Índice

1. Método de Subestructuración

1.1. Complemento de Schur

Consideremos el problema dado por la Ec.

$$\begin{aligned} \mathcal{L}u &= f \quad \text{en } \Omega \\ u &= g \quad \text{en } \partial\Omega \end{aligned} \tag{1}$$

donde

$$\mathcal{L}u = -\nabla \cdot \underline{\underline{a}} \cdot \nabla u + cu \tag{2}$$

como un caso particular del operador elíptico de orden dos.en el dominio Ω, el cual es subdividido en E subdominios Ω_i, $i = 1, 2, ..., E$ sin traslape, es decir

$$\Omega_i \cap \Omega_j = \varnothing \quad \forall i \neq j \quad \text{y} \quad \overline{\Omega} = \bigcup_{i=1}^{E} \overline{\Omega}_i, \tag{3}$$

y al conjunto

$$\Sigma = \bigcup_{i=1}^{E} \Sigma_i, \quad \text{si } \Sigma_i = \partial\Omega_i \backslash \partial\Omega \tag{4}$$

lo llamaremos la frontera interior de los subdominios, un ejemplo se muestra en la figura:

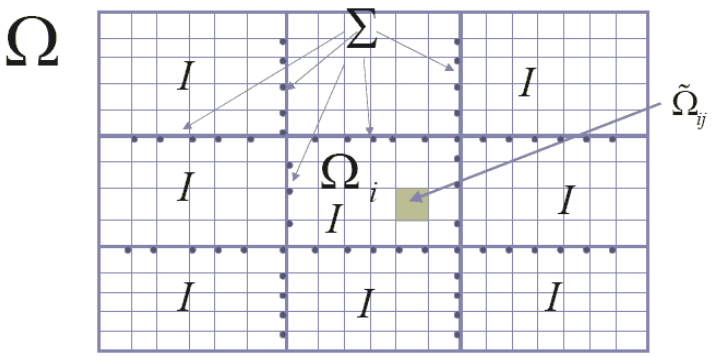

Figura 1: Dominio Ω descompuesto en subdominios Ω_i, con $i = 1, 2, ..., 9$.

Sin perdida de generalidad tomemos $g = 0$ en $\partial\Omega$. Primeramente sea $D \subset H_0^1(\Omega)$ un espacio lineal de funciones de dimensión finita N, en el cual esté definido un producto interior denotado para cada $u, v \in D$ por

$$u \cdot v = \langle u, v \rangle \tag{5}$$

además sean $\tilde{D}_I, \tilde{D}_\Sigma$ y $\bar{D}_\Sigma$ subespacios lineales de D con la propiedad de que $\tilde{D}_\Sigma$ y $\bar{D}_\Sigma$ cada uno por separado sean los complementos algebraicos con respecto a D de $\tilde{D}_I$. Más explícitamente

$$D = \tilde{D}_I + \tilde{D}_\Sigma \quad \text{y} \quad \tilde{D}_I \cap \tilde{D}_\Sigma = \{0\} \tag{6}$$

y

$$D = \tilde{D}_I + \bar{D}_\Sigma \quad \text{y} \quad \tilde{D}_I \cap \bar{D}_\Sigma = \{0\}\,. \tag{7}$$

Adicionalmente $\bar{D}_\Sigma$ es ortogonal a $\tilde{D}_I$, i.e. $\bar{D}_\Sigma$ es el complemento ortogonal de $\tilde{D}_I$, como se muestra en la figura:

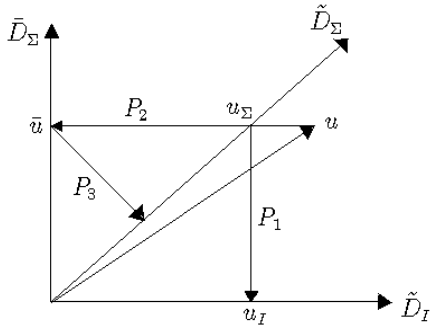

Figura 2: Esquematización de los subespacios $\tilde{D}_I, \tilde{D}_\Sigma$ y $\bar{D}_\Sigma$ del espacio D.

Sean

$$\xi_I = \left\{ w_I^i \mid i = 1, 2, ..., N_I \right\}, \quad \xi_\Sigma = \{ w_\Sigma^\alpha \mid \alpha = 1, 2, ..., N_\Sigma \} \tag{8}$$

bases linealmente independientes de $\tilde{D}_I$ y $\tilde{D}_\Sigma$ respectivamente, tales que

$$\xi = \xi_I \cup \xi_\Sigma \tag{9}$$

sea una base del espacio D, donde $N = N_I + N_\Sigma$ y sin perdida de generalidad podemos suponer que ξ es una base ordenada, es decir, primero están los vectores linealmente independientes de ξ_I y después los de ξ_Σ. Denotaremos al dual algebraico de D por D^*, el cual será el espacio lineal de funciones definidas en D.

Definimos también los operadores proyección P_1, P_2 y P_3 de $\tilde{D}_I, \bar{D}_\Sigma$ y $\tilde{D}_\Sigma$ respectivamente, notemos que podemos definir una biyección entre los subespacios $\bar{D}_\Sigma$ y $\tilde{D}_\Sigma$ por medio de las proyecciones $P_2 : \tilde{D}_\Sigma \to \bar{D}_\Sigma$ y $P_3 : \bar{D}_\Sigma \to \tilde{D}_\Sigma$, al tener la misma cardinalidad y ser complementos algebraicos con respecto a D de $\tilde{D}_I$.

Como trabajaremos con espacios de dimensión finita, cuando $\xi_I \subset \tilde{D}_I$ y $\xi_\Sigma \subset \tilde{D}_\Sigma$ se mantengan fijos, se puede asociar con cada $u \in D$ un único vector

$\underline{u} \equiv (u_I, u_\Sigma)$ donde $u_I = \left(u_{I_1}, ..., u_{I_{N_I}}\right)$ y $u_\Sigma = \left(u_{\Sigma_1}, ..., u_{\Sigma_{N_\Sigma}}\right)$, entonces $\underline{u} = (u_1, ..., u_N) \in \mathbb{R}^N$ tal que

$$u = \sum_{i=1}^{N} u_i w^i \tag{10}$$

donde $w^i \in \xi$ y $u_i \in \mathbb{R}$ para $i = 1, ..., N$. En especial cuando $u_\Sigma \in \tilde{D}_\Sigma$ y $u_I \in \tilde{D}_I$ podemos asociar un único vector $\underline{u_\Sigma} \in \mathbb{R}^{N_\Sigma}$ y $\underline{u_I} \in \mathbb{R}^{N_I}$ respectivamente tal que

$$\underline{u_\Sigma} = \sum_{\alpha=1}^{N} u_\alpha w_\Sigma^\alpha \quad \text{y} \quad \underline{u_I} = \sum_{i=1}^{N} u_i w_I^i. \tag{11}$$

Las funciones de $D \to \mathbb{R}^N$ definidas de está manera son una biyeción a la cual nos referiremos como la *inmersión natural* de $\mathbb{R}^N$ en D. Adicionalmente, las imágenes bajo la inmersión natural de $\tilde{D}_I$ y $\tilde{D}_\Sigma$ son isomorfos a $\mathbb{R}^{N_I}$ y $\mathbb{R}^{N_\Sigma}$ respectivamente, sin perdida de generalidad podemos suponer que la base $(u_1, ..., u_N) \in \mathbb{R}^N$ es una base ordenada, es decir, suponemos que primero aparecen los vectores de $(v_1, ..., v_{N_I}) \in \mathbb{R}^{N_I}$ y después los vectores $(w_1, ..., w_{N_\Sigma}) \in \mathbb{R}^{N_\Sigma}$, entonces la base para $\mathbb{R}^N$ será $(v_1, ..., v_{N_I}, w_1, ..., w_{N_\Sigma})$.

El producto interior Euclidiano en $\mathbb{R}^N$, para cada par $\underline{u} \in \mathbb{R}^N$ y $\underline{v} \in \mathbb{R}^N$, denotado por $\underline{u} \cdot \underline{v}$, será definido por

$$\underline{u} \cdot \underline{v} \equiv \sum_{i=1}^{N} u_i v_i, \tag{12}$$

el cual no necesariamente coincide con el producto interior del espacio D definido en la Ec. (5).

Si adicionalmente suponemos que existe una familia ortogonal $\tilde{D}_{I_i}(\Omega_i)$ de subespacios linealmente independientes del subespacio de $\tilde{D}_I(\Omega)$, con $i = 1, ..., E$, es decir $\left\{\tilde{D}_{I_1}(\Omega_1), ..., \tilde{D}_{I_E}(\Omega_E)\right\}$ tales que

$$\tilde{D}_I = \sum_{i=1}^{E} \tilde{D}_{I_i} \tag{13}$$

y denotamos por $P_{1_i}, i = 1, ..., E$, al operador proyección sobre $\tilde{D}_{I_i}$, entonces

$$P_1 = \sum_{i=1}^{E} P_{1_i} \tag{14}$$

además, para cada $j = 1, ..., E$, sea

$$\xi_{I_j} \equiv \left\{w_I^j \mid i = 1, ..., E\right\} \tag{15}$$

tal que ξ_{I_j} es una base de $\tilde{D}_{I_j}$ (las funciones w_I^j pueden ser las ϕ_i usadas en el método de elemento finito o cualquier otro tipo de funciones base). En adición, asumimos que

$$\xi_I \equiv \bigcup_{j=1}^{E} \xi_{I_j} \equiv \xi_{I_1} \cup \xi_{I_2} \cup ... \cup \xi_{I_E} \tag{16}$$

y que el orden del conjunto $\xi_I \equiv \left\{ w_I^i \mid i = 1, ..., N_I \right\}$ y es la dada por la Ec. (16). Obsérvese la siguiente implicación lógica

$$\text{Si } w_{I_\delta}^i, w_{I_{\delta'}}^{i'} \in \xi_I \quad \text{y} \quad \delta' \neq \delta \Longrightarrow \left\langle w_{I_\delta}^i, w_{I_{\delta'}}^{i'} \right\rangle = 0. \tag{17}$$

Entonces definiendo para toda $\delta = 1, ..., E$, la matriz de $N_\delta \times N_\delta$

$$\underline{\underline{A}}_\delta^{II} \equiv \left[\left\langle w_I^i, w_I^j \right\rangle \right] \tag{18}$$

que sólo esta definida en cada subespacio (subdominio Ω_δ). Entonces, la matriz virtual $\underline{\underline{A}}^{II}$ es dada por la matriz diagonal de la forma

$$\underline{\underline{A}}^{II} \equiv \begin{bmatrix} \underline{\underline{A}}_1^{II} & & & \\ & \underline{\underline{A}}_2^{II} & & \\ & & \ddots & \\ & & & \underline{\underline{A}}_E^{II} \end{bmatrix} \tag{19}$$

donde el resto de la matriz fuera de la diagonal en bloques es cero.

De forma similar definimos

$$\underline{\underline{A}}_\delta^{I\Sigma} \equiv \left[\left\langle w_I^i, w_\Sigma^\alpha \right\rangle \right], \quad \underline{\underline{A}}_\delta^{\Sigma I} \equiv \left[\left\langle w_\Sigma^\alpha, w_I^i \right\rangle \right] \tag{20}$$

y

$$\underline{\underline{A}}_\delta^{\Sigma\Sigma} \equiv \left[\langle w_\Sigma^\alpha, w_\Sigma^\alpha \rangle \right] \tag{21}$$

para toda $\delta = 1, ..., E$, obsérvese que $\underline{\underline{A}}_\delta^{I\Sigma} = \left(\underline{\underline{A}}_\delta^{\Sigma I} \right)^T$. Entonces las matrices virtuales $\underline{\underline{A}}^{\Sigma I}, \underline{\underline{A}}^{\Sigma I}$ y $\underline{\underline{A}}^{\Sigma\Sigma}$ quedarán definidas como

$$\underline{\underline{A}}^{I\Sigma} \equiv \begin{bmatrix} \underline{\underline{A}}_1^{I\Sigma} \\ \underline{\underline{A}}_2^{I\Sigma} \\ \vdots \\ \underline{\underline{A}}_E^{I\Sigma} \end{bmatrix} \tag{22}$$

$$\underline{\underline{A}}^{\Sigma I} \equiv \begin{bmatrix} \underline{\underline{A}}_1^{\Sigma I} & \underline{\underline{A}}_2^{\Sigma I} & \cdots & \underline{\underline{A}}_E^{\Sigma I} \end{bmatrix} \tag{23}$$

y

$$\underline{\underline{A}}^{\Sigma\Sigma} \equiv \left[\sum_{i=1}^{E} \underline{\underline{A}}_i^{\Sigma\Sigma} \right] \tag{24}$$

donde $\left[\sum_{i=1}^{E} \underline{\underline{A}}_i^{\Sigma\Sigma}\right]$ es construida sumando las $\underline{\underline{A}}_i^{\Sigma\Sigma}$ según el orden de los nodos globales versus los nodos locales.

También consideremos al vector $\underline{u} \equiv (u_1, u_E)$ el cual puede ser escrito como $\underline{u} = (\underline{u_I}, \underline{u_\Sigma})$ donde $\underline{u_I} = (u_1, ..., _{N_I})$ y $\underline{u_\Sigma} = (u_1, ..., _{N_\Sigma})$.

Así, el sistema virtual

$$\begin{aligned} \underline{\underline{A}}^{II}\underline{u_I} + \underline{\underline{A}}^{I\Sigma}\underline{u_\Sigma} &= \underline{b_I} \\ \underline{\underline{A}}^{\Sigma I}\underline{u_I} + \underline{\underline{A}}^{\Sigma\Sigma}\underline{u_\Sigma} &= \underline{b_\Sigma} \end{aligned} \tag{25}$$

quedando expresado como

$$\begin{bmatrix} \underline{\underline{A}}_1^{II} & & \\ & \ddots & \\ & & \underline{\underline{A}}_E^{II} \end{bmatrix} \begin{bmatrix} \underline{u_I}_1 \\ \vdots \\ \underline{u_I}_E \end{bmatrix} + \begin{bmatrix} \underline{\underline{A}}_1^{I\Sigma} \\ \vdots \\ \underline{\underline{A}}_E^{I\Sigma} \end{bmatrix} \begin{bmatrix} \underline{u_\Sigma}_1 \\ \vdots \\ \underline{u_\Sigma}_E \end{bmatrix} = \begin{bmatrix} \underline{b_I}_1 \\ \vdots \\ \underline{b_I}_E \end{bmatrix}$$

$$\begin{bmatrix} \underline{\underline{A}}_1^{\Sigma I} & \cdots & \underline{\underline{A}}_E^{\Sigma I} \end{bmatrix} \begin{bmatrix} \underline{u_I}_1 \\ \vdots \\ \underline{u_I}_E \end{bmatrix} + \begin{bmatrix} \underline{\underline{A}}^{\Sigma\Sigma} \end{bmatrix} \begin{bmatrix} \underline{u_\Sigma}_1 \\ \vdots \\ \underline{u_\Sigma}_E \end{bmatrix} = \begin{bmatrix} \underline{b_\Sigma}_1 \\ \vdots \\ \underline{b_\Sigma}_E \end{bmatrix}$$

o más compactamente como $\underline{\underline{A}}\underline{u} = \underline{b}$.

Si del sistema anterior eliminamos $\underline{u_I}$ nos queda

$$\left(\underline{\underline{A}}^{\Sigma\Sigma} - \underline{\underline{A}}^{\Sigma I}\left(\underline{\underline{A}}^{II}\right)^{-1}\underline{\underline{A}}^{I\Sigma}\right)\underline{u_\Sigma} = \underline{b_\Sigma} - \underline{\underline{A}}^{\Sigma I}\left(\underline{\underline{A}}^{II}\right)^{-1}\underline{b_I} \tag{26}$$

a la matriz

$$\underline{\underline{S}} = \underline{\underline{A}}^{\Sigma\Sigma} - \underline{\underline{A}}^{\Sigma I}\left(\underline{\underline{A}}^{II}\right)^{-1}\underline{\underline{A}}^{I\Sigma} \tag{27}$$

se le llama el complemento de Schur global.

En nuestro caso, tenemos definidas las matrices $\underline{\underline{A}}_i^{\Sigma\Sigma}, \underline{\underline{A}}_i^{\Sigma I}, \underline{\underline{A}}_i^{I\Sigma}$ y $\underline{\underline{A}}_i^{II}$ de manera local, por ello definimos el complemento de Schur local como

$$\underline{\underline{S}}_i = \underline{\underline{A}}_i^{\Sigma\Sigma} - \underline{\underline{A}}_i^{\Sigma I}\left(\underline{\underline{A}}_i^{II}\right)^{-1}\underline{\underline{A}}_i^{I\Sigma} \tag{28}$$

adicionalmente definimos

$$\underline{b}_i = \underline{b_\Sigma}_i - \underline{\underline{A}}_i^{\Sigma I}\left(\underline{\underline{A}}_i^{II}\right)^{-1}\underline{b_I}_i \tag{29}$$

en cada subespacio $i = 1, 2, ..., E$. Notemos que las matrices $\underline{\underline{A}}_i^{\Sigma\Sigma}, \underline{\underline{A}}_i^{\Sigma I}, \underline{\underline{A}}_i^{I\Sigma}$ son matrices dispersas y $\underline{\underline{A}}_i^{II}$ es una matriz bandada.

El sistema dado por la Ec. (26) lo escribimos como

$$\underline{\underline{S}}\underline{u_\Sigma} = \underline{b} \tag{30}$$

y queda definido de manera virtual a partir de

$$\left[\sum_{i=1}^{E} \underline{\underline{S}}_i\right]\underline{u_\Sigma} = \left[\sum_{i=1}^{E} \underline{b}_i\right] \tag{31}$$

donde $\left[\sum_{i=1}^{E} \underline{\underline{S}}_i\right]$ y $\left[\sum_{i=1}^{E} \underline{b}_i\right]$ podrían ser construida sumando las S_i y b_i respectivamente según el orden de los nodos globales versus los nodos locales.

El sistema lineal virtual obtenido de esta forma (30) se resuelve eficientemente usando el método de gradiente conjugado, para ello no es necesario construir la matriz $\underline{\underline{S}}$ con las contribuciones de cada S_i co-rrespondientes al subdominio i, lo que hacemos es pasar a cada subdominio el vector $\underline{u_\Sigma}^i$ correspondiente a la i-ésima iteración del método de gradiente conjugado para que en cada subdominio se evalué $\underline{\tilde{u}_\Sigma}^i = \underline{\underline{S}}_i \underline{u_\Sigma}^i$ localmente y con el resultado se forma el vector $\underline{\tilde{u}_\Sigma} = \sum_{i=1}^{E} \underline{\tilde{u}_\Sigma}^i$ y se continué con los demás pasos del método. Esto es ideal para una implementación en paralelo del método de gradiente conjugado.

Observación 1 *Notemos que el normalmente las matrices locales* $\underline{\underline{S}}_i$ *y* $\left(\underline{\underline{A}}_i^{II}\right)^{-1}$ *no se construyen, ya que estas serian matrices densas y su construcción es computacionalmente muy costosa. Y como sólo nos interesa el producto* $\underline{\underline{S}}\underline{y_\Sigma}$*, o más precisamente* $\left[\sum_{i=1}^{E} \underline{\underline{S}}_i\right] \underline{y_\Sigma}$*, entonces si llamamos* $\underline{y_\Sigma}_i$ *al vector correspondiente al subdominio* i*, entonces tendremos*

$$\underline{z} = \left(\underline{\underline{A}}^{\Sigma\Sigma} - \underline{\underline{A}}^{\Sigma I}\left(\underline{\underline{A}}^{II}\right)^{-1}\underline{\underline{A}}^{I\Sigma}\right)\underline{y_\Sigma}_i. \tag{32}$$

Para evaluar eficientemente esta expresión, realizamos las siguientes operaciones equivalentes

$$\begin{aligned} \underline{x1} &= \underline{\underline{A}}^{\Sigma\Sigma}\underline{y_\Sigma}_i \\ \underline{x2} &= \left(\underline{\underline{A}}^{\Sigma I}\left(\underline{\underline{A}}^{II}\right)^{-1}\underline{\underline{A}}^{I\Sigma}\right)\underline{y_\Sigma}_i \\ \underline{z} &= \underline{x1} - \underline{x2} \end{aligned} \tag{33}$$

la primera y tercera expresión no tienen ningún problema en su evaluación, para la segunda expresión tendremos que hacer

$$\underline{x3} = \underline{\underline{A}}^{I\Sigma}\underline{y_\Sigma}_i \tag{34}$$

con este resultado intermedio, deberíamos calcular

$$\underline{x4} = \left(\underline{\underline{A}}^{II}\right)^{-1}\underline{x3} \tag{35}$$

pero como no contamos con $\left(\underline{\underline{A}}^{II}\right)^{-1}$*, entonces multiplicamos la expresión por* $\underline{\underline{A}}^{II}$ *obteniendo*

$$\underline{\underline{A}}^{II}\underline{x4} = \underline{\underline{A}}^{II}\left(\underline{\underline{A}}^{II}\right)^{-1}\underline{x3} \tag{36}$$

al simplificar, tenemos

$$\underline{\underline{A}}^{II}\underline{x4} = \underline{x3}. \tag{37}$$

Esta última expresión puede ser resuelta usando Factorización LU o Gradiente Conjugado (cada una de estas opciones tiene ventajas y desventajas que

deben ser evaluadas al momento de implementar el código para un problema particular). Una vez obtenido $\underline{x3}$*, podremos calcular*

$$\underline{x2} = \underline{\underline{A}}^{\Sigma I} \underline{x3} \tag{38}$$

completando la secuencia de operaciones necesaria para obtener $\underline{\underline{S}}_i \underline{y_\Sigma}$.

Una vez resuelto el sistema de la Ec. (31) en el que hemos encontrado la solución para los nodos de la frontera interior $\underline{u_\Sigma}$, entonces debemos resolver localmente los $\underline{u_{I_i}}$ correspondientes a los nodos interiores para cada subespacio Ω_i, para esto empleamos

$$\underline{u_{I_i}} = \left(\underline{\underline{A}}_i^{II}\right)^{-1} \left(\underline{b_{I_i}} - \underline{\underline{A}}_i^{I\Sigma} \underline{u_{\Sigma_i}}\right) \tag{39}$$

para cada $i = 1, 2, ..., E$, quedando así resuelto el problema $\underline{\underline{A}}\,\underline{u} = \underline{b}$ tanto en los nodos interiores $\underline{u_{I_i}}$ como en los de la frontera interior $\underline{u_{\Sigma_i}}$ correspondientes a cada subespacio Ω_i.

Observación 2 *En la evaluación de* $\underline{u_{I_i}} = \left(\underline{\underline{A}}_i^{II}\right)^{-1} \left(\underline{b_{I_i}} - \underline{\underline{A}}_i^{I\Sigma} \underline{u_{\Sigma_i}}\right)$*, esta nuevamente involucrado* $\left(\underline{\underline{A}}_i^{II}\right)^{-1}$*, por ello deberemos de usar el siguiente procedimiento para evaluar eficientemente esta expresión, realizando las siguientes operaciones equivalentes*

$$\begin{aligned} \underline{x4} &= \underline{b_{I_i}} - \underline{\underline{A}}_i^{I\Sigma} \underline{u_{\Sigma_i}} \\ \underline{u_{I_i}} &= \left(\underline{\underline{A}}_i^{II}\right)^{-1} \underline{x4} \end{aligned} \tag{40}$$

multiplicando por $\underline{\underline{A}}_i^{II}$ *a la última expresión, obtenemos*

$$\underline{\underline{A}}_i^{II} \underline{u_{I_i}} = \underline{\underline{A}}_i^{II} \left(\underline{\underline{A}}_i^{II}\right)^{-1} \underline{x4} \tag{41}$$

simplificando, tenemos

$$\underline{\underline{A}}_i^{II} \underline{u_{I_i}} = \underline{x4} \tag{42}$$

esta última expresión puede ser resuelta usando Factorización LU o Gradiente Conjugado.

Como se indico en las dos últimas observaciones, para resolver el sistema $\underline{\underline{A}}_i^{II} \underline{x} = \underline{b}$ podemos usar Factorización LU, Gradiente Conjugado o cualquier otro método para resolver sistemas lineales, pero deberá de usarse aquel que proporcione la mayor velocidad en el cálculo o que consuma la menor cantidad de memoria (ambas condicionantes son mutuamente excluyentes), por ello la decisión de que método usar deberá de tomarse al momento de tener que resolver un problema particular en un equipo dado.

Si la matriz $\underline{\underline{A}}_i^{II}$ es de banda b, para usar el método de Factorización LU, se deberá primeramente de factorizar la matriz bandada $\underline{\underline{A}}_i^{II}$ en una matriz $\underline{\underline{LU}}$

cuya banda será $2b+1$, pero esta operación sólo se deberá de realizar una vez por cada subdominio, y para solucionar los diversos sistemas lineales $\underline{\underline{A}}_i^{II}\underline{x} = \underline{b}$ sólo será necesario evaluar los sistemas

$$\begin{aligned} \underline{\underline{L}}\underline{y} &= \underline{b} \\ \underline{\underline{U}}\underline{x} &= \underline{y} \end{aligned} \tag{43}$$

en donde $\underline{y}$ es un vector auxiliar. Esto proporciona una manera muy eficiente de evaluar el sistema lineal pero el consumo en memoria para un problema particular puede ser excesivo.

Por ello si el problema involucra una gran cantidad de nodos interiores y el equipo en el que se implantará la ejecución del programa tiene una cantidad de memoria muy limitada, es recomendable usar el método de Gradiente Conjugado, este consume una cantidad de memoria adicional muy pequeña, pero la eficiencia en tiempo de ejecución depende del tamaño del sistema y no tanto de los métodos usados, para ver cual tiene mejor desempeño en un problema dado, hay que evaluar rendimiento de ambos métodos.

De esta forma, es posible adaptar el código para tomar en cuenta la implementación de este en un equipo de cómputo en particular y poder sacar el máximo provecho al método de Subestructuración en la resolución de problemas elípticos de gran envergadura.

En lo que resta del presente trabajo, se asume que el método empleado para resolver $\underline{\underline{A}}_i^{II}\underline{x} = \underline{b}$ en sus respectivas variantes necesarias para evitar el cálculo de $\left(\underline{\underline{A}}_i^{II}\right)^{-1}$ es la Factorización LU.

Para más detalles de la forma como se presento este método ver [10] y para ver otras formulaciones ver [5], [4] y [2].

Observaciones:

- La precisión del método es la misma que la usada por los interpoladores en la descomposición de los subdominios.
- El método es paralelizable permitiendo que cada subdominio sea manipulado por un procesador y además es posible usar en cada subdominio para el manejo de las matrices generadas diversos esquemas de paralelización para aumentar el rendimiento de los procesadores.
- Hay que notar que por cada subdominio (supóngase E) se resuelven sólo los nodos de la frontera interior k, esto significa que en promedio se usan menos de k iteraciones en el método de gradiente conjugado para resolver el sistema lineal asociado (estas iteraciones son en cantidad menor que las realizadas por el método de Schwarz para un sólo subdominio, si Ω_i es tomado aproximadamente del mismo tamaño en ambos métodos). Y como la solución de los nodos interiores no conllevan iteraciones adicionales, este método resulta más económico computacionalmente que el método de Schwarz. Está

economía se hace aún más patente cuando se usa un buen precondicionador, logrando bajar hasta en un orden de magnitud el número de iteraciones del caso no precondicionado

1.1.1. Discretización Usando Rectángulos

Suponiendo que cada subdominio $\Omega_\alpha \subset \Omega \subset \mathbb{R}^2$, entonce el mallado del dominio, la selección de las funciones base, el proceso de discretización que da origen a la solución aproximada de la ecuación y su posterior implementación computacional se explican a continuación.

Mallado del dominio Para comenzar con el método, dividimos el dominio $\Omega_\alpha \subset \mathbb{R}^2$ en E subdominios o elementos Ω_e llamados elementos finitos, tal que

$$\overline{\Omega}_\alpha = \bigcup_{e=1}^{E} \overline{\Omega}_e$$

donde:

- Cada Ω_e es un polígono (rectángulo o triángulo) con interior no vacío ($\mathring{\Omega}_e \neq \emptyset$).
- $\Omega_i \cap \Omega_j = \emptyset$ para cada $i \neq j$.
- El diámetro $Diam(\Omega_e) \leq h$ para cada $e = 1, 2, ..., E$.
- Los vértices de cada Ω_e son llamados nodos, teniendo N de ellos.

Funciones Base A continuación describiremos la manera de construir las funciones base usada por el método de elemento finito. En este procedimiento debemos tener en cuenta que las funciones base están definidas en un subespacio de $V = H^1(\Omega)$ para problemas de segundo orden que satisfacen las condiciones de frontera.

Las funciones base deberán satisfacer las siguientes propiedades:

i) Las funciones base ϕ_i son acotadas y continuas, i.e $\phi_i \in C(\Omega_e)$.

ii) Existen ℓ funciones base por cada nodo del polígono Ω_e, y cada función ϕ_i es no cero solo en los elementos contiguos conectados por el nodo i.

iii) $\phi_i = 1$ en cada i nodo del polígono Ω_e y cero en los otros nodos.

iv) La restricción ϕ_i a Ω_e es un polinomio, i.e. $\phi_i \in \mathbb{P}_k[\Omega_e]$ para alguna $k \geq 1$ donde $\mathbb{P}_k[\Omega_e]$ es el espacio de polinomios de grado a lo más k sobre Ω_e.

Decimos que $\phi_i \in \mathbb{P}_k[\Omega_e]$ es una base de funciones y por su construcción es evidente que estas pertenecen a $H^1(\Omega_\alpha)$. Al conjunto formado por todas las

funciones base definidas para todo Ω_e de Ω_α será el espacio $\mathbb{P}^h[k]$ de funciones base, i.e.

$$\mathbb{P}^h[k] = \bigcup_{e=1}^{E} \mathbb{P}_k[\Omega_e]$$

estas formarán las funciones base.

Solución aproximada Para encontrar la solución aproximada elegimos el espacio $\mathbb{P}^h[k]$ de funciones base, como el espacio de funciones lineales ϕ_i definidas por pedazos de grado menor o igual a k (en nuestro caso $k = 1$), entonces el espacio a trabajar es

$$V^h = Generado\left\{\phi_i \in \mathbb{P}^h[k] \;\mid \phi_i(x) = 0 \text{ en } \partial\Omega_\alpha\right\}. \tag{44}$$

La solución aproximada de la Ec. (1) queda en términos de

$$\int_{\Omega_\alpha} \left(\nabla\phi_i \cdot \underline{\underline{a}} \cdot \nabla\phi_j - c\phi_i\phi_j\right) dxdy = \int_{\Omega_\alpha} f_\Omega \phi_j dxdy \tag{45}$$

si definimos el operador bilineal

$$K_{ij} \equiv a\left(\phi_i, \phi_j\right) = \int_{\Omega_\alpha} \left(\nabla\phi_i \cdot a_{ij} \cdot \nabla\phi_j - c\phi_i\phi_j\right) dxdy \tag{46}$$

y la funcional lineal

$$F_j \equiv \left\langle f, \phi_j \right\rangle = \int_{\Omega_\alpha} f_\Omega \phi_j dxdy \tag{47}$$

entonces la matriz $\underline{\underline{K}} \equiv [K_{ij}]$, los vectores $\underline{u} \equiv (u_1, ..., u_N)$ y $\underline{F} \equiv (F_1, ..., F_N)$ definen el sistema lineal (que es positivo definido).

1.1.2. Discretización Usando Rectángulos

Para resolver la Ec. (1), usando una discretización con rectángulos en el subdominio Ω_α, primero dividimos el dominio $\Omega_\alpha \subset \mathbb{R}^2$ en N_x nodos horizontales por N_y nodos verticales, teniendo $E = (N_x - 1)(N_y - 1)$ subdominios o elementos rectangulares Ω_e tales que $\overline{\Omega} = \cup_{e=1}^{E} \overline{\Omega_e}$ y $\overline{\Omega_i} \cap \overline{\Omega_j} \neq \varnothing$ si son adyacentes, con un total de $N = N_x N_y$ nodos.

Donde las funciones lineales definidas por pedazos en Ω_e en nuestro caso serán polinomios de orden uno en cada variable separadamente y cuya restricción de ϕ_i a Ω_e es $\phi_i^{(e)}$. Para simplificar los cálculos en esta etapa, supondremos que la matriz $\underline{\underline{a}} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, entonces se tiene que la integral del lado izquierdo de la Ec. (45) queda escrita como

$$\int_{\Omega_\alpha} \left(a\nabla\phi_i \cdot \nabla\phi_j + c\phi_i\phi_j\right) dxdy = \int_{\Omega_\alpha} f_\Omega \phi_j dxdy \tag{48}$$

donde usando una discretización con rectángulos, primero dividimos el dominio $\Omega_i \subset \mathbb{R}^2$ en N_x nodos horizontales por N_y nodos verticales, teniendo $E = (N_x-1)(N_y-1)$ elementos rectangulares Ω_e tales que $\overline{\Omega} = \cup_{e=1}^{E} \overline{\Omega_e}$ y $\overline{\Omega_i} \cap \overline{\Omega_j} \neq \varnothing$ si son adyacentes, con un total de $N = N_x N_y$ nodos.

Donde las funciones lineales definidas por pedazos en Ω_e en nuestro caso serán polinomios de orden uno en cada variable separadamente y cuya restricción de ϕ_i a Ω_e es $\phi_i^{(e)}$.

Sea

$$\begin{aligned} K_{ij} &= \int_{\Omega} \left(a \nabla \phi_i \cdot \nabla \phi_j + c \phi_i \phi_j \right) dxdy \qquad (49) \\ &= \sum_{e=1}^{E} \int_{\Omega_e} \left(a \nabla \phi_i^{(e)} \cdot \nabla \phi_j^{(e)} + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \\ &= \sum_{e=1}^{E} \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \end{aligned}$$

y el lado derecho

$$\begin{aligned} F_j &= \int_{\Omega} f_{\Omega} \phi_j dxdy \qquad (50) \\ &= \sum_{e=1}^{E} \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dxdy. \end{aligned}$$

Para cada Ω_e de Ω, la submatriz de integrales (matriz de carga local)

$$K_{ij} = \int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \qquad (51)$$

tiene la estructura

$$\begin{bmatrix} K_{1,1}^{(e)} & K_{1,2}^{(e)} & K_{1,3}^{(e)} & K_{1,4}^{(e)} \\ K_{2,1}^{(e)} & K_{2,2}^{(e)} & K_{2,3}^{(e)} & K_{2,4}^{(e)} \\ K_{3,1}^{(e)} & K_{3,2}^{(e)} & K_{3,3}^{(e)} & K_{3,4}^{(e)} \\ K_{4,1}^{(e)} & K_{4,2}^{(e)} & K_{4,3}^{(e)} & K_{4,4}^{(e)} \end{bmatrix}$$

la cual deberá ser ensamblada en la matriz de carga global que corresponda a la numeración de nodos locales del elemento Ω_e con respecto a la numeración global de los elementos en Ω.

De manera parecida, para cada Ω_e de Ω se genera el vector de integrales (vector de carga local)

$$F_j = \int_{\Omega_e} f_{\Omega} \phi_j^{(e)} dxdy \qquad (52)$$

con la estructura

$$\begin{bmatrix} F_1^{(e)} \\ F_2^{(e)} \\ F_3^{(e)} \\ F_4^{(e)} \end{bmatrix}$$

el cual también deberá ser ensamblado en el vector de carga global que corresponda a la numeración de nodos locales al elemento Ω_e con respecto a la numeración global de los elementos de Ω.

Montando los $K_{ij}^{(e)}$ en la matriz $\underline{\underline{\mathbb{K}}}$ (correspondientes a cada una de las matrices $\underline{\underline{A}}_i^{\Sigma\Sigma}, \underline{\underline{A}}_i^{\Sigma I}, \underline{\underline{A}}_i^{I\Sigma}$ y $\underline{\underline{A}}_i^{II}$) y los $F_j^{(e)}$ en el vector $\underline{\mathbb{F}}$ (Correspondiente a $\underline{b_\Sigma}_i$ y $\underline{b_I}_i$) según la numeración de nodos locales al subdominio.

Para implementar numéricamente en cada Ω_e las integrales

$$\int_{\Omega_e} \left(a \left[\frac{\partial \phi_i^{(e)}}{\partial x} \frac{\partial \phi_j^{(e)}}{\partial x} + \frac{\partial \phi_i^{(e)}}{\partial y} \frac{\partial \phi_j^{(e)}}{\partial y} \right] + c \phi_i^{(e)} \phi_j^{(e)} \right) dxdy \tag{53}$$

y

$$\int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy, \tag{54}$$

teniendo en mente el simplificar los cálculos computacionales, se considera un elemento de referencia $\hat{\Omega}$ en los ejes coordenados (ε, η) cuyos vértices están el $(-1,-1),(1,-1),(1,1)$ y $(-1,1)$ respectivamente, en el cual mediante una función afín será proyectado cualquier elemento rectangular Ω_e cuyos vértices $(x_1^{(e)}, y_1^{(e)}), (x_2^{(e)}, y_2^{(e)}), (x_3^{(e)}, y_3^{(e)})$ y $(x_4^{(e)}, y_4^{(e)})$ están tomados en sentido contrario al movimiento de las manecillas del reloj como se muetra en la figura

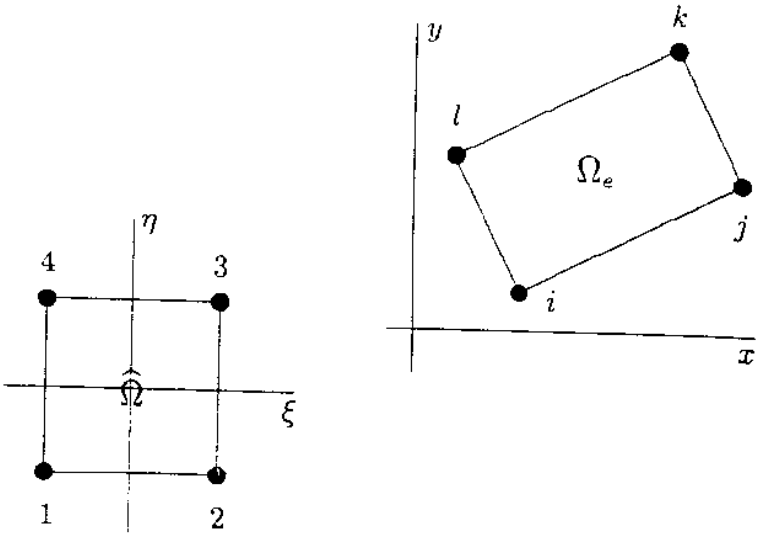

Figura 3:

mediante la transformación $f(x,y) = \underline{\underline{T}}(\varepsilon,\eta) + \underline{b}$, quedando dicha transforma-

ción como

$$\begin{aligned} x &= \frac{x_2^{(e)} - x_1^{(e)}}{2}\varepsilon + \frac{y_2^{(e)} - y_1^{(e)}}{2}\eta \\ y &= \frac{x_4^{(e)} - x_1^{(e)}}{2}\varepsilon + \frac{y_4^{(e)} - y_1^{(e)}}{2}\eta \end{aligned} \tag{55}$$

en la cual la matriz $\underline{\underline{T}}$ está dada por

$$\underline{\underline{T}} = \begin{pmatrix} \frac{x_2^{(e)}-x_1^{(e)}}{2} & \frac{y_2^{(e)}-y_1^{(e)}}{2} \\ \frac{x_4^{(e)}-x_1^{(e)}}{2} & \frac{y_4^{(e)}-y_1^{(e)}}{2} \end{pmatrix} \tag{56}$$

y el vector $\underline{b}$= (b_1, b_2) es la posición del vector centroide del rectángulo Ω_e, también se tiene que la transformación inversa es

$$\begin{aligned} \varepsilon &= \frac{x - b_1 - \frac{y_2^{(e)}-y_1^{(e)}}{2}\left[\frac{y-b_2}{\left(\frac{x_4^{(e)}-x_1^{(e)}}{2}\right)\left(\frac{x-b_1-\frac{y_2^{(e)}-y_1^{(e)}}{2}}{\frac{x_2^{(e)}-x_1^{(e)}}{2}}\right)}\right]}{\frac{x_2^{(e)}-x_1^{(e)}}{2}} \\ \eta &= \frac{y - b_2}{\left(\frac{x_4^{(e)}-x_1^{(e)}}{2}\right)\left(\frac{x-b_1-\frac{y_2^{(e)}-y_1^{(e)}}{2}}{\frac{x_2^{(e)}-x_1^{(e)}}{2}}\right) + \frac{y_4^{(e)}-y_1^{(e)}}{2}}. \end{aligned} \tag{57}$$

Entonces las $\phi_i^{(e)}$ quedan definidas en términos de $\hat{\phi}_i$ como

$$\begin{aligned} \hat{\phi}_1(\varepsilon,\eta) &= \frac{1}{4}(1-\varepsilon)(1-\eta) \\ \hat{\phi}_2(\varepsilon,\eta) &= \frac{1}{4}(1+\varepsilon)(1-\eta) \\ \hat{\phi}_3(\varepsilon,\eta) &= \frac{1}{4}(1+\varepsilon)(1+\eta) \\ \hat{\phi}_4(\varepsilon,\eta) &= \frac{1}{4}(1-\varepsilon)(1+\eta) \end{aligned} \tag{58}$$

y las funciones $\phi_i^{(e)}$ son obtenidas por el conjunto $\phi_i^{(e)}(x,y) = \hat{\phi}_i(\varepsilon,\eta)$ con (x,y) y (ε,η) relacionadas por la Ec. (55), entonces se tendrian las siguientes integrales

$$\begin{aligned} K_{ij}^{(e)} &= \int_{\Omega_e}\left(a\left[\frac{\partial\phi_i^{(e)}}{\partial x}\frac{\partial\phi_j^{(e)}}{\partial x} + \frac{\partial\phi_i^{(e)}}{\partial y}\frac{\partial\phi_j^{(e)}}{\partial y}\right] + c\phi_i^{(e)}\phi_j^{(e)}\right)dxdy \\ &= \int_{\hat{\Omega}}\left(\left[a\left(\frac{\partial\hat{\phi}_i}{\partial\varepsilon}\frac{\partial\varepsilon}{\partial x} + \frac{\partial\hat{\phi}_i}{\partial\eta}\frac{\partial\eta}{\partial x}\right)\left(\frac{\partial\hat{\phi}_j}{\partial\varepsilon}\frac{\partial\varepsilon}{\partial x} + \frac{\partial\hat{\phi}_j}{\partial\eta}\frac{\partial\eta}{\partial x}\right) + \right.\right. \\ &\quad \left.\left.\left(\frac{\partial\hat{\phi}_i}{\partial\varepsilon}\frac{\partial\varepsilon}{\partial y} + \frac{\partial\hat{\phi}_i}{\partial\eta}\frac{\partial\eta}{\partial y}\right)\left(\frac{\partial\hat{\phi}_j}{\partial\varepsilon}\frac{\partial\varepsilon}{\partial y} + \frac{\partial\hat{\phi}_j}{\partial\eta}\frac{\partial\eta}{\partial y}\right)\right] + c\hat{\phi}_i\hat{\phi}_j\right)|J|\,d\varepsilon d\eta \end{aligned} \tag{59}$$

donde el índice i y j varia de 1 a 4. En está última usamos la regla de la cadena y $dxdy = |J|\, d\varepsilon d\eta$ para el cambio de variable en las integrales, aquí $|J| = \det T$, donde T está dado como en la Ec. (56). Para resolver $\int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy$ en cada Ω_e se genera las integrales

$$\begin{aligned} F_j^{(e)} &= \int_{\Omega_e} f_\Omega \phi_j^{(e)} dxdy \\ &= \int_{\hat{\Omega}} f_\Omega \hat{\phi}_j \, |J| \, d\varepsilon d\eta \end{aligned} \tag{60}$$

donde el índice i y j varia de 1 a 4.

Para realizar el cálculo numérico de las integrales en el rectángulo de referencia $\hat{\Omega} = [-1,1] \times [-1,1]$, debemos conocer $\frac{\partial \phi i}{\partial \varepsilon}$, $\frac{\partial \phi i}{\partial \eta}$, $\frac{\partial \varepsilon}{\partial x}$, $\frac{\partial \varepsilon}{\partial y}$, $\frac{\partial \eta}{\partial x}$ y $\frac{\partial \eta}{\partial y}$, entonces realizando las operaciones necesarias a la Ec. (58) obtenemos

$$\begin{array}{ll} \frac{\partial \phi_1}{\partial \varepsilon} = -\frac{1}{4}(1-\eta) & \frac{\partial \phi_1}{\partial \eta} = -\frac{1}{4}(1-\varepsilon) \\ \frac{\partial \phi_2}{\partial \varepsilon} = \frac{1}{4}(1-\eta) & \frac{\partial \phi_2}{\partial \eta} = -\frac{1}{4}(1+\varepsilon) \\ \frac{\partial \phi_3}{\partial \varepsilon} = \frac{1}{4}(1+\eta) & \frac{\partial \phi_3}{\partial \eta} = \frac{1}{4}(1+\varepsilon) \\ \frac{\partial \phi_4}{\partial \varepsilon} = -\frac{1}{4}(1+\eta) & \frac{\partial \phi_4}{\partial \eta} = \frac{1}{4}(1-\varepsilon) \end{array} \tag{61}$$

y también

$$\begin{array}{ll} \frac{\partial \varepsilon}{\partial x} = \left(\frac{y_4^{(e)} - y_1^{(e)}}{2 \det T} \right) & \frac{\partial \varepsilon}{\partial y} = \left(\frac{x_4^{(e)} - x_1^{(e)}}{2 \det T} \right) \\ \frac{\partial \eta}{\partial x} = \left(\frac{y_2^{(e)} - y_1^{(e)}}{2 \det T} \right) & \frac{\partial \eta}{\partial y} = \left(\frac{x_2^{(e)} - x_1^{(e)}}{2 \det T} \right) \end{array} \tag{62}$$

las cuales deberán de ser sustituidas en cada $\underline{\underline{K_{ij}^{(e)}}}$ y $\underline{F_j^{(e)}}$ para calcular las integrales en el elemento Ω_e. Estas integrales se harán en el programa usando cuadratura Gaussiana, permitiendo reducir el número de cálculos al mínimo pero manteniendo el balance entre precisión y número bajo de operaciones necesarias para realizar las integraciones..

1.2. El Operador de Laplace y la Ecuación de Poisson

Consideramos como modelo matemático el problema de valor en la frontera (BVP) asociado con el operador de Laplace en dos dimensiones, el cual en general es usualmente referido como la ecuación de Poisson, con condiciones de frontera Dirichlet, definido en Ω como:

$$\begin{aligned} -\nabla^2 u &= f_\Omega \text{ en } \Omega \\ u &= g_{\partial\Omega} \text{ en } \partial\Omega. \end{aligned} \tag{63}$$

Se toma está ecuación para facilitar la comprensión de las ideas básicas. Es un ejemplo muy sencillo, pero gobierna los modelos de muchos sistemas de la ingeniería y de la ciencia, entre ellos el flujo de agua subterránea a través de un acuífero isotrópico, homogéneo bajo condiciones de equilibrio y es muy usada en múltiples ramas de la física. Por ejemplo, gobierna la ecuación de la conducción de calor en un sólido bajo condiciones de equilibrio.

En particular consideramos el problema con Ω definido en:

$$\Omega = [-1, 1] \times [0, 1] \tag{64}$$

donde

$$f_\Omega = 2n^2\pi^2 \sin(n\pi x) * \sin(n\pi y) \quad \text{y} \quad g_{\partial\Omega} = 0 \tag{65}$$

cuya solución es

$$u(x, y) = \sin(n\pi x) * \sin(n\pi y). \tag{66}$$

Para las pruebas de rendimiento en las cuales se evalúa el desempeño de los programas realizados se usa $n = 10$, pero es posible hacerlo con $n \in \mathbb{N}$ grande. Por ejemplo para $n = 4$, la solución es $u(x, y) = \sin(4\pi x) * \sin(4\pi y)$, cuya gráfica se muestra a continuación:

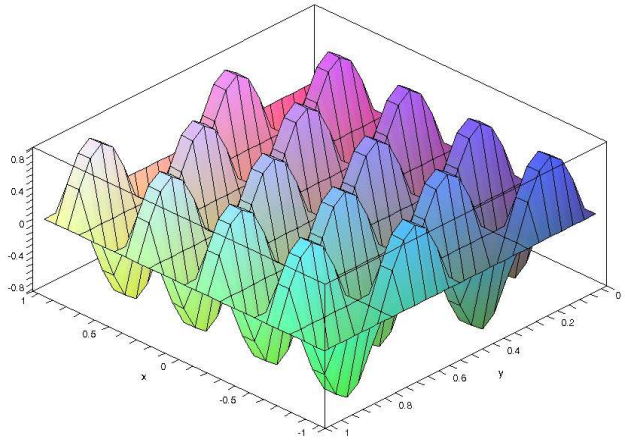

Figura 4: Solución a la ecuación de Poisson para n=4.

Hay que hacer notar que al implementar la solución numérica por el método del elemento finito y el método de subestructuración secuencial en un procesador, un factor limitante para su operación es la cantidad de memoria disponible en la computadora, ya que el sistema algebraico de ecuaciones asociado a esté problema crece muy rápido (del orden de n^2), donde n es el número de nodos en la partición. Es por ello que la elección de un buen manejador de matrices será determinante en la eficiencia alcanzada por las distintas implementaciones de los programas.

Actualmente existen múltiples bibliotecas que permiten manipular operaciones de matrices tanto en forma secuencial como en paralelo (hilos y Pipeline) para implementarlas tanto en procesadores con memoria compartida como distribuida. Pero no están presentes en todas las arquitecturas y sistemas operativos. Por ello en este trabajo se implementaron todas las operaciones necesarias usando clases que fueron desarrolladas sin usar ninguna biblioteca externa a las proporcionadas por el compilador de C++ de GNU, permitiendo la operación de los programas desarrollados en múltiples sistemas operativos del tipo Linux, Unix y Windows.

En cuanto a las pruebas de rendimiento que mostraremos en las siguientes secciones se usaron para la parte secuencial el equipo:

- Computadora Pentium IV HT a 2.8 GHtz con 1 GB de RAM corriendo bajo el sistema operativo Linux Debian Stable con el compilador g++ de GNU.

Para la parte paralela se usaron los equipos siguientes:

- Cluster homogéneo de 10 nodos duales Xeon a 2.8 GHtz con 1 GB de RAM por nodo, unidos mediante una red Ethernet de 1 Gb, corriendo bajo el sistema operativo Linux Debian Stable con el compilador mpiCC de MPI de GNU.
- Cluster heterogéneo con el nodo maestro Pentium IV HT a 3.4 GHtz con 1 GB de RAM y 7 nodos esclavos Pentium IV HT a 2.8 GHtz con 0.5 GB de RAM por nodo, unidos mediante una red Ethernet de 100 Mb, corriendo bajo el sistema operativo Linux Debian Stable con el compilador mpiCC de MPI de GNU.

A estos equipos nos referiremos en lo sucesivo como equipo secuencial, cluster homogéneo y cluster heterogéneo respectivamente.

El tiempo dado en los resultados de las distintas pruebas de rendimiento de los programas y mostrado en todas las tablas y gráficas fue tomado como un promedio entre por lo menos 5 corridas, redondeado el resultado a la unidad siguiente. En todos los cálculos de los métodos numéricos usados para resolver el sistema lineal algebraico asociado se usó una tolerancia mínima de 1×10^{-10}.

1.3. Método de Subestructuración Secuencial

A partir de la formulación del método de subestructuración visto en la sección (1) se generan las matrices locales $\underline{\underline{A}}_i^{II}, \underline{\underline{A}}_i^{I\Sigma}, \underline{\underline{A}}_i^{\Sigma I}$ y $\underline{\underline{A}}_i^{\Sigma\Sigma}$ y con ellas se

construyen $\underline{\underline{S}}_i = \underline{\underline{A}}_i^{\Sigma\Sigma} - \underline{\underline{A}}_i^{\Sigma I}\left(\underline{\underline{A}}_i^{II}\right)^{-1}\underline{\underline{A}}_i^{I\Sigma}$ y $\underline{b}_i = \underline{\underline{A}}_i^{\Sigma I}\left(\underline{\underline{A}}_i^{II}\right)^{-1}\underline{b}_{I_i}$ que son problemas locales a cada subdominio Ω_i, con $i = 1, 2, ..., E$. Generando de manera virtual el sistema lineal $\underline{\underline{S}}\,\underline{u_\Sigma} = \underline{b}$ a partir de

$$\left[\sum_{i=1}^{E}\underline{\underline{S}}_i\right]\underline{u_\Sigma} = \left[\sum_{i=1}^{E}\underline{b}_i\right] \tag{67}$$

donde $\underline{\underline{S}} = \left[\sum_{i=1}^{E}\underline{\underline{S}}_i\right]$ y $\underline{b} = \left[\sum_{i=1}^{E}\underline{b}_i\right]$ podría ser construida sumando las $\underline{\underline{S}}_i$ y $\underline{b}_i$ respectivamente según el orden de los nodos globales versus los nodos locales a cada subdominio.

El sistema lineal virtual resultante

$$\underline{\underline{S}}\,\underline{u_\Sigma} = \underline{b} \tag{68}$$

es resuelto usando el método de gradiente conjugado visto en la sección (**??**), para ello no es necesario construir la matriz $\underline{\underline{S}}$ con las contribuciones de cada S_i correspondientes al subdominio i. Lo que hacemos es pasar a cada subdominio el vector $\underline{u_\Sigma}^i$ correspondiente a la i-ésima iteración del método de gradiente conjugado para que en cada subdominio se evalué $\underline{\tilde{u}_\Sigma}^i = \underline{\underline{S}}_i\underline{u_\Sigma}^i$ localmente y con el resultado se forma el vector $\underline{\tilde{u}_\Sigma} = \sum_{i=1}^{E}\underline{\tilde{u}_\Sigma}^i$ y se continué con los demás pasos del método.

La implementación computacional que se desarrolló tiene una jerarquía de clases en la cual se agregan las clases *FEM2D Rectángulos* y *Geometría*, además de heredar a la clase *Problema*. De esta forma se rehusó todo el código desarrollado para *FEM2D Rectángulos*, la jerarquía queda como:

> La clase *DDM2D* realiza la partición gruesa del dominio mediante la clase *Geometría* y controla la partición de cada subdominio mediante un objeto de la clase de *FEM2D Rectángulos* generando la partición fina del dominio. La resolución de los nodos de la frontera interior se hace mediante el método de gradiente conjugado, necesaria para resolver los nodos internos de cada subdominio.

Así, el dominio Ω es descompuesto en una descomposición gruesa de $n \times m$ subdominios y cada subdominio Ω_i se parte en $p \times q$ subdominios, generando la participación fina del dominio como se muestra en la figura:

El método de descomposición de dominio se implementó realizando las siguientes tareas:

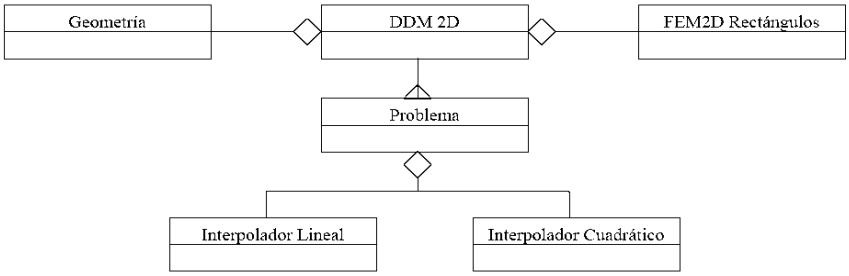

Figura 5: Jerarquía de clases para el método de subestructuración secuencial

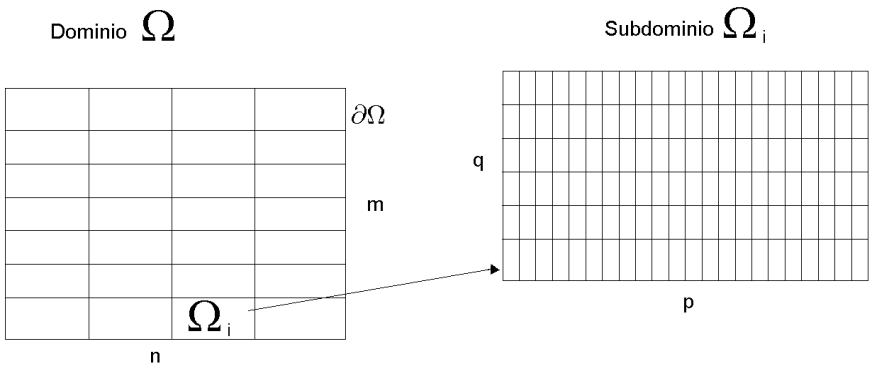

Figura 6: Descomposición del dominio Ω en $E = n \times m$ subdominios y cada subdominio Ω_i en $p \times q$ subdominios

A) La clase *DDM2D* genera la descomposición gruesa del dominio mediante la agregación de un objeto de la clase *Geometría* (supongamos particionado en $n \times m$ subdominios, generando $s = n * m$ subdominios Ω_i, $i = 1, 2, ..., E$).

B) Con esa geometría se construyen los objetos de *FEM2D Rectángulos* (uno por cada subdominio Ω_i), donde cada subdominio es particionado (supongamos en $p \times q$ subdominios) y regresando las coordenadas de los nodos de frontera del subdominio correspondiente a la clase DDM2D.

C) Con estas coordenadas, la clase *DDM2D* conoce a los nodos de la frontera interior (son estos los que resuelve el método de descomposición de dominio). Las coordenadas de los nodos de la frontera interior se dan a conocer a los objetos *FEM2D Rectángulos*, transmitiendo sólo aquellos que están en su subdominio.

D) Después de conocer los nodos de la frontera interior, cada ob-

jeto *FEM2D Rectángulos* calcula las matrices $\underline{\underline{A}}_i^{\Sigma\Sigma}, \underline{\underline{A}}_i^{\Sigma I}, \underline{\underline{A}}_i^{I\Sigma}$ y $\underline{\underline{A}}_i^{II}$ necesarias para construir el complemento de Schur local $\underline{\underline{S}}_i = \underline{\underline{A}}_i^{\Sigma\Sigma} - \underline{\underline{A}}_i^{\Sigma I}\left(\underline{\underline{A}}_i^{II}\right)^{-1}\underline{\underline{A}}_i^{I\Sigma}$ sin realizar comunicación alguna. Al terminar de calcular las matrices se avisa a la clase *DDM2D* de la finalización de los cálculos.

E) Mediante la comunicación de vectores del tamaño del número de nodos de la frontera interior entre la clase *DDM2D* y los objetos *FEM2D Rectángulos*, se prepara todo lo necesario para empezar el método de gradiente conjugado y resolver el sistema lineal virtual $\left[\sum_{i=1}^{E}\underline{\underline{S}}_i\right]\underline{u_\Sigma} = \left[\sum_{i=1}^{E}\underline{b}_i\right]$.

F) Para usar el método de gradiente conjugado, se transmite un vector del tamaño del número de nodos de la frontera interior para que en cada objeto se realicen las operaciones pertinentes y resolver así el sistema algebraico asociado, esta comunicación se realiza de ida y vuelta entre la clase *DDM2D* y los objetos *FEM2D Rectángulos* tantas veces como iteraciones haga el método. Resolviendo con esto los nodos de la frontera interior $\underline{u_{\Sigma_i}}$.

G) Al término de las iteraciones se pasa la solución $\underline{u_{\Sigma_i}}$ de los nodos de la frontera interior que pertenecen a cada subdominio dentro de cada objeto *FEM2D Rectángulos* para que se resuelvan los nodos interiores $\underline{u_{I_i}} = \left(\underline{\underline{A}}_i^{II}\right)^{-1}\left(\underline{b_{I}}_i - \underline{\underline{A}}_i^{I\Sigma}\underline{u_{\Sigma_i}}\right)$, sin realizar comunicación alguna en el proceso, al concluir se avisa a la clase *DDM2D* de ello.

I) La clase *DDM2D* mediante un último mensaje avisa que se concluya el programa, terminado así el esquema maestro-esclavo secuencial.

Por ejemplo, para resolver la Ec. (63), usando 513×513 nodos (igual al ejemplo de *FEM2D Rectángulos* secuencial), podemos tomar alguna de las siguientes descomposiciones:

Descomposición	Nodos Interiores	Subdominios	Elementos Subdominio	Total Nodos Subdominio	Nodos Desconocidos Subdominio
2x2 y 256x256	260100	4	65536	66049	65025
4x4 y 128x128	258064	16	16384	16641	16129
8x8 y 64x64	254016	64	4096	4225	3969
16x16 y 32x32	246016	256	1024	1089	961
32x32 y 16x16	230400	1024	256	289	225

Cada una de las descomposiciones genera un problema distinto. Usando el equipo secuencial y evaluando el desempeño del método de subestructuración secuencial se obtuvieron los siguientes resultados:

Partición	Nodos Frontera Interior	Iteraciones	Tiempo Total
2x2 y 256x256	1021	139	5708 seg.
4x4 y 128x128	3057	159	2934 seg.
8x8 y 64x64	7105	204	1729 seg.
16x16 y 32x32	15105	264	1077 seg.
32x32 y 16x16	30721	325	1128 seg

Nótese que aún en un solo procesador es posible encontrar una descomposición que desminuya los tiempos de ejecución (la descomposición de 2x2 y 256x256 concluye en 5708 seg. versus los 6388 seg. en el caso de *FEM2D Rectángulos)*, ello es debido a que al descomponer el dominio en múltiples subdominios, la complejidad del problema es también disminuida y esto se ve reflejado en la disminución del tiempo de ejecución.

En la última descomposición, en lugar de disminuir el tiempo de ejecución este aumenta, esto se debe a que se construyen muchos objetos *FEM2D Rectángulos* (1024 en este caso), con los cuales hay que hacer comunicación resultando muy costoso computacionalmente.

Finalmente las posibles mejoras de eficiencia para el método de subestructuración secuencial para disminuir el tiempo de ejecución son las mismas que en el caso del método de elemento finito pero además se tienen que:

- Encontrar la descomposición pertinente entre las posibles descomposiciones que consuma el menor tiempo de cálculo.

Adicionalmente si se cuenta con un equipo con más de un procesador con memoria compartida es posible usar bibliotecas para la manipulación de matrices y vectores que paralelizan o usan Pipeline como una forma de mejorar el rendimiento del programa. Este tipo de mejoras cuando es posible usarlas disminuyen sustancialmente el tiempo de ejecución, ya que en gran medida el consumo total de CPU está en la manipulación de matrices, pero esto no hace paralelo al método de subestructuración secuencial.

1.4. Método de Subestructuración en Paralelo

La computación en paralelo es una técnica que nos permite distribuir una gran carga computacional entre muchos procesadores. Y es bien sabido que una de las mayores dificultades del procesamiento en paralelo es la coordinación de las actividades de los diferentes procesadores y el intercambio de información entre los mismos.

Para hacer una adecuada coordinación de actividades entre los diferentes procesadores, el programa que soporta el método de subestructuración paralelo, usa la misma jerarquía de clases que el método de subestructuración secuencial. Este se desarrolló para usar el esquema maestro-esclavo, de forma tal que el nodo maestro mediante la agregación de un objeto de la clase de *Geometría* genere la descomposición gruesa del dominio y los nodos esclavos creen un conjunto de objetos *FEM2D Rectángulos* para que en estos objetos se genere la participación fina y mediante el paso de mensajes (vía MPI) puedan comunicarse los nodos esclavos con el nodo maestro, realizando las siguientes tareas:

A) El nodo maestro genera la descomposición gruesa del dominio (supongamos particionado en $n \times m$ subdominios) mediante la agregación de un objeto de la clase *Geometría*, esta geometría es pasada a los nodos esclavos.

B) Con esa geometría se construyen los objetos *FEM2D Rectángulos* (uno por cada subdominio), donde cada subdominio es particionado (supongamos en $p \times q$ subdominios). Cada objeto de *FEM2D Rectángulos* genera la geometría solicitada, regresando las coordenadas de los nodos de frontera del subdominio correspondiente al nodo maestro.

C) Con estas coordenadas, el nodo maestro conoce a los nodos de la frontera interior (son estos los que resuelve el método de descomposición de dominio). Las coordenadas de los nodos de la frontera interior se dan a conocer a los objetos *FEM2D Rectángulos* en los nodos esclavos, transmitiendo sólo aquellos que están en su subdominio.

D) Después de conocer los nodos de la frontera interior, cada objeto *FEM2D Rectángulos* calcula las matrices $\underline{\underline{A}}_i^{\Sigma\Sigma}, \underline{\underline{A}}_i^{\Sigma I}, \underline{\underline{A}}_i^{I\Sigma}$ y $\underline{\underline{A}}_i^{II}$ necesarias para construir el complemento de Schur local $\underline{\underline{S}}_i = \underline{\underline{A}}_i^{\Sigma\Sigma} - \underline{\underline{A}}_i^{\Sigma I}\left(\underline{\underline{A}}_i^{II}\right)^{-1}\underline{\underline{A}}_i^{I\Sigma}$ sin realizar comunicación alguna. Al terminar de calcular las matrices se avisa al nodo maestro de la finalización de los cálculos.

E) Mediante la comunicación de vectores del tamaño del número de nodos de la frontera interior entre el nodo maestro y los objetos *FEM2D Rectángulos*, se prepara todo lo necesario para empezar el método de gradiente conjugado y resolver el sistema lineal virtual $\left[\sum_{i=1}^{E} \underline{\underline{S}}_i\right] \underline{u_\Sigma} = \left[\sum_{i=1}^{E} \underline{b}_i\right]$.

F) Para usar el método de gradiente conjugado, se transmite un vector del tamaño del número de nodos de la frontera interior para que en cada objeto se realicen las operaciones pertinentes y resolver así el sistema algebraico asociado, esta comunicación se realiza de ida y vuelta entre el nodo maestro y los objetos *FEM2D Rectángulos* tantas veces como iteraciones haga el método. Resolviendo con esto los nodos de la frontera interior $\underline{u_{\Sigma_i}}$.

G) Al término de las iteraciones se pasa la solución $\underline{u_{\Sigma_i}}$ de los nodos de la frontera interior que pertenecen a cada subdominio dentro de cada objeto *FEM2D Rectángulos* para que se resuelvan los nodos interiores $\underline{u_{I_i}} = \left(\underline{\underline{A}}_i^{II}\right)^{-1}\left(\underline{b_{I}}_i - \underline{\underline{A}}_i^{I\Sigma}\underline{u_{\Sigma_i}}\right)$, sin realizar comunicación alguna en el proceso, al concluir se avisa al nodo maestro de ello.

I) El nodo maestro mediante un último mensaje avisa que se concluya el programa, terminado así el esquema maestro-esclavo.

Del algoritmo descrito anteriormente hay que destacar la sincronía entre el nodo maestro y los objetos *FEM2D Rectángulos* contenidos en los nodos esclavos, esto es patente en las actividades realizadas en los incisos A, B y C, estas consumen una parte no significativa del tiempo de cálculo.

Una parte importante del tiempo de cálculo es consumida en la generación de las matrices locales descritas en el inciso D que se realizan de forma independiente en cada nodo esclavo, esta es muy sensible a la discretización particular del dominio usado en el problema.

Los incisos E y F del algoritmo consumen la mayor parte del tiempo total del ejecución al resolver el sistema lineal que dará la solución a los nodos de la frontera interior. La resolución de los nodos interiores planteada en el inciso G consume muy poco tiempo de ejecución, ya que sólo se realiza una serie de cálculos locales previa transmisión del vector que contiene la solución a los nodos de la frontera interior.

Este algoritmo es altamente paralelizable ya que los nodos esclavos están la mayor parte del tiempo ocupados y la fracción serial del algoritmo esta principalmente en las actividades que realiza el nodo maestro, estas nunca podrán ser eliminadas del todo pero consumirán menos tiempo del algoritmo conforme se haga más fina la malla en la descomposición del dominio.

Para resolver la Ec. (63), usando 513×513 nodos (igual al ejemplo de *FEM2D Rectángulos* secuencial), en la cual se toma una partición rectangular gruesa de 4×4 subdominios y cada subdominio se descompone en 128×128 subdominios.

Usando para los cálculos en un procesador el equipo secuencial y para la parte paralela el cluster heterogéneo resolviendo por el método de gradiente conjugado sin precondicionador, la solución se encontró en 159 iteraciones obteniendo los siguientes valores:

Procesadores	Tiempo	Factor de Aceleración	Eficiencia	Fracción Serial
1	2943 seg.			
2	2505 seg.	1.17485	0.58742	0.70234
3	1295 seg.	2.27258	0.75752	0.16004
4	1007 seg.	2.92254	0.73063	0.12289
5	671 seg.	4.38599	0.87719	0.03499
6	671 seg.	4.38599	0.73099	0.07359
7	497seg.	5.92152	0.84593	0.03035
8	497 seg.	5.92152	0.74019	0.05014
9	359 seg.	8.19777	0.91086	0.01223
10	359 seg.	8.19777	0.81977	0.02442
11	359 seg.	8.19777	0.74525	0.03441
12	359 seg.	8.19777	0.68314	0.04216
13	359 seg.	8.19777	0.63059	0.04881
14	359 seg.	8.19777	0.58555	0.05444
15	359 seg.	8.19777	0.54651	0.05926
16	359 seg.	8.19777	0.51236	0.06344
17	188 seg	15.65425	0.92083	0.00537

Estos resultados pueden ser apreciados mejor de manera gráfica como se muestra a continuación:

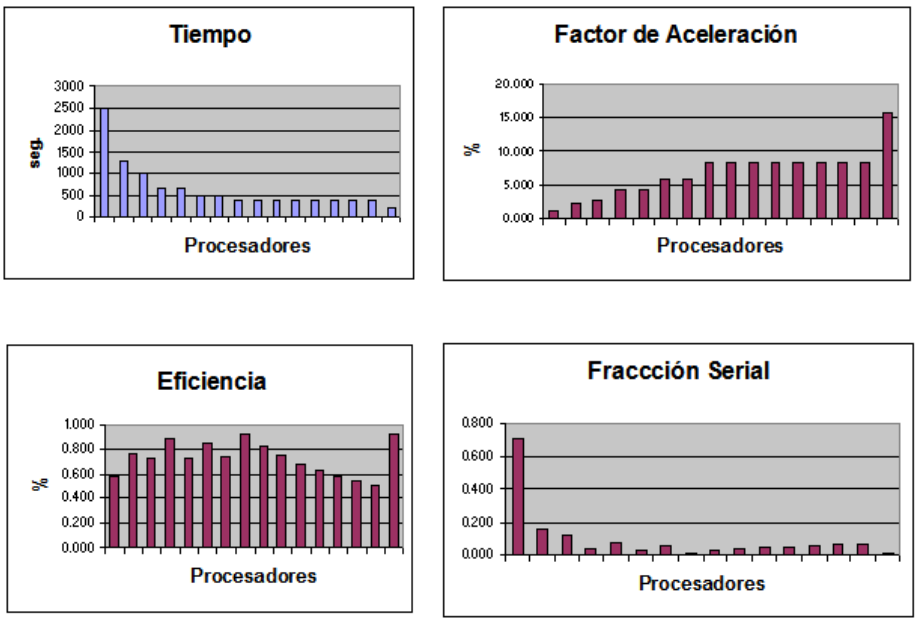

Figura 7: Métricas de desempeño de 2 a 17 procesadores

Primeramente notemos que existe mal balanceo de cargas. La descomposición adecuada del dominio para tener un buen balanceo de cargas se logra cuando se descompone en $n \times m$ nodos en la partición gruesa, generándose $n * m$ subdominios y si se trabaja con P procesadores (1 para el nodo maestro y $P-1$ para los nodos esclavos), entonces el balance de cargas adecuado será cuando $(P-1) \mid (n*m)$.

En nuestro caso se logra un buen balanceo de cargas cuando se tienen $2, 3, 5, 9, 17$ procesadores, cuyas métricas de desempeño se muestran a continuación:

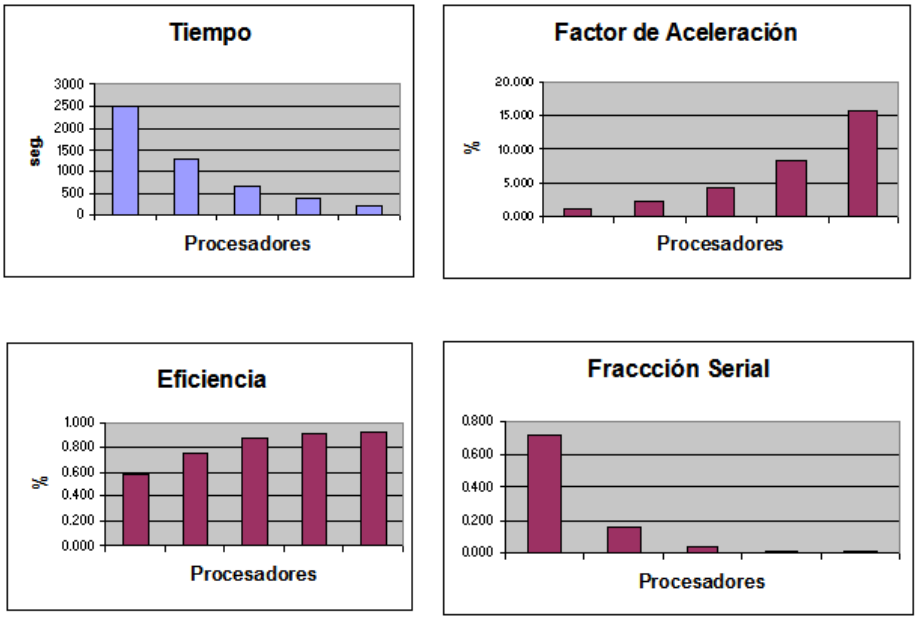

Figura 8: Métricas de desempeño mostrando sólo cuando las cargas están bien balanceadas (2, 3, 5, 9 y 17 procesadores).

En cuanto a las métricas de desempeño, obtenemos que el factor de aceleración en el caso ideal debería de aumentar de forma lineal al aumento del número de procesadores, que en nuestro caso no es lineal pero cumple bien este hecho si están balanceadas las cargas de trabajo.

El valor de la eficiencia deberá ser cercano a uno cuando el hardware es usado de manera eficiente, como es en nuestro caso cuando se tiene un procesador por cada subdominio.

Y en la fracción serial su valor debiera de tender a cero en el caso ideal, siendo este nuestro caso si están balanceadas las cargas de trabajo, de aquí se puede concluir que la granularidad del problema es gruesa, es decir, no existe una sobrecarga en los procesos de comunicación siendo el cluster una buena

herramienta de trabajo para este tipo de problemas.

Finalmente las posibles mejoras de eficiencia para el método de subestructuración en paralelo para disminuir el tiempo de ejecución pueden ser:

- Balanceo de cargas de trabajo homogéneo.
- Al compilar los códigos usar directivas de optimización.
- Usar bibliotecas que optimizan las operaciones en el manejo de los elementos de la matriz usando punteros en las matrices densas o bandadas.
- El cálculo de las matrices que participan en el complemento de Schur pueden ser obtenidas en paralelo

2. Análisis de Rendimiento

Uno de los grandes retos del área de cómputo científico es poder analizar a priori una serie consideraciones dictadas por factores externos al problema de interés que repercuten directamente en la forma de solucionar el problema, estas consideraciones influirán de manera decisiva en la implementación computacional de la solución numérica. Algunas de estas consideraciones son:

- Número de Procesadores Disponibles
- Tamaño y Tipo de Partición del Dominio
- Tiempo de Ejecución Predeterminado

Siendo común que ellas interactúan entre si, de forma tal que normalmente el encargado de la implementación computacional de la solución numérica tiene además de las complicaciones técnicas propias de la solución, el conciliarlas con dichas consideraciones.

Esto deja al implementador de la solución numérica con pocos grados de libertad para hacer de la aplicación computacional una herramienta eficiente y flexible que cumpla con los lineamientos establecidos a priori y permita también que esta sea adaptable a futuros cambios de especificaciones -algo común en ciencia e ingeniería-.

En este capítulo haremos el análisis de los factores que merman el rendimiento de la aplicación y veremos algunas formas de evitarlo, haremos una detallada descripción de las comunicaciones entre el nodo principal y los nodos esclavos, la afectación en el rendimiento al aumentar el número de subdominios en la descomposición y detallaremos algunas consideraciones generales para aumentar el rendimiento computacional. También daremos las conclusiones generales a este trabajo y veremos las diversas ramificaciones que se pueden hacer en trabajos futuros.

2.1. Análisis de Comunicaciones

Para hacer un análisis de las comunicaciones entre el nodo principal y los nodos esclavos en el método de subestructuración es necesario conocer qué se trasmite y su tamaño, es por ello detallaremos en la medida de lo posible las comunicaciones existentes (hay que hacer mención que entre los nodos esclavos no hay comunicación alguna).

Tomando la descripción del algoritmo detallado en el método de subestructuración en paralelo visto en la sección (1.4), suponiendo una partición del dominio Ω en $n \times m$ y $p \times q$ subdominios, las comunicaciones correspondientes a cada inciso son:

A) El nodo maestro transmite 4 coordenadas (en dos dimensiones) correspondientes a la delimitación del subdominio.

B) $2 * p * q$ coordenadas transmite cada objeto *FEM2D Rectángulos* al nodo maestro.

C) A lo más $n * m * 2 * p * q$ coordenadas son las de los nodos de la frontera interior, y sólo aquellas correspondientes a cada subdominio son trasmitidas por el nodo maestro a los objetos *FEM2D Rectángulos* siendo estas a lo más $2 * p * q$ coordenadas.

D) Sólo se envía un aviso de la conclusión del cálculo de las matrices.

E) A lo más $2 * p * q$ coordenadas son transmitidas a los objetos *FEM2D Rectángulos* desde el nodo maestro y los nodos esclavos transmiten al nodo maestro esa misma cantidad información.

F) A lo más $2 * p * q$ coordenadas son transmitidas a los objetos *FEM2D Rectángulos* en los nodos esclavos y estos retornan un número igual al nodo maestro por iteración del método de gradiente conjugado.

G) A lo más $2 * p * q$ valores de la solución de la frontera interior son transmitidas a los objetos *FEM2D Rectángulos* desde el nodo maestro y cada objeto transmite un único aviso de terminación.

I) El nodo maestro manda un aviso a cada objeto *FEM2D Rectángulos* para concluir con el esquema.

La transmisión se realiza mediante paso de arreglos de enteros y números de punto flotante que varían de longitud pero siempre son cantidades pequeñas de estos y se transmiten en forma de bloque, por ello las comunicaciones son eficientes.

2.2. Afectación del Rendimiento al Aumentar el Número de Subdominios en la Descomposición

Una parte fundamental a considerar es la afectación del rendimiento al aumentar el número de subdominios en descomposición, ya que el complemento de Schur local $\underline{\underline{S}}_i = \underline{\underline{A}}_i^{\Sigma\Sigma} - \underline{\underline{A}}_i^{\Sigma I} \left(\underline{\underline{A}}_i^{II}\right)^{-1} \underline{\underline{A}}_i^{I\Sigma}$ involucra el generar las matrices $\underline{\underline{A}}_i^{II}, \underline{\underline{A}}_i^{\Sigma\Sigma}, \underline{\underline{A}}_i^{\Sigma I}, \underline{\underline{A}}_i^{I\Sigma}$ y calcular de alguna forma $\left(\underline{\underline{A}}_i^{II}\right)^{-1}$.

Si el número de nodos interiores en el subdominio es grande entonces obtener la matriz anterior será muy costoso computacionalmente, como se ha mostrado en el transcurso de las últimas secciones del capítulo anterior.

Al aumentar el número de subdominios en una descomposición particular, se garantiza que las matrices a generar y calcular sean cada vez más pequeñas y fáciles de manejar.

Pero hay un límite al aumento del número de subdominio en cuanto a la eficiencia de ejecución, este cuello de botella es generado por el esquema maestro-esclavo y es reflejado por un aumento del tiempo de ejecución al aumentar el número de subdominios en una configuración de hardware particular.

Esto se debe a que en el esquema maestro-esclavo, el nodo maestro deberá de atender todas las peticiones hechas por cada uno de los nodos esclavos, esto

toma especial relevancia cuando todos o casi todos los nodos esclavos compiten por ser atendidos por el nodo maestro.

Por ello se recomienda implementar este esquema en un cluster heterogéneo en donde el nodo maestro sea más poderoso computacionalmente que los nodos esclavos. Si a éste esquema se le agrega una red de alta velocidad y de baja latencia, se le permitirá operar al cluster en las mejores condiciones posibles, pero este esquema se verá degradado al aumentar el número de nodos esclavos inexorablemente. Por ello hay que ser cuidadosos en cuanto al número de nodos esclavos que se usan en la implementación en tiempo de ejecución versus el rendimiento general del sistema al aumentar estos.

2.3. Descomposición Óptima para un Equipo Paralelo Dado.

Otra cosa por considerar es que normalmente se tiene a disposición un número fijo de procesadores, con los cuales hay que trabajar, así que es necesario encontrar la descomposición adecuada para esta cantidad de procesadores. No es posible hacer un análisis exhaustivo, pero mediante pruebas podemos determinar cual es la mejor descomposición en base al tiempo de ejecución.

Para el análisis, consideremos pruebas con $3, 4, 5$ y 6 procesadores y veremos cual es la descomposición más adecuada para esta cantidad de procesadores tomando como referencia el resolver la Ec. (63), usando 513×513 nodos.

Usando para estos cálculos el cluster homogéneo, al resolver por el método de gradiente conjugado precondicionado para cada descomposición se obtuvieron los siguientes resultados:

Partición	Tiempo en 3 Procesadores	Tiempo en 4 Procesadores	Tiempo en 5 Procesadores	Tiempo en 6 Procesadores
2×2 y 256×256	2576 seg.	2084 seg.	1338 seg.	–
4×4 y 128×128	1324 seg.	1071 seg.	688 seg.	688 seg.
8×8 y 64×64	779 seg.	630 seg.	405 seg.	405 seg.
16×16 y 32×32	485 seg.	391 seg.	251 seg.	251 seg.

De estas pruebas se observa que el mal balanceo de cargas es reflejado en los tiempos de ejecución, pese a contar con más procesadores no hay una disminución del tiempo de ejecución.

Ahora para las mismas descomposiciones, usando el cluster heterogéneo para cada descomposición se obtuvieron los siguientes resultados:

Partición	Tiempo en 3 Procesadores	Tiempo en 4 Procesadores	Tiempo en 5 Procesadores	Tiempo en 6 Procesadores
2×2 y 256×256	2342 seg.	1895 seg.	1217 seg.	–
4×4 y 128×128	1204 seg.	974 seg.	626 seg.	626 seg.
8×8 y 64×64	709 seg.	573 seg.	369 seg.	369 seg.
16×16 y 32×32	441 seg.	356 seg.	229 seg.	229 seg.

Primeramente hay que destacar que los nodos esclavos de ambos clusters son comparables en poder de cómputo, pero aquí lo que hace la diferencia es que el nodo maestro del segundo ejemplo tiene mayor rendimiento. Es por ello que al disminuir la fracción serial del problema y atender mejor las comunicaciones que se generan en el esquema maestro-esclavo con todos los objetos *FEM2D Rectángulos* creados en cada uno de los nodos esclavos mejora sustancialmente el tiempo de ejecución.

En ambas pruebas el mal balanceo de cargas es un factor determinante del rendimiento, sin embargo el usó de un cluster en el que el nodo maestro sea más poderoso computacionalmente, hará que se tenga una mejora sustancial en el rendimiento.

Para evitar el mal balanceo de cargas se debe de asignar a cada nodo esclavo una cantidad de subdominios igual. La asignación mínima del número de nodos por subdominio queda sujeta a la velocidad de los procesadores involucrados para disminuir en lo posible los tiempos muertos, obteniendo así el máximo rendimiento.

La asignación máxima del número de nodos por subdominio a cada nodo esclavo, estará en función de la memoria que consuman las matrices que contienen cada uno de los objetos de *FEM2D Rectángulos*. La administración de ellos y las comunicaciones no son un factor limitante y por ello se pueden despreciar.

2.4. Descomposición Fina del Dominio

Supongamos ahora que deseamos resolver el problema de una descomposición fina del dominio Ω en 65537×65537 nodos, este tipo de problemas surgen cotidianamente en la resolución de sistemas reales y las opciones para implantarlo en un equipo paralelo son viables, existen y son actualmente usadas. Aquí las opciones de partición del dominio son muchas y variadas, y la variante seleccionada dependerán fuertemente de las características del equipo de cómputo paralelo del que se disponga, es decir, si suponemos que una descomposición de 1000×1000 nodos en un subdominio consume 1 GB de RAM y el consumo de memoria crece linealmente con el número de nodos, entonces algunas posibles descomposiciones son:

Procesadores	Descomposición	Nodos Subdominio	RAM Mínimo
5	2×2 y 32768×32768	32768×32768	≈33.0 GB
257	16×16 y 4096×4096	4096×4096	≈4.0 GB
1025	32×32 y 2048×2048	2048×2048	≈2.0 GB
4097	64×64 y 1024×1024	1024×1024	≈1.0 GB

Notemos que para las primeras particiones, el consumo de RAM es excesivo y en las últimas particiones la cantidad de procesadores en paralelo necesarios es grande (pero ya de uso común en nuestros días). Como en general, contar con equipos paralelos de ese tamaño es en extremo difícil, ¿es posible resolver este tipo de problemas con una cantidad de procesadores menor al número sugerido y donde cada uno de ellos tiene una memoria muy por debajo de lo sugerido?, la respuesta es si.

Primero, notemos que al considerar una descomposición del tipo 64×64 y 1024×1024 subdominios requerimos aproximadamente 1.0 GB de RAM mínimo por nodo, si suponemos que sólo tenemos unos cuantos procesadores con memoria limitada (digamos 2 GB), entonces no es posible tener en memoria de manera conjunta a las matrices generadas por el método.

Una de las grandes ventajas de los métodos de descomposición de domino es que los subdominios son en principio independientes entre si y que sólo están acoplados a través de la solución en la interfaz de los subdominios que es desconocida.

Como sólo requerimos tener en memoria la información de la frontera interior, es posible bajar a disco duro todas las matrices y datos complementarios (que consumen el 99 % de la memoria del objeto *FEM2D Rectángulos*) generados por cada subdominio que no se requieran en ese instante para la operación del esquema maestroesclavo.

Recuperando del disco duro solamente los datos del subdominio a usarse en ese momento (ya que el proceso realizado por el nodo maestro es secuencial) y manteniéndolos en memoria por el tiempo mínimo necesario. Así, es posible resolver un problema de una descomposición fina, usando una cantidad de procesadores fija y con una cantidad de memoria muy limitada por procesador.

En un caso extremo, la implementación para resolver un dominio Ω descompuesto en un número de nodos grande es posible implementarla usando sólo dos procesos en un procesador, uno para el proceso maestro y otro para el proceso esclavo, en donde el proceso esclavo construiría las matrices necesarias por cada subdominio y las guardaría en disco duro, recuperándolas conforme el proceso del nodo maestro lo requiera. Nótese que la descomposición del domino Ω estará sujeta a que cada subdominio Ω_i sea soportado en memoria conjuntamente con los procesos maestro y esclavo.

De esta forma es posible resolver un problema de gran envergadura usando recursos computacionales muy limitados, sacrificando velocidad de procesamiento en aras de poder resolver el problema. Está es una de las grandes ventajas de los métodos de descomposición de dominio con respecto a los otros métodos de discretización tipo diferencias finitas y elemento finito.

El ejemplo anterior nos da una buena idea de las limitantes que existen en la resolución de problemas con dominios que tienen una descomposición fina y nos pone de manifiesto las características mínimas necesarias del equipo paralelo para soportar dicha implantación.

2.5. Consideraciones para Aumentar el Rendimiento

Algunas consideraciones generales para aumentar el rendimiento son:

a) Balanceo de cargas de trabajo homogéneo, si se descompone en $n \times m$ subdominios en la partición gruesa se y si se trabaja con P procesadores, entonces el balance de cargas adecuado será cuando $(P-1) \mid (n*m)$.), ya que de no hacerlo el rendimiento se ve degradado notoriamente.

b) Usar el mejor precondicionador a priori disponible para el problema en particular, ya que esto disminuye sustancialmente el número de iteraciones (hasta en un orden de magnitud cuando el precondicionador es óptimo).

c) Usar la biblioteca Lapack++ de licencia GNU que optimiza las operaciones en el manejo de los elementos de la matriz usando punteros en donde se usan matrices densas y bandadas.

d) Si se cuenta con un equipo en que cada nodo del cluster tenga más de un procesador, usar bibliotecas (PLAPACK por ejemplo) que permitan paralelizar mediante el uso de procesos con memoria compartida, Pipeline o hilos, las operaciones que involucren a vectores y matrices; como una forma de mejorar el rendimiento del programa.

e) Siempre usar al momento de compilar los códigos, directivas de optimización (estas ofrecen mejoras de rendimiento en la ejecución de 30 % aproximadamente en las pruebas realizadas), pero existen algunos compiladores con optimizaciones especificas para ciertos procesadores (Intel compiler para 32 y 64 bits) que pueden mejorar aun más este rendimiento (más de 50 %).

Todas estas mejoras pueden ser mayores si se usa un nodo maestro del mayor poder computacional posible aunado a una red en el cluster de de 1 Gb o mayor y de ser posible de baja latencia, si bien las comunicaciones son pocas, estas pueden generar un cuello de botella sustancial.

Por otro lado, hay que hacer notar que es posible hacer usó de múltiples etapas de paralelización que pueden realizarse al momento de implantar el método de descomposición de dominio, estas se pueden implementar conforme se necesite eficiencia adicional, pero implica una cuidadosa planeación al momento de hacer el análisis y diseño de la aplicación y una inversión cuantiosa en tiempo para implantarse en su totalidad, estas etapas se describen a continuación:

- El propio método de descomposición de dominio ofrece un tipo particular y eficiente de paralelización al permitir dividir el dominio en múltiples subdominios independientes entre si, interconectados sólo por la frontera interior.
- A nivel subdominio otra paralelización es posible, específicamente en el llenado de las matrices involucradas en el método de descomposición de dominio, ya que varias de ellas son independientes.
- A nivel de los cálculos, entre matrices y vectores involucrados en el método también se pueden paralelizar de manera muy eficiente.
- A nivel del compilador es posible generar el ejecutable usando esquemas de paralelización automático y opciones para eficientizar la ejecución.

Por lo anterior es posible usar una serie de estrategias que permitan realizar estas etapas de paralelización de manera cooperativa y aumentar la eficiencia

en una factor muy significativo, pero esto implica una programación particular para cada una de las etapas y poder distribuir las tareas paralelas de cada etapa en uno o más procesadores distintos a los de las otras etapas.

Notemos finalmente que si se toma el programa de elemento finito y se paraleliza usando sólo directivas de compilación, el aumento del rendimiento es notorio pero este se merma rápidamente al aumentar del número de nodos (esto es debido al aumento en las comunicaciones para mantener y acceder a la matriz del sistema algebraico asociado al método). Pero es aun más notorio cuando el método de descomposición de dominio serial usando las mismas directivas de compilación se paraleliza (sin existir merma al aumentar el número de nodos siempre y cuando las matrices generadas estén el la memoria local del procesador).

Esto se debe a que en el método de elemento finito la matriz estará distribuida por todos los nodos usando memoria distribuida, esto es muy costoso en tiempo de cómputo ya que su manipulación requiere de múltiples comunicaciones entre los procesadores, en cambio el el método de descomposición de dominio ya están distribuidas las matrices en los nodos y las operaciones sólo involucran transmisión de un vector entre ellos, minimizando las comunicaciones entre procesadores.

Pero aún estos rendimientos son pobres con respecto a los obtenidos al usar el método de descomposición de dominio paralelizado conjuntamente con bliotecas para manejo de matrices densas y dispersas en equipos con nodos que cuenten con más de un procesador, en donde mediante el uso de memoria compartida se pueden usar el resto de los procesadores dentro del nodo para efectuar en paralelo las operaciones en donde estén involucradas las matrices.

3. Bibliografía

Referencias

[1] A. Quarteroni, A. Valli; *Domain Decomposition Methods for Partial Differential Equations*. Clarendon Press Oxford 1999.

[2] A. Toselli, O. Widlund; *Domain Decomposition Methods - Algorithms and Theory*. Springer, 2005.

[3] B. D. Reddy; *Introductory Functional Analysis - With Applications to Boundary Value Problems and Finite Elements*. Springer 1991.

[4] B. F. Smith, P. E. Bjørstad, W. D. Gropp; *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[5] B. I. Wohlmuth; *Discretization Methods and Iterative Solvers Based on Domain Decomposition*. Springer, 2003.

[6] I. Foster; *Designing and Building Parallel Programs*. Addison-Wesley Inc., Argonne National Laboratory, and the NSF, 2004.

[7] G. Herrera; Análisis de Alternativas al Método de Gradiente Conjugado para Matrices no Simétricas. Tesis de Licenciatura, Facultad de Ciencias, UNAM, 1989.

[8] I. Herrera, M. Díaz; *Modelación Matemática de Sistemas Terrestres* (Notas de Curso en Preparación). Instituto de Geofísica, (UNAM).

[9] I. Herrera; *Un Análisis del Método de Gradiente Conjugado*. Comunicaciones Técnicas del Instituto de Geofísica, UNAM; Serie Investigación, No. 7, 1988.

[10] I. Herrera; *Método de Subestructuración* (Notas de Curso en Preparación). Instituto de Geofísica, (UNAM).

[11] J. II. Bramble, J. E. Pasciak and A. II Schatz. *The Construction of Preconditioners for Elliptic Problems by Substructuring*. I. Math. Comput., 47, 103-134,1986.

[12] J. L. Lions & E. Magenes; *Non-Homogeneous Bonduary Value Problems and Applications Vol. I*, Springer-Verlag Berlin Heidelber New York 1972.

[13] K. Hutter & K. Jöhnk; *Continuum Methods of Physical Modeling.* Springer-Verlag Berlin Heidelber New York 2004.

[14] L. F. Pavarino, A. Toselli; *Recent Developments in Domain Decomposition Methods*. Springer, 2003.

[15] M.B. Allen III, I. Herrera & G. F. Pinder; *Numerical Modeling in Science And Engineering.* John Wiley & Sons, Inc . 1988.

[16] M. Diaz; *Desarrollo del Método de Colocación Trefftz-Herrera Aplicación a Problemas de Transporte en las Geociencias.* Tesis Doctoral, Instituto de Geofísica, UNAM, 2001.

[17] M. Diaz, I. Herrera; *Desarrollo de Precondicionadores para los Procedimientos de Descomposición de Dominio.* Unidad Teórica C, Posgrado de Ciencias de la Tierra, 22 pags, 1997.

[18] P.G. Ciarlet, J. L. Lions; *Handbook of Numerical Analysis, Vol. II.* North-Holland, 1991.

[19] R. L. Burden y J. D. Faires; *Análisis Numérico.* Math Learning, 7 ed. 2004.

[20] S. Friedberg, A. Insel, and L. Spence; *Linear Algebra*, 4th Edition, Prentice Hall, Inc. 2003.

[21] W. Gropp, E. Lusk, A. Skjellem, *Using MPI, Portable Parallel Programming Whit the Message Passing Interface.* Scientific and Engineering Computation Series, 2ed, 1999.

[22] W. Rudin; *Principles of Mathematical Analysis.* McGraw-Hill International Editions, 1976.

[23] X. O. Olivella, C. A. de Sacribar; *Mecánica de Medios Continuos para Ingenieros.* Ediciones UPC, 2000.

[24] Y. Saad; *Iterative Methods for Sparse Linear Systems.* SIAM, 2 ed. 2000.

[25] Y. Skiba; *Métodos y Esquemas Numéricos, un Análisis Computacional.* UNAM, 2005.