Rev. Int. Mét. Num. Cálc. Dis. Ing. (2010) 26: 283-292

Aspectos computacionales del método *Local Discontinuous Galerkin* para mallas no conformes en 2D

Paul E. Castillo

Recibido: Abril 2010, Aceptado: Agosto 2010

Resumen En este artículo se describe una implementación del método *Local Discontinuous Galerkin* (LDG) aplicado a problemas elípticos en 2D para mallas no conformes con elementos rectilíneos. Esta implementación toma ventaja de algunas de las propiedades intrínsecas del método, en particular el uso de aproximaciones de orden variable y de mallas no conformes con un número arbitrario fijo de nodos colgantes. Estructuras de datos eficientes que permiten un rápido ensamblado del sistema lineal en su formulación mixta son descritas en detalle. Esta implementación puede ser utilizada como base para el desarrollo de códigos *hp* adaptativos.

COMPUTATIONAL ASPECTS OF THE *LOCAL DISCONTINUOUS GALERKIN* FOR NON CONFORMING MESHES IN 2D

Summary This article describes an implementation of the *Local Discontinuous Galerkin* (LDG) method applied to a general elliptic boundary value problem in 2D. The implementation takes advantage of some intrinsic properties of the method, in particular, the use of non conforming meshes with an arbitrary but fixed number of hanging nodes and polynomial approximations with variable degree. A detailed description of efficient data structures yielding a fast assembly of the linear system in its mixed formulation is presented. This implementation can be used as a model for *hp* adaptive codes.

Paul E. Castillo
Department of Mathematical Sciences
University of Puerto Rico at Mayagüez
Luis Monzón Building, Room 215
HWY 2, Post 259 North, Mayagüez, Puerto Rico 00681
Tel: 787 832 4040 ext 2661; Fax: 787 265 5454
email: paul.castillo@upr.edu

1. Introducción

En las últimas décadas varios métodos de elemento finito basados en aproximaciones discontinuas han sido propuestos. Estos métodos han ganado mucha popularidad debido a su gran flexibilidad. En primer lugar, al no imponer continuidad entre celdas adyacentes, aproximaciones de orden variable pueden ser utilizadas celda por celda. Por otro lado, estos métodos no se restringen a mallas conformes como las que se usan en los métodos de elemento finito clásicos. Estos permiten mallas con nodos colgantes. En este artículo se discuten varios aspectos computacionales de uno de estos métodos: el *Local Discontinuous Galerkin*, (LDG), propuesto en [13] para problemas de convección-difusión en el caso transitorio. El análisis de estabilidad y convergencia del método para problemas lineales estacionarios fue presentado en [11,16] y en [3] para el caso de difusión no lineal. Un estudio teórico y numérico del condicionamiento espectral de la matriz de rigidez se realizó en [6]. En [7] se comprobó la existencia de puntos de super convergencia en el caso lineal para problemas unidimensionales; y en [12] se analizó la super convergencia del gradiente en mallas Cartesianas en 2D y 3D.

La mayoría de los artículos han sido dedicados al estudio y análisis de las propiedades del método. Desafortunadamente muy poco o casi nada se ha escrito sobre la implementación de los métodos discontinuos y en particular sobre el método LDG. El propósito de este artículo es el de describir una implementación del

método la cual permite aproximaciones de alto orden y el uso de mallas no conformes que poseen un número arbitrario pero fijo de nodos colgantes. Se proponen estructuras de datos eficientes apropiadas para la versión *hp* del método y que permiten un ensamblado rápido del sistema lineal en su forma mixta.

El artículo está organizado como sigue: en la Sección 2 se presenta brevemente la formulación del método aplicado a un problema modelo; en la Sección 3 se discuten los operadores más importantes del método LDG; la Sección 4 está dedicada al cálculo de los operadores de arista y al diseño de las estructuras de datos que dan soporte a estos operadores. En la Sección 5 se presentan algunos experimentos numéricos con el propósito de validar el código y finalmente, se presentan algunas conclusiones en la Sección 6.

2. El método *Local Discontinuous Galerkin*

Para la descripción de las ideas generales del método se considera el siguiente problema elíptico en 2D

$$\begin{aligned} -\nabla\cdot\mathcal{D}\nabla u &= f, \quad \text{en } \Omega,\\ u &= g_{\mathcal{D}}, \quad \text{sobre } \partial\Omega_{\mathcal{D}},\\ \frac{\partial u}{\partial n} &= g_{\mathcal{N}}, \quad \text{sobre } \partial\Omega_{\mathcal{N}}, \end{aligned}$$

donde Ω es un dominio acotado de $\mathbb{R}^2$, $\partial\Omega = \partial\Omega_{\mathcal{D}} \cup \partial\Omega_{\mathcal{N}}$ y $\mathcal{D}$ es una matriz simétrica definida positiva definida a trozos. El método LDG se obtiene, primero, introduciendo una nueva variable, $\boldsymbol{q} = \mathcal{D}\nabla u$, y luego, reformulando el problema elíptico como un sistema de ecuaciones de primer orden en las variables $\boldsymbol{q}$ y u de la siguiente forma

$$\begin{aligned} \mathcal{D}^{-1}\boldsymbol{q} &= \nabla u, \quad \text{en } \Omega,\\ -\nabla\cdot\boldsymbol{q} &= f, \quad \text{en } \Omega,\\ u &= g_{\mathcal{D}}, \quad \text{sobre } \partial\Omega_{\mathcal{D}},\\ \frac{\partial u}{\partial n} &= g_{\mathcal{N}}, \quad \text{sobre } \partial\Omega_{\mathcal{N}}. \end{aligned}$$

La introducción de la nueva variable $\boldsymbol{q}$ no es casual. Por ejemplo, en medios porosos esta variable representa la velocidad del fluido, conocida como la velocidad de Darcy (signo opuesto) y la variable u representa la presión.

Sea $\mathcal{T}_h$ una triangulación de Ω y, sea $\mathcal{V}_h$ y $\mathcal{M}_h$ los espacios de elemento finito definidos por

$$\mathcal{V}_h = \prod_{T\in\mathcal{T}_h} \mathcal{V}_{h,T} \quad \text{y} \quad \mathcal{M}_h = \prod_{T\in\mathcal{T}_h} \mathcal{M}_{h,T},$$

donde $\mathcal{M}_{h,T} = \mathcal{P}_p(T)$, $\mathcal{V}_{h,T} = \mathcal{P}_p(T)^2$ y $\mathcal{P}_p(T)$ es el conjunto de polinomios de grado no mayor que p, con $p \geq 1$. Es importante destacar que la definición de estos espacios no impone ningún tipo de continuidad entre celdas. Esto permite, de manera natural, el uso de aproximaciones polinomiales de grado variable. Sin embargo, en la presente exposición asumimos que el grado es el mismo en todas las celdas que componen la triangulación $\mathcal{T}_h$. La solución $(u_h, \boldsymbol{q}_h) \in \mathcal{M}_h \times \mathcal{V}_h$ del problema discreto de elemento finito satisface para todo $T \in \mathcal{T}_h$ y para todo $(v, \boldsymbol{r}) \in \mathcal{M}_{h,T} \times \mathcal{V}_{h,T}$ las siguientes ecuaciones:

$$\int_T \mathcal{D}_T^{-1}\boldsymbol{q}_h\cdot\boldsymbol{r} \; - \; \oint_{\partial T} \widehat{u_h}_{\{e,T\}}\boldsymbol{r}\cdot\mathbf{n}_T + \int_T u_h\nabla\cdot\boldsymbol{r} = 0, \tag{1}$$

$$\int_T \boldsymbol{q}_h\cdot\nabla v \; - \; \oint_{\partial T} v\widehat{\boldsymbol{q}_h}_{\{e,T\}}\cdot\mathbf{n}_T \; + \oint_{\partial T} v\alpha_e(u_h)\cdot\mathbf{n}_T = \int_T fv, \tag{2}$$

donde $\widehat{u_h}_{\{e,T\}}$ y $\widehat{\boldsymbol{q}_h}_{\{e,T\}}$ son llamados *flujos numéricos*. La definición de estos será discutida en la sección 3.2. El término $\alpha_e(u_h)$ cuya definición se presenta en la sección 4.1 es un término de estabilización el cual garantiza la existencia de la solución del problema discreto. Observe que a diferencia del método clásico de elemento finito la formulación se realiza integrando celda por celda en lugar de integrar sobre todo el dominio de manera global. Por otro lado, esta formulación no se restringe a mallas conformes como las que se utilizan en el método de elemento finito clásico.

3. Sistema lineal en su forma mixta

La formulación discreta obtenida de las ec. (1) y (2) conduce al siguiente sistema lineal para las variables globales $\boldsymbol{q}_h$ y u_h:

$$\begin{pmatrix} D & B\\ -B^T & C \end{pmatrix}\begin{pmatrix} \boldsymbol{q}_h\\ u_h \end{pmatrix} = \begin{pmatrix} b_q\\ b_u \end{pmatrix}. \tag{3}$$

En la práctica se resuelve para la variable primaria (o potencial) u_h. Esto se puede hacer mediante eliminación Gaussiana por bloques, obteniendo el siguiente sistema para el complemento de Schur $\mathcal{S}$:

$$\mathcal{S}u_h = b_u + B^TD^{-1}b_q, \quad \text{donde } \mathcal{S} = C + B^TD^{-1}B.$$

En [11] se demostró que $\mathcal{S}$ es simétrica definida positiva y en [5,6] el autor demostró que su condicionamiento espectral $\kappa(\mathcal{S}) = \mathcal{O}(h^{-2})$ donde $h = \max\{diam(T) : T \in \mathcal{T}_h\}$. El comportamiento asintótico de $\kappa(\mathcal{S})$ es igual al del método de elemento finito clásico. Como se verá en la Sección 3.1 la matriz D puede ser invertida explícitamente, lo cual resulta muy conveniente para la obtención del complemento de Schur.

En esta sección se describe en detalle el cálculo de las matrices D, B y C tanto para mallas conformes como no conformes. 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 $\phi_k(x) = \mathcal{J}_k x + b$ tal que $\phi_k(\hat{T}) = T_k$, donde el Jacobiano $\mathcal{J}_k$ satisface $|\mathcal{J}_k| = \det \mathcal{J}_k > 0$.

Para cada $T_k \in \mathcal{T}_h$ denotamos por $\boldsymbol{\Phi} = \{\phi_i^k\}$ una base para el espacio local de polinomios $\mathcal{M}_{h,T_k}$ (variable primaria o potencial) y por $\boldsymbol{\Psi} = \boldsymbol{\Phi} \times \boldsymbol{\Phi}$ una base para el espacio local de polinomios $\mathcal{V}_{h,T_k}$ (variable secundaria o gradiente). Obsérvese que estas bases pueden ser construidas a partir de bases en el elemento de referencia. Finalmente, se denota por N el número total de celdas en $\mathcal{T}_h$.

3.1. Representación matricial del operador de difusión

Sea $\mathcal{D}_k$ el tensor de difusión de la celda T_k, el cual se asume constante en cada celda. Entonces la representación matricial del operador discreto de difusión relativo a la base $\boldsymbol{\Psi}$ es una matriz diagonal de $N \times N$ bloques. Además, como $\boldsymbol{\Psi}$ 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

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

donde $M = [\int_{\hat{T}} \phi_i \phi_j]$ es la matriz de masa asociada a la celda de referencia $\hat{T}$. Observe que siendo D una matriz diagonal por bloques esta puede ser invertida facilmente. Además la inversa de cada bloque de la diagonal es también un producto Kronecker:

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

De manera más explícita sea

$$\mathcal{D}_k \;=\; \begin{pmatrix} k_{11} & k_{12} \\ k_{11} & k_{12} \end{pmatrix}.$$

Entonces

$$D_{kk}^{-1} \;=\; |\mathcal{J}_T|^{-1} \begin{pmatrix} k_{11}M^{-1} & k_{12}M^{-1} \\ k_{21}M^{-1} & k_{22}M^{-1} \end{pmatrix}$$

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

$$N \dim \mathcal{V}_{h,\hat{T}} \times N \dim \mathcal{V}_{h,\hat{T}} = 2N \dim \mathcal{M}_{h,\hat{T}} \times 2N \dim \mathcal{M}_{h,\hat{T}}$$

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. Representación matricial del operador 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 $\dim \mathcal{V}_{h,T_k} \times \dim \mathcal{M}_{h,T_k}$. El operador gradiente asociado a la celda T_k se obtiene a partir de la siguiente expresión:

$$\int_{T_k} u_h \nabla \cdot \boldsymbol{r} \;-\; \oint_{\partial T_k} \widehat{u_h}_{\{e,T_k\}} \boldsymbol{r} \cdot \mathbf{n}_k. \tag{4}$$

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 ec. (4) puede se calculado como sigue

$$\left[\int_{T_k} \phi_j \nabla \cdot \boldsymbol{\psi}_i\right] \;=\; |\mathcal{J}_k| \begin{bmatrix} (\partial_x \hat{x})\, DX + (\partial_x \hat{y})\, DY \\ (\partial_y \hat{x})\, DX + (\partial_y \hat{y})\, DY \end{bmatrix},$$

donde

$$DX = \left[\int_{\hat{T}} \phi_j \partial_{\hat{x}} \psi_i\right] \quad \text{y} \quad DY = \left[\int_{\hat{T}} \phi_j \partial_{\hat{y}} \psi_i\right].$$

Las matrices DX y DY se calculan solamente para la celda de referencia. Los demás coeficientes pueden ser obtenidos de la información geométrica de la celda T_k,

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

Ahora se extiende el procedimiento utilizado por el autor en [5], para el cálculo de las integrales en la frontera de la celda de la ec. (4), a mallas con nodos colgantes como la que se muestra en la Figura 1.

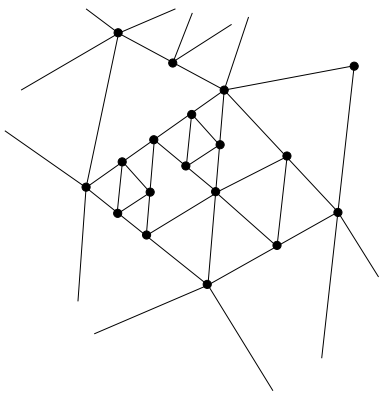

Figura 1. Mallas no conformes con varios nodos colgantes por lado

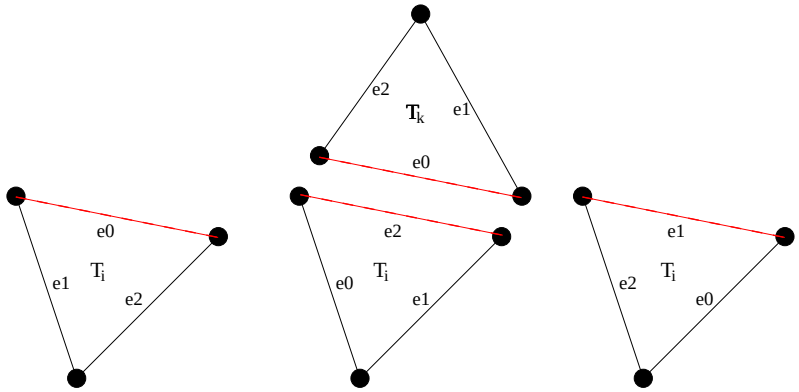

Figura 2. Combinaciones de todas las posiciones relativas

Sea e una arista compartida por las celdas T_k y T_i. Obsérvese que dos celdas pueden compartir más de una arista ya que se permiten mallas no conformes. u_h^k y u_h^i denotan la aproximación de la solución en las celdas T_k y T_i respectivamente. El flujo numérico $\widehat{u_h}_{\{e,T_k\}}$ se define como una combinación convexa de la siguiente manera:

$$\widehat{u_h}_{\{e,T_k\}} = \alpha_i u_h^k + \zeta_i u_h^i,$$

donde $\alpha_i \geq 0$, $\zeta_i \geq 0$ y para garantizar la consistencia del método $\alpha_i + \zeta_i = 1$. La combinación $(\alpha_i, \zeta_i) = (1, 0)$ o $(\alpha_i, \zeta_i) = (0, 1)$ es de gran interés ya que reduce el número de bloques no nulos fuera de la diagonal principal. Esta selección particular de parámetros fue utilizada en [10] donde el autor propone varias estrategias para minimizar el número total de bloques en el complemento de Schur. Para una arista interior e se tiene

$$\oint_e \widehat{u_h}_{\{e,T_k\}} \boldsymbol{r} \cdot \mathbf{n}_k \;=\; \alpha_i \oint_e u_h^k \boldsymbol{r}^k \cdot \mathbf{n}_k \;+\; \zeta_i \oint_e u_h^i \boldsymbol{r}^k \cdot \mathbf{n}_k.$$

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 arista y contribuye exclusivamente al bloque B_{ki} que se encuentra fuera de la diagonal.

La representación matricial correspondiente es

$$\left[\alpha_i \oint_e u_h^k \boldsymbol{r}^k \cdot \mathbf{n}_k\right] \;=\; \alpha_i \mathbf{n}_k \otimes [FI]_e \quad \text{y}$$

$$\left[\zeta_i \oint_e u_h^i \boldsymbol{r}^k \cdot \mathbf{n}_k\right] \;=\; \zeta_i \mathbf{n}_k \otimes [FE]_e$$

donde las matrices $[FI]_e$, operador interior de arista, y $[FE]_e$, operador exterior de arista, son matrices de tamaño $\dim \mathcal{M}_{h,T_k} \times \dim \mathcal{M}_{h,T_k}$ y $\dim \mathcal{M}_{h,T_k} \times \dim \mathcal{M}_{h,T_i}$ respectivamente y están definidas de la siguiente manera

$$[FI]_e(m,n) \;=\; \oint_e \phi_n^k \boldsymbol{\phi}_m^k \;\text{ y }\; [FE]_e(m,n) \;=\; \oint_e \phi_n^i \boldsymbol{\phi}_m^k. \tag{5}$$

Siguiendo la definición de flujo númerico propuesta en [11], se tiene $\widehat{u_h}_{\{e,T_k\}} = g_{\mathcal{D}}$ para cualquier arista de borde e donde se imponen condiciones de Dirichlet. Por lo tanto e no afecta la matriz B. Por otro lado, si se tienen condiciones de tipo Neumann en la arista e se tiene $\widehat{u_h}_{\{e,T_k\}} = u_h$, por lo que se puede fijar $\alpha_i = 1$ y $\zeta_i = 0$.

Como la malla consiste de celdas linealmente equivalentes, todas las integrales pueden ser precalculadas en la celda de referencia. Para el operador FE_e, las funciones de la base son evaluadas en ambos lados de la arista, por lo que se consideran todas las combinaciones posibles de pares de lados en la celda de referencia. En la Figura 2 se presentan todas las posibles combinaciones para la arista marcada en puntillas. Dentro de la celda T_k, la arista corresponde al lado 0, sin embargo dentro de la celda T_i esta arista puede ser etiquetada de tres formas distintas. En el Tabla 1 se resume el número total de combinaciones incluyendo celdas de diferentes tipos. Esta idea fue utilizada por primera vez en [5]. Aquí se extiende esa misma idea a mallas no conformes que poseen un número arbitrario pero fijo de niveles. De esta forma se obtiene una mayor flexibilidad en el código adaptativo.

Tabla 1. Combinaciones en función del tipo de celda

Forma de la celda	Forma del vecino	Número de combinaciones
Triángulo	Triángulo	3×3
Triángulo	Cuadrilátero	3×4
Cuadrilátero	Cuadrilátero	4×4
Cuadrilátero	Triángulo	4×3

De acuerdo a [11] el parámetro de estabilización $\alpha_e(\cdot)$ se define como sigue

$$\alpha_e(u_h) \;=\; \eta_e \left(u_h^k \mathbf{n}_k + u_h^i \mathbf{n}_i\right).$$

De ahí que la matriz de estabilización C es una matriz esparcida de $N \times N$ bloques, donde los bloques C_{kk} y C_{ki} se leen

$$C_{kk} \;=\; \sum_{e \in \partial T_k} \eta_e [FI]_e \quad \text{y} \quad C_{ki} \;=\; -\eta_e [FE]_e$$

La forma para escoger el parámetro de estabilización fue ampliamente discutido en [11] para mallas arbitrarias compuesta por celdas triangulares y en [12] para mallas Cartesianas, cuyas celdas son rectángulos paralelos a los ejes del sistema de referencia.

3.3. Término fuente

El vector global del sistema lineal (3) se descompone en dos vectores: el primero, $b_{\boldsymbol{q}}$, se obtiene a partir de las condiciones de frontera de tipo Dirichlet. Si $e \in \partial T_k \cap \Omega_{\mathcal{D}}$, la contribución de la arista e a este vector $b_{\boldsymbol{q}}$ se obtiene por el vector local

$$\left[\oint_e g_{\mathcal{D}} \boldsymbol{r} \cdot \mathbf{n}_k \right] \;=\; \mathbf{n}_k \otimes \left[\oint_e \phi_m^k g_{\mathcal{D}} \right].$$

El tamaño de este vector es igual a $2 \dim \mathcal{M}_{h,T_k}$. La segunda componente es el vector b_u que depende del término fuente f, así como de ambos tipos de condiciones de frontera. La contribución final de la celda T_k al vector b_u es un vector de dimensión $\dim \mathcal{M}_{h,T_k}$ el cual puede ser calculado de la siguiente manera

$$\left[\int_{T_k} f \phi_m^k \right] + \sum_{e \in \partial T_k \cap \Omega_{\mathcal{D}}} \left[\eta_e \oint_e g_{\mathcal{D}} \phi_m^k \right] + \sum_{e \in \partial T_k \cap \Omega_{\mathcal{N}}} \left[\oint_e g_{\mathcal{N}} \phi_m^k \right].$$

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

4. Estructuras de datos

En esta sección se describen las estructuras de datos y algoritmos más relevantes para nuestra implementación del método LDG en mallas no conformes. Primero se introduce cierta notación y algunas convenciones. A continuación se diferencia el término *lado* del término *arista*. Un polígono está delimitado por n lados. Por ejemplo una celda triangular tiene tres lados, una celda rectangular contiene cuatro. Un lado es la unión de un conjunto de aristas. Denotamos por S_i el i-ésimo lado de una celda. La enumeración se hace en sentido contrario a las manecillas del reloj.

Las técnicas de refinamiento local utilizadas en códigos adaptativos están basadas, por lo general, en la división de celdas triangulares por algún tipo de bisección [2, 14, 15, 17, 18]. Una de las desventajas de esta estrategia es que para mantener conformidad en la malla el proceso de refinamiento no es estrictamente local y puede expandirse fuera de la zonas de interés. Como el método LDG, y en general cualquier método discontinuo, puede utilizar mallas que poseen nodos colgantes, se puede utilizar métodos de refinamiento basados en la descomposición de una celda en cuatro celdas posiblemente congruentes. Sin embargo, las pocas implementaciones existentes, por ejemplo el entorno desarrollado en [1], utilizan lo que se denomina "la regla de la diferencia de nivel 1", la cual establece que la diferencia de nivel entre dos celdas adyacentes es a lo más 1. Esta se puede explicar de la siguiente manera. A cada celda se le asigna un nivel, empezando por 0 en la malla inicial. Al refinar una celda se le asigna a las cuatro celdas nuevas el nivel de la celda madre incrementado en uno. La regla establece que, en cualquier momento, la malla no puede tener dos celdas adyacentes con una diferencia de nivel mayor a 1. Para evitar que esto ocurra se aplica de manera recursiva el refinamiento sobre aquellas celdas adyacentes que violan dicha propiedad. En ocasiones esta regla hace que el procedimiento de refinamiento no sea tan local. En el código desarrollado para este trabajo se consideran diferencias de nivel arbitrarias pero prefijadas por el usuario. Esta flexibilidad permite que el proceso de refinamiento sea más local.

Como se desea un procedimiento de ensamblado rápido, se asume una descomposición regular. Para una diferencia de nivel, (l), cada lado se particiona en 2^l aristas de igual tamaño. $\mathcal{E}_l$ denota el conjunto formado por estas aristas. Una vez que la diferencia de nivel máxima, L_{max} es seleccionada, el conjunto de aristas válidas es la unión de las aristas de nivel l, para $l = 0, \ldots, L_{max}$. El nivel 0 corresponde a una malla conforme (sin nodos colgantes) el tipo de mallas permitidas por los métodos de elemento finito clásicos. La Figura 3 presenta el conjunto de aristas válidas para un nivel de diferencia igual a 2.

4.1. Representación matricial del operador de estabilización

La enumeración de las aristas se realiza de la siguiente manera. Toda arista es representada por un par de índices $e(i, j)$, donde j indica el lado al cual la arista pertenece y el índice i indica su posición local en ese lado. La Figura 4 ilustra la enumeración local utilizada. Un esquema similar fue utilizado para celda de tipo cuadrangular.

Se debe enfatizar que la limitación de sólo permitir posiciones específicas para los nodos colgantes es exclusivamente por razones de desempeño. Como el código

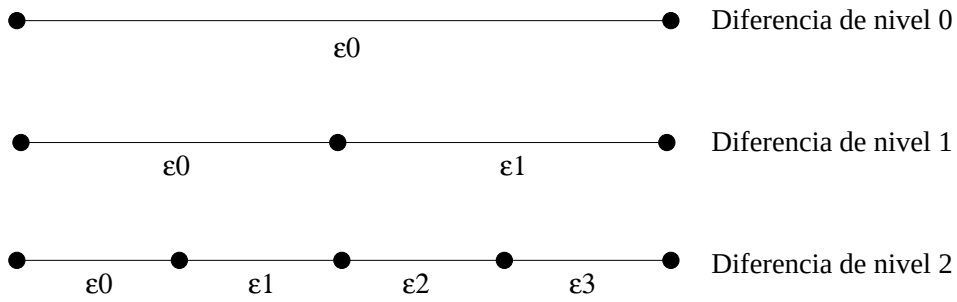

Figura 3. Conjunto de aristas válidas para una diferencia de nivel igual a 2

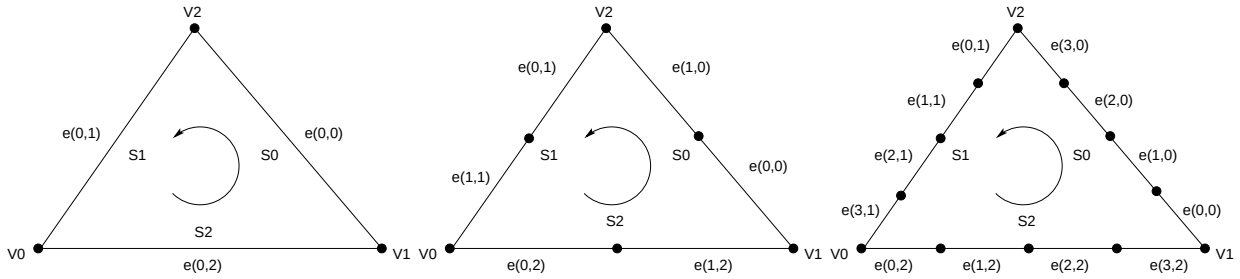

Figura 4. Enumeración de aristas para la celda de nivel 0 (arriba e izquierda), nivel 1 (arriba y derecha) y nivel 2 (abajo)

maneja mallas dinámicas, es decir mallas que pueden cambiar durante el cálculo, se requiere procedimientos eficientes para un ensamblado rápido de los operadores discretos, por lo tanto el uso de operadores precalculados es más eficiente para problemas lineales de convección difusión. Para mallas un poco más generales, donde la posición relativa de los nodos colgantes no se conoce *a priori*, se deben calcular todas las entradas de las matrices FI_e o FE_e para cada arista e. Esto incrementará el tiempo de ensamblado del sistema lineal, en particular, el desempeño se verá severamente afectado en problemas que dependen del tiempo.

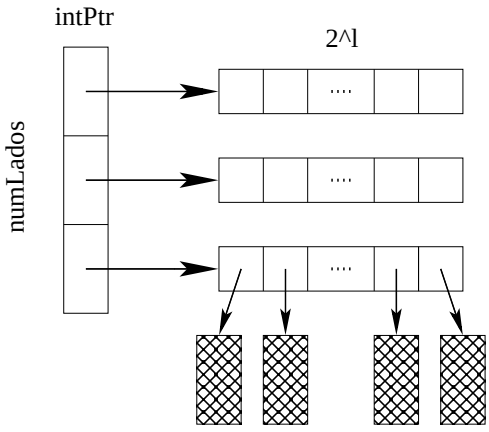

Figura 5. Estructura de datos para una diferencia de nivel dada

4.2. Operador interior de arista

Para los operadores interiores de arista, FI_e, ambas funciones de base son evaluadas por el mismo lado de la celda (celda de referencia). Se tiene un array de punteros $intPtr$, de tamaño $numLados$, cada uno apuntando hacia un array de longitud 2^l. Una entrada en este array apunta hacia uno de las matrices FI_e definida en la ec. (5) para cada arista $e \in \mathcal{E}_l$. Siendo más específico, para cada arista $e(i,j)$ el valor $intPtr[j][i]$ apuntará hacia la matriz asociada a la i-ésima arista del lado j. La Figura 5 ilustra esta estructura de datos. Para cada diferencia de nivel, $l = 0, \ldots, L_{max}$ se almacena una de estas estructuras de datos.

La Figura 6 presenta una posible configuración de celdas con su respectivos niveles. La numeración de la arista α relativo a la celda de nivel 0 es $e(0,j)$ donde j es el número del lado en esa celda. Como la diferencia de niveles entre las celdas que comparten la arista α es 1, la matriz FI_α puede ser obtenida de la estructura de datos correspondiente al de la diferencia de nivel 1, mediante el puntero almacenado en $intPtr[j][0]$. Conforme al esquema de enumeración de aristas presentado en la Figura 4, como la posición local de la arista β es 2 dentro de los lados de nivel 2 su etiqueta es $e(2,j)$. Por otro lado la diferencia de nivel es 2, por lo que la matriz FI_β se obtiene de la estructura de datos asociada a la diferencia de nivel 2 mediante el puntero $intPtr[j][2]$. Obsérvese que en cualquier caso para obtener el ope-

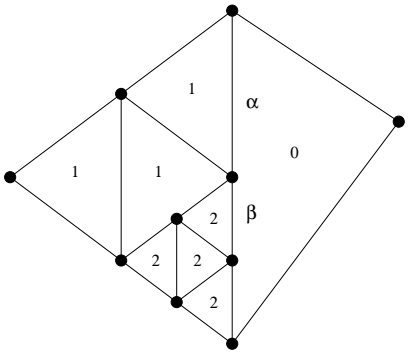

Figura 6. Distribución de niveles

rador, se ha utilizado el número local del lado al cual pertenece la arista así como su posición local en ese lado y la diferencia de nivel. Todas estas cantidades pueden ser obtenidas de la estructura de datos utilizada para almacenar la malla.

4.3. Operador exterior de arista

Para una diferencia de nivel específica se utiliza la estructura de datos ilustrada en la Figura 5 para almacenar los operadores exteriores. Sin embargo se debe crear dos estructuras para cada combinación de par de lados. Esto se explica de la siguiente manera. Considere la arista α de la Figura 6. Contrario al operador FI_α, el operador FE_α requiere de la evaluación de una de las funciones base por la celda de nivel 0 y otra por la celda de nivel 1. La dificultad es que la posición local de la arista relativa a las celdas que comparten esta arista no corresponde al mismo nivel. En la celda que se encuentra a la izquierda de α, el número de la arista es $e(0, j)$, este es relativo a la diferencia de nivel 0; pero en la celda que se encuentra a la derecha, el número es $e(0, j')$ el cual es relativo a la diferencia de nivel 1. Por lo tanto para cada diferencia de nivel l, y para cada combinación de pares de lados, se crea una estructura donde la celda T posee un nivel inferior que el de su vecino K y otra para el caso opuesto. En ambos casos $|T.level - K.level| = l$.

Se utilizan dos clases para los operadores de arista: la clase `EdgeOperatorStruct` que sirve como contenedor para los operadores interiores y exteriores de aristas en un nivel específico l; y, la clase `Operator` la cual encapsula los operadores de arista para todos sus niveles hasta la diferencia de nivel máxima indicada por el usuario.

Como el propósito de la clase `EdgeOperatorStruct` es el de almacenar y calcular de forma númerica las matrices FI_e y FE_e para toda $e \in \mathcal{E}_l$, sólo es necesario uno de estos objetos. Para la creación de dicha estructura de datos se necesita la diferencia de nivel, una base de polinomios y una cuadratura en una dimensión que sea exacta para polinomios de grado doble del máximo grado permitido. La siguiente declaración en C++ muestra la interfaz para esta clase

```
class EdgeOperatorStruct {
public :

   EdgeOperatorStruct();
  ~EdgeOperatorStruct();
   void setup(uint, const Basis2D*,
                            const Quad1D* );

private :

   uint     level_;
   double** intPtr_;
   double** extPtr0_;
   double** extPtr1_;
   friend class Operator;
};
```

Un objeto perteneciente a la clase `Operator` puede ser utilizado para el manejo de los operadores de arista. Esta clase contiene un array de objetos de tipo `EdgeOperatorStruct` para cada una de las diferencias de nivel $l = 0, \ldots, L_{max}$. Tal como se muestra en la siguiente declaración se proveen dos métodos para obtener el operador deseado. Estos métodos requieren la diferencia de nivel entre las celdas que comparten la arista, la cual puede ser negativa, el índice de la arista y su respectivo número de lado local relativo a la celda.

```
class Operator {
public:

    Operator();
    ~Operator();
    void setup(uint, const Basis2D*,
                        const Quad1D*);
    const double* interiorOperator
              (int, uint, uint) const;
    const double* exteriorOperator
          (int, uint, uint, uint) const;

 private:

    uint Lmax_;
    uint optrBlockSize_;
    EdgeOperatorStruct* edgeOptrStruct_;
};
```

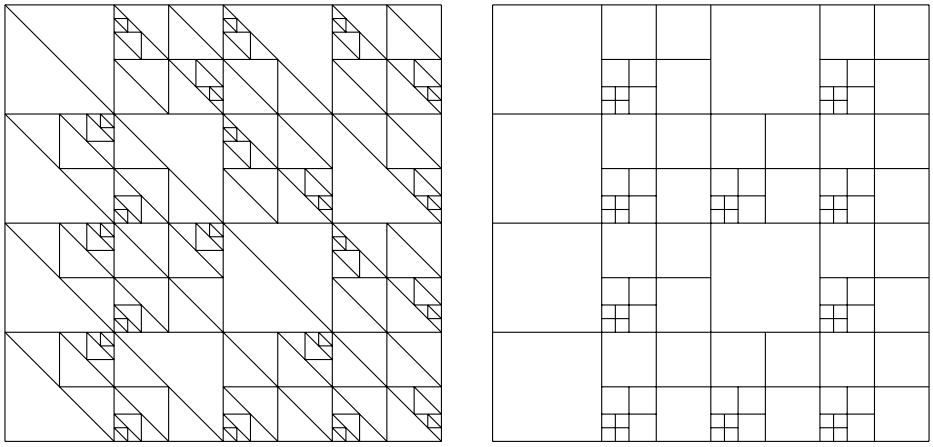

Figura 7. Mallas no conformes con diferencia de nivel variable

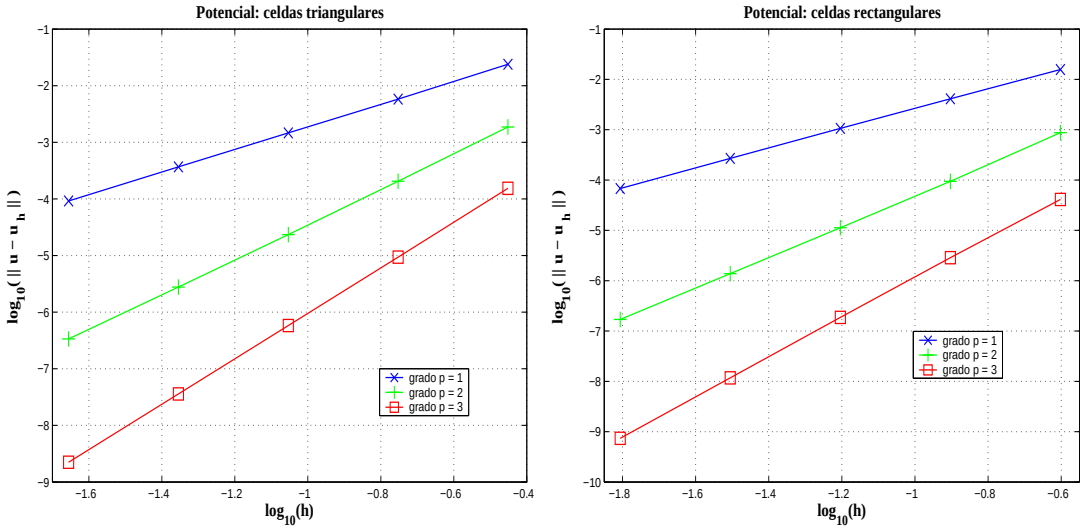

Figura 8. Error en la variable de presión: celdas triangulares (izquierda) y celdas rectangulares (derecha)

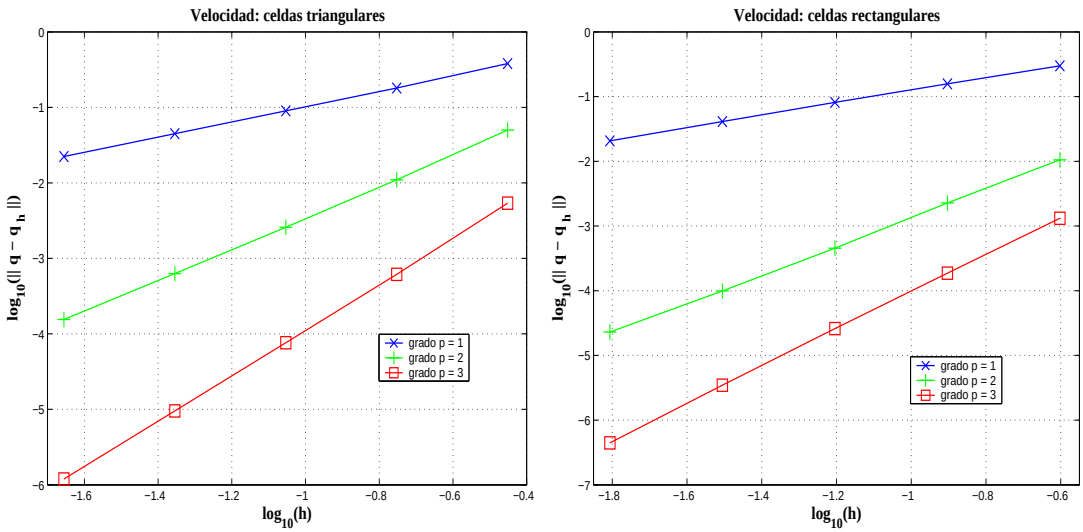

Figura 9. Error en la variable de la velocidad: celdas triangulares (izquierda) y celdas rectangulares (derecha)

4.4. Estructura de datos para la malla

La malla contiene un conjunto de quad-trees cuyos nodos representan las celdas de la triangulación $\mathcal{T}_h$. `Cell` es una clase abstracta de la cual se derivan las clases concretas `Triangle` y `Quadrilateral`. La raíz de cada quad-tree representa una celda de nivel 0; cada nodo que no es una hoja posee cuatro punteros a sus descendientes y, para permitir el recorrido hacia arriba cada nodo que no es raíz posee un puntero a su nodo padre. La clase `Cell` tiene la información de su nivel y sus clases derivadas contienen un array de punteros a objetos de tipo `Edge` de tamaño igual al número de lados.

La malla también contiene un conjunto de árboles binarios cuyos nodos representan las aristas, objetos de la clase `Edge`. Así como las celdas, a cada arista se le asocia un nivel. La raíz de cada uno de estos árboles representa una arista de la malla inicial y el nivel asociado es igual a 0. Cada arista que no es una hoja en el árbol posee dos punteros los cuales apuntan a sus descendientes y cada arista que no es una raíz contiene un puntero a su padre. Además, las aristas interiores poseen dos punteros a celdas que comparten esta arista. En el caso de una arista de borde indicamos la única celda que la contiene y el tipo de condición de borde.

Con estas estructuras de datos, dada una celda, para cada lado, el puntero de arista correspondiente provee la información de las celdas vecinas y de sus niveles haciendo un recorrido del árbol binario asociado a la arista. Además, dada una arista, sus índices i y j, definidos en las secciones 4.2 y 4.3, pueden obtenerse mediante un recorrido del árbol en dirección ascendente. Para acceder cualquier celda o arista en tiempo constante, esta implementación mantiene una lista de hojas de celdas y otra de hojas de aristas.

5. Validación del código

A continuación se presentan algunos experimentos numéricos para validar la implementación descrita previamente, en especial, las tasas de convergencia para varios grados de aproximación en mallas las cuales poseen una diferencia de nivel variable entre celdas adyacentes. Para ello considérese el problema modelo de la ec. (1) en el dominio $\Omega = (0,1)\times(0,1)$ con difusión uniforme $\mathcal{D} = I_d$ y condiciones de frontera homogénea. La función f se escoge de tal forma que la solución exacta sea la función

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

En la Figura 7 se muestran dos mallas no conformes, una que utiliza exclusivamente triángulos y la otra sólo rectángulos. En ambas mallas la diferencia de nivel entre celdas adyacentes varía en el rango $\{0,1,2,3\}$. El problema se resuelve en varias mallas, las cuales se obtienen por refinamiento global partiendo de las mallas que se muestran en la Figura 7. El comportamiento del error en la norma L_2 para la variable del potencial se muestra en la Figura 8 y para la velocidad en la Figura 9. Tal como lo predicen los estimados de error *a priori* analizados en [11], para ambos tipos de mallas, se obtienen tasas de convergencia óptimas de orden $p+1$ para el potencial y de orden p para la velocidad cuando se utilizan polinomios de grado p.

El objetivo principal de este tipo de códigos es para dar soporte a técnicas adaptativas las cuales permiten obtener una mejor aproximación de la solución sin tener que recurrir a un refinamiento global de la malla. Para valorar la calidad de la aproximación ya sea local o global estas técnicas asignan a cada una de las celdas un indicador de error. Si el valor de este es grande entonces el error local, en la celda, es grande, por lo tanto conviene dividir esa celda de acuerdo a algún criterio con la esperanza de reducir el error. Este procedimiento se repite hasta alcanzar la precisión deseada.

Para el método LDG se han propuesto algunas estrategias adaptativas basadas en estimados de error *a posteriori*. Por ejemplo, en [4] se analizó un estimado de error en la norma de la energía, el cual se basa en una descomposición de Helmholtz y en [8,9] se analizó otro en la norma L_2, el cual está basado en el argumento clásico de dualidad de Aubin y Nitsche. Para validar ambas estrategias adaptativas se utilizaron códigos diferentes. El refinamiento local utilizado en [4] no permite una diferencia de nivel mayor a 1 mientras que el utilizado en [9] utiliza la implementación descrita en este artículo.

6. Conclusiones

En este artículo se han descrito algunas de las estructuras de datos más relevantes para una implementación eficiente del método LDG utilizando aproximaciones de alto orden y mallas no conformes en 2D. Esta implementación puede ser utilizada en mallas que tienen un número arbitrario pero fijo de nodos colgantes en cada lado. El diseño de la estructura de datos para almacenar la malla provee los métodos necesarios para obtener la información que el método requiere. Estas estructuras de datos también pueden ser utilizadas en otros métodos de elemento finito discontinuos. Además, como estas no dependen de propiedades intrínsecas de un lenguaje de programación específico pueden ser implementadas en otros lenguajes como Fortran o C. Finalmente se han presentado algunos experimentos nu-

méricos los cuales corroboran que los estimados de error *a priori* pueden ser obtenidos en mallas no conformes que no obedecen necesariamente a la regla de diferencia de nivel 1.

Referencias

1. Bangerth W., Hartmann R., Kanschat G. (2007) Deal.II: a general purpose object oriented finite element library. *ACM. Trans. Math. Software.* 33(4):24:1–24:27.
2. Bnsch E. (1991) Local mesh refinement in 2 and 3 dimenesions. *Impact of Computing in Science and Engineering.* 3:181–191.
3. Bustinza R., Gatica G.N. (2004) A Local Discontinuous Galerkin method for nonlinear diffusion problems with mixed boundary conditions. *SIAM J. Sci. Comput.* 26(1):152–177.
4. Bustinza R., Gatica G.N., Cockburn B. (2005) An *a posteriori* error estimate for the local discontinuous galerkin method applied to linear and non-linear diffusion problems. *J. Scientific Computing.* 22-23:147–185.
5. Castillo P. (2001) *Local Discontinuous Galerkin methods for convection-diffusion and elliptic problems.* PhD thesis. University of Minnesota. Minneapolis. MN. May 2001.
6. Castillo P. (2002) Performance of discontinuous Galerkin methods for elliptic PDE's. *SIAM J. Sci. Comput.* 24(2):524–547.
7. Castillo P. (2003) A superconvergence result for the Local Discontinuous Galerkin method applied to elliptic problems. *Comput. Methods Appl. Mech. Engrg.* 41-42:4675–4685.
8. Castillo P. (2005) An *a posteriori* error estimate for the Local Discontinuous Galerkin method. *J. Scientific Computing.* 22-23:187–204.
9. Castillo P. (2008) An adaptive strategy for the Local Discontinuous Galerkin method applied to porous media problems. *Computer Aided Civil and Infrastructure Engineering.* 23:238–252.
10. Castillo P. (2010) Stencil reduction algorithms for the Local Discontinuous Galerkin method. *Internat. J. Numer. Methods Engrg.* 81:1475–1491.
11. Castillo P., Cockburn B., Perugia I., Schtzau D. (2000) An a priori error analysis of the Local Discontinuous Galerkin method for elliptic problems. *SIAM J. Num. Anal.* 38(5):1676–1706.
12. Cockburn B., Kanschat G., Perugia I., Schtzau D. (2001) Superconvergence of the Local Discontinuous Galerkin method for elliptic problems on Cartesian grids. *SIAM J. Num. Anal.* 39(1):264–285.
13. Cockburn B., Shu C.W. (1998) The Local Discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Num. Anal.* 35:2440–2463.
14. Hempel D. (1993) Local mesh refinement in two space dimensions. *Impact of Computing in Science and Engineering.* 5:181–191.
15. Kossaczk I. (1994) A recursive approach to local mesh refinement in two and three dimensions. *J. Comput. Applied. Math.* 55:275–288.
16. Perugia I., Schtzau D. (2002) An *hp* analysis of the Local Discontinuous Galerkin method for diffusion problems. *J. Scientific Computing.* 17:561–571.
17. Rivara M.C. (1984) Mesh refinement processes based on the generalized bisection for simplices. *SIAM J. Num. Anal.* 21:604–613.
18. Sewell E.G. (1979) A finite element program with automatic user-controlled mesh grading. In R. Vichnevetsky and R.S. Stepleman Eds., *Advances in Computer Methods for Partial Differential Equations III.* IMACS:8–10. New Brunswick, NJ.