}} \varphi_n^k \varphi_m^k\right] \quad \text{y} \quad [EF]_{\mathcal{S}} = \left[\oint_{\mathcal{S}} \varphi_n^i \varphi_m^k\right]. \tag{3.3}$$

En el caso de una cara de borde, de acuerdo a la Tabla 2–1 se tiene $\widehat{u_h}_{\{\mathcal{S},T_k\}} = g_{\mathcal{D}}$ para cualquier cara de borde $\mathcal{S}$ donde se imponen condiciones de Dirichlet. Por lo

tanto s no afecta la matriz B. Por otro lado, si se tienen condiciones de tipo Neumann o Robin en la cara s se tiene $\widehat{u_h}_{\{s,T_k\}} = u_h$, por lo que se puede fijar $\gamma_i = 1$ y $\zeta_i = 0$.

Como la malla consiste de celdas linealmente equivalentes, las integrales pueden ser todas precalculadas en la celda de referencia. Para el operador $[EF]_s$, las funciones de la base son evaluadas en ambos lados de la cara, por lo que se consideran todas las combinaciones posibles de pares de caras en la celda de referencia, así como sus rotaciones, para un total de 48 posibilidades. En la Figura 3–4 se presentan todas las posibles rotaciones para la cara sombreada. Dentro de la celda superior, la cara corresponde al lado 3, sin embargo dentro de la celda inferior esta cara puede ser etiquetada de cuatro formas distintas.

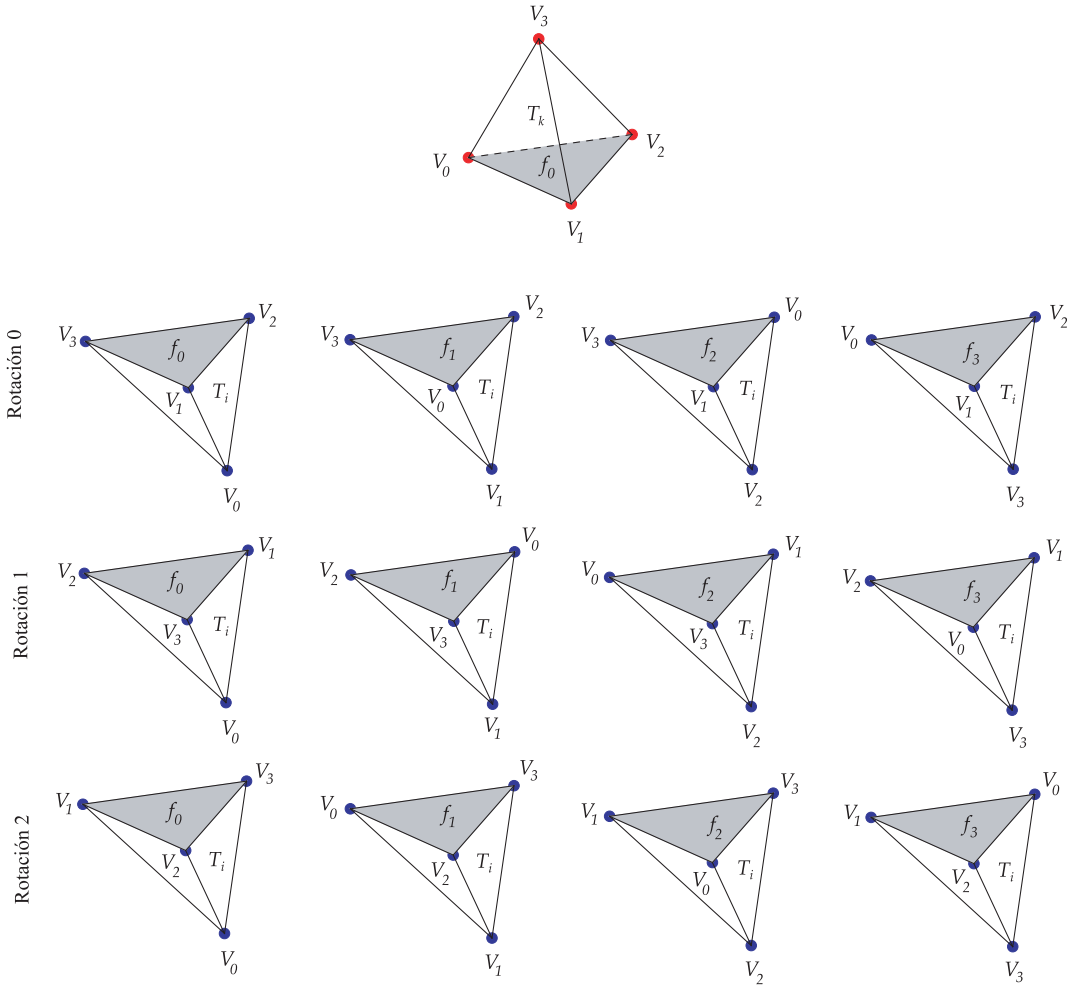


Figura 3–4: Posibles rotaciones de una cara compartida

3.3 Operador de estabilidad

De acuerdo a [4] el parámetro de estabilidad $\alpha_{\mathcal{S}}(\cdot)$ se define como sigue

$$\alpha_{\mathcal{S}}(u_h) \;=\; \eta_{\mathcal{S}}\left(u_h^k\vec{\boldsymbol{n}}_k + u_h^i\vec{\boldsymbol{n}}_i\right) \;=\; \eta_{\mathcal{S}}\left(u_h^k - u_h^i\right)\vec{\boldsymbol{n}}_k.$$

De donde se tiene que la matriz de estabilidad S es una matriz esparcida de $N \times N$ bloques, donde los bloques S_{kk} y S_{ki} se leen

$$S_{kk} \;=\; \sum_{\mathcal{S}\in\partial T_k} \eta_{\mathcal{S}}[IF]_{\mathcal{S}} \quad \text{y} \quad S_{ki} \;=\; -\eta_{\mathcal{S}}[EF]_{\mathcal{S}},$$

cuando $\mathcal{s}$ es una cara interior. Para una cara de borde $\mathcal{s}$ con condiciones de Dirichlet, esta contribuye al bloque S_{kk} de igual manera a la anterior. Cuando a $\mathcal{s}$ se le imponen condiciones de Neumann, $\mathcal{s}$ no afecta la matriz S. Para $\mathcal{s}$ con condiciones de Robin, se tiene $\eta_{\mathcal{S}} = a$ y $\mathcal{s}$ contribuye al bloque S_{kk} análogamente a una cara interior.

3.4 Término fuente

El vector global del sistema lineal (3.1) se descompone en dos vectores: el primero, b_q, se obtiene a partir de las condiciones de frontera de tipo Dirichlet. Si $\mathcal{s}$ $\in \partial T_k \cap \partial\Omega_{\mathcal{D}}$, la contribución de la cara $\mathcal{s}$ a este vector b_q se obtiene por el vector local

$$\left[\oint_{\mathcal{s}} g_{\mathcal{D}}\boldsymbol{\psi}_m^k \cdot \vec{\boldsymbol{n}}_k\right] \;=\; \vec{\boldsymbol{n}}_k \otimes \left[\oint_{\mathcal{s}} g_{\mathcal{D}}\varphi_m^k\right].$$

El tamaño de este vector es igual a 3dim $\mathcal{V}_h(T_k)$. La segunda componente es el vector b_u que depende del término fuente f, así como de los dos tipos de condiciones de frontera. La contribución final de la celda T_k al vector b_u es un vector de dimensión dim $\mathcal{V}_h(T_k)$ el cual puede ser calculado de la siguiente manera

$$\left[\int_{T_k} f\varphi_m^k\right] + \sum_{\mathcal{S}\in\partial T_k\cap\partial\Omega_{\mathcal{D}}}\left[\eta_{\mathcal{S}}\oint_{\mathcal{s}} g_{\mathcal{D}}\varphi_m^k\right] + \sum_{\mathcal{S}\in\partial T_k\cap\partial\Omega_{\mathcal{R}}}\left[\oint_{\mathcal{s}} g_{\mathcal{R}}\varphi_m^k\right].$$

El primer término se debe a la fuente, función f, el segundo y tercer término corresponden a condiciones de frontera de tipo Dirichlet y Robin, respectivamente.

Las condiciones de Neumann contribuyen al vector b_u de la misma manera que las condiciones de Robin, tomando $g_{\mathcal{R}} = g_{\mathcal{N}}$.

3.5 Ensamblaje rápido del complemento de Schur

En la práctica se resuelve para la variable primaria u_h. Esto se puede hacer mediante eliminación Gaussiana por bloques, obteniendo el siguiente sistema mejor conocido como el complemento de Schur $\mathcal{A}$:

$$\mathcal{A}u_h = b_u + B^T D^{-1} b_q, \text{ donde } \mathcal{A} = S + B^T D^{-1} B. \tag{3.4}$$

Como se vío en la Sección 3.1, la matriz D puede ser invertida explícitamente, lo cual resulta muy conveniente para la obtención del complemento de Schur de manera directa.

Una vez calculados los operadores D, B y S, se obtiene el complemento de Schur $\mathcal{A}$, de la forma $\mathcal{A} = S + B^T D^{-1} B$. Sin embargo, en la práctica no es necesario calcular los operadores D, B y S de manera global. Se puede reducir el almacenamiento necesario del complemento de Schur calculando este de manera local en cada celda $T_k \in \mathcal{T}_h$, por medio del siguiente algoritmo:

Algoritmo 3.5: FastSchurComplementAssembly

```
1  A ← S
2  para cada T_k ∈ T_h hacer
3      Calcular bloques D_kk^{-1} y B_ki para i ∈ N_k
4      para cada i ∈ N_k hacer
5          para cada j ∈ N_k hacer
6              A_ij ← A_ij + B_ki D_kk^{-1} B_kj
```

Donde $\mathcal{N}_k$ es el conjunto de índices formado por k y el conjunto de índices de las celdas adjacentes a la celda T_k, $\mathcal{V}_k$, conocido como *el vecindario de* T_k, es decir, $\mathcal{N}_k = \{k\} \cup \mathcal{V}_k$. En la línea 1 del algoritmo se inicializa $\mathcal{A}$ como el operador S de manera global, es decir, es necesario ensamblar S primero, para luego agregar las contribuciones locales de los demás operadores.

Cuando se tiene un esténcil completo en mallas compuestas por tetrahedros, este algoritmo es $\mathcal{O}(25N)$. La complejidad de este algoritmo es lineal en términos del número de celdas de la malla. La simetría de la matriz $\mathcal{A}$ reduce la complejidad del algoritmo, al disminuir los cálculos de la línea 6. Una vez calculada la contribución $C = B_{ik}^T D_{kk} B_{jk}$ del bloque $\mathcal{A}_{ki}$, se puede actualizar de igual manera el bloque $\mathcal{A}_{ik}$, por la contribución $C^T = B_{jk}^T D_{kk} B_{ik}$. De esta forma, la complejidad del algoritmo en mallas compuestas por tetrahedros es $\mathcal{O}(15N)$.

A manera de resumen, en la Tabla 3–2 se muestran las dimensiones que posee cada operador, donde n representa la dimensión del dominio.

Tabla 3–2: Dimensiones de las matrices

Matriz	Dimensión por bloques	Tamaño de bloque
D	$N \times N$	$(n \dim \mathcal{V}_{h,T}) \times (n \dim \mathcal{V}_{h,T})$
B	$N \times N$	$(n \dim \mathcal{V}_{h,T}) \times \dim \mathcal{V}_{h,T}$
S	$N \times N$	$\dim \mathcal{V}_{h,T} \times \dim \mathcal{V}_{h,T}$
$\mathcal{A}$	$N \times N$	$\dim \mathcal{V}_{h,T} \times \dim \mathcal{V}_{h,T}$

CAPÍTULO 4
ESTRUCTURA DE DATOS

Las estructuras que se diseñaron para la implementación del método LDG, están basadas en el paradigma de la programación orientada a objectos (OOP por sus siglas en ingles). Actualmente existen varios lenguajes de programación para trabajar en computación científica que permiten OOP. En nuestra implementación elegimos el lenguaje C++ [13] debido a que existen estudios que prueban que ha superado a Java, en cuanto al desempeño en cálculos numéricos [14, 15], además que facilita enlazar librerías hechas en FORTRAN, por ejemplo, LAPACK [16].

En general, para implementar un método DG, se deben desarrollar módulos para las principales necesidades, las cuales son:

- Malla,
- Geometría,
- Matrices esparcidas,
- Bases de polinomios,
- Cuadraturas.

En la Figura 4–1 se muestra un esquema de módulos para un método DG en general, basta con cambiar los *Operadores* para tener un método DG diferente.

4.1 La malla

La malla no adaptiva contiene una lista de elementos que representan las celdas de la discretización $\mathcal{T}_h$. La clase **Mesh3D** (Tabla 4–1) tiene como atributos la lista de nodos, celdas, caras interiores y caras de borde, asignandoles así un índice global a cada uno de estos elementos en la malla.

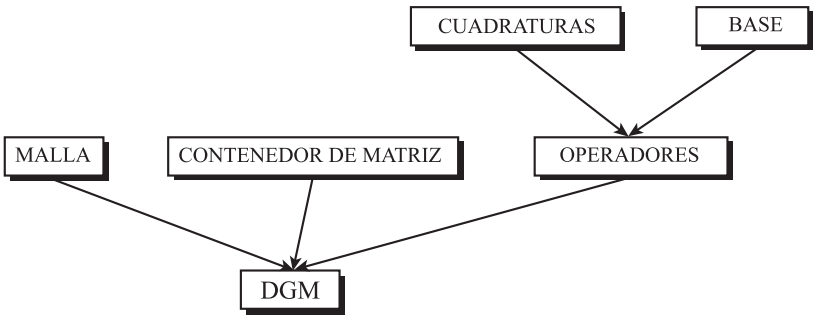


Figura 4–1: Esquema de operadores para DGM

Tabla 4–1: Estructura de datos de la malla Mesh3D

```
class Mesh3D {
public:
  Mesh3D();
 ~Mesh3D();
  uint          getNumNodes() const;
  uint          getNumFaces() const;
  uint          getNumCells() const;
  uint          getNumBoundaryFaces() const;
  Element3D*    getGeometry(uint) const;
  const Cell3D& getCell(uint) const;
  const Face3D& getFace(uint) const;
  const Face3D& getBoundaryFace(uint) const;
  void          read(const string);

private:
   ...
};
```

La topología de cada celda, es representada por la clase **Cell3D** (Tabla 4–2) cuyos atributos son cuatro índices a sus nodos, cuatro índices a sus caras y la información del material de la celda. La numeración local de los nodos sigue una orientación para que al calcular las normales en cada cara, éstas vayan hacia afuera de la celda, como se muestra en la Tabla 4–3. Además, garantiza que el determinante del jacobiano en la celda sea positivo.

Tabla 4–2: Estructura de datos de la topología de la celda Cell3D

```
class Cell3D {
public:
  Cell3D();
 ~Cell3D();
  uint getNumNodes() const;
  uint getNumFaces() const;
  uint getMaterialId() const;
  uint getGlobalNodeId(uint) const;
  void getLocalFace(uint, uint[]) const;
  void getGlobalFace(uint,uint[]) const;
  uint getGlobalFaceId(uint) const;

private:
   ...
};
```

Tabla 4–3: Orientación de los nodos en las caras

Cara	Orientación de los nodos
0	V_1, V_2, V_3
1	V_0, V_3, V_2
2	V_0, V_1, V_3
3	V_0, V_2, V_1

La clase **Face3D** (Tabla 4–4) contiene información sobre una cara, es decir, sus atributos son el índice global en la malla, los tres índices de sus nodos, dos índices de las celdas que comparten dicha cara, además del lado que tiene la cara en cada celda, así como su rotación. Una cara compartida por dos celdas podría representarse de distintas maneras en cada una de ellas, debido a la orientación que sigue la cara en cada celda. Sin embargo, por la orientación misma, existe un nodo en la cara que es localmente etiquetado de la misma forma por ambas celdas. Si el nodo común, por ejemplo, es el nodo 0 entonces se dice que esa cara tiene *rotación 0*. Así, cada cara tiene tres posibles rotaciones. En la Figura 4–2 se muestran las

posibles rotaciones para una cara compartida. En el caso de una cara de borde, se almacena la condición de frontera asociada a dicha cara.

Tabla 4–4: Estructura de datos de la cara Face3D

```
class Face3D {
public:
  Face3D();
 ~Face3D();
  uint getNumNodes() const;
  uint getFaceId() const;
  uint getCellId() const;
  uint getNeighborId(uint) const;
  uint getSideId() const;
  bool isBoundaryFace() const;
  uint getBoundaryId() const;
  uint getRotation() const;

private:
   ...
};
```

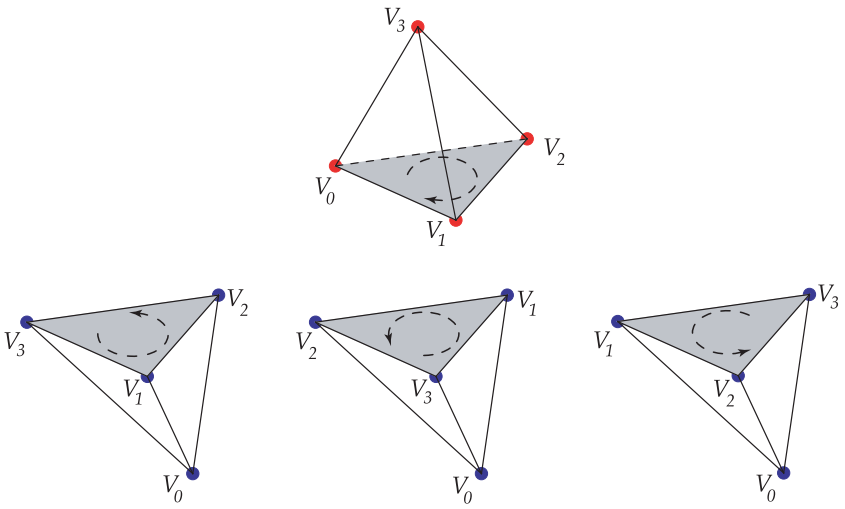


Figura 4–2: Posibles rotaciones de una cara compartida

4.2 Geometría

La clase **Element3D** (Tabla 4–5), representa la información geométrica de una celda. Por ejemplo, la información del diámetro y el volumen de la celda. Además,

provee información de las caras, como ser el área y el vector normal a la cara. Esta clase retorna la matriz Jacobiana ($\mathcal{J}_k$) del mapeo ϕ_k, el cual permite tranformar los operadores, previamente calculados en la celda de referencia, a la celda actual T_k. El método *getPoint()* mapea un punto en la celda de referencia a la celda actual.

Tabla 4–5: Estructura de datos de la geometría

Element3D

```
class Element3D {
public:
  Element3D();
 ~Element3D();
  void            setVertices(const Point3D[]);
  Point3D         getPoint(const Point3D&) const;
  const Mat3x3&   getJacobian() const;
  const Vector3D& getFaceNormal(uint) const;
  double          getFaceArea(uint) const;
  double          getVolume() const;
  double          getDiameter() const;
  uint            getNumNodes() const;
  uint            getNumFaces() const;

private:
    ...
};
```

4.3 Matrices esparcidas

Para el manejo de matrices D, B S y $\mathcal{A}$ utilizamos una versión por bloques del formato "Compressed Sparse Row", o CSR [17], el cual se utiliza comúnmente para el almacenamiento de matrices esparcidas. Esta versión es ideal para la versión p del método: por un lado se asume una estructura de bloques la cual es natural para aproximaciones de alto orden y por otro, se almacena solamente los bloques no nulos de la matriz utilizando una estrategia similar a la del formato CSR. En lugar de utilizar librerías existentes, como por ejemplo la librería BPKIT desarrollada en [18], se implementó una versión más modesta, enfocada a las necesidades del método. En la Figura 4–3 se muestra la estructura utilizada, en la cual se considera una matriz

A esparcida por bloques, que tiene $M \times N$ bloques, y que las dimensiones del bloque i, j son $m_i \times n_j$, es decir, los bloques pueden ser de tamaños no uniformes. La lista iA representa las M filas de la matriz A, donde cada fila es una nueva lista jA_i, que almacena los bloques no nulos presentes en la fila i.

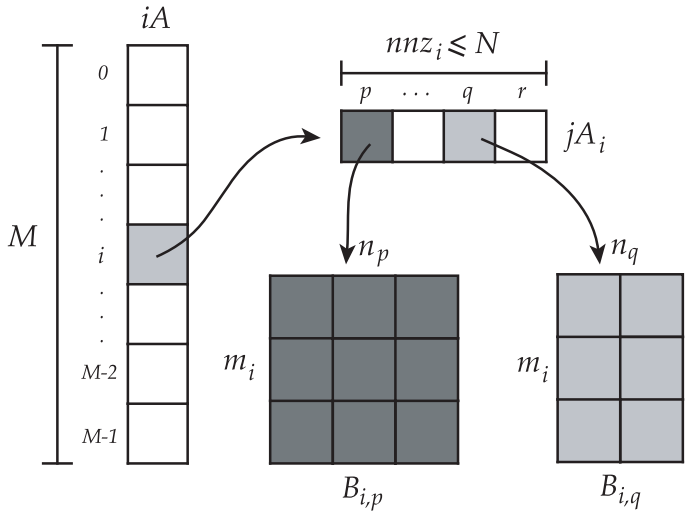


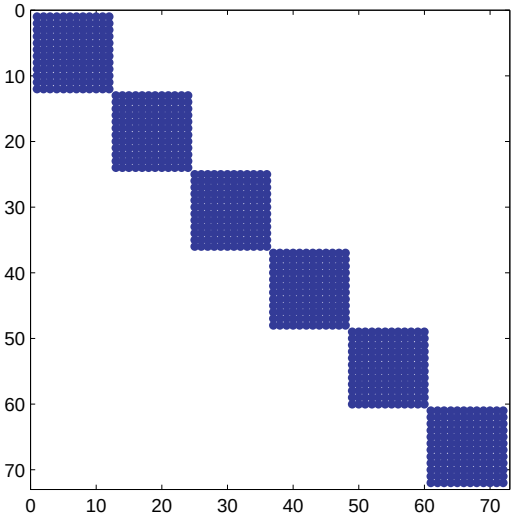
Figura 4–3: Estructura de datos para una matriz esparcida por bloques

Las figuras 4–4 y 4–5 muestran los patrones de esparcidad de los operadores D, B, S y $\mathcal{A}$ para una malla de seis celdas, usando grado $p = 1$ (izquierda) y $p = 2$ (derecha). A nivel de bloques, el patrón de esparcidad en las matrices de grado $p = 1$ es el mismo que las de grado $p = 2$. Sin embargo, los bloques para las matrices con grado $p = 2$ son más densos que las de grado $p = 1$, debido al incremento en la dimensión del espacio de polinomios. Similarmente, en la Figura 4–6 se presentan los patrones de esparcidad de $\mathcal{A}$ para $p = 1$ y $p = 2$, pero para una malla de 854 celdas. El patrón de esparcidad a nivel de bloques es el mismo en ambas matrices.

4.4 Bases de polinomios

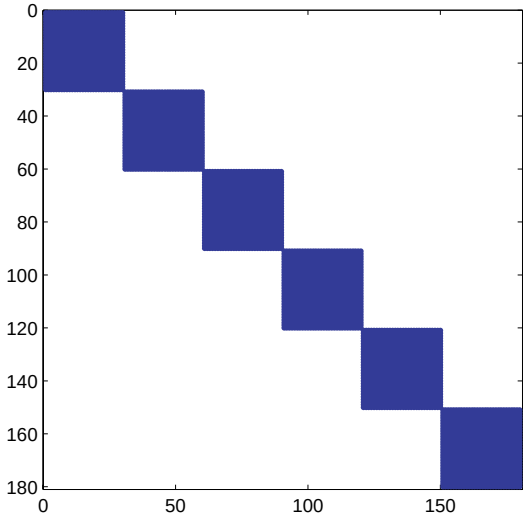
Las base de los polinomios para la celda de referencia, es representada mediante la interfaz **Basis3D** (Tabla 4–6), la cual contiene la información de la dimensión del espacio de polinomios, y permite evaluar los elementos de la base en un punto dado del elemento de referencia, y de la misma manera, permite evaluar en sus gradientes.

Matriz D para $p = 1$



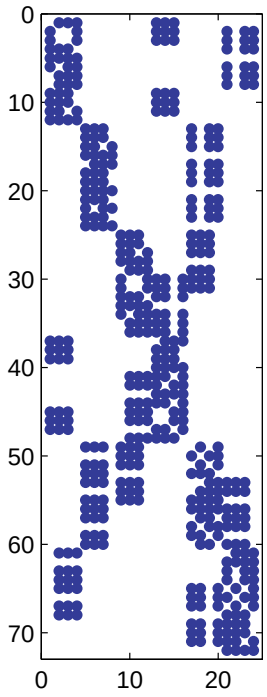
Dimensión de los bloques: 12×12

Matriz D para $p = 2$



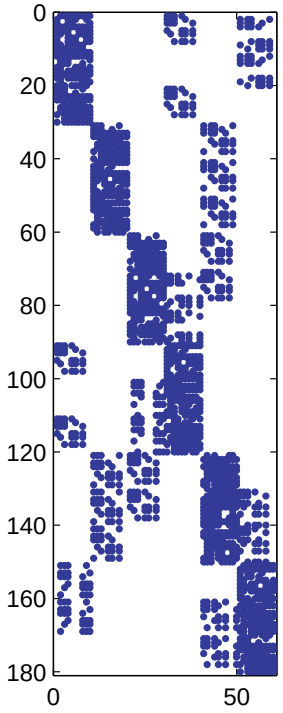
Dimensión de los bloques: 30×30

Matriz B para $p = 1$



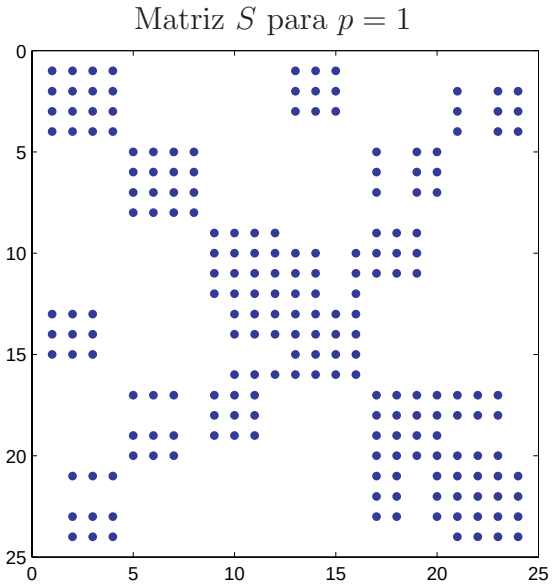
Dimensión de los bloques: 12×4

Matriz B para $p = 2$

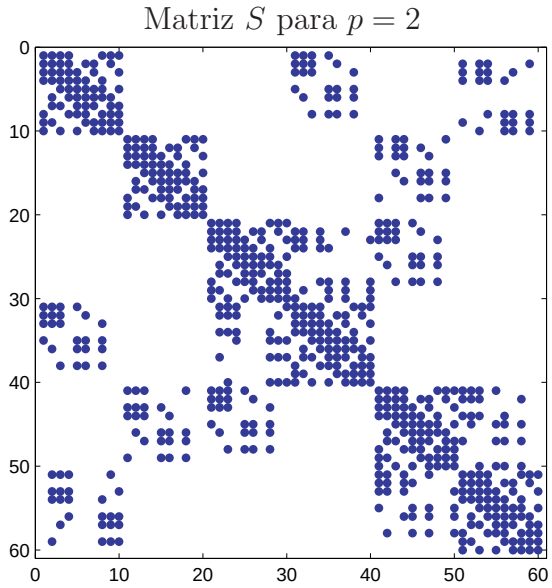


Dimensión de los bloques: 30×10

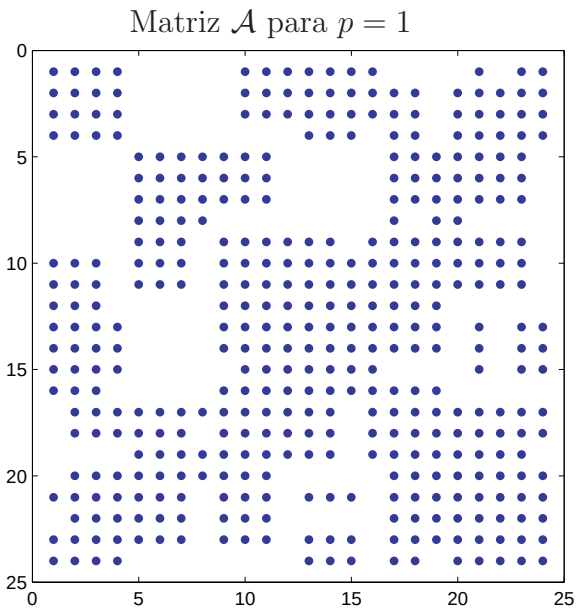
Figura 4–4: Patrón de esparcidad de los operadores D y B para grados $p = 1, 2$



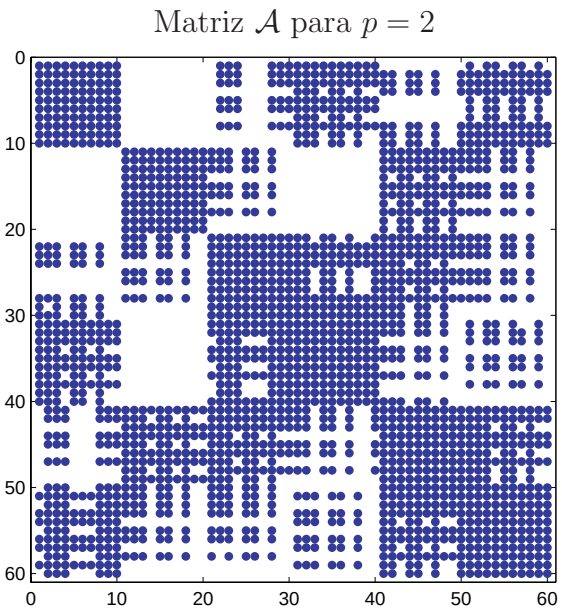
Dimensión de los bloques: 4×4



Dimensión de los bloques: 10×10



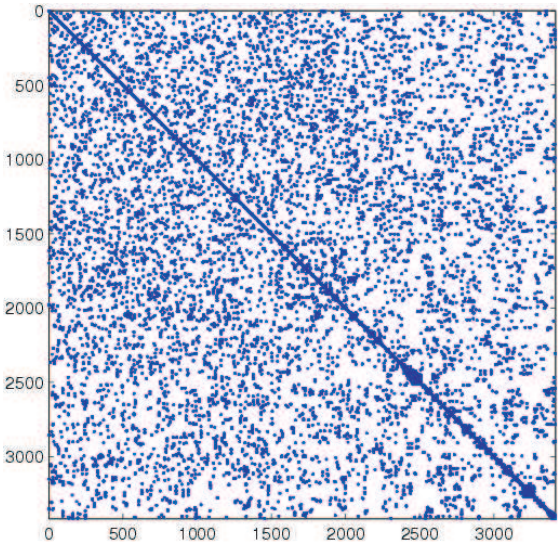
Dimensión de los bloques: 4×4



Dimensión de los bloques: 10×10

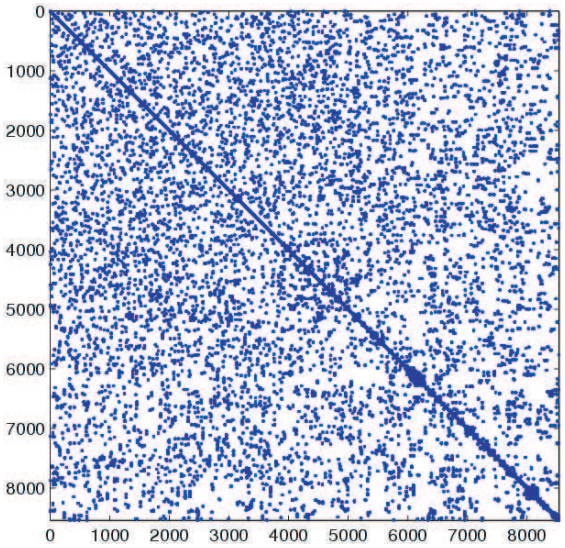
Figura 4–5: Patrón de esparcidad de los operadores S y $\mathcal{A}$ para grados $p = 1, 2$

Matriz $\mathcal{A}$ para $p = 1$



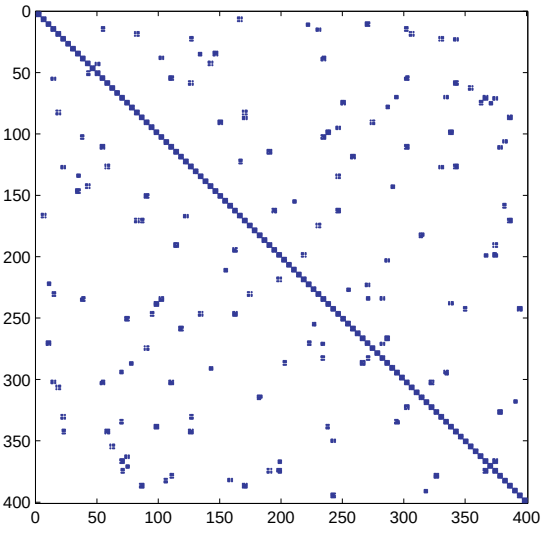
Dimensión de los bloques: 4×4

Matriz $\mathcal{A}$ para $p = 2$



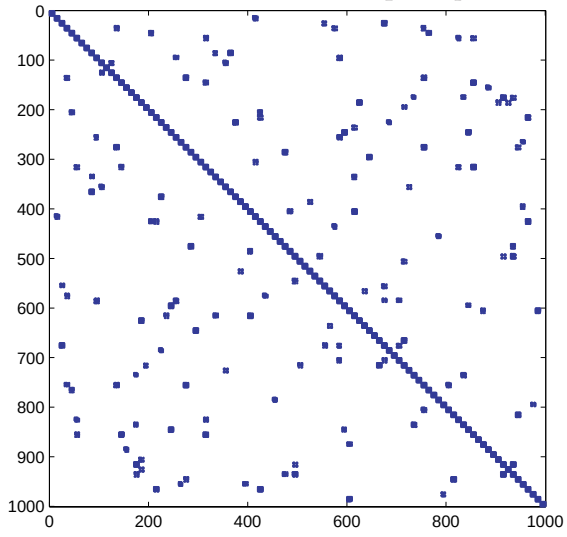
Dimensión de los bloques: 10×10

Zoom a la matriz $\mathcal{A}$ para $p = 1$



Dimensión de los bloques: 4×4

Zoom a la matriz $\mathcal{A}$ para $p = 2$



Dimensión de los bloques: 10×10

Figura 4–6: Patrón de esparcidad del operador $\mathcal{A}$ para grados $p = 1, 2$

Tabla 4–6: Estructura de datos de la base Basis3D

```
class Basis3D {
public:
  virtual ~Basis3D(){}
          uint getDegree() const
               { return( degree ); }
          uint getDimension() const
               { return( dimension ); }
  virtual bool isHierarchical() const = 0;
  virtual void eval(const Point3D&,
                    double[]) const = 0;
  virtual void evalGradient(const Point3D&,
                      Vector3D[]) const = 0;
  virtual void evalGradient(const Point3D&,
                        double[], double[],
                        double[]) const = 0;

protected:
  Basis3D(){}
  uint degree;
  uint dimension;
};
```

A partir de la interfaz **Basis3D** se pueden diseñar clases para las bases que se deseen, por ejemplo, para la base canónica o la base interpolatoria de Lagrange, ya sea para tetrahedros o hexahedros. El fragmento de código de la Tabla 4–7 corresponde al diseño de la clase **LBasisTET** que representa la base de Lagrange para el tetrahedro de referencia.

4.5 Cuadraturas

En general, los operadores discretos de un método DG involucran integrales de superficie y de volumen, por lo que es necesario cuadraturas en 2D y 3D. Estas son representadas por las clases **Quad2D** y **Quad3D** respectivamente. Ambas clases, contienen listas para los pesos y los puntos de cuadratura. En la Tabla 4–8 se muestra el diseño de la clase **Quad3D**, el cual es similar al de la clase **Quad2D**,

Tabla 4–7: Estructura de datos de la base de Lagrange

LBasisTET

```
class LBasisTET : public Basis3D {
public:
  LBasisTET(unit, const Point3D* = 0);
 ~LBasisTET();
  bool isHierarchical() const;
  void eval(const Point3D&, double[]) const;
  void evalGradient(const Point3D&,
                    Vector3D[]) const;
  void evalGradient(const Point3D&, double[],
                    double[], double[]) const;
  void getInterpolationNodes(Point3D[]) const;

private:
   ...
};
```

exceptuando el método *getPoint()* que retorna un objeto de tipo **Point2D**. Los métodos *read()* y *write()* permiten leer y guardar en un archivo el listado de puntos y pesos de la cuadratura, respectivamente. De esta manera el código es más flexible, pues permite el uso de cualquier listado de puntos y pesos que se deseen sin alterar de ninguna manera el código.

Evaluar un punto $(\hat{x}, \hat{y})$ de la cuadratura 2D en los elementos de la base, requiere que se mapee el punto de cuadratura $(\hat{x}, \hat{y})$ del triángulo, a un punto (x, y, z) en la cara de la celda de referencia $\hat{T}$ (Figura 4–7). Para ello, se calculan las coordenadas baricéntricas del punto $(\hat{x}, \hat{y})$, denotadas $(\lambda_0, \lambda_1, \lambda_2)$, que cumplen que

$$(\hat{x}, \hat{y}) \;=\; \lambda_0 \hat{V}_0 + \lambda_1 \hat{V}_1 + \lambda_2 \hat{V}_2,$$

para luego encontrar el punto (x, y, z) en la cara de la celda de la forma:

$$(x, y, z) \;=\; \lambda_0 V_{\sigma_0} + \lambda_1 V_{\sigma_1} + \lambda_2 V_{\sigma_2},$$

Tabla 4–8: Estructura de datos de la cuadratura Quad3D

```
class Quad3D {
public:
  Quad3D();
 ~Quad3D();
  uint            getNumPoints() const;
  const Point3D&  getPoint(uint) const;
  double          getWeight(uint) const;
  void            read(const string&);
  void            write(const string&) const;

private:
   ...
};
```

donde σ_0, σ_1 y σ_2 son los índices locales de los nodos en la cara, como se muestra en la Tabla 4–3, tal que $\hat{V}_i$ es mapeado a V_{σ_i}, para $i = 0, 1, 2$.

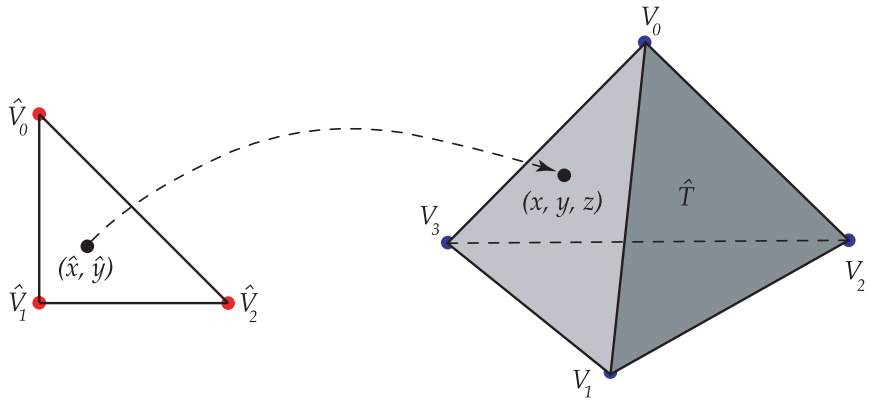


Figura 4–7: Mapear un punto en el triángulo a un punto en el tetrahedro

4.6 Operadores en la cara

Para la implementación de los operadores numéricos se consideraron dos estructuras. La clase **Operator2D** almacena exclusivamente los operadores que requieren integrales de superficie, por ejemplo $[IF]$, $[EF]$ y los operadores para las condiciones de frontera. La clase **Operator3D** contiene los operadores que se construyen a partir de integrales de volumen, por ejemplo la matriz de masa local y su inversa, y los operadores diferenciales DX, DY y DZ. El método *setData()* inicializa las tablas

internas en ambas clases y requiere de un apuntador para la interfaz de bases de polinomios **Basis3D** y de un apuntador para la interfaz de cuadraturas en 2D **Quad2D** o **Quad3D** en 3D. De esta forma el código no depende ni de las bases de polinomios ni de las cuadraturas.

En los fragmentos de código de las Tablas 4–9 y 4–10 se ilustra ambas estructuras de datos. Para obtener el operador $[IF]$ de una cara se utiliza el método *Operador2D::setIF()* el cual requiere como parámetro de entrada el índice local de la cara. En cambio para obtener el operador $[EF]$ se utiliza el método *Operador2D::setEF()*, el cual tiene tres párametros de entrada: los índices locales de la cara y el índice de la rotación.

Tabla 4–9: Estructura de datos del operador 2D

Operator2D

```
class Operator2D {
public:
  Operator2D();
 ~Operator2D();
  void   setData(const Basis3D*, const Quad2D*);
  Block* getIF(uint);
  Block* getEF(uint, uint, uint);
  void   getProjection(const Element3D*, uint, uint,
                       Function3D,  double*);

private:
   ...
};
```

En las Figuras 4–8 y 4–9 se muestran los diagramas de las estructuras de datos implementadas para almacenar los operadores $[IF]$ y $[EF]$ respectivamente.

Tabla 4–10: Estructura de datos del operador 3D

Operator3D

```
class Operator3D {
public:
  Operator3D();
 ~Operator3D();
  void   setData(const Basis3D*, const Quad3D*);
  Block* getMassMatrixInv();
  Block* getDX();
  Block* getDY();
  Block* getDZ();
  void   getProjection(const Element3D*, Function3D,
                       double*);

private:
   ...
};
```

La estructura para almacenar los operadores interiores consiste de un apuntador a los cuatro operadores $[IF]$ del tetrahedro de referencia, uno por cada cara. Mientras que la estructura de almacenamiento para los operadores exteriores es más complicada, debido a que se tienen 48 operadores $[EF]$ distintos. Para obtener un operador exterior es necesario conocer la rotación de la cara, su índice local en el primer tetrahedro (primer lado) y su índice local en el segundo tetrahedro (segundo lado).

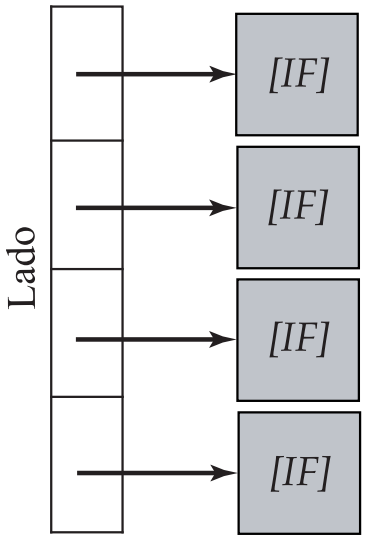


Figura 4–8: Estructura de datos para almacenar los operadores interiores a la cara

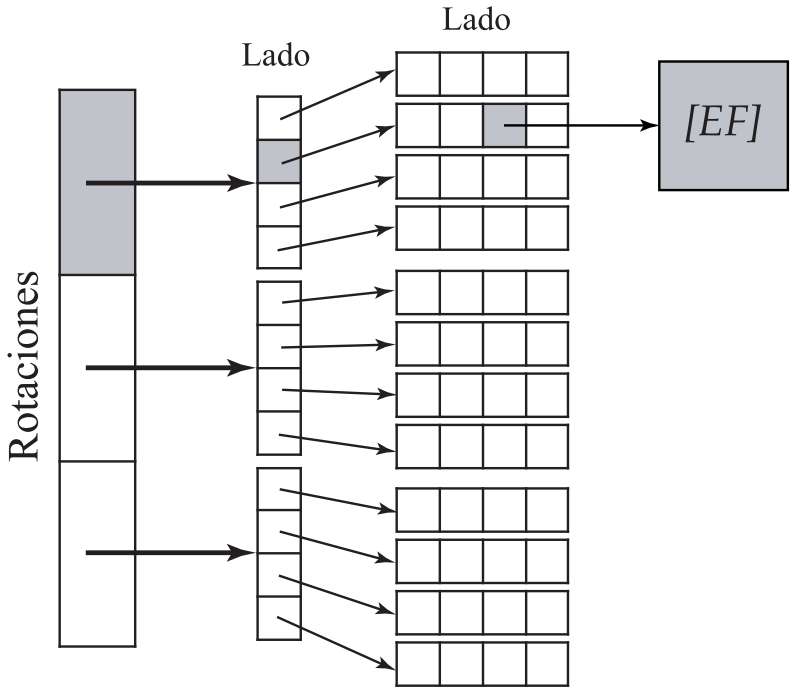


Figura 4–9: Estructura de datos para almacenar los operadores exteriores a la cara

CAPÍTULO 5
VALIDACIÓN DE CÓDIGO

En este capítulo se presentan algunos experimentos numéricos para validar la implementación descrita en los capítulos 3 y 4, en especial, las tasas de convergencia para varios grados de aproximación. Además, se ilustra el cambio en el esténcil del complemento de Shur, al variar el parámetro $\boldsymbol{\beta}_s$ del flujo numérico. En todos los ejemplos se utilizó el software TETGEN, desarrollado por H. Si [19], para la generación de las mallas no estructuradas en 3D. En la Figura 5–1 se muestran algunas de las mallas generadas por TETGEN utilizadas en los experimentos numéricos. Estas mallas fueron obtenidas mallando el cubo unitario, y luego por medio de refinamiento, se obtienen mallas más finas.

La base de polinomios utilizada corresponde a la interpolatoria de Lagrange, [20]. En el caso de la selección de los parámetros del flujo numérico, η_s y $\boldsymbol{\beta}_s$, se tomó $\eta_s = 1$ y $\boldsymbol{\beta}_s = (0.5, 0.5)$. Los sistemas lineales obtenidos en cada experimento, se resolvieron mediante el método de Gradiente Conjugado (CG por sus siglas en íngles) [21]. En cuanto a los criterios de parada de CG, se consideró un máximo de 5000 iteraciones y una tolerancia residual relativa de 10^{-13}. De esta manera, se resuelven los sistemas con una buena presición.

Todos los cálculos presentados en este capítulo y en el capítulo 6, se realizaron en una computadora portátil con procesador AMD Athlon-X2 de 64 bits, 2GB de memoria RAM y 120GB de disco duro. Los tamaños de las mallas, y los grados de aproximación usados fueron seleccionados teniendo en cuenta las limitaciones de esta computadora.

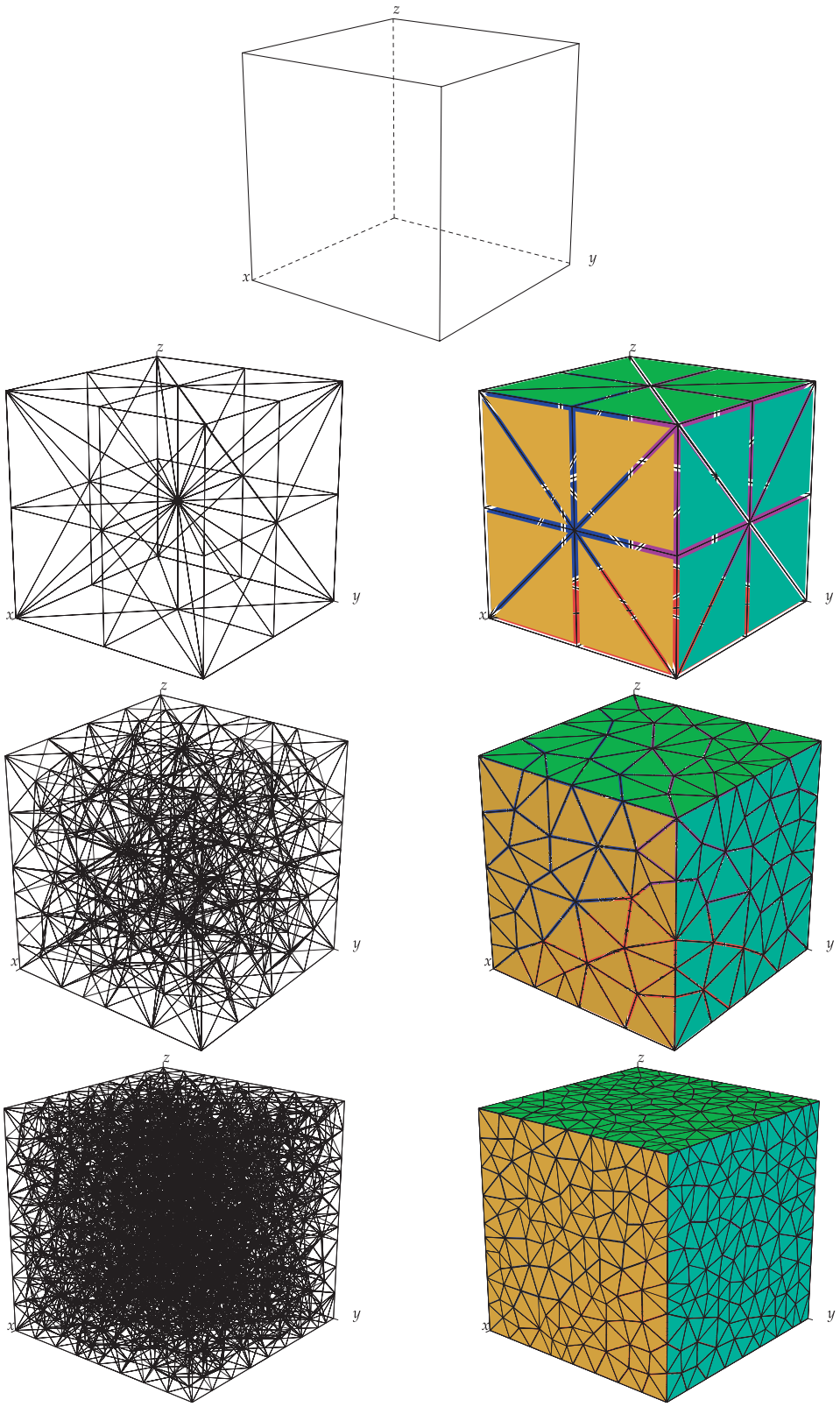


Figura 5–1: Ejemplo de malla (izquierda) y frontera (derecha)

5.1 Convergencia del método LDG en 3D

En [4] se muestra el siguiente estimado de error *a priori* para método LDG,

Teorema 1. *Sea $u \in H^{s+2}(\Omega)$ la solución y p el grado de aproximación, entonces*

$$\begin{aligned} \| u - u_h \|_{0,\Omega} &= \mathcal{O}\left(h^{mín\{s+1,p\}+1}\right) \\ \| (\boldsymbol{q} - \boldsymbol{q}_h, u - u_h) \|_{\mathcal{A}} &= \mathcal{O}\left(h^{mín\{s+1,p\}}\right) \end{aligned}$$

□

donde $s+2$ es la regularidad de la solución u. Si $s+1 > p$ entonces las tasas de convergencia para las variables u y q son $\mathcal{O}\left(h^{p+1}\right)$ y $\mathcal{O}\left(h^{p}\right)$ respectivamente. A continuación se presentan diferentes problemas que muestran numéricamente que el Teorema 1 se cumple.

5.1.1 Soluciones polinomiales

Se considera el problema modelo (2.1) en el dominio $\Omega = (0,1) \times (0,1) \times (0,1)$ con condiciones de frontera de tipo Dirichlet y difusión isotrópica $\mathcal{D} = I_d$. La función f se define de tal forma que la solución exacta sea la función

$$u(x,y,z) = x^2 + y^2 + z^2.$$

En las Tablas 5–1 y 5–2 se muestra la norma L_2 del error para el potencial, $u - u_h$, y para el gradiente, $\nabla u - \nabla u_h$, respectivamente, al utilizar aproximaciones de grado $p = 1, 2, 3$. Se puede apreciar que según se aumenta el tamaño de la malla el error va decreciendo. Además, como la solución exacta es un polinomio de grado $p = 2$ cuando se utilizan aproximaciones $p \geq 2$, la solución discreta es exactamente la solución exacta, por lo que el error es cero. También se presenta el orden de convergencia entre dos mallas consecutivas. Para el error en el potencial se obtiene un comportamiento asintótico de $\mathcal{O}\left(h^{p+1}\right)$, mientras que para el error en el gradiente, el comportamiento es $\mathcal{O}\left(h^{p}\right)$ tal como lo predicen los estimados de error *a priori* propuesto en el Teorema 1.

Tabla 5–1: Error y orden para el potencial

$p=1$		$p=2$		$p=3$	
Error	Orden	Error	Orden	Error	Orden
1.21e-01	-	4.37e-14	-	1.12e-13	-
2.78e-02	2.1	5.51e-14	-	1.86e-13	-
5.26e-03	2.1	3.78e-13	-	5.69e-13	-
1.40e-03	2.2	9.02e-13	-	1.20e-12	-
8.77e-04	2.0	1.40e-12	-	2.31e-12	-

Tabla 5–2: Error y orden para el gradiente

$p=1$		$p=2$		$p=3$	
Error	Orden	Error	Orden	Error	Orden
8.97e-01	-	5.04e-13	-	2.19e-12	-
3.93e-01	1.2	1.01e-12	-	6.07e-12	-
1.58e-01	1.1	5.52e-12	-	8.51e-12	-
7.83e-02	1.1	1.44e-11	-	1.96e-11	-
6.22e-02	1.0	1.79e-11	-	3.03e-11	-

Como un segundo ejemplo se considera el problema

$$-\nabla \cdot \mathcal{D}\nabla u + \alpha u \;=\; f,$$

en $\Omega = (0,1)\times(0,1)\times(0,1)$ con condiciones de frontera de tipo Dirichlet, tomando $\alpha = 6$ y difusión anisotrópica

$$\mathcal{D} \;=\; \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}.$$

La función f se define de tal forma que la solución exacta sea la función

$$u(x,y,z) \;=\; x^3 + y^3 + z^3.$$

Análogamente al ejemplo anterior, en las Tablas 5–3 y 5–4 se muestra la norma L_2 de los errores $u-u_h$ y $\nabla u-\nabla u_h$ respectivamente. Según se incrementa el tamaño de la malla, el error decrece con el orden predicho en el Teorema 1. Además, como la solución exacta es un polinomio de grado $p=3$ cuando se utilizan aproximaciones $p \geq 3$, el error es cero.

Tabla 5–3: Error y orden para el potencial

$p=1$		$p=2$		$p=3$	
Error	Orden	Error	Orden	Error	Orden
1.85e-01	-	2.76e-02	-	1.02e-14	-
4.71e-02	2.0	2.90e-03	3.3	7.21e-14	-
8.58e-03	2.1	2.30e-04	3.1	4.71e-13	-
2.35e-03	2.1	3.04e-05	3.3	1.88e-12	-
1.47e-03	2.0	1.53e-05	2.9	1.85e-12	-

Tabla 5–4: Error y orden para el gradiente

$p=1$		$p=2$		$p=3$	
Error	Orden	Error	Orden	Error	Orden
1.45e+00	-	3.33e-01	-	2.08e-13	-
6.98e-01	1.1	8.00e-02	2.1	2.68e-12	-
2.80e-01	1.1	1.46e-02	2.1	9.31e-12	-
1.41e-01	1.1	3.81e-03	2.2	2.88e-11	-
1.11e-01	1.0	2.43e-03	1.9	2.67e-11	-

5.1.2 Condiciones de Dirichlet no cero

En este ejemplo se considera el problema modelo (2.1) en el dominio $\Omega = (0,1)\times(0,1)\times(0,1)$ con condiciones de frontera de tipo Dirichlet no cero y difusión isotrópica $\mathcal{D} = I_d$. La función f se define de tal forma que la solución exacta sea la función

$$u(x,y,z) \;=\; e^{x+y+z}.$$

En las Figuras 5–2 y 5–3 se muestra la norma L_2 de los errores $u - u_h$ y $\nabla u - \nabla u_h$ respectivamente, al utilizar aproximaciones de grado $p = 1, 2, 3$. Mediante regresión lineal se obtienen las pendientes 2.23, 3.40, 4.64 para $\| u - u_h \|$; y, 1.20, 2.31, 3.59 para $\| \nabla u - \nabla u_h \|$ con $p = 1, 2, 3$, respectivamente, lo cual muestra un comportamiento asintótico de orden $\mathcal{O}(h^{p+1})$ para el error en el potencial y de orden $\mathcal{O}(h^p)$ para el gradiente tal como lo predicen los estimados de error *a priori* propuesto en el Teorema 1.

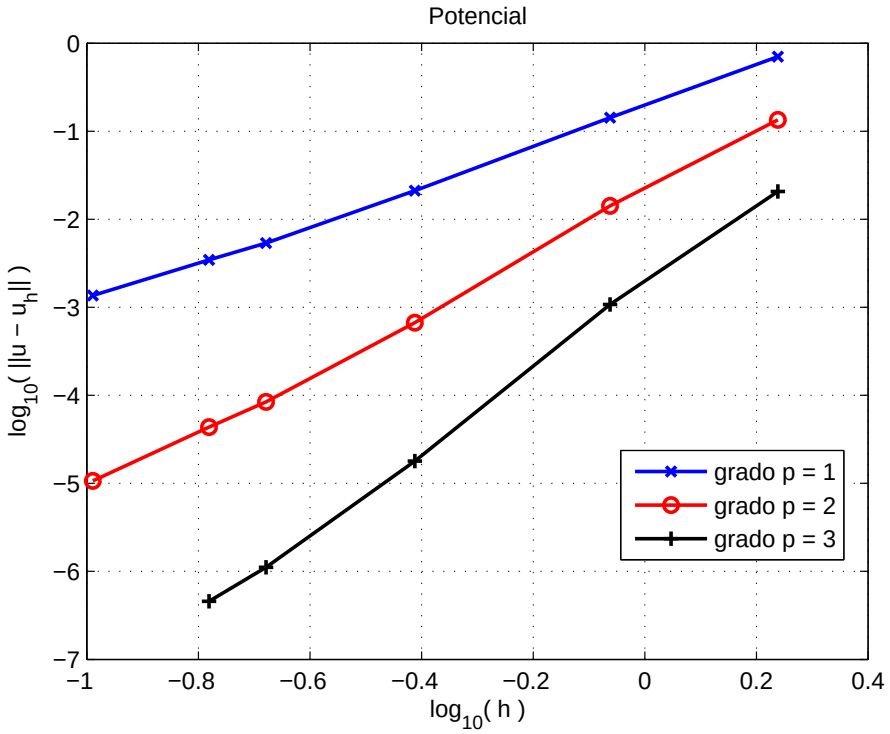


Figura 5–2: Condiciones de Dirichlet no cero: Error del potencial

5.1.3 Condiciones de Robin

En este ejemplo se considera nuevamente el problema modelo con difusión isotrópica, $\mathcal{D} = I_d$ con condiciones de tipo Robin en la cara que se encuentra dentro del plano $x = 1$ del cubo Ω, mientras que para las caras restantes se tiene Dirichlet no cero. La función f se define de tal forma que la solución exacta sea la función

$$u(x, y, z) \;=\; e^{x+y+z}.$$

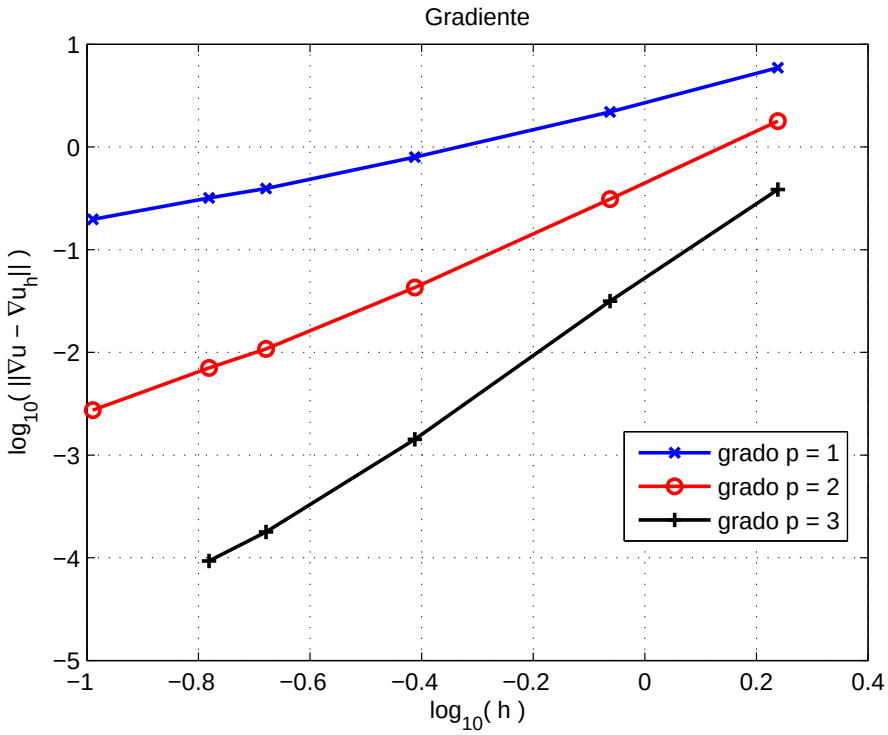


Figura 5–3: Condiciones de Dirichlet no cero: Error del gradiente

En las Figuras 5–4 y 5–5 se aprecia que la norma L_2 de los errores $u - u_h$ y $\nabla u - \nabla u_h$ se comporta de manera asintótica como dice el Teorema 1. Se tienen las pendientes 2.19, 3.38, 4.64 para $\| u - u_h \|$; y, 1.14, 2.26, 3.52 para $\| \nabla u - \nabla u_h \|$ con $p = 1, 2, 3$, respectivamente.

5.1.4 Problema anisotrópico

En este ejemplo se considera nuevamente el problema modelo con condiciones de frontera de tipo Dirichlet y difusión anisotrópica. La matriz de difusión es

$$\mathcal{D} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}.$$

La función f se escoge de tal forma que la solución exacta sea la función

$$u(x, y, z) = \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

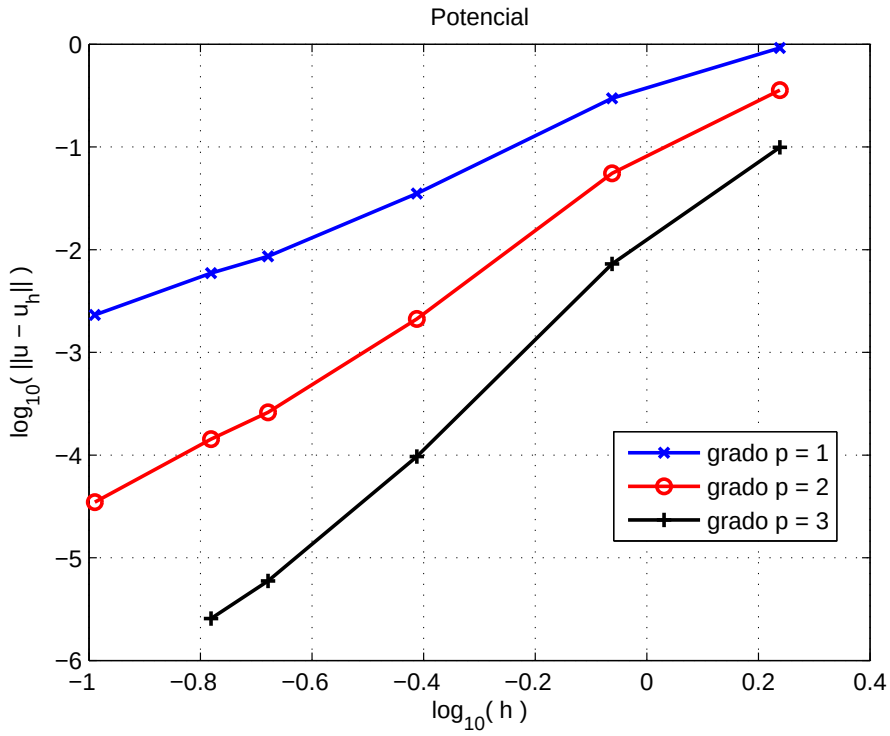


Figura 5–4: Condiciones de Robin: Error del potencial

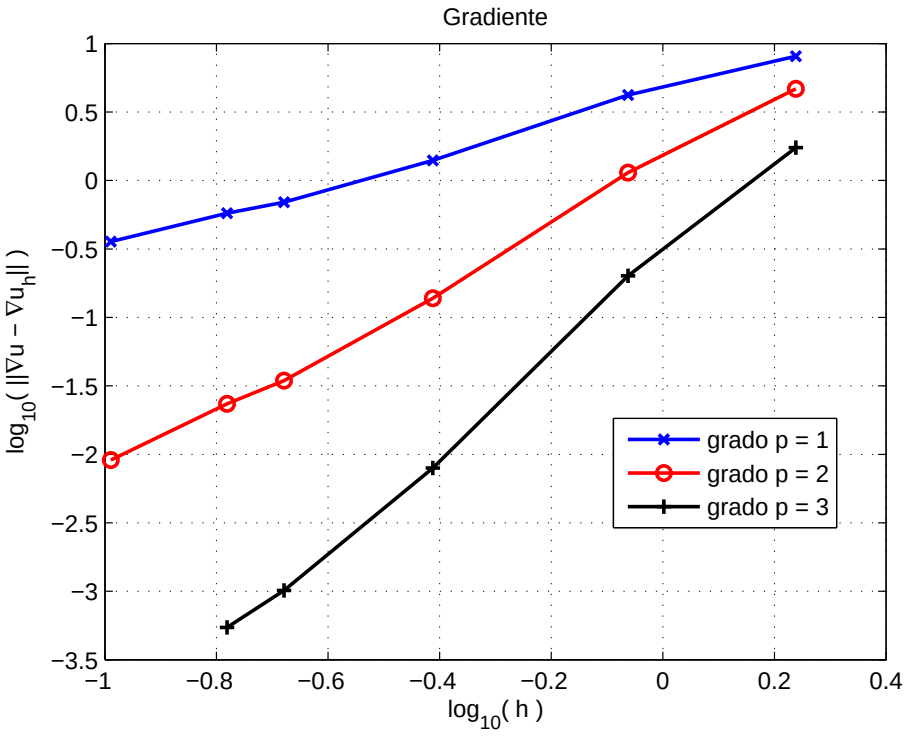


Figura 5–5: Condiciones de Robin: Error del gradiente

En las Figuras 5–6 y 5–7 se muestra la norma L_2 de los errores $u - u_h$ y $\nabla u - \nabla u_h$ respectivamente, al utilizar aproximaciones de grado $p = 1, 2, 3$. Se tienen las pendientes 1.92, 3.00, 4.32 para $\| u - u_h \|$; y, 0.90, 1.92, 3.20 para $\| \nabla u - \nabla u_h \|$ con $p = 1, 2, 3$, respectivamente, lo que muestra un comportamiento asintótico de orden $\mathcal{O}\left(h^{p+1}\right)$ para el error en el potencial y de orden $\mathcal{O}\left(h^{p}\right)$ para el gradiente tal como lo predicen los estimados de error del Teorema 1.

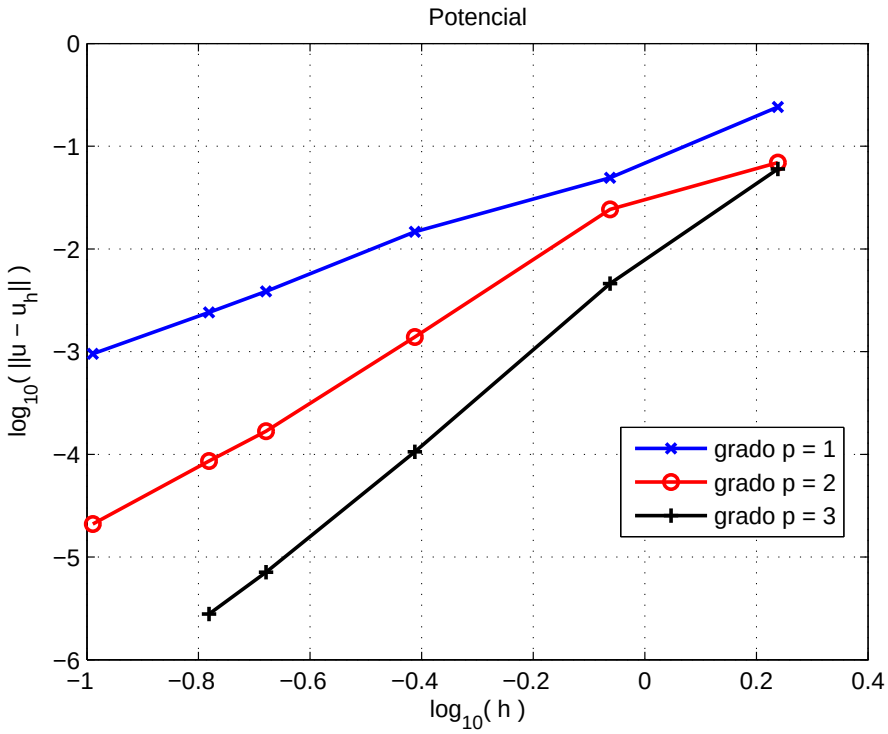


Figura 5–6: Problema anisotrópico: Error del potencial

5.1.5 Problema en un dominio con forma L

El siguiente ejemplo consiste de una solución de baja regularidad en un dominio con forma de L, [22]. Se considera este problema en la región $[-1, 1] \times [-1, 1] - [0, 1] \times [-1, 0]$, y se extiende el dominio a 3D, tomando $0 \leq z \leq 1$, como se muestra en la Figura 5–8. Se imponen condiciones de tipo Neumann cero en las caras de los planos $z = 0$ y $z = 1$.

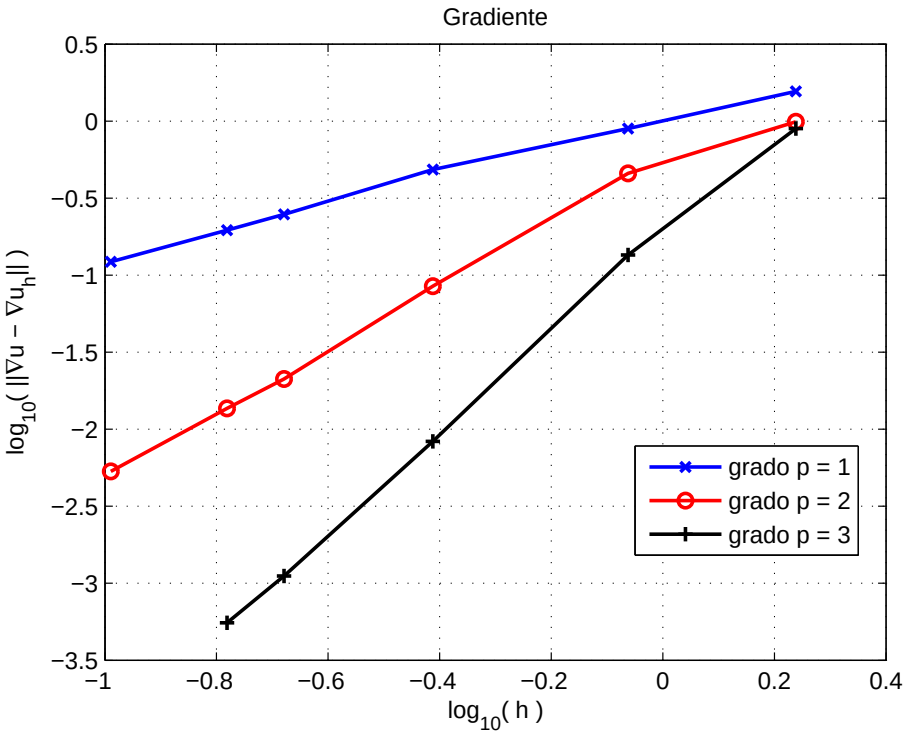


Figura 5–7: Problema anisotrópico: Error del gradiente

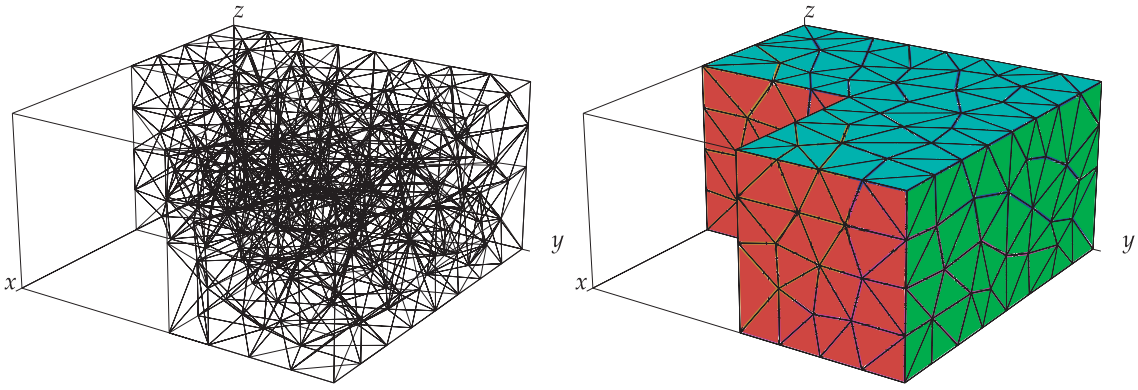


Figura 5–8: Ejemplo de malla (izquierda) y frontera (derecha)

La solución exacta para este problema es dada por la función

$$u(r,\theta) \;=\; r^{\frac{2}{3}} \sin\left(\frac{2}{3}\theta\right), \text{ en 2D}$$

para la cual el orden de convergencia de la norma L_2 de los errores $u-u_h$ y $\nabla u-\nabla u_h$ es $\frac{4}{3}-\varepsilon$ y $\frac{2}{3}-\varepsilon$ respectivamente, para todo $\varepsilon > 0$, [4]. En las Figuras 5–9 y 5–10

se muestran los errores para el potencial y el gradiente respectivamente, donde se puede apreciar que el error disminuye según se incrementa el tamaño de la malla. En las Tablas 5–5 y 5–6 se muestran las tasas de convergencia para el potencial y el gradiente, respectivamente. Se tiene que

$$\tau_i \;=\; \frac{\log_{10} e_{i+1} - \log_{10} e_i}{\log_{10} h_{i+1} - \log_{10} h_i}$$

donde e_i representa la norma L_2 del error $u - u_h$ en la malla i, y h_i es el tamaño de la malla. Se puede apreciar que las tasas de convergencia cumplen con lo propuesto en el Teorema 1.

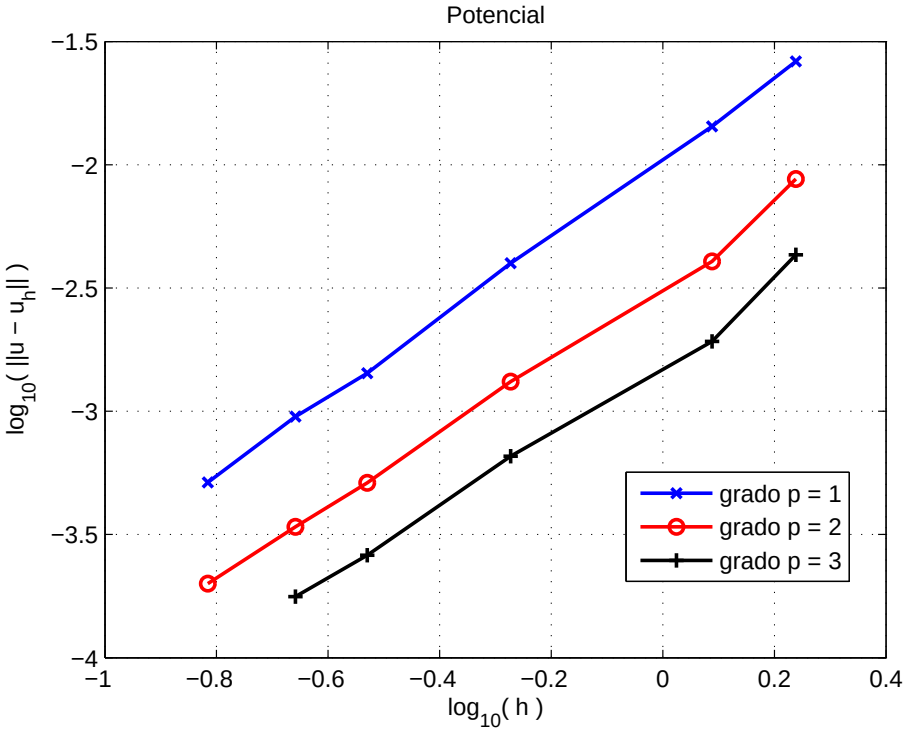


Figura 5–9: Problema en el dominio L: Error del potencial

Tabla 5–5: Tasas de convergencia para el potencial

p	τ_1	τ_2	τ_3	τ_4	τ_5
1	1.7533	1.5366	1.7377	1.3730	1.6961
2	2.2259	1.3505	1.5993	1.3893	1.4633
3	2.3377	1.2916	1.5609	1.3090	-

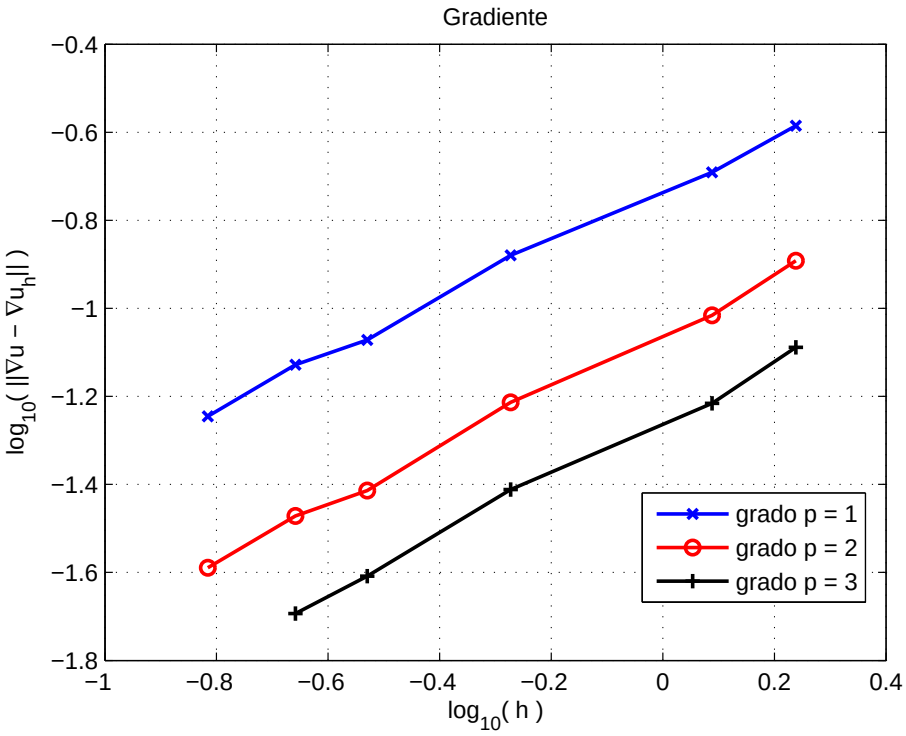


Figura 5–10: Problema en el dominio L: Error del gradiente

Tabla 5–6: Tasas de convergencia

p	τ_1	τ_2	τ_3	τ_4	τ_5
1	0.7048	0.5214	0.7484	0.4371	0.7474
2	0.8236	0.5472	0.7801	0.4509	0.7491
3	0.8469	0.5429	0.7647	0.6606	-

5.2 Esténcil de $\mathcal{A}$: mallas no estructuradas

Se puede reducir el esténcil de la matriz de rigidez $\mathcal{A}$, proveniente de la discretización del método LDG, de acuerdo a la selección del parámetro $\boldsymbol{\beta}_s$. El peor de los escenario es cuando el esténcil es *extendido*, resultado de tener el flujo en cada cara por ambas direcciones. Mientras que el escenario ideal, el esténcil *compacto*, representa la menor cantidad de entradas no cero en la matriz $\mathcal{A}$, reduciendo así el espacio de memoria necesario para almacenar dicha matriz. En la Tabla 5–7 se muestran los diferentes esténciles para mallas compuestas por triángulos y cuadriláteros, en el caso de 2D y para mallas compuestas por tetrahedros y hexahedros en 3D. El

esténcil compacto no necesariamente se alcanza para mallas no estructuradas. En [12] se presentan algunas heurísticas para reducir el esténcil, basados en el grafo que representa la malla.

Tabla 5–7: Esténcil extendido y esténcil compacto

	2D		3D	
	Triángulos	Cuadriláteros	Tetrahedros	Hexahedros
Extendido	10	13	17	25
Compacto	4	5	5	7

El parámetro $\boldsymbol{\beta}_s$ afecta las constantes γ_i y ζ_i de la ecuación 2.7. En la Tabla 5–8 se muestran los valores de (γ_i, ζ_i) obtenidos de algunos valores dados a $\boldsymbol{\beta}_s$.

Tabla 5–8: Selección de parámetros

Etiqueta	γ_s	ζ_s
β_0	1.0	0.0
β_1	0.0	1.0
β_2	0.5	0.5
β_{rnd}	elegir al azar en $[0, 1]$	$1.0 - \gamma_s$

El esténcil extendido es obtenido para el caso de β_2. En la Tabla 5–9 se puede observar cómo se reduce el número de bloques en el complemento de Schur, en cada uno de los casos descritos en la Tabla 5–8 para $p = 1$. Para $p > 1$ el patrón de esparcidad de $\mathcal{A}$ no cambia, por lo que cantidad de bloques es la misma que para $p = 1$. Por otro lado, en la Tabla 5–10 se puede apreciar que la convergencia del método LDG no es afectada por la selección del parámetro $\boldsymbol{\beta}_s$.

Tabla 5–9: Bloques no cero en el complemento de Schur

β_0	β_1	β_2	β_{rnd}
24	22	32	22
264	264	432	254
6328	6188	10910	5824
46245	45785	82783	42899
93653	93187	168157	86805
379278	377656	683928	351200

Tabla 5–10: Comportamiento para $p = 1$

β_0		β_1		β_2		β_{rnd}	
Error	Orden	Error	Orden	Error	Orden	Error	Orden
2.46e-01	-	2.54e-01	-	2.44e-01	-	2.48e-01	-
6.73e-02	1.9	6.85e-02	1.9	4.51e-02	2.4	5.43e-02	2.2
1.74e-02	1.7	1.89e-02	1.6	1.49e-02	1.4	1.72e-02	1.4
4.76e-03	2.1	5.03e-03	2.2	3.92e-03	2.2	4.56e-03	2.2
2.99e-03	2.0	3.16e-03	2.0	2.45e-03	2.0	2.84e-03	2.0
1.19e-03	1.9	1.26e-03	1.9	9.60e-04	1.9	1.13e-03	1.9

En la Figura 5–11 se presenta el número de filas que tienen una determinada cantidad de bloques no ceros de la matriz de rigidez $\mathcal{A}$, al usar los diferentes valores de $\boldsymbol{\beta}_s$ propuestos en la Tabla 5–8, para una malla de 854 celdas y grado de la aproximación $p = 1$. En la Figura 5–12 se muestra la distribución de las entradas no cero de cada una de estas matrices en cada valor de $\boldsymbol{\beta}_s$. Se puede observar cómo la matriz de β_2 es más densa que las demás.

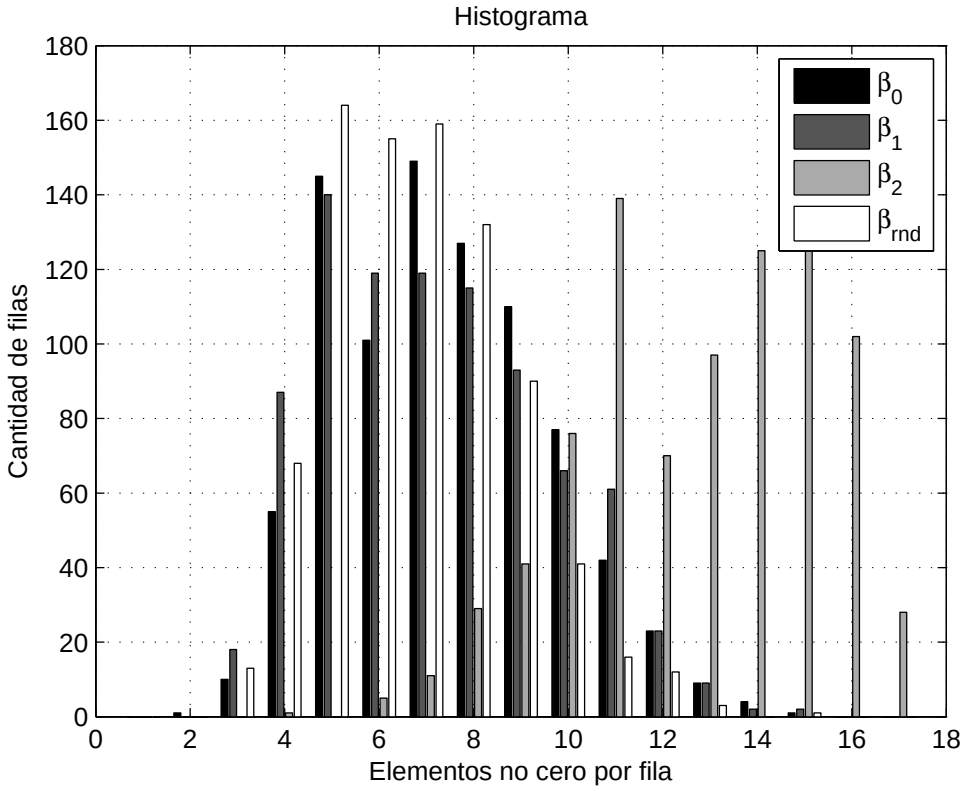
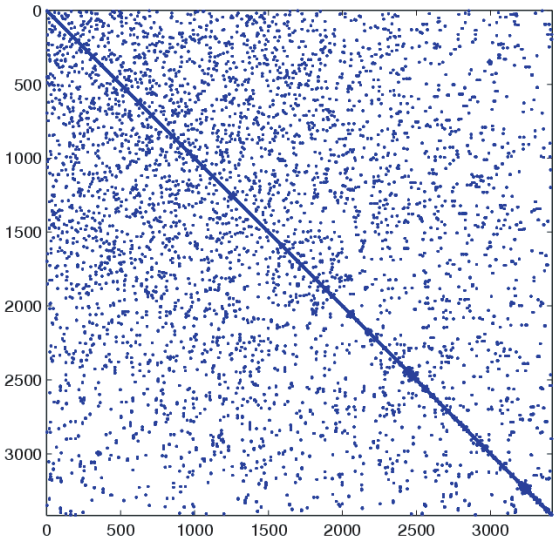


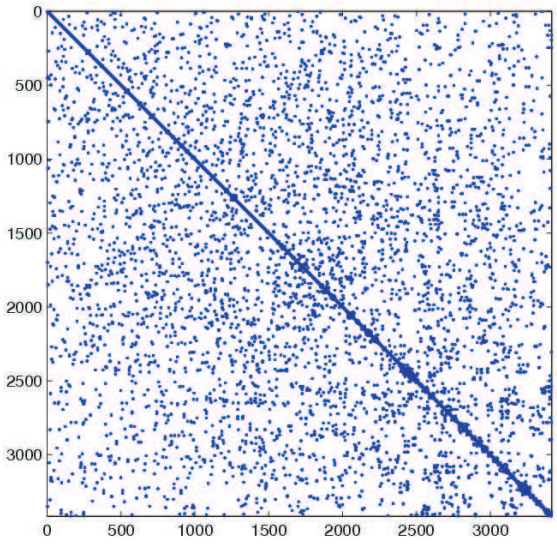
Figura 5–11: Distribución de los bloques no ceros por fila

Matriz $\mathcal{A}$ para β_0



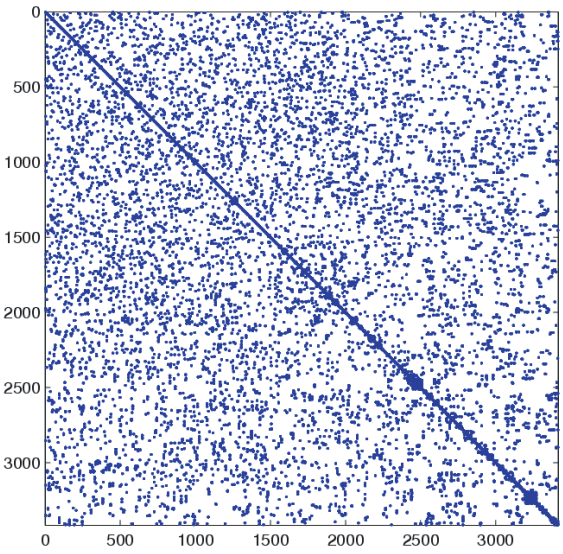
Bloques no cero: 6328

Matriz $\mathcal{A}$ para β_1



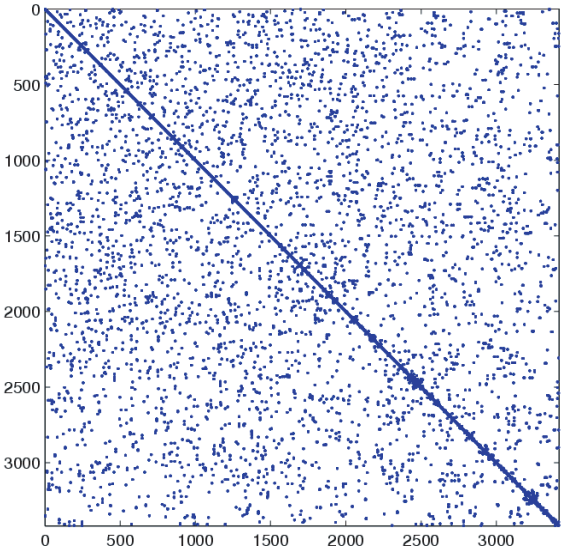
Bloques no cero: 6188

Matriz $\mathcal{A}$ para β_2



Bloques no cero: 10910

Matriz $\mathcal{A}$ para β_{rnd}



Bloques no cero: 5824

Figura 5–12: Patrón de esparcidad del operador $\mathcal{A}$ para β_0, β_1, β_2 y β_{rnd}

CAPÍTULO 6
PRECONDICIONADORES MULTINIVELES

En la práctica el sistema lineal 3.4 posee una gran cantidad de incógnitas, por lo que resolverlo mediante eliminación Gaussiana resulta bastante lento, debido a que eliminación Gaussiana tiene una complejidad de $\mathcal{O}(n^3)$, [17]. En [4] se demostró que $\mathcal{A}$ es una matriz esparcida simétrica definida positiva, por lo tanto, el método del Gradiente Conjugado (CG por sus siglas en ingles) introducido en [21] es el más apropiado para aproximar la solución del sistema. En el Teorema 2 se presenta el estimado de error para el método CG.

Teorema 2. *Sea x_m la aproximación obtendida de la m-ésima iteración de Gradiente Conjugado aplicado al sistema $Ax = b$, y x^* la solución exacta, entonces*

$$\|x^* - x_m\|_A \;\leq\; 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \|x^* - x_0\|_A,$$

donde κ es el condicionamiento espectral de A, es decir, $\kappa = \lambda_{máx}/\lambda_{mín}$, con $\lambda_{mín}$ y $\lambda_{máx}$ el menor y mayor valor propio de A, respectivamente. □

Se puede apreciar que cuando $\kappa \gg 1$, el término $(\sqrt{\kappa}-1)/(\sqrt{\kappa}+1)$ se comporta como $1 - 2/(\sqrt{\kappa}+1)$, el cual converge hacia 1, por lo que el Teorema 2 predice una convergencia lenta del CG. En [2] se demostró que el condicionamiento espectral para la matriz $\mathcal{A}$ se comporta de la siguiente manera

Teorema 3. *Considere*

$$C^1_{\beta,\eta}\,\|u_h\|^2_{0,\Omega} \;\leq\; \mathcal{A}_h(u_h,u_h) \;\leq\; \frac{C^2_{\beta,\eta}}{h^2}\,\|u_h\|^2_{0,\Omega}\,,$$

donde $C^1_{\beta,\eta} = \left(C_\beta máx\{1, \frac{1}{\eta}\}\right)^{-1}$, *y* $C^2_{\beta,\eta} = C_1 + \eta C_2$; *y* C_β, C_1, C_2 *son constantes positivas que dependen de* β, σ, k. *Entonces,*

$$\kappa(\mathcal{A}_h) \leq R^* \left(C_1 + \eta C_2\right) \left(C_\beta máx\left\{1, \frac{1}{\eta}\right\}\right) h^{-2}$$

donde $h = máx\{diam(T) : T \in \mathcal{T}_h\}$. □

El Teorema 3 muestra que el comportamiento asintótico de $\kappa(\mathcal{A})$ es $\mathcal{O}\left(h^{-2}\right)$, igual al del método de elemento finito clásico. Esta cota fué comprobada numéricamente para discretizaciones con mallas no estructuradas en 2D. A continuación se muestra que dicho comportamiento también se verifica para mallas no estructuradas en 3D. En la Figura 6–1 se muestra el comportamiento del condicionamiento con respecto al tamaño de la malla. Mediante regresión lineal se obtienen las siguientes pendientes -2.74, -2.27 y -2.11 para $p = 1, 2, 3$ respectivamente, lo cual muestra el comportamiento asintótico de orden $\mathcal{O}\left(h^{-2}\right)$ independientemente del grado de aproximación utilizado.

En la Figura 6–2 se muestra el condicionamiento espectral κ como función del parámetro de estabilidad η. Se observa un comportamiento de $\mathcal{O}\left(1/\eta\right)$ para $\eta \ll 1$ y de $\mathcal{O}\left(\eta\right)$ para $\eta \gg 1$, lo que muestra que la cota propuesta en el Teorema 3 también es precisa para mallas no estructuradas en 3D.

El Teorema 3 presenta que $\kappa(\mathcal{A})$ depende en forma desconocida del parámetro $\boldsymbol{\beta}_{\mathcal{S}}$. En la sección 5.2 se muestra que el esténcil de $\mathcal{A}$ se puede reducir dependiendo de la selección de este parámetro. Además, como se muestra en la Tabla 6–1, el condicionamiento de $\mathcal{A}$ también es afectado por $\boldsymbol{\beta}_{\mathcal{S}}$, de tal manera que al reducir el esténcil de $\mathcal{A}$ su condicionamiento aumenta.

En la práctica, el valor de h es bastante pequeño, por lo que $\kappa(\mathcal{A}) \gg 1$. Por esta razón, es necesario el uso de técnicas que reduzcan el condicionamiento de $\mathcal{A}$. Estas técnicas consisten en construir una matriz M no singlar, la cual se llama *precondicionador*, tal que la matriz $M^{-1}A$ del sistema equivalente $M^{-1}A = M^{-1}b$

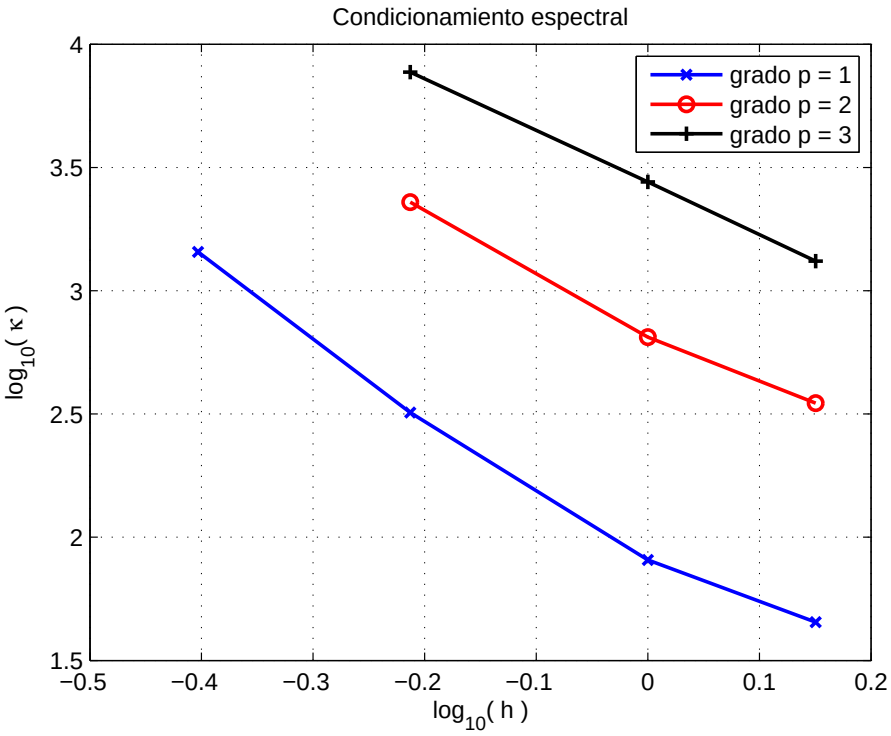


Figura 6–1: Condicionamiento espectral: $\kappa(h)$ para $p = 1, 2, 3$

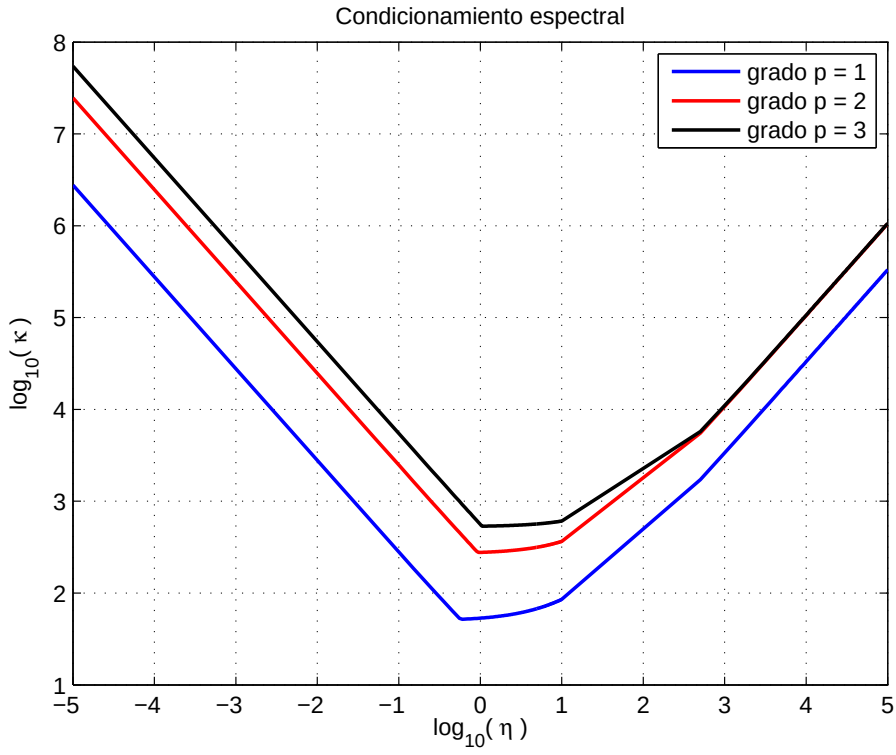


Figura 6–2: Condicionamiento espectral: $\kappa(\eta)$ para $p = 1, 2, 3$

Tabla 6–1: Condicionamiento de la matriz $\mathcal{A}$ para $p = 1$

β_0	β_1	β_2	β_{rnd}
6.67e+01	6.43e+01	2.06e+01	7.08e+01
1.49e+02	1.53e+02	5.33e+01	1.37e+02
2.08e+03	1.73e+03	2.60e+02	1.63e+03
4.64e+03	3.48e+03	8.73e+02	3.99e+03

posee un mejor condicionamiento espectral que el de la matriz A del sistema original. Nótese que si $M = A$ el sistema es resuelto de manera inmediata. En el caso de CG, que requiere simetría en la matriz, se considera el precondicionador de la forma $M^{-1}AM^{-T}$, el cual es simétrico. Al resolver un sistema lineal por CG utilizando algún precondicionador se le conoce como Gradiente Conjugado Precondicionado (PCG por sus siglas en ingles). Algunos de los precondicionadores más comunes para sistemas simétricos son Jacobi, Gauss-Seidel, Cholesky y técnicas multiniveles [17].

Las técnicas de multiniveles son las más óptimas, debido a que el número de iteraciones no depende del tamaño de la matriz. Éstas consisten en reducir el tamaño de la matriz del sistema lineal, por medio de una secuencia de mallas anidadas. La matriz del sistema es reducida al fusionar o eliminar entradas de la matriz. Una vez generada la nueva matriz, el nuevo sistema también es reducido de manera recursiva con ayuda de la siguiente malla, hasta finalizar con todas las mallas. Esta secuencia de matrices son conocidas como los niveles del precondicionador. En la Figura 6–3 se muestra una secuencia de mallas Cartesianas que cumple el hecho se estar anidadas. Similarmente en la Figura 6–4, se presenta una secuencia de mallas no estructuradas anidadas. A este precondicionador multiniveles se le conoce como *precondicionador multinivel geométrico* [23].

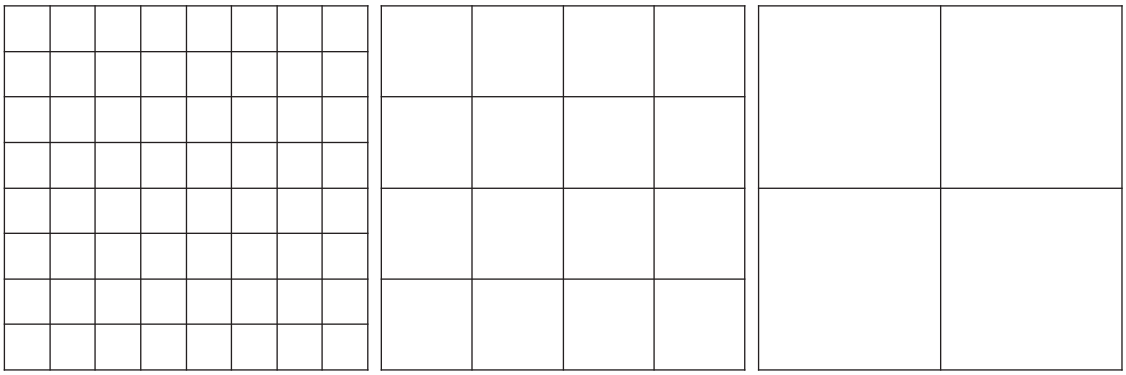


Figura 6–3: Secuencia de mallas cartesianas

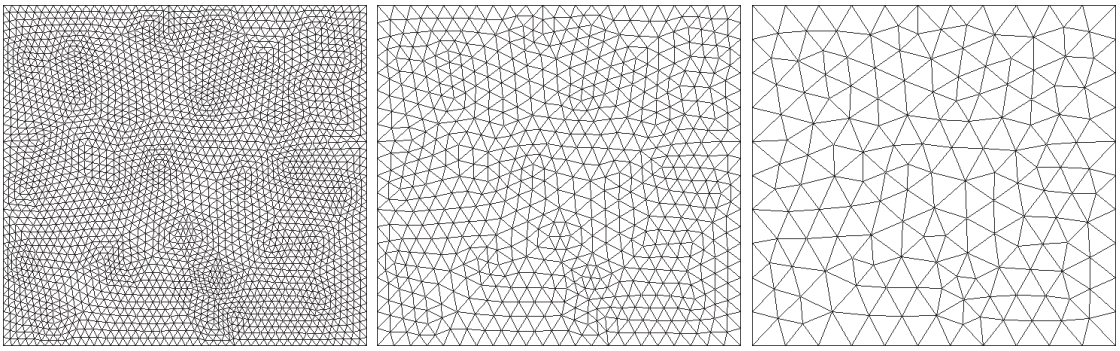


Figura 6–4: Secuencia de mallas no estructuradas

Una vez generados todos los niveles el precondicionador, para resolver el sistema lineal se realizan algunos pasos de un método de relajación, como Jacobi o Gauss-Seidel. La aproximación obtenida es *restringida*, para ser usada como valor inicial del sistema lineal auxiliar dado por el siguiente nivel inferior. De igual manera se sigue este proceso hasta el último nivel. Luego, cuando el sistema $l+1$ es resuelto, la solución es *prolongada* al nivel l, donde nuevamente se realizan algunos pasos de un método de relajación antes de pasar al siguiente nivel. A todo este proceso se le conoce como *ciclo-V*, el cual se ilustra en la Figura 6–5.

Desafortunadamente, en la práctica no se cuenta con esta secuencia de mallas anidadas, solamente se tiene la malla más fina. Por tal motivo, es necesario generar esta secuencia a través de la malla fina mediante la aglomeración de celdas. En mallas Cartesianas este proceso es más sencillo que en mallas no estructuradas, donde en la mayoría de los casos la aglomeración de celdas no es posible. Por ejemplo, en el caso de una malla de triángulos, la aglomeración de triángulos no siempre da

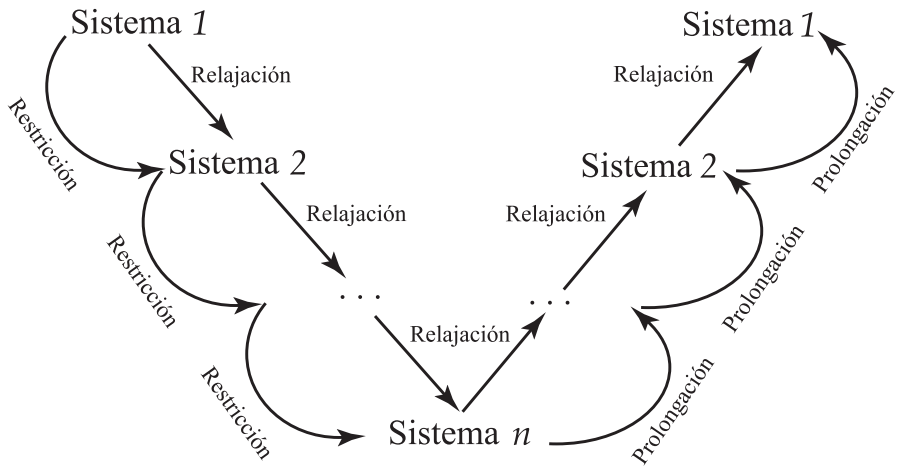


Figura 6–5: Diagrama de un Ciclo-V

como resultado otro triángulo, esto se ilustra en la Figura 6–6. Una alternativa para solventar este problema consiste en crear el siguiente nivel, mediante el grafo que representa la matriz del sistema y no por medio del grafo que representa la malla. A este proceso se le conoce como *multinivel algebraico* y fué introducido por Ruge y Stüben en [24]. Algunos ejemplos de precondicionadores multiniveles algebraicos se presentan en [25, 26].

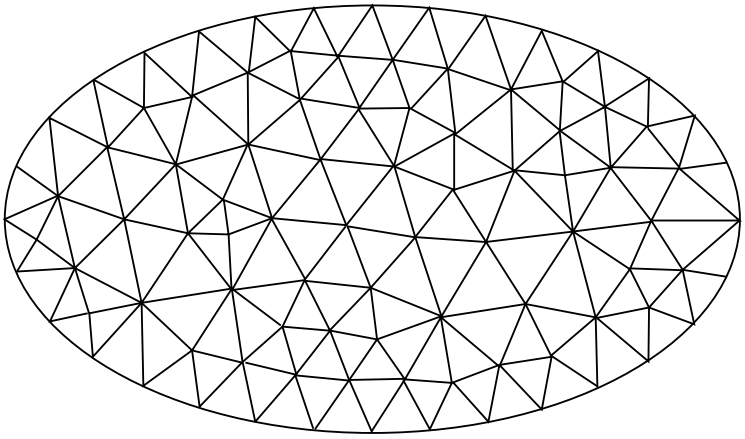


Figura 6–6: Malla no estructurada, en la que no se puede aglomerar celdas

En [8] se estudian algunos precondicionadores algebraicos para la formulación del LDG en 2D. Además, se considera un precondicionador basado en las idea de aproximar el espacio donde se encuentra la solución del problema original, por un *espacio auxiliar* de menor dimensión. El uso de un espacio auxiliar fue propuesto

por J. Xu en [27]. Esto se puede interpretar como un precondicionador de dos niveles, donde el primer nivel es dado por el espacio donde se aproxima la solución del sistema original, mientras el segundo nivel es dado por un espacio auxiliar. Al sistema generado por el espacio auxiliar se le conoce como el *problema auxiliar*. En la Sección 6.1 se presentan dos precondicionadores multiniveles algebraicos, propuestos para resolver el problema auxiliar resultante en el precondicionador *multinivel semi-algebraico* presentado en la Sección 6.2.

6.1 Precondicionador multinivel algebraico

En esta sección se presenta la descripción de dos precondicionadores multiniveles algebraicos. El primero de ellos es un precondicionador AMG ("Algebraic Multigrid"), propuesto por R. Beck en [25], el cual genera un operador de prolongación para cada nivel, por medio del grafo que representa la matriz. El segundo, referido como ASM ("Algebraic Symmetric Multigrid") consiste en un precondicionador basado en la factorización LU incompleta [17], propuesto por Y. Saad en [26]. Ambos precondicionadores, son construidos de manera simétrica. La matriz A_l representa la matriz en el nivel l y $A_0 = \mathcal{A}$. Finalmente, denotamos $l_{\text{máx}}$ el último nivel del precondicionador.

6.1.1 AMG basado en gáfos

El AMG presente en esta sección es el propuesto en [25], el cual utiliza ciclos-V con un número variado de pasos de relajación en cada nivel, donde se mostró numéricamente que al incrementar el número de pasos de relajación el rendimiento mejora, disminuyendo el número de iteraciones necesarias para resolver el sistema. La reducción de cada matriz es llevada acabo por medio de una selección de entradas en la matriz, la cual es realizada por una estrategia de *engrosamiento* sobre el grafo de la matriz del nivel actual. Luego, para cada nivel $l \neq l_{\text{máx}}$, se considera la matriz P_l, y definimos

$$A_{l+1} \;=\; P_l^T A_l P_l,$$

la matriz para el siguiente nivel. Note que A_{l+1} es simétrica, y además su tamaño es menor que el de A_l. Luego se sigue este proceso hasta obtener $A_{l_{\text{máx}}}$. En este contexto, P_l es conocido como el *operador de prolongación*, mientras que P_l^T se le llama *operador de restricción*.

Como la matriz $A_{l_{\text{máx}}}$ es de tamaño pequeño, el sistema $A_{l_{\text{máx}}} x_{l_{\text{máx}}} = r_{l_{\text{máx}}}$ se resuelve por medio de eliminación Gaussiana. Una vez ya definidas las diferentes matrices para cada nivel, se puede formular un precondicionador multiniveles algebraico, usando un ciclo-V con un número variado de pasos de relajación en los distintos niveles, como se muestra en el Algoritmo 6.1.1.

Algoritmo 6.1.1: AMG(x_0, b)

```
1  r_0 ← b − A x_0
2  x_0 = e_0 ← 0
3  para l ← 0 aumentar hasta l_{máx−1} hacer
4        μ pasos de un método de relajación al sistema A_l e_l = r_l
5        r_{l+1} ← P_l^T (r_l − A_l e_l)
6        e_{l+1} ← 0
7  resolver A_{l_máx} e_{l_máx} = r_{l_máx} por eliminación Gaussiana
8  para l ← l_{máx−1} decrecer hasta 0 hacer
9        e_l ← e_l + P_l e_{l+1}
10       μ pasos de un método de relajación al sistema A_l e_l = r_l
```

Para los experimientos de la Sección 6.3, la definición del operador de prolongación es presentada en [25]. La construcción de P_l es mediante el grafo asociado a la matriz A_l, por medio de la selección de *nodos maestros* y *nodos esclavos*. Esta selección de nodos, se le conoce como una estrategia de *engrosamiento*.

6.1.2 ASM basado en fractorizaciones incompletas

El ASM consiste en un precondicionador multinivel obtenido de la factorización LU incompleta (ILUM) presentado en [26], el cual reduce el tamaño de una matriz al aplicar eliminación Gaussiana por bloques, y así obtener la matriz para el siguiente nivel. Basados en las ideas de selección de nodos maestros y nodos esclavos en [25], se realiza una permutación simetría sobre A_l, de tal forma que:

$$P_l^T A_l P_l \;=\; \begin{pmatrix} D_l & E_l \\ E_l^T & C_l \end{pmatrix},$$

donde D_l es una matriz diagonal. Luego, se reduce el sistema por medio de eliminación Gaussiana por bloques, para obtener el complemento de Schur

$$A_{l+1} \;=\; C_l - E_l^T D_l^{-1} E_l, \tag{6.1}$$

y se continúa el proceso hasta un nivel máximo $l_{\text{máx}}$. La nueva matriz $P_l^T A_l P_l$ se puede escribir como

$$P_l^T A_l P_l \;=\; \begin{pmatrix} D_l & E_l \\ E_l^T & C_l \end{pmatrix} \;=\; \begin{pmatrix} I_d & 0 \\ E_l^T D_l^{-1} & I_d \end{pmatrix} \begin{pmatrix} D_l & E_l \\ 0 & A_{l+1} \end{pmatrix},$$

donde I_d representa la matriz identidad. El vector de las incógnitas es dividido de la siguiente manera

$$x_l \;=\; \begin{pmatrix} x_{l,f} \\ \\ x_{l,c} \end{pmatrix}.$$

En el Algoritmo 6.1.2 se define el precondicionador multinivel.

Algoritmo 6.1.2: ASM(x_0, b)

$x_0 \leftarrow b$
para $l \leftarrow 0$ **aumentar hasta** $l_{\text{máx}-1}$ **hacer**
    $x_l \leftarrow P_l^T x_l$
    $x_{l+1} \leftarrow x_{l,c} - E_l^T D_l^{-1} x_{l,f}$
$b_{l_{\text{máx}}} \leftarrow x_{l_{\text{máx}}}$
resolver $A_{l_{\text{máx}}} x_{l_{\text{máx}}} = b_{l_{\text{máx}}}$ **por eliminación Gaussiana**
para $l \leftarrow l_{\text{máx}-1}$ **decrecer hasta** 0 **hacer**
    $x_{l,c} \leftarrow x_{l+1,c}$
    $x_{l,f} \leftarrow D_l^{-1}(x_{l+1,f} - E_l x_{l+1,c})$
    $x_l \leftarrow P_l x_l$

La matriz A_l es esparcida, por lo que las matrices D_l, E_l y C_l, también lo son. Por otro lado, en cada nivel, la matriz $A_{l+1} = C_l - E_l^T D_l^{-1} E_l$ se va haciendo más densa, lo cual causa que el rendimiento del precondicionador ASM sea bajo. Para solucionar esto, se consideran las ideas de *umbrales* presentadas en [17], para así aproximar A_{l+1} por una matriz esparcida y con ello, mejorar el tiempo en la construcción del precondicionador. Sin embargo, como se verá en la Sección 6.3 el precondicionador es poco robusto, ya que construir el ASM dependerá de un parámetro adicional τ.

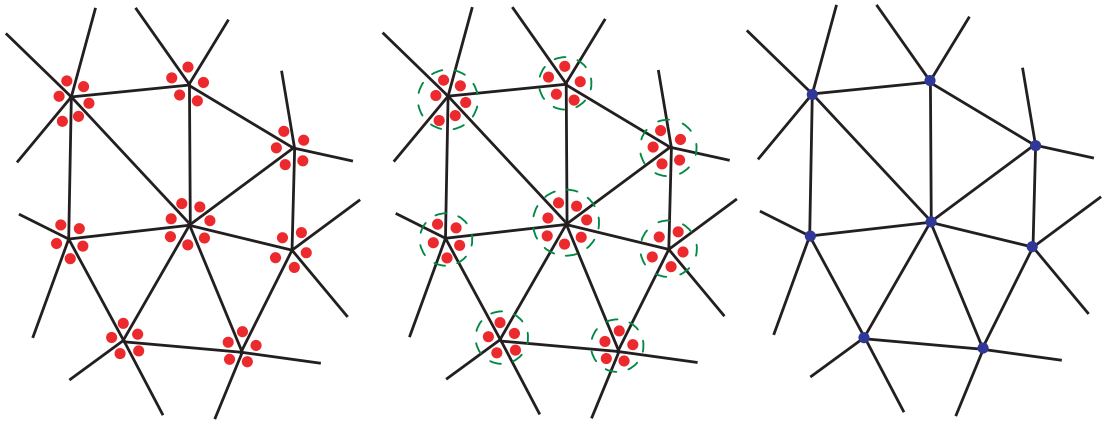
6.2 Precondicionador multinivel semi-algebraico

En [27] se presenta un marco abstracto para un precondicionador de dos niveles, utilizando un espacio auxiliar. El objectivo es reducir la dimensión del espacio donde se encuentra la solución original, por un espacio de menor dimensión. Por ejemplo, esta técnica puede ser aplicada para resolver aproximaciones de elemento finito de alto orden, por medio de un espacio de bajo orden. Para el caso del LDG cuando se usan aproximaciones $p = 1$, se puede considerar el espacio de funciones continuas

$$\mathcal{V}_1 \;=\; \{\phi \in C^0(\Omega) : \phi|_T \in \mathcal{P}_1(T), \forall\, T \;\in \mathcal{T}_h\},$$

como nuestro espacio auxiliar. En [8] se describe una técnica para obtener el espacio auxiliar $\mathcal{V}_1$. La idea consiste en colapsar los grados de libertad asociados a un nodo, como un único grado de libertad para dicho nodo, como se muestra en la Figura 6–7.

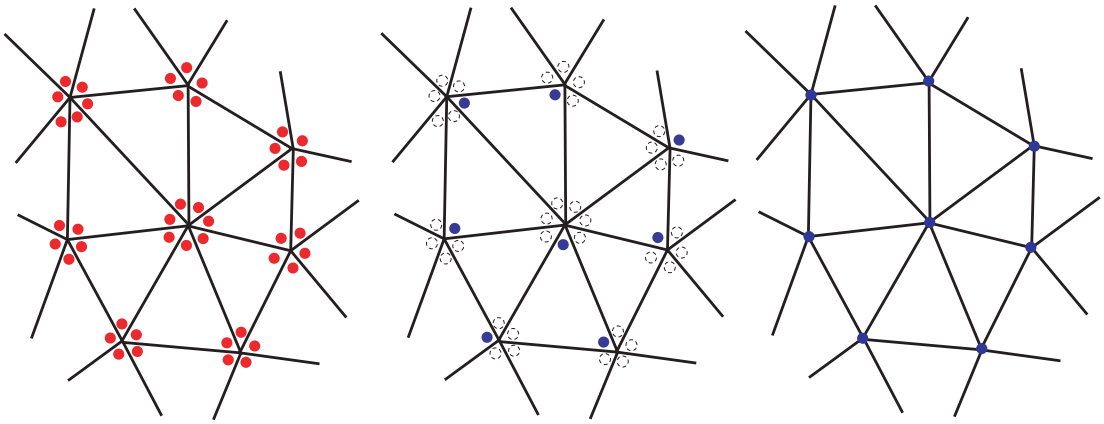
La técnica de colapsar los nodos mostró ser eficiente en el caso de 2D. En la Sección 6.3 se muestran algunos experimentos numéricos que muestran el comportamiento en 3D. Además, a manera de una segunda técnica para obtener el espacio auxiliar $\mathcal{V}_1$, se considera la idea de designar un único grado de libertad para cada nodo del conjunto de grados de libertad asociados a ese nodo. Se puede interpretar



Original del LDG Colapsar Espacio auxiliar

Figura 6–7: Representación de los grados de libertad

como una exclusión de los grados de libertad en un nodo, conservando sólo uno de manera aleatoria. En la Figura 6–8 se ilustra esta técnica de exclusión.



Original del LDG Excluir Espacio auxiliar

Figura 6–8: Representación de los grados de libertad

Se considera $A_1 x_1 = d_1$ la representación matricial del problema auxiliar, obtenido del espacio auxiliar $\mathcal{V}_1$. Este problema se resuelve mendiante algún método multinivel algebraico y su solución se proyecta a la aproximación del problema original $A_0 x_0 = b$. De esta forma, en el Algoritmo 6.2 se define el precondicionador multinivel semi-algebraico.

Para resolver la línea 4 del Algortimo 6.2, se pueden considerar los dos precondicionadores multinieveles algebraicos presentados en la Sección 6.1. Para obtener el

Algoritmo 6.2: C^0-AMG(x_0, b)

1 **aplicar** μ pasos de un método de relajación al sistema $A_0 x_0 = b$
2 $d_0 \leftarrow b - A_0 x_0$
3 **reducir** la aproximación al problema auxiliar, $d_1 \leftarrow P_0^T d_0$
4 **resolver** el problema auxiliar, $A_1 e_1 = d_1$
5 **prolongar** la aproximación al problema original, $e_0 \leftarrow P_0 e_1$
6 $x_0 \leftarrow x_0 + e_0$
7 **aplicar** μ pasos del método de relajación al sistema $A_0 x_0 = b$

espacio auxiliar se tienen las técnicas de colapsar y excluir nodos de las Figuras 6–7 y 6–8, respectivamente. Además, al considerar aplicar en el fragmento de código anterior un método de relajación puntual o bien por bloques, se pueden definir ocho precondicionadores multinieveles semi-algebraicos, los cuales se presentan en la Tabla 6–2.

Tabla 6–2: Definición de los precondicionadores multiniveles semi-algebraicos

Nombre del precondicionador	Definición del problema auxiliar	Método para resolver el sistema auxiliar	Método de relajación
C^0-AMG	Figura 6–7	AMG	Puntual
BC^0-AMG	Figura 6–7	AMG	Por bloques
C^0-ASM	Figura 6–7	ASM	Puntual
BC^0-ASM	Figura 6–7	ASM	Por bloques
C^0-AMG$_2$	Figura 6–8	AMG	Puntual
BC^0-AMG$_2$	Figura 6–8	AMG	Por bloques
C^0-ASM$_2$	Figura 6–8	ASM	Puntual
BC^0-ASM$_2$	Figura 6–8	ASM	Por bloques

6.3 Experimentos numéricos

A continuación se presentan algunos experimentos numéricos para estudiar la eficiencia y lo robusto de los precondicionadores descritos previamente. Para ello se consideran problemas donde el condicionamiento de la matriz de rigidez $\mathcal{A}$ sea alto. Esto se logra, por ejemplo, al reducir el esténcil o al presentarse varios materiales en el dominio. Los experiementos en esta sección siguen las características de los presentados en el Capítulo 5, es decir, todas las mallas fueron generadas utilizando

TETGEN, la base de polinomios es la interpolatoria de Lagrange. De los valores para $\boldsymbol{\beta}_{\mathcal{S}}$ propuestos en la Tabla 5–8, solamente se consideran los casos de β_2 por ser el que mejor condicionamiento presenta en la Tabla 6–1 y de β_{rnd} por ser el parámetro que más disminuye el esténcil de $\mathcal{A}$.

6.3.1 Problema con un material

En este ejemplo se considera el problema modelo (2.1) en el dominio $\Omega = (0,1)\times(0,1)\times(0,1)$, con condiciones de frontera de tipo Dirichlet y difusión isotrópica igual a la matriz identidad. Las mallas para este ejemplo son las mismas de la Figura 5–1. La función f se escoge de tal forma que la solución exacta sea la función

$$u(x,y,z) \;=\; \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

Este problema es resuelto utilizando aproximaciones de grado $p = 1$ y con tres mallas de 5913, 46692 y 372238 celdas. Además, se considera los parámetros β_2 y β_{rnd} para el esténcil de $\mathcal{A}$. Como resultado se obtienen seis sistemas lineales que representan la discretización del problema deseado. En las Tablas 6–3 y 6–4 se presenta el número de interaciones que le tomó al PCG resolver los seis sistemas, utilizando los precondicionadores AMG, con un número máximo de 1000 iteraciones permitidas y una tolerancia relativa de 10^{-13}.

Tabla 6–3: Número de iteraciones de los precondicionadores AMG para β_2

ndofs	μ	AMG	C^0-AMG	BC^0-AMG	C^0-AMG$_2$	BC^0-AMG$_2$
23652	4	25	19	12	41	40
	8	20	13	9	26	28
	16	25	9	8	16	19
186768	4	29	21	13	77	78
	8	25	14	11	49	54
	16	20	9	9	31	37
1488952	4	35	21	16	149	152
	8	31	15	14	95	105
	16	26	15	12	59	73

Tabla 6–4: Número de iteraciones de los precondicionadores AMG para β_{rnd}

ndofs	μ	AMG	C^0-AMG	BC^0-AMG	C^0-AMG$_2$	BC^0-AMG$_2$
23652	4	96	89	17	101	60
	8	66	62	13	69	41
	16	46	44	9	47	28
186768	4	153	134	19	161	114
	8	104	95	14	110	78
	16	71	65	10	75	54
1488952	4	227	189	20	248	223
	8	156	135	15	168	153
	16	107	96	12	114	106

En la Tabla 6–3 se presentan las tres matrices con el esténcil de β_2, mientras en la Tabla 6–4 las tres matrices de β_{rnd}. El parámetro μ representa la cantidad de pasos del método de relajación Gauss-Seidel. Para un nivel $l = 1, \ldots, l_{\text{máx}}$, la cantidad de pasos de relajación es $\mu + l - 1$. Finalmente, *ndofs* se refiere a la cantidad de incógnitas del sistema, mejor conocidos como los *grados de libertad*.

Se puede apreciar que al aumentar el número de pasos de relajación μ, el número de iteraciones disminuye. Los precondicionadores C^0-AMG y BC^0-AMG muestran pocos cambios en las iteraciones para las primeras tres matrices. Por otro lado, para las últimas tres el número de iteraciones es mayor, debido al cambio en el condicionamiento, consecuencia de la disminución en el esténcil. Se muestra que para estas matrices, el BC^0-AMG mantiene su comportamiento al reducir el número de iteraciones, de manera contraria al C^0-AMG.

Análogamente a los precondicionadores AMG, en las Tablas 6–5 y 6–6 se presentan las iteraciones al resolver los sistemas con los precondicionadores ASM, donde C denota aproximar el complemento de Schur 6.1 sólo con el bloque C_l, es decir, tomando $E_l^T D_l^{-1} E_l = 0$. Por otro lado, por medio de las estrategias de umbrales, se puede aproximar A_{l+1} por $\tilde{A}_{l+1}$, la cual posee una menor cantidad de elementos no cero que la matriz A_{l+1}, obtenida al eliminar entradas en la fila w_i de A_{l+1}, por medio de un parámetro τ, de tal forma que el elemento w_{ij} es eliminado de w_i si este

cumple que $|w_{ij}| < \tau\|w_i\|_2$. Es claro, que si $\tau \to 0$, entonces $\tilde{A}_{l+1} \to A_{l+1}$. Para las Tablas 6–5 y 6–6 se consideran los siguientes valores para τ, $\tau_1 = 0.01$ y $\tau_2 = 0.005$. Finalmente, con respecto a C^0-ASM, BC^0-ASM, C^0-ASM$_2$ y BC^0-ASM$_2$ se realizaron cuatro pasos de Gauss-Seidel, es decir, $\mu = 4$.

Tabla 6–5: Número de iteraciones de los precondicionadores ASM para β_2

ndofs	τ	ASM	C^0-ASM	BC^0-ASM	C^0-ASM$_2$	BC^0-ASM$_2$
23652	C	105	24	23	41	40
	τ_1	67	19	14	44	43
	τ_2	49	19	12	44	43
186768	C	196	46	45	77	78
	τ_1	128	29	27	82	83
	τ_2	90	24	24	82	79
1488952	C	381	88	87	149	152
	τ_1	266	50	49	149	152
	τ_2	–	43	43	149	152

Tabla 6–6: Número de iteraciones de los precondicionadores ASM para β_{rnd}

ndofs	τ	ASM	C^0-ASM	BC^0-ASM	C^0-ASM$_2$	BC^0-ASM$_2$
23652	C	318	93	26	101	60
	τ_1	309	90	18	101	63
	τ_2	118	89	18	101	63
186768	C	494	149	50	161	114
	τ_1	1000	143	29	161	114
	τ_2	583	142	26	161	114
1488952	C	835	223	97	248	223
	τ_1	1000	215	52	248	223
	τ_2	1000	212	45	248	223

Se puede notar que en la mayoría de los precondicionadores se mostró mejoras en el número de iteraciones. Sin embargo, similarmente al BC^0-AMG, el BC^0-ASM muestra mejores resultados sobre los demás precondionadores ASM. En general, los mejores resultados se presentan para los precondicionadores AMG en comparación a los ASM. Además, en ambas tablas se puede apreciar que la estrategia de colapsar los nodos es más eficiente que la de excluir los nodos, por lo que las ideas presentadas en [8], muestran ser eficientes para LDG, tanto en 2D como en 3D.

En las Tablas 6–7 y 6–8 se presentan los tiempos de los precondicionadores AMG. Similarmente, en las Tablas 6–9 y 6–10 se presentan los tiempos de los precondicionadores ASM. Se considera el tiempo necesario para construir el precondicionador (setup) y el tiempo que le tomó a dicho precondicionador resolver los sistemas lineales anteriores. En ambas tablas, se muestra que al aumentar μ y disminuir τ, el tiempo al resolver también se incrementa. Además, se puede observar que el tiempo en resolver los tres últimos sistemas, donde el esténcil es dado por β_{rnd}, es menor para BC^0-AMG y BC^0-ASM (con τ_2), por lo que la estrategia de colapsar los nodos, junto con las técnicas de relajación a nivel de bloques, muestran ser las más eficientes. Con respecto a resolver el sistema auxiliar, se puede apreciar que los mejores resultados son al resolver dicho sistema con el AMG, ya que para el ASM es necesario la selección de un valor de τ adecuado, que permita reducir el número de iteraciones y el tiempo al resolver, lo cual muestra que entre el BC^0-AMG y el BC^0-ASM, el primero de ellos es más robusto.

Tabla 6–7: Tiempo (segs) de los precondicionadores AMG para β_2

		AMG		C^0-AMG		BC^0-AMG		C^0AMG$_2$		BC^0-AMG$_2$	
ndofs	μ	setup	resol.	setup	resol.	setup	resol.	setup	resol.	setup	resol.
23652	4	0.1	3.2	0.09	2.5	0.2	2.1	0.1	5.1	0.2	7.1
	8	0.1	4.5	0.09	2.9	0.2	2.7	0.1	5.6	0.2	8.5
	16	0.1	6.2	0.09	3.8	0.2	4.4	0.1	6.3	0.2	10.6
186768	4	1.9	34.5	1.2	24.9	2.1	26.4	0.2	85.6	1.4	157.3
	8	1.9	50.7	1.2	28.5	2.1	36.9	0.2	92.7	1.4	184.0
	16	1.9	74.6	1.2	34.5	2.1	55.8	0.2	108.3 1	1.4	230.6
1488952	4	67.6	370.8	44.5	218.6	63.6	309.3	1.5	1437.8	17.3	2956.7
	8	67.6	552.1	44.5	269.9	63.6	456.7	1.5	1560.6	17.3	3454.6
	16	67.6	848.1	44.5	393.6	63.6	713.3	1.5	1773.0	17.3	4346.5

En las Figuras 6–9, 6–10, 6–11 y 6–12 se presentan los historiales de los residuos del PCG por iteración, al usar los precondicionadores anteriores para las matrices con 186768 grados de libertad. A la izquierda se muestra el historial de residuos para las matrices con esténcil β_2, mientras que a la derecha las matrices con esténcil β_{rnd}.

Tabla 6–8: Tiempo (segs) de los precondicionadores AMG para β_{rnd}

ndofs	μ	AMG		C^0-AMG		BC^0-AMG		C^0AMG$_2$		BC^0-AMG$_2$	
		setup	resol.	setup	resol.	setup	resol.	setup	resol.	setup	resol.
23652	4	0.1	7.8	0.1	6.9	0.2	1.7	0.1	7.5	0.1	6.3
	8	0.1	9.4	0.1	8.2	0.2	2.2	0.1	8.8	0.1	7.3
	16	0.1	11.8	0.1	10.5	0.2	2.8	0.1	10.9	0.1	9.2
186768	4	3.8	112.3	1.1	93.6	1.7	20.7	0.1	106.1	1.0	126.5
	8	3.8	129.9	1.1	111.9	1.7	26.2	0.1	123.7	1.0	146.9
	16	3.8	161.8	1.1	139.3	1.7	34.9	0.1	154.5	1.0	185.5
1488952	4	216.9	1524.0	42.9	1151.5	47.6	210.0	1.3	1419.6	8.2	2376.5
	8	216.9	1787.5	42.9	1387.3	47.6	267.4	1.3	1636.2	8.2	2758.5
	16	216.9	2226.9	42.9	1789.9	47.6	395.2	1.3	2034.8	8.2	3480.7

Tabla 6–9: Tiempo (segs) de los precondicionadores ASM para β_2

ndofs	τ	ASM		C^0-ASM		BC^0-ASM		C^0ASM$_2$		BC^0-ASM$_2$	
		setup	resol.	setup	resol.	setup	resol.	setup	resol.	setup	resol.
23652	C	0.5	3.1	0.2	3.1	0.3	4.2	0.1	5.1	0.2	7.1
	τ_1	1.5	2.2	0.1	2.5	0.3	2.6	0.1	5.4	0.2	7.9
	τ_2	2.4	1.9	0.1	2.5	0.3	2.3	0.1	5.4	0.2	7.5
186768	C	3.9	61.5	0.7	52.1	2.0	93.1	0.2	85.0	1.4	156.7
	τ_1	14.9	46.5	0.8	33.2	2.1	56.5	0.2	90.6	1.4	166.4
	τ_2	24.1	40.8	0.8	27.7	2.1	50.2	0.2	90.5	1.4	158.7
1488952	C	52.2	1112.2	6.5	865.2	22.9	1719.0	1.6	1420.3	19.5	2959.9
	τ_1	139.7	948.5	7.8	493.6	23.7	976.7	1.7	1426.9	17.6	2957.4
	τ_2	–	–	8.4	425.2	25.0	861.8	1.5	1421.1	17.3	2947.7

El número de iteraciones es mayor para las matrices con esténcil β_{rnd}. El número máximo de iteraciones permitidas es 300, con una tolerancia relativa de 10^{-13}. En estas gráficas se puede apreciar que al incrementar el número de pasos μ del método de relajación, los residuos dismuyen con mayor rapidez. Para los precondicionadores ASM, tomar τ muy cerca de cero también contribuye a que los residuos decrezcan con mayor velocidad.

En la Figura 6–13 se presenta el historial de residuos de los precondicionadores con $\mu = 16$ y $\tau = \tau_2$, es decir, los que mejores resultados presentan. Se muestra que los precondicionadores BC^0-AMG y BC^0-ASM sobresalen ante demás. En la Figura

Tabla 6–10: Tiempo (segs) de los precondicionadores ASM para β_{rnd}

ndofs	τ	ASM		C^0-ASM		BC^0-ASM		C^0ASM$_2$		BC^0-ASM$_2$	
		setup	resol.	setup	resol.	setup	resol.	setup	resol.	setup	resol.
23652	C	0.3	6.3	0.1	7.1	0.2	2.8	0.1	7.5	0.1	6.1
	τ_1	0.9	7.3	0.1	6.9	0.2	1.9	0.1	7.5	0.2	6.4
	τ_2	1.3	3.2	0.1	6.9	0.2	1.9	0.1	7.5	0.1	6.4
186768	C	2.0	100.2	0.6	99.9	1.4	57.0	0.2	105.6	1.0	126.3
	τ_1	9.2	270.8	0.7	96.2	1.5	33.7	0.2	105.6	1.0	126.0
	τ_2	13.0	179.0	0.7	95.5	1.6	30.3	0.2	105.7	1.0	126.4
1488952	C	17.1	1548.2	5.1	1301.1	11.8	1044.4	1.3	1405.3	8.5	2366.1
	τ_1	83.2	2620.2	6.4	1255.5	13.4	570.1	1.5	1398.2	8.4	2343.9
	τ_2	120.7	3062.1	6.9	1236.2	14.4	491.0	1.4	1399.1	8.5	2343.9

6–14 se considera el precondicionador AMG y ASM que presentan los menores residuos en la Figura 6–13, donde se muestra que los residuos del BC^0-AMG disminuyen con mayor rapidez.

6.3.2 Problema con dos materiales

En este ejemplo se considera el problema modelo con condiciones de Dirichlet y difusión isotrópica, además de tener un dominio con dos materiales distintos. El dominio, consiste en dos cubos, el primero, el cubo $(0,1)\times(0,1)\times(0,1)$ el cual tiene difusión $\mathcal{D} = I_d$ y dentro de este cubo, un segundo cubo $\left(\frac{1}{4},\frac{3}{4}\right)\times\left(\frac{1}{4},\frac{3}{4}\right)\times\left(\frac{1}{4},\frac{3}{4}\right)$ con difusión $\mathcal{D} = (10^{-10}\cdot I_d)$. En la Figura 6–15 se muestra una de las mallas para este dominio.

Las condiciones de borde son Dirichlet en dos caras opuestas, una con Dirichlet cero y la otra con Dirichlet uno. Las caras restante tiene condiciones de Neumann cero. Para la fuente se considera $f = 0$. Este problema es resuelto con los precondicionadores C^0-AMG, BC^0-AMG, C^0-ASM y BC^0-ASM por ser los que mejores resultados presentaron en la Seción 6.3.1. Se consideran dos mallas de 191050 y 383819 celdas, además de los esténciles generados de los parámetros β_2 y β_{rnd}. Las matrices de estos cuatro problemas presentan un alto condicionamiento debido al

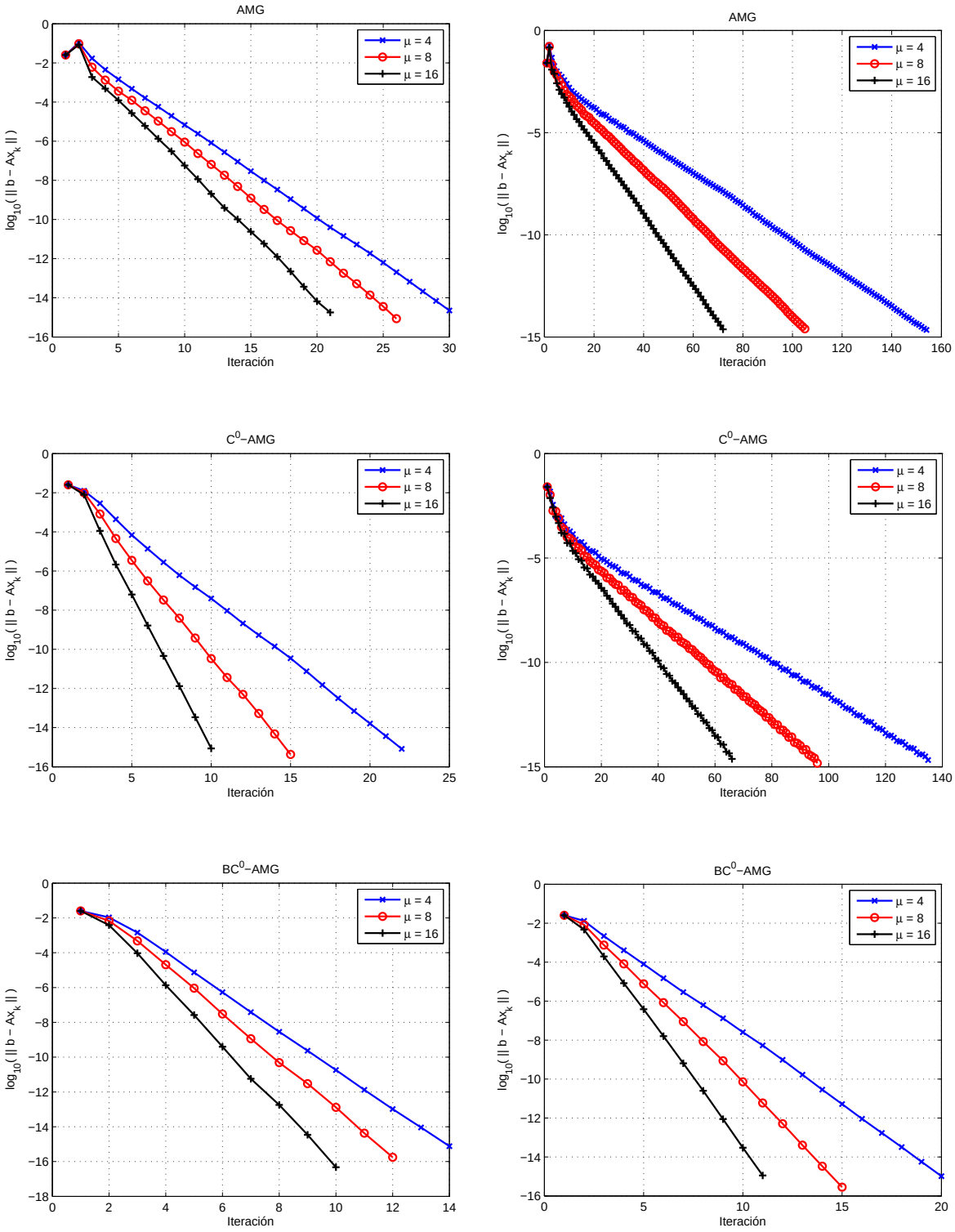


Figura 6–9: Historial de residuos para CG precondicionado con AMG, C^0-AMG y BC^0-AMG, para β_2 (izquierda) y β_{rnd} (derecha)

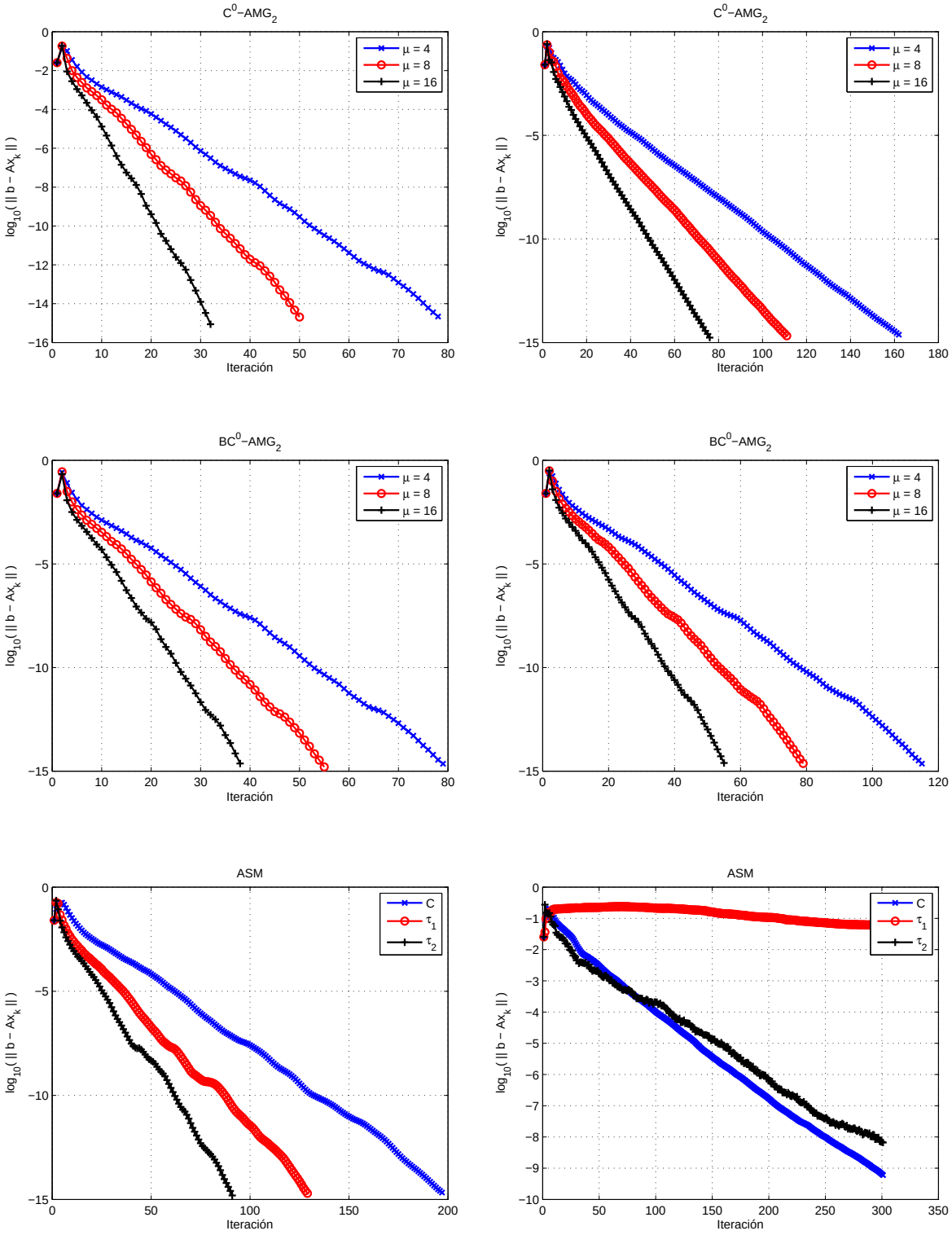


Figura 6–10: Historial de residuos para CG precondicionado con C^0-AMG$_2$, BC^0-AMG$_2$ y ASM para β_2 (izquierda) y β_{rnd} (derecha)

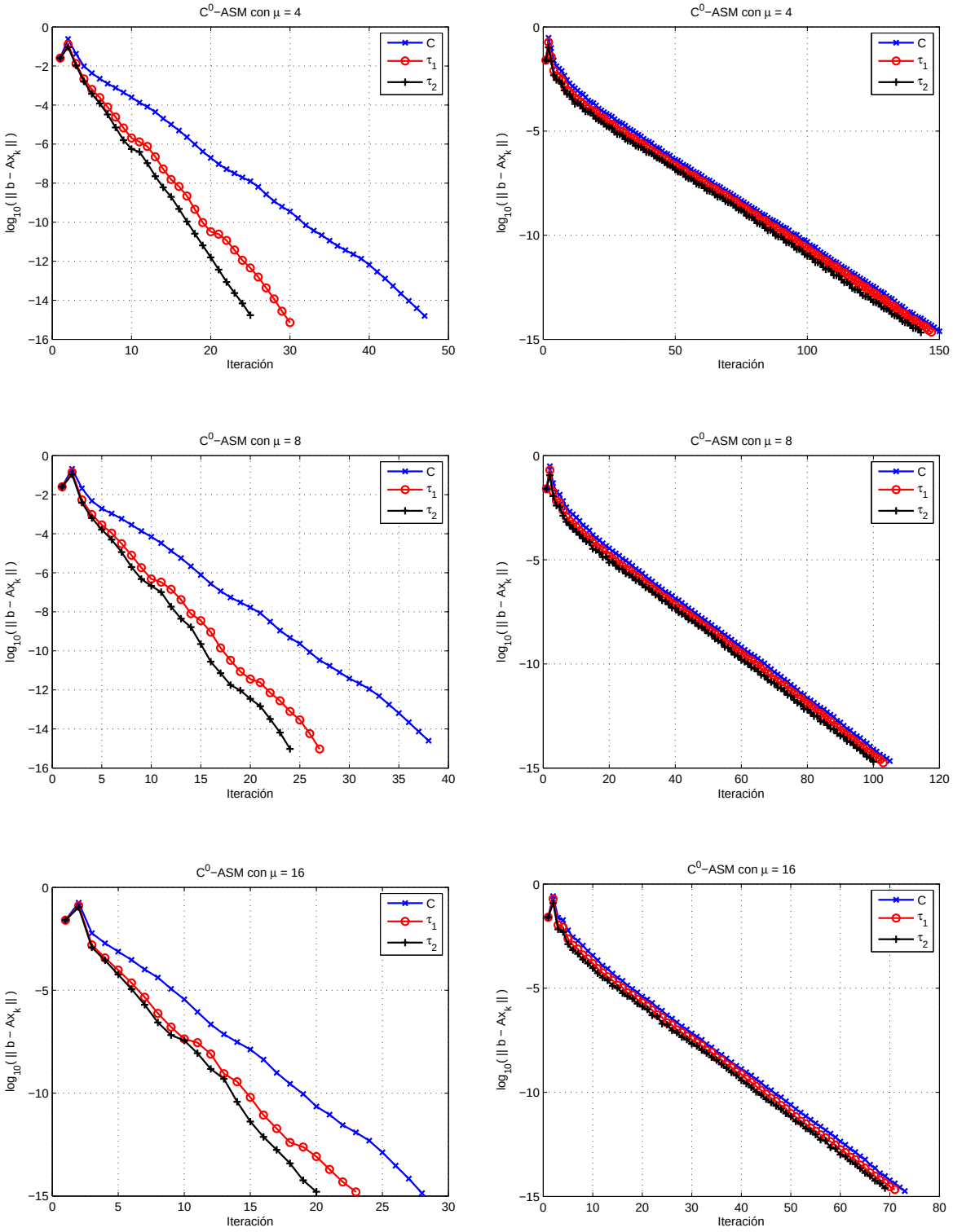


Figura 6–11: Historial de residuos para CG precondicionado con C^0-ASM para $\mu = 4, 8, 16$ y β_2 (izquierda) y β_{rnd} (derecha)

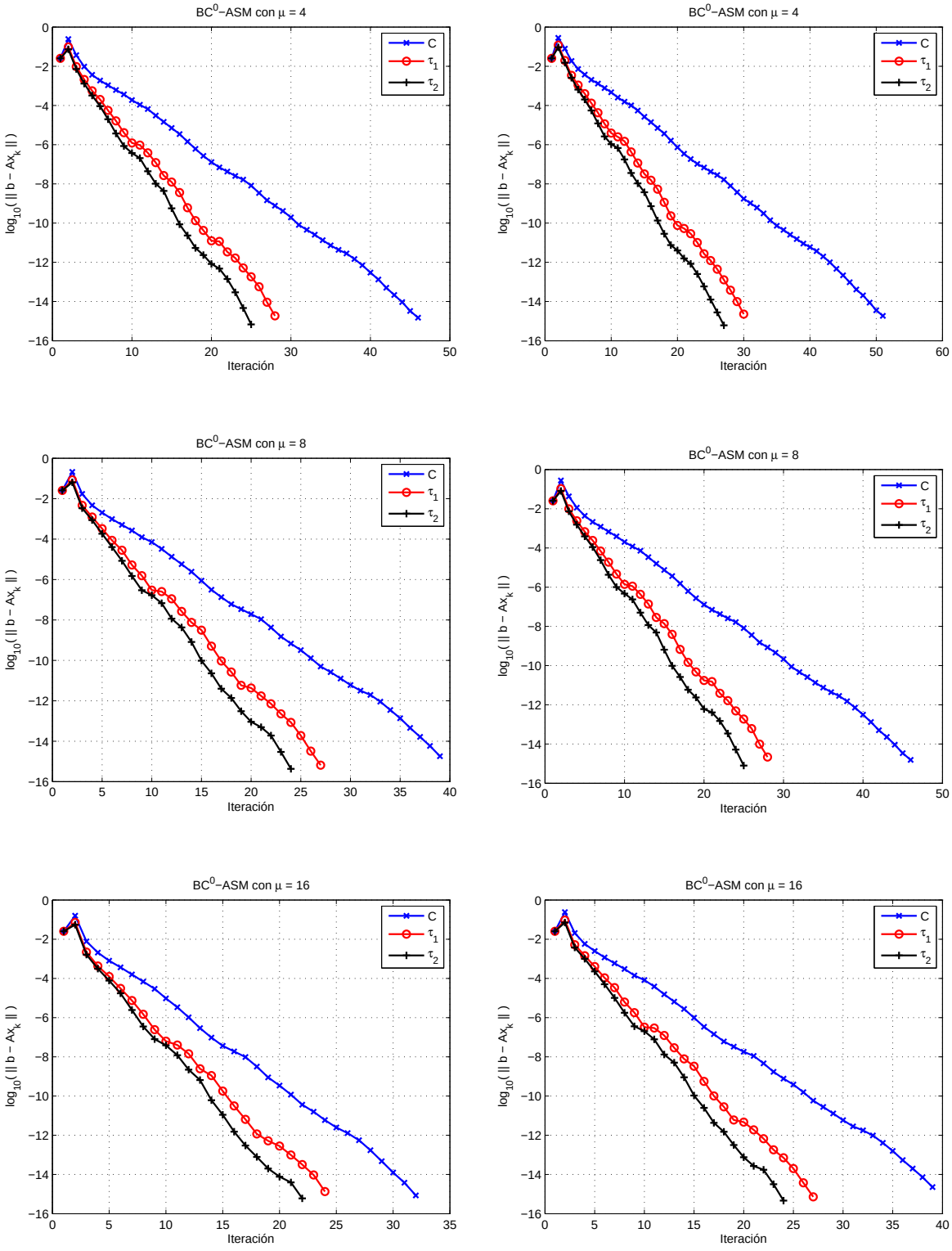


Figura 6–12: Historial de residuos para CG precondicionado con BC^0-ASM para $\mu = 4, 8, 16$ y β_2 (izquierda) y β_{rnd} (derecha)

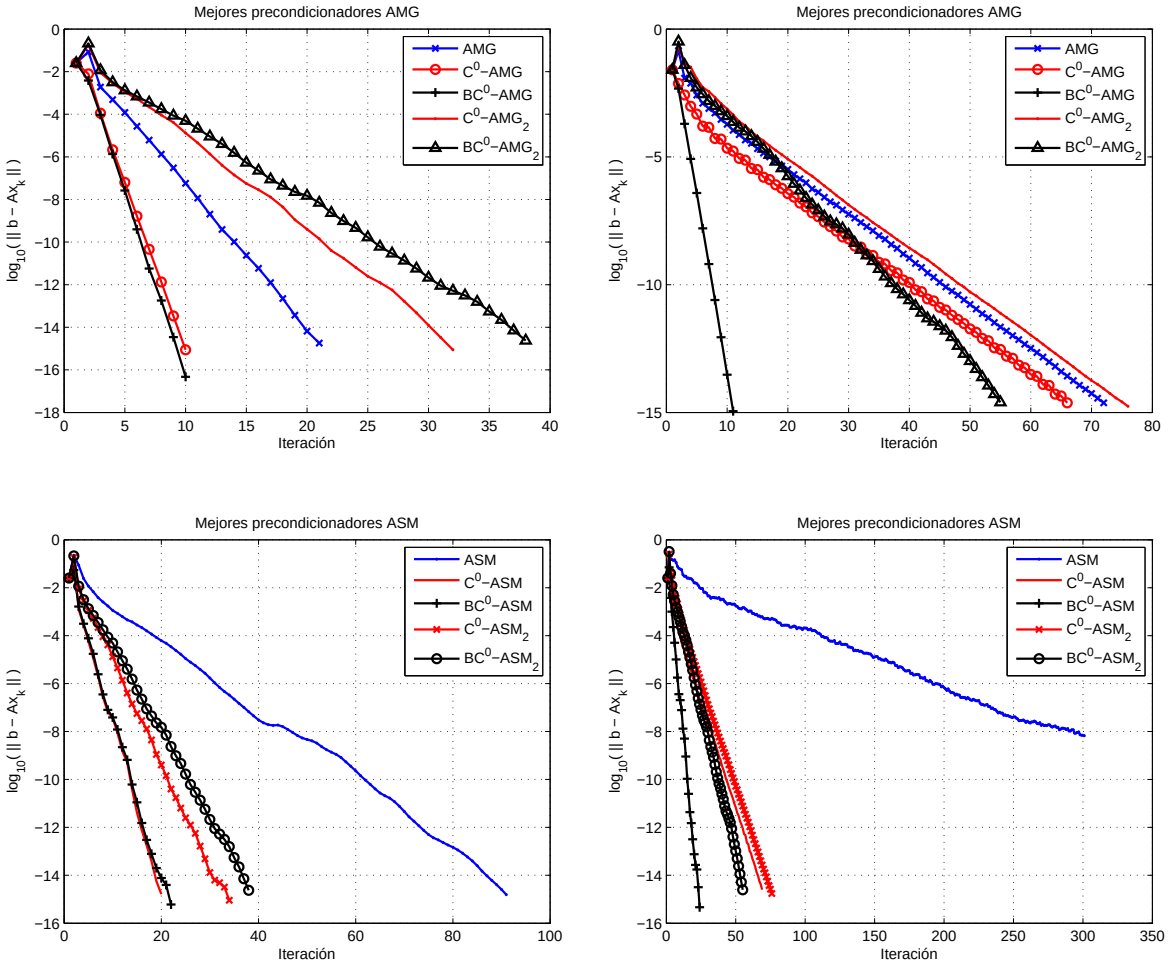


Figura 6–13: Historial de residuos para CG precondicionado con los mejores precondicionadores AMG y ASM, para β_2 (izquierda) y β_{rnd} (derecha)

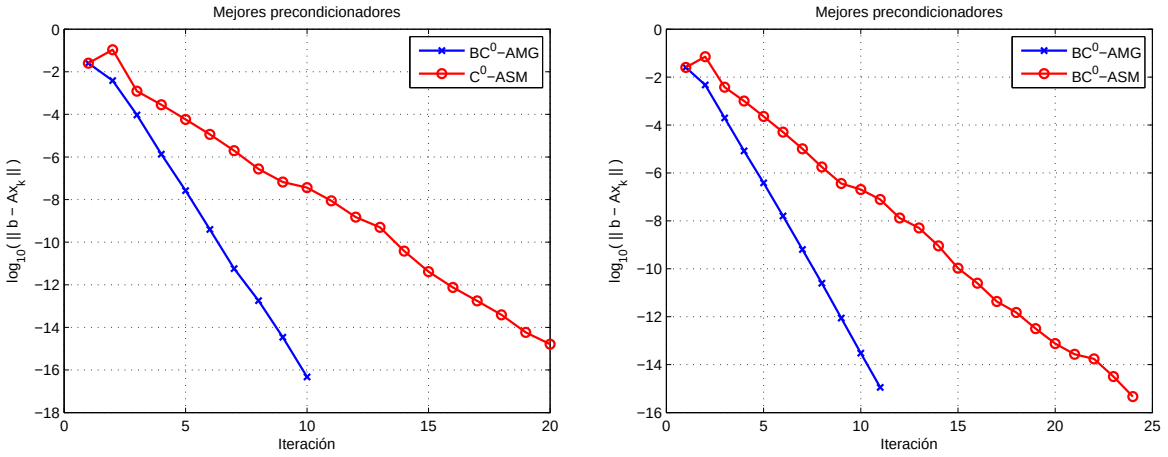


Figura 6–14: Historial de residuos para CG precondicionado con los mejores precondicionadores, para β_2 (izquierda) y β_{rnd} (derecha)

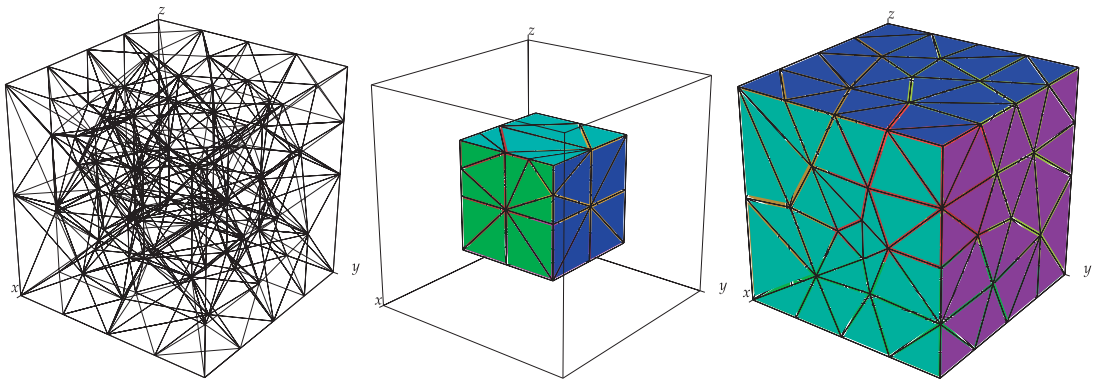


Figura 6–15: Ejemplo de malla (izquierda), cubo interior (centro) y frontera (derecha)

cambio brusco en los ordenes de las difusiones. En las Tablas 6–11 y 6–12 se muestra el número de iteraciones al resolver los cuatro sistemas con PCG utilizando un número máximo de 250 iteraciones permitidas y una tolerancia relativa de 10^{-13}. Para los precondicionadores ASM, se considera $\mu = 4$ pasos de Gauss-Seidel.

Tabla 6–11: Número de iteraciones de los precondicionadores AMG

		β_2		β_{rnd}	
ndofs	μ	C^0-AMG	BC^0-AMG	C^0-AMG	BC^0-AMG
764200	4	20	18	149	19
	8	16	16	104	16
	16	14	13	73	14
1535276	4	22	20	176	20
	8	19	18	120	18
	16	16	15	84	15

Tabla 6–12: Número de iteraciones de los precondicionadores ASM

		β_2		β_{rnd}	
ndofs	τ	C^0-ASM	BC^0-ASM	C^0-ASM	BC^0-ASM
764200	C	104	103	173	115
	τ_1	59	59	169	61
	τ_2	50	50	163	51
1535276	C	140	130	204	144
	τ_1	79	75	198	77
	τ_2	67	63	193	64

Se puede apreciar que el AMG no converge a la tolerancia deseada, para un máximo de 250 iteraciones. Por otro lado, el BC^0-AMG muestra ser el precondicionador más eficiente entre los considerados, además de no variar mucho el número de iteraciones cuando se incrementa el tamaño de la matriz del sistema. En las Tablas 6–13 y 6–14 se presentan los tiempos para estos precondicionadores.

Tabla 6–13: Tiempo (segs) de los precondicionadores AMG

		β_2				β_{rnd}			
		C^0-AMG		BC^0-AMG		C^0-AMG		BC^0-AMG	
ndofs	μ	setup	resol.	setup	resol.	setup	resol.	setup	resol.
764200	4	3.0	99.7	8.4	178.7	2.4	442.5	5.9	104.0
	8	3.0	134.6	8.4	266.6	2.4	517.5	5.9	147.7
	16	3.0	214.7	8.4	397.0	2.4	658.7	5.9	235.9
1535276	4	7.2	233.8	27.0	435.5	5.5	1107.3	12.6	239.5
	8	7.2	336.7	27.0	655.3	5.5	1271.0	12.6	362.5
	16	7.2	518.7	27.0	1006.4	5.5	1608.9	12.6	549.2

Tabla 6–14: Tiempo (segs) de los precondicionadores ASM

		β_2				β_{rnd}			
		C^0-ASM		BC^0-ASM		C^0-ASM		BC^0-ASM	
ndofs	τ	setup	resol.	setup	resol.	setup	resol.	setup	resol.
764200	C	3.1	485.1	8.4	964.7	2.5	493.9	5.9	593.5
	τ_1	3.6	277.8	8.7	560.5	3.0	483.0	6.5	317.1
	τ_2	3.9	236.1	9.0	473.0	3.2	467.6	6.8	266.0
1535276	C	7.4	1383.3	24.4	2676.7	5.5	1226.2	12.5	1611.3
	τ_1	8.7	788.1	25.5	1561.0	6.9	1194.7	14.3	862.5
	τ_2	9.3	664.2	28.4	1310.1	7.6	1165.6	15.0	719.9

El precondicionador BC^0-AMG muestra mejores resultados en tiempo para las matrices con un esténcil de β_{rnd}, las cuales poseen mayor condicionamiento que las demás. Es decir, se puede reducir el esténcil de la matriz de rigidez, sin importar el incremento en el condicionamiento de la misma, ya que el precondicionador BC^0-AMG muestra ser eficiente para estas matrices.

6.3.3 Problema con diez materiales

En este ejemplo se considera un problema de conducción de calor, el cual es resuelto en un dominio con conductividad térmal anisotrópica multiple [28], es decir, un dominio que presenta varios coeficientes de difusión anisotrópica $\mathcal{D}$. El dominio consiste del paralelepípedo $\Omega = (-3, 3) \times (0, 10) \times (0, 1)$, donde la región $(-3, 3) \times (k-1, k) \times (0, 1)$, con $k = 1, \ldots, 10$ posee su propio material. En la Figura 6–16 se muestra el dominio de este problema, además de un ejemplo de la malla para Ω.

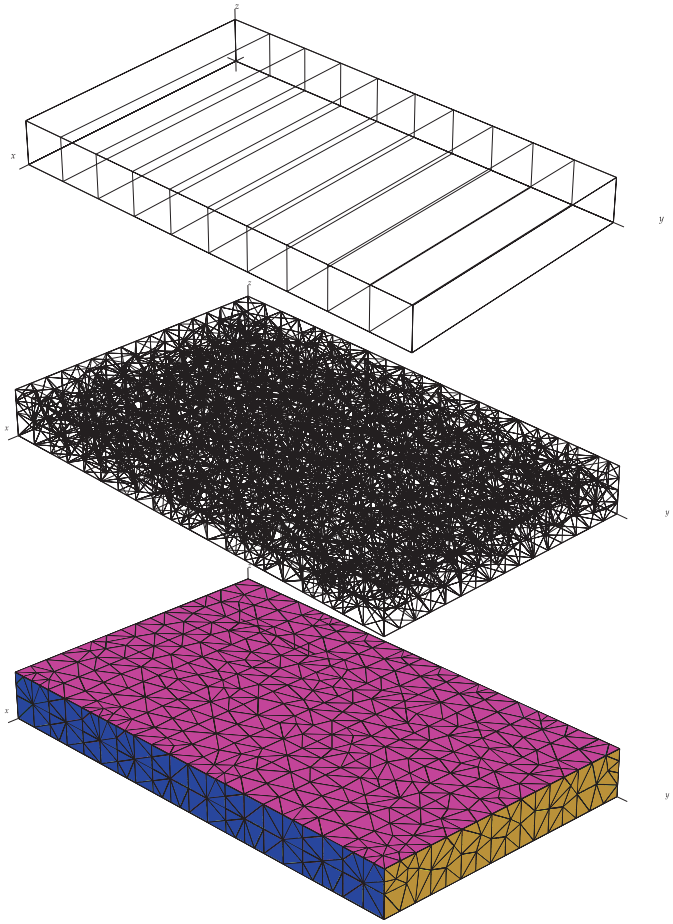


Figura 6–16: Dominio (superior), malla (centro) y frontera (inferior)

A continuación se describen las condiciones de borde. Para los planos $x = -3$, $x = 3$, $z = 0$ y $z = 1$ se consideran condiciones de tipo Neumann cero. Para el plano $y = 10$ se tiene Dirichlet cero, y para el plano $y = 0$ se considera Dirichlet cero cuando $x \in (-3, -1) \cup (1, 3)$, mientras que cuando $x \in [-1, 1]$ se tiene condiciones

de tipo Dirichlet igual a la constante T_0, conocida como la temperatura concentrada. En este caso, se toma $T_0 = 1$ y en la Tabla 6–15 se presentan los materiales para cada región. Finalmente la fuente corresponde a $f = 0$.

Tabla 6–15: Descripción de la difusión en las regiones

	Conductividad térmica (W/m K)					
Región	k_{11}	k_{12}	k_{22}	k_{13}	k_{23}	k_{33}
1	44.01	11.91	85.28	0.0	0.0	1.0
2	76.56	20.63	52.73	0.0	0.0	1.0
3	30.65	3.37	28.82	0.0	0.0	1.0
4	83.61	18.12	20.84	0.0	0.0	1.0
5	33.67	0.0	33.67	0.0	0.0	1.0
6	52.73	20.63	76.56	0.0	0.0	1.0
7	28.82	3.37	30.65	0.0	0.0	1.0
8	83.61	18.12	20.84	0.0	0.0	1.0
9	33.67	0.0	33.67	0.0	0.0	1.0
10	85.28	11.91	44.01	0.0	0.0	1.0

En [28] se resuelve este problema numéricamente para el problema en 2D, donde la solución obtenida es ilustrada en la Figura 6–17. La solución para el caso de 3D, es presentada en la Figura 6–18.

Para resolver el problema en 3D, se consideran dos mallas formadas por 118237 y 375111 celdas. Además, de los esténciles β_2 y β_{rnd}. Los cuatro sistemas son resueltos utilizando Gradiente Conjugado precondicionado con AMG, C^0-AMG, BC^0-AMG, C^0-ASM y BC^0-ASM, con un número máximo de 250 iteraciones permitidas y una tolerancia relativa de 10^{-13}. En las Tablas 6–16 y 6–17 se presentan las iteraciones realizadas por el PCG, mientras que el tiempo es considerado en las Tablas 6–18 y 6–19.

De manera similar a los ejemplos anteriores, se puede observar que el precondionador BC^0-AMG muestra ser el más eficiente en cuanto al número de iteraciones. Con respecto al tiempo, es el mejor al resolver los problemas grandes, donde el esténcil no es el extendido, por lo que se puede reducir el esténcil de la matriz

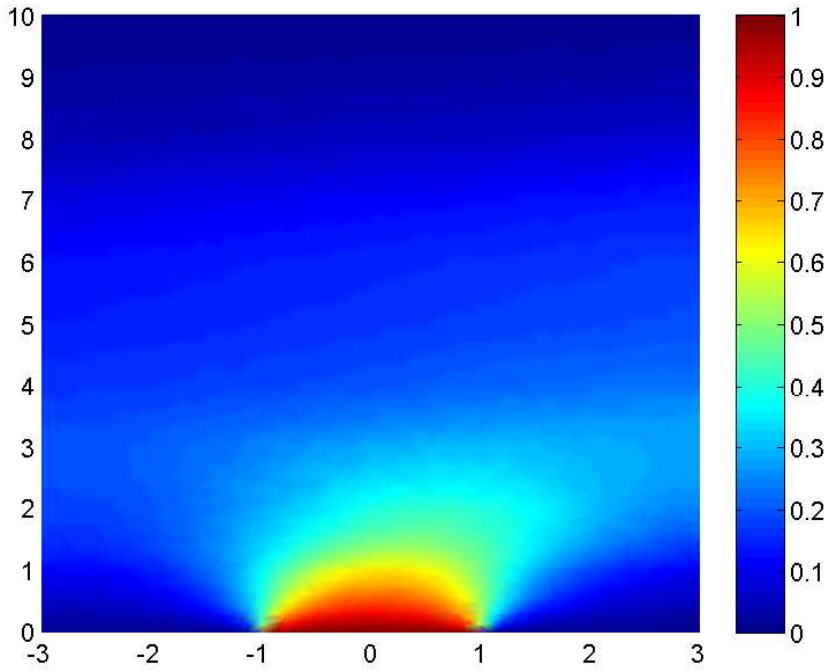


Figura 6–17: Aproximación de la solución en 2D

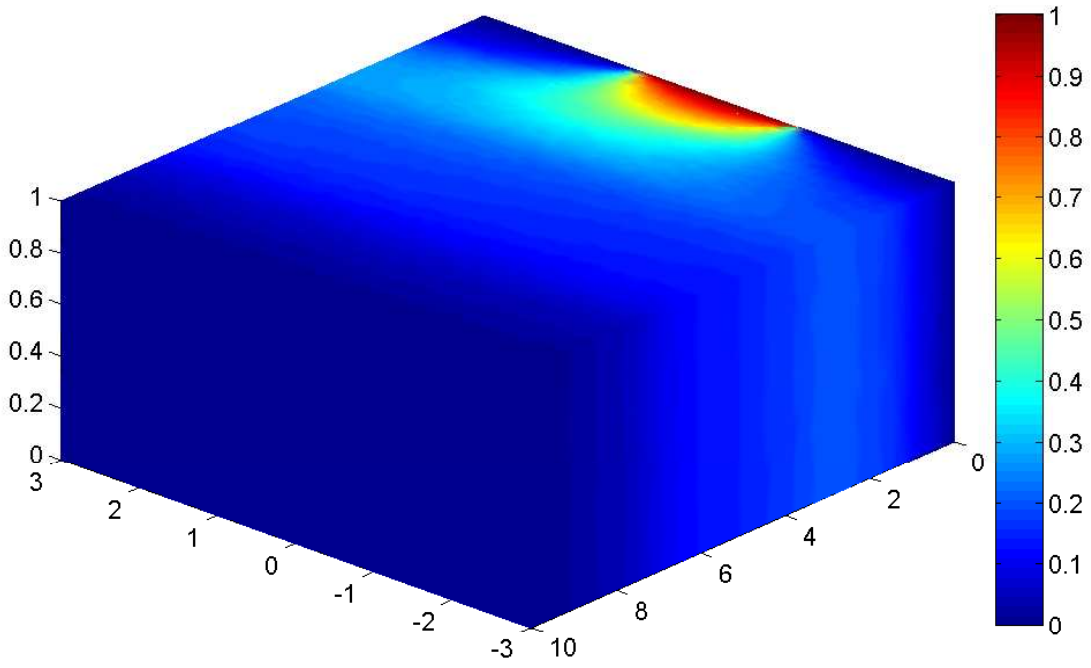


Figura 6–18: Aproximación de la solución en 3D

Tabla 6–16: Número de iteraciones de los precondicionadores AMG

ndofs	μ	β_2		β_{rnd}	
		C^0-AMG	BC^0-AMG	C^0-AMG	BC^0-AMG
472948	4	35	34	250	38
	8	29	28	250	30
	16	23	23	237	24
1500444	4	41	39	250	47
	8	34	33	250	37
	16	28	28	250	29

Tabla 6–17: Número de iteraciones de los precondicionadores ASM

ndofs	τ	β_2		β_{rnd}	
		C^0-ASM	BC^0-ASM	C^0-ASM	BC^0-ASM
472948	C	152	151	250	168
	τ_1	86	84	250	89
	τ_2	68	68	250	70
1500444	C	224	222	250	246
	τ_1	111	106	250	114
	τ_2	91	97	250	103

Tabla 6–18: Tiempo (segs) de los precondicionadores AMG

ndofs	μ	β_2				β_{rnd}			
		C^0-AMG		BC^0-AMG		C^0-AMG		BC^0-AMG	
		setup	resol.	setup	resol.	setup	resol.	setup	resol.
472948	4	1.7	93.7	4.8	166.8	1.4	406.0	3.4	106.1
	8	1.7	130.3	4.8	232.0	1.4	679.7	3.4	141.7
	16	1.7	187.2	4.8	346.4	1.4	1159.1	3.4	206.7
1500444	4	6.8	394.1	26.0	800.6	5.4	1456.5	11.9	524.8
	8	6.8	546.6	26.0	1131.7	5.4	2411.8	11.9	693.6
	16	6.8	815.0	26.0	1737.1	5.4	4356.3	11.9	987.0

de rigidez, y el precondicionador sigue siendo eficiente, lo que en la práctica es el escenario que interesa resolver. Finalmente, en las Figuras 6–19, 6–20 y 6–21, se consideran los historiales de los residuos al resolver la matriz de 1500444 grados de libertad. Mientras que en la Figura 6–22 se presenta el historial de residuos de los

Tabla 6–19: Tiempo (segs) de los precondicionadores ASM

		β_2				β_{rnd}			
		C^0-ASM		BC^0-ASM		C^0-ASM		BC^0-ASM	
ndofs	τ	setup	resol.	setup	resol.	setup	resol.	setup	resol.
472948	C	1.7	392.0	4.8	705.3	1.4	393.0	3.5	449.7
	τ_1	2.0	221.5	5.2	395.3	1.7	393.0	3.9	238.0
	τ_2	2.3	176.7	5.4	319.9	1.9	393.1	4.2	190.6
1500444	C	6.8	2058.4	27.4	4305.5	5.2	1376.6	12.0	2620.1
	τ_1	8.4	1027.5	26.3	2088.8	6.9	1388.8	14.1	1221.0
	τ_2	9.3	844.2	25.1	1891.0	7.6	1387.2	15.0	1100.0

precondicionadores con $\mu = 16$ y $\tau = \tau_2$. Es decir, los que mejores resultados presentan. Se muestra que los precondicionadores BC^0-AMG y BC^0-ASM sobresalen ante los demás. En la Figura 6–23 se considera el precondicionador AMG y ASM que presentan los menores residuos en la Figura 6–22, donde se muestra que los residuos del BC^0-AMG disminuyen con mayor rapidez.

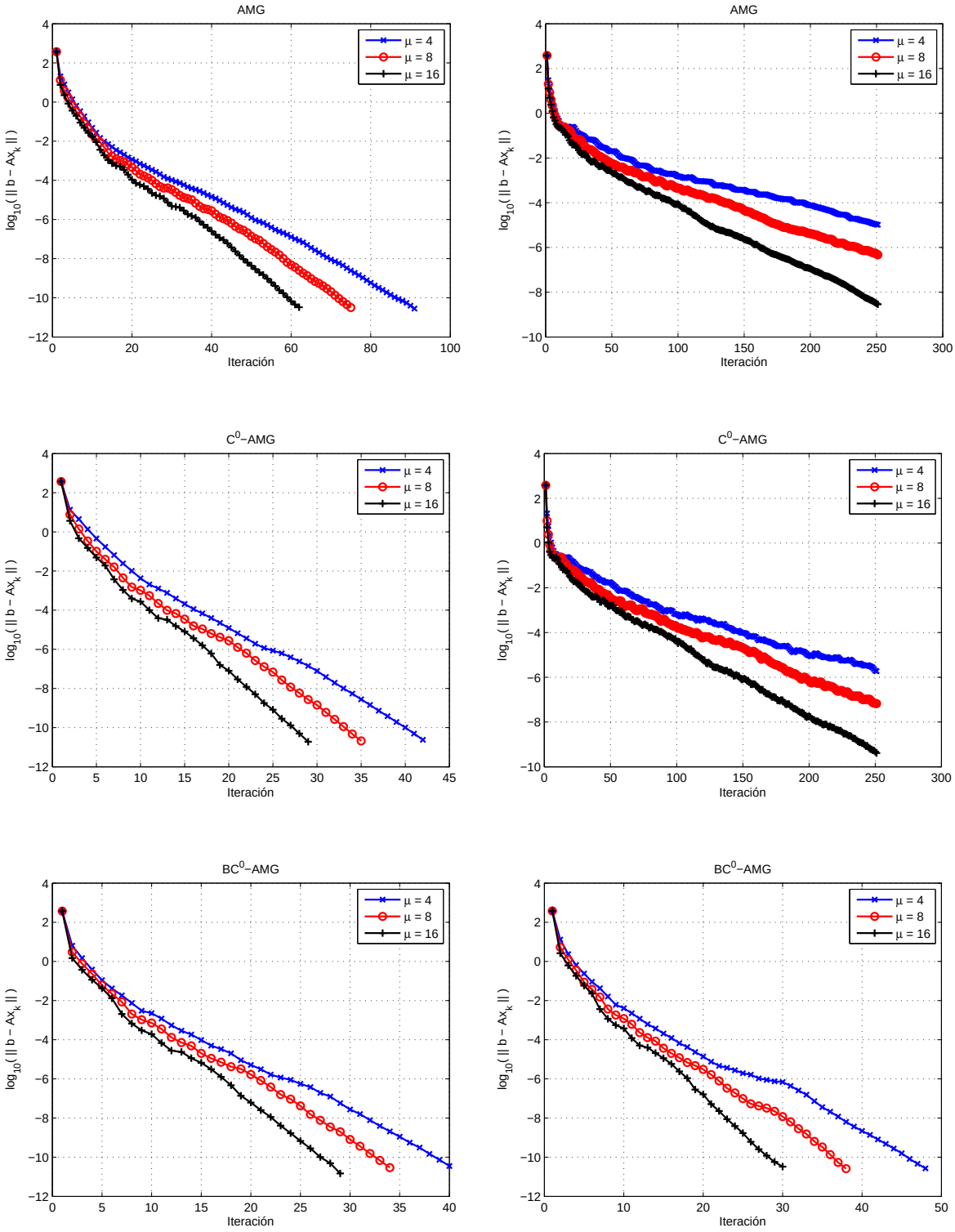


Figura 6–19: Historial de residuos para CG precondicionado con AMG, C^0-AMG y BC^0-AMG, para β_2 (izquierda) y β_{rnd} (derecha)

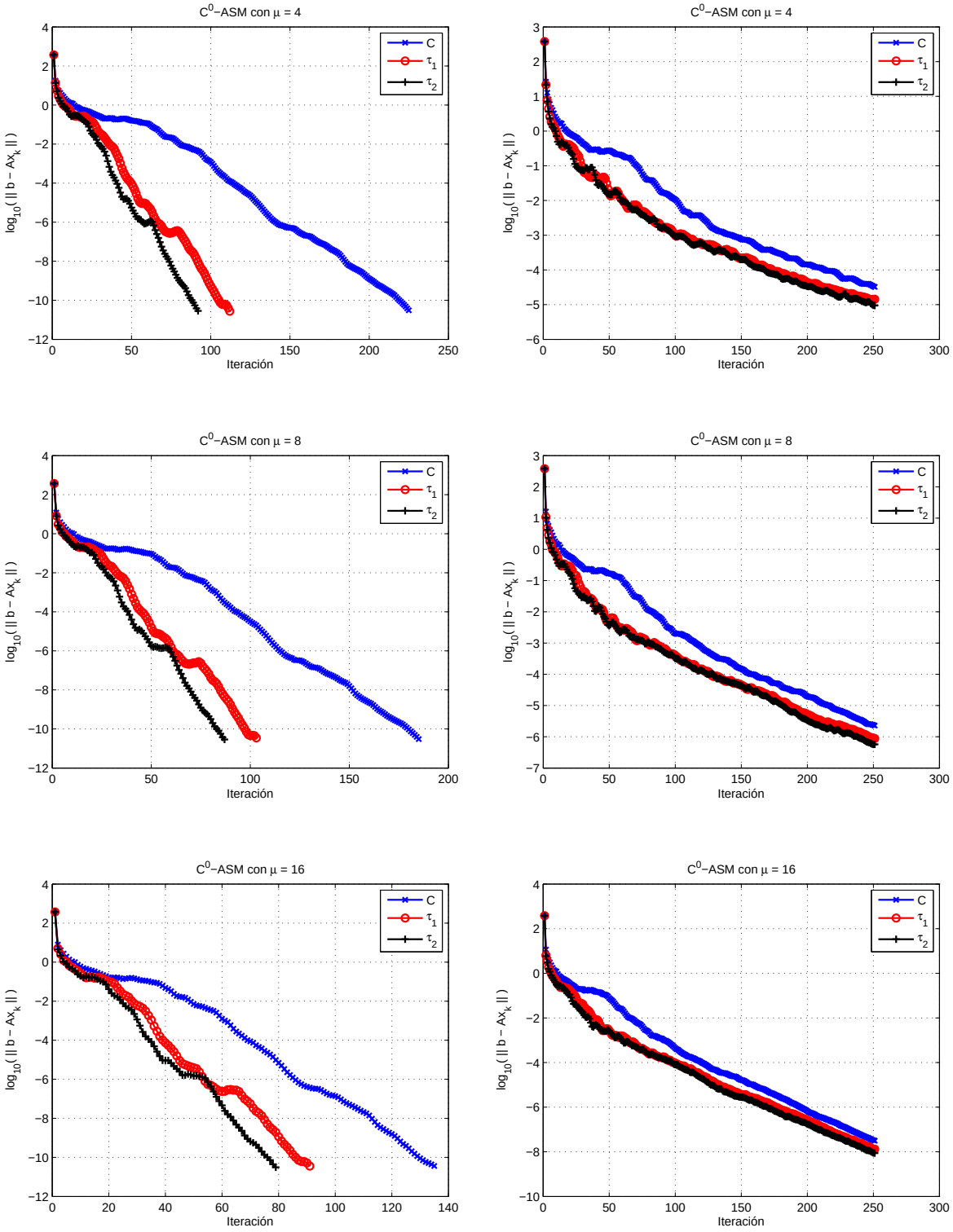


Figura 6–20: Historial de residuos para CG precondicionado con C^0-ASM para $\mu = 4, 8, 16$ y β_2 (izquierda) y β_{rnd} (derecha)

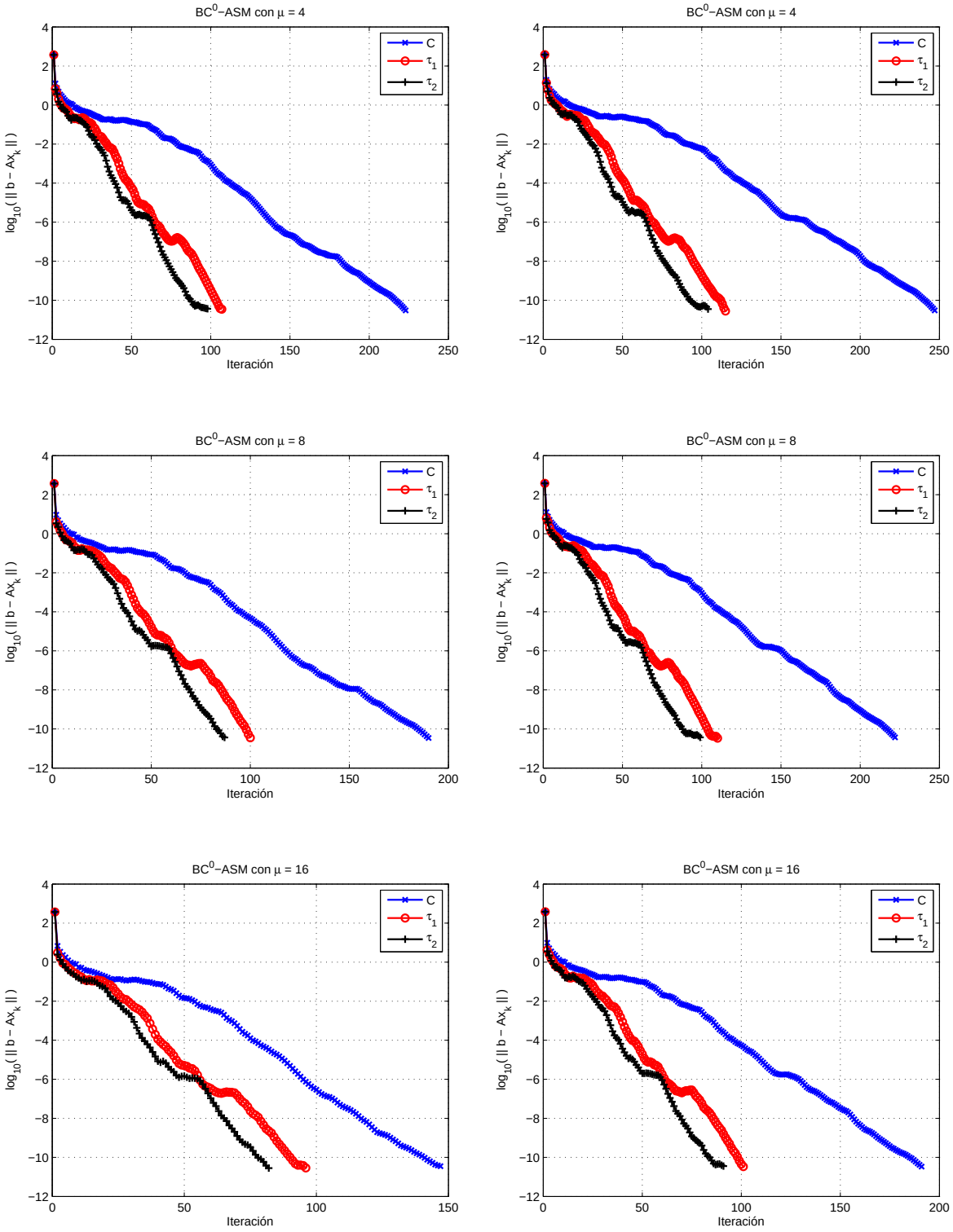


Figura 6–21: Historial de residuos para CG precondicionado con BC^0-ASM para $\mu = 4, 8, 16$ y β_2 (izquierda) y β_{rnd} (derecha)

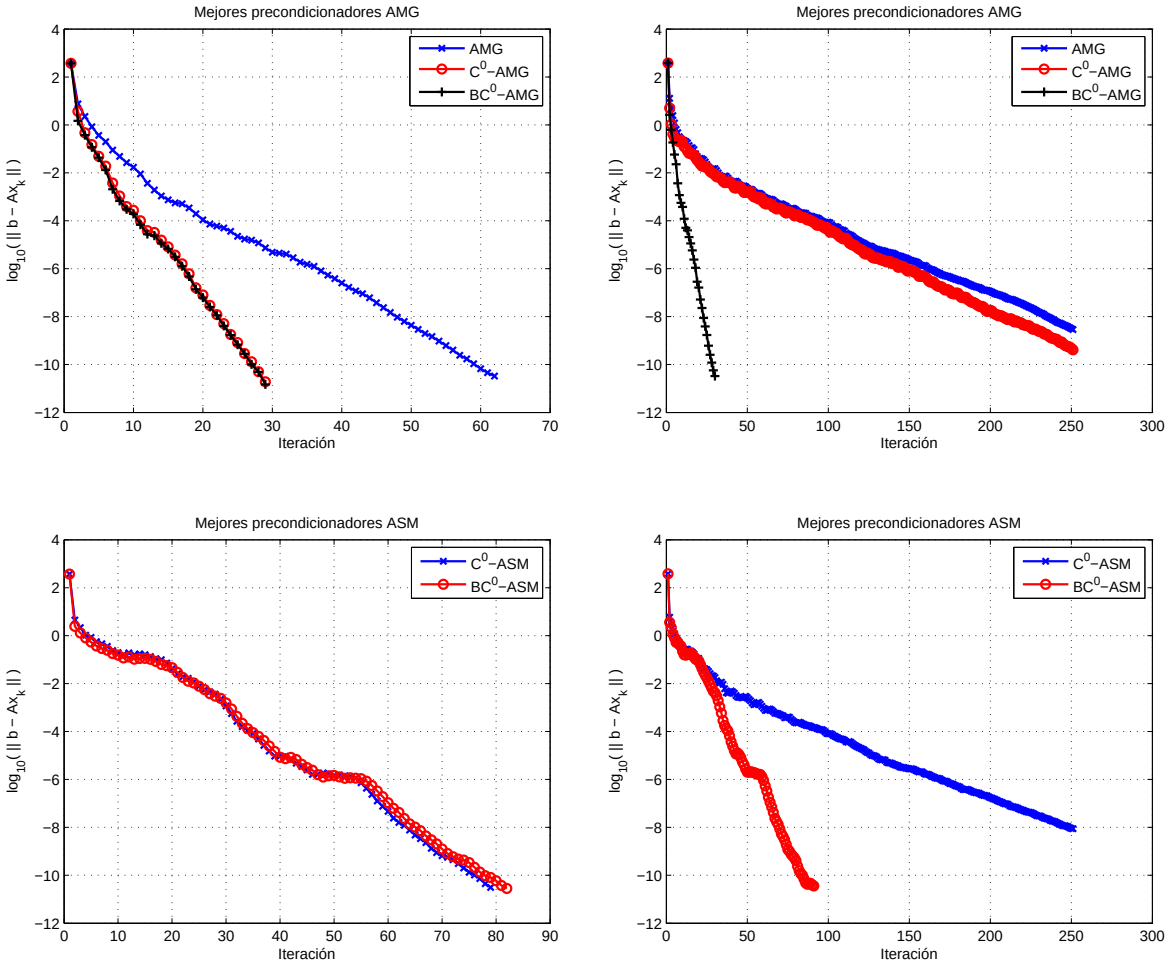


Figura 6–22: Historial de residuos para CG precondicionado con los mejores precondicionadores AMG y ASM, para β_2 (izquierda) y β_{rnd} (derecha)

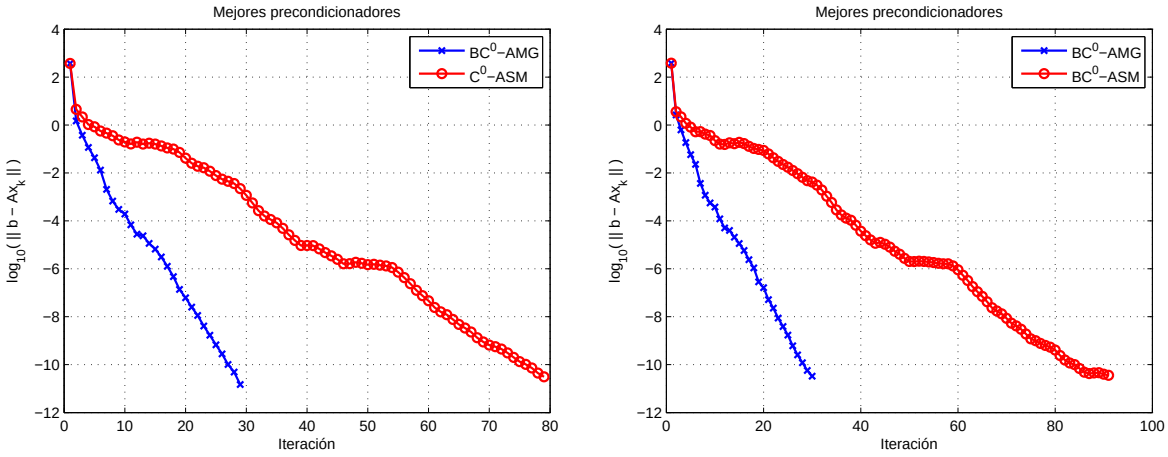


Figura 6–23: Historial de residuos para CG precondicionado con los mejores precondicionadores, para β_2 (izquierda) y β_{rnd} (derecha)

CAPÍTULO 7
CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se han descrito algunas de las estructuras de datos más relevantes para una implementación eficiente del método LDG aplicada a problemas lineales elípticos, utilizando aproximaciones de alto orden y mallas no estructuradas en 3D. Por medio de dichas estructuras, basadas en el paradigma de programación orientada a objectos, es presentada la primera implementación conocida del método LDG para mallas no estructuradas en 3D. La manera en que son descritas las estructuras permite una implementación limpia, eficiente y natural del método. Además, el código desarrollado, posee las siguientes características:

- Resuelve problemas lineales de difusión,
- Permite aproximaciones de alto orden, es decir, que se puede aproximar la solución del problema por polinomios del grado que se deseen, sin afectar la programación,
- No depende de bases particulares, en otras palabras, el código no esta sujeto al uso de una base específica,
- No depende de cuadraturas que poseen una distribución de puntos simétricos,
- La abstracción de clases permite que el usuario puede proveer sus propias bases y cuadraturas,
- Ensamblado del sistema tradicional en su forma mixta y además en su forma reducida mediante el complemento de Schur de manera directa.

Se desarrollaron y estudiaron algunas estrategias para precondicionadores multiniveles algebraicos y semi-algebraicos, con el objetivo de reducir el tiempo y el

número de iteraciones al aproximar la solución del sistema lineal generado de la discretización del método LDG, para aproximaciones de grado uno, cuando se utiliza la base interpolatoria de Lagrange. Por medio de una serie de experimentos numéricos presentados, se corroboraron los estimados de error a priori y comportamiento del condicionamiento espectral en mallas no estructuradas en 3D. Por otro lado, se puede apreciar que las ideas para precondicionar descritas en [8] se pueden extender a dominios en 3D. Además, el uso de suavizadores a nivel de bloques muestran tener mejores resultados que los realizados a nivel puntual.

Con respecto a trabajos futuros, para resolver problemas utilizando aproximaciones de orden variable, es decir, por medio de refinamiento-p, es necesaria una base de polinomios que sea jerárquica, pues para efectos de eficiencia en la implementación, se pueden reducir los cálculos si se trabaja con una base de este tipo, como se menciona en el Capítulo 3. Además, la implementación para mallas no conformes: refinamiento-h, necesita del diseño de una estructura eficiente para recorrer los operadores interiores y exteriores a la cara, ya que para este tipo de mallas, el número de estos operadores se incrementa.

El tener una base de polinomios jerárquica, también puede ayudar a construir precondicionadores multiniveles eficientes para alto orden. Por ejemplo, si se tiene un sistema proveniente de una discretización de grado p, es posible aproximar la discretización de grado $p-1$ por medio de la de grado p, usando la propiedad de que la base usada es jerárquica, y seguir este proceso recursivo hasta reducir el sistema a uno que aproxime la discretización de grado uno. Finalmente, si además de ser jerárquica, la base para polinomios de grado uno tiene la característica de asociar los grados de libertadad a los nodos de la malla, como en la base de Lagrange, entonces podrían utilizarse las técnicas de precondicionamiento descritas en el Capítulo 6.

Se puede extender el código para resolver problemas lineales de convección-difusión. Donde es necesario calcular la contribución del operador de convección al

complemento de Schur, utilizando los operadores previos descritos en este trabajo. Como resultado, se obtiene un sistema lineal no simétrico, por lo que se requiere un método iterativo adecuado a este tipo de sistemas, como por ejemplo, GMRES [29]. Además, de estudiar y desarrollar nuevas técnicas de precondicionamiento no simétricas robustas y eficientes, [26, 30–32].

Finalmente, se puede realizar una implementación paralela del método LDG en 3D. Teniendo en consideración que entre la memoria distribuida y la memoria compartida, la segunda no requiere particionar la malla, lo cual en memoria distribuida provocaría una gran cantidad de comunicaciones al construir los operadores en el elemento de referencia.

REFERENCIAS BIBLIOGRÁFICAS

[1] D.N. Arnold, F. Brezzi, B. Cockburn, and D. Marini. A unified analysis for discontinuous Galerkin methods for elliptic problems. *SIAM J. Num. Anal.*, 39(5):1749–1779, 2002.

[2] P. Castillo. Performance of discontinuous Galerkin methods for elliptic PDE's. *SIAM J. Sci. Comput.*, 24(2):524–547, 2002.

[3] B. Cockburn and C.W. Shu. The Local Discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Num. Anal.*, 35:2440–2463, 1998.

[4] P. Castillo, B. Cockburn, I. Perugia, and D. Schötzau. An a priori error analysis of the Local Discontinuous Galerkin method for elliptic problems. *SIAM J. Num. Anal.*, 38(5):1676–1706, 2000.

[5] I. Perugia and D. Schötzau. An *hp* analysis of the Local Discontinuous Galerkin method for diffusion problems. *J. Scientific Computing.*, 17:561–571, 2002.

[6] R. Bustinza and G.N Gatica. A Local Discontinuous Galerkin method for nonlinear diffusion problems with mixed boundary conditions. *SIAM J. Sci. Comput.*, 26(1):152–177, 2004.

[7] W. Bangerth, R. Hartmann, and G. Kanschat. Deal.II: a general purpose object oriented finite element library. *ACM. Trans. Math. Software.*, 33(4):24:1–24:27, 2007.

[8] P. Castillo and E. Velázquez. A numerical study of a semi-algebraic multilevel preconditioner for the Local Discontinuous Galerkin method. *Internat. J. Numer. Methods Engrg.*, 79:255–268, 2008.

[9] J. Bear. *Dynamics of Fluids in Porous Media.* Dover Publications, 1988.

[10] J.D. Jackson. *Classical Electrodynamics*. Wiley, third edition, 1998.

[11] C. Schwab. *p -and hp-Finite Element Methods*. Oxford Publications, 1998.

[12] P. Castillo. Stencil reduction algorithms for the Local Discontinuous Galerkin method. *Internat. J. Numer. Methods Engrg.*, 81:1475–1491, 2010.

[13] B. Strouttrup. *The C++ Programming Language*. Addison-Wesley, Reading, MA, second edition, 1991.

[14] J.M. Bull, L.A. Smith, L. Pottage, and R. Freeman. Benchmarking Java against C and Fortran for Scientific Applications. *ACM Java Grande/ISCOPE Conference*, pages 97–105, 2001.

[15] R.F. Boisvert, J. Moreira, M. Philippsen, and R. Pozo. Java and Numerical Computing. Technical Report 3, Math. & Comput. Sci. Div., Nat. Inst. of Stand. & Technol., 2001.

[16] M. Anderson, Z. Bai, C. Bischof, S. Blacford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKennes, and D. Sorensen. *LAPACK User's Guide*. Society for Industrial and Applied Mathematics, USA, third edition, 1999.

[17] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, USA, second edition, 2003.

[18] E. Chow and M.A. Heroux. An object-oriented framework for block preconditioning. *ACM Trans. Math. Softw.*, 24:159–183, 1998.

[19] H. Si. TETGEN: A quality tetrahedral mesh generator and three dimensional Delaunay triangulator v.1.3 user's manual. Technical Report 9, Weierstrass Institute for Applied Analysis and Stochastics, Berlin, 2004.

[20] E. Süli and D.F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

[21] M. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[22] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Dover Publications, 2009.

[23] U. Trottenberg, C.W. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2000.

[24] J.W. Ruge and K. Stüben. Algebraic Multigrid (AMG). In S.F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[25] R. Beck. Graph-based algebraic multigrid for Lagrange-type finite elements on simplicial meshes. Technical report, Konrad Zuse Zentrum für Informationstechnik, Berlin, 1999.

[26] Y. Saad. ILUM: a multi-elimination ILU preconditioner for general sparse matrices. *SIAM J. Sci. Comput.*, 17:830–847, 1996.

[27] J. Xu. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing*, 56:215–235, 1996.

[28] C.-C. Ma and S.-W. Chang. Analytical exact solutions of heat conduction problems for anisotropic multi-layered media. *Heat and Mass Transfer*, 47:1643–1655, 2004.

[29] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–859, 1986.

[30] Y. Saad. ILUT: a dual threshold incomplete ILU preconditioner. *Numer. Linear Alg. Appl*, 1:387–402, 1994.

[31] Y. Saad and J. Zhang. BILUTM: A domain-based multi-level block ILUT preconditioner for general sparse matrices. *SIAM J. Matrix Anal. Appl*, 21:279–299, 1998.

[32] Y. Saad and B. Suchomel. ARMS: An Algebraic Recursive Multilevel Solver for general sparse linear systems. Technical report, Num. Lin. Alg. Appl, 1999.

ASPECTOS COMPUTACIONALES DEL MÉTODO "LOCAL DISCONTINUOUS GALERKIN" PARA MALLAS NO ESTRUCTURADAS EN 3D

Filánder de los Ángeles Sequeira Chavarría
filander.sequeira@upr.edu
Departamento de Ciencias Matemáticas
Consejero: Paul Castillo, Ph.D
Grado: Maestría en Ciencias
Fecha de Graduación: Noviembre 2010

En este trabajo se describe una implementación del método "Local Discontinuous Galerkin" (LDG) aplicado a problemas elípticos en 3D. Se discute la implementación de los principales operadores. En particular el uso de aproximaciones de alto orden y de mallas no estructuradas. Estructuras de datos eficientes que permiten un rápido ensamblado del sistema lineal en su formulación mixta son descritas en detalle. Se ha probado que el condicionamiento espectral de la matriz de rigidez muestra un comportamiento asintótico de $\mathcal{O}(h^{-2})$ para mallas estructuradas y no estructuradas, donde h es el tamaño de la malla. Se presentan precondicionadores multiniveles algebraicos y semi-algebraicos usando una base de funciones del tipo interpolatoria de Lagrange. Se muestran experimentos numéricos que ilustran lo eficiente y robustas que son las técnicas propuestas.