Esta tesis doctoral contiene un índice que enlaza a cada uno de los capítulos de la misma.
Existen asimismo botones de retorno al índice al principio y final de cada uno de los capítulos.

Ir directamente al índice

Para una correcta visualización del texto es necesaria la versión de Adobe Acrobat Reader 7.0 o posteriores

Aquesta tesi doctoral conté un índex que enllaça a cadascun dels capítols. Existeixen així mateix botons de retorn a l'index al principi i final de cadascun deis capítols .

Anar directament a l'index

Per a una correcta visualització del text és necessària la versió d' Adobe Acrobat Reader 7.0 o posteriors.

Universidad de Alicante

Departamento de Ciencia de la Computación e Inteligencia Artificial

Métodos iterativos paralelos para la resolución de sistemas lineales hermíticos y definidos positivos

Memoria presentada para optar al grado de Doctora Ingeniera en Informática por

María Jesús Castel de Haro,

subvencionada por el proyecto DGSIC PB98-0977

Alicante, mayo de 2000.

Doña VIOLETA MIGALLÓN GOMIS y Don JOSÉ PENADÉS MARTÍNEZ, Titulares de Universidad del Departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante,

CERTIFICAN:

Que la presente memoria *Métodos iterativos paralelos para la resolución de sistemas lineales hermíticos y definidos positivos*, ha sido realizada bajo su dirección, en el Departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante, por la Licenciada María Jesús Castel de Haro, y constituye su tesis para optar al grado de Doctora Ingeniera en Informática.

Para que conste, en cumplimiento de la legislación vigente, autorizan la presentación de la referida tesis doctoral ante la comisión de Doctorado de la Universidad de Alicante, firmando el presente certificado.

Alicante, mayo de 2000.

Fdo.: Violeta Migallón Gomis. José Penadés Martínez.

A mi padre,
que siempre estará en mi recuerdo.

A mi madre, gracias por todo.

A Pepe, por quererme y estar a mi lado.

A María, mi estrellita de la suerte.

Quiero expresar mi más sincero agradecimiento a los profesores Violeta Migallón Gomis y José Penadés Martínez por su constante orientación, sin cuya ayuda y dirección habría sido imposible la realización de esta memoria, así como a los restantes miembros del Grupo de Computación en Paralelo del Departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante por haberme transmitido el ambiente agradable de trabajo que existe entre ellos, y muy especialmente al profesor Josep Arnal García con el que he compartido alegrías y penas al haber elaborado su tesis doctoral simultáneamente a la mía.

Asimismo, quiero agradecer al profesor Daniel B. Szyld la lectura previa de esta memoria y sus inestimables consejos.

Además, deseo agradecer a los técnicos Ginés López Sevilla y Francisco Martínez su gran ayuda y asesoramiento en los problemas surgidos en la utilización de las máquinas paralelas.

Índice

Prólogo

Con la aparición de la computación paralela, se ha logrado cubrir una necesidad producida por la gran demanda de computadores de gran velocidad destinados a realizar grandes cálculos tanto en el campo de la ciencia como en el de la industria. Este tipo de computación, tiene una de sus aplicaciones más importantes en el contexto matemático, en particular, en el área del Álgebra Lineal Numérica. La introducción del paralelismo en este área, ha hecho que se diseñaran diversos métodos y algoritmos, como por ejemplo, algoritmos para cálculos matriciales básicos, factorizaciones básicas del álgebra lineal, métodos de resolución de ecuaciones lineales y algoritmos para el cálculo de valores propios.

Centrando nuestra atención en los métodos iterativos, parte de los avances en la programación paralela se producen en el diseño de algoritmos paralelos para ejecutar los métodos secuenciales existentes. Para ello, se han utilizado técnicas como la descomposición por bloques, y también se han desarrollado técnicas para la distribución de datos y cálculos entre los distintos procesadores

que intervienen en su ejecución (ver por ejemplo [81]). Además, motivados por las ventajas de la computación paralela, se han desarrollado nuevos algoritmos diseñados expresamente para su ejecución en paralelo, como podemos ver, por ejemplo, en [15], [18], [19] y [67].

En esta memoria, considerando las ventajas que supone la resolución de sistemas de ecuaciones lineales en paralelo, estudiamos métodos no estacionarios basados en la técnica de multipartición y en la técnica de dos etapas en el contexto de las matrices hermíticas y definidas positivas. Estos resultados nos permitirán, además, establecer la base necesaria para la construcción de precondicionadores paralelos para el método del gradiente conjugado basados en la técnica de dos etapas.

El Capítulo 1, *Métodos iterativos secuenciales*, está compuesto de ocho secciones, donde se describen y citan distintos resultados que utilizaremos en el desarrollo de esta memoria. Así, después de una breve introducción, en la Sección 1.2 se describen los conceptos relativos a normas vectoriales y matriciales; en la Sección 1.3, tenemos los conceptos de partición de una matriz y de esquema iterativo; los métodos iterativos clásicos como el de Jacobi, Gauss-Seidel, SOR y SSOR aparecen en la Sección 1.4; por otro lado, el método iterativo en dos etapas y los métodos del gradiente conjugado y gradiente conjugado precondicionado, que serán básicos en el desarrollo de esta memoria, se expondrán en las Secciones 1.5, 1.6 y 1.7, respectivamente. Para terminar este capítulo, en la Sección 1.8 se describen dos precondicionadores secuenciales, los precondicionadores polinomiales y los basados en la factorización incompleta de Choleski.

El Capítulo 2, *Arquitecturas paralelas*, contiene una pequeña introducción a los sistemas de computación paralela, dando una descripción de las máquinas y

la programación desde sus comienzos hasta llegar a nuestros días con la introducción del paralelismo.

En el Capítulo 3, *Métodos iterativos paralelos. Planteamiento y objetivos*, se revisan distintas técnicas de distribución de incógnitas, para realizar la implementación de métodos iterativos secuenciales, en paralelo. En particular, nos centraremos en los métodos iterativos basados en la técnica de multipartición, (ver Secciones 3.2 y 3.3). La Sección 3.4, se dedica a diversos precondicionadores paralelos, entre ellos, los precondicionadores polinomiales aditivos y los precondicionadores por bloques. Para terminar, en la Sección 3.5 se plantean los objetivos de esta memoria vistos desde la perspectiva del conocimiento de los capítulos previos.

Los Capítulos 4, 5, 6 y 7, constituyen la parte principal de nuestra investigación. Los Capítulos 4, 5 y 6 (*Métodos de multipartición no estacionarios*, *Métodos paralelos en dos etapas* y *Precondicionadores basados en los métodos en dos etapas*, respectivamente), son los que contienen los resultados teóricos presentados en esta memoria. Los Capítulos 4 y 5 tienen básicamente la misma estructura. En ellos se hace, en primer lugar, una introducción en la que se describe el método que es motivo de estudio en dicho capítulo, junto con los resultados de convergencia existentes que han motivado el estudio del método en cuestión. Después se analizan dichos métodos cuando la matriz del sistema es hermítica y definida positiva, y por último, se escriben las conclusiones obtenidas en cada uno de ellos.

El Capítulo 6 se centra en el estudio de precondicionadores paralelos, concretamente y después de una breve motivación, en la Sección 6.2, se construye un precondicionador basado en el método en dos etapas que sigue la idea del

precondicionamiento por series truncadas. Posteriormente, en la Sección 6.3 se obtiene un precondicionador basado en el de la Sección 6.2, en el que la partición interna se obtiene mediante la factorización incompleta de Choleski.

En el Capítulo 7, *Experimentos numéricos*, se hace un estudio experimental de los métodos estudiados en los capítulos anteriores. La descripción de los dos multiprocesadores utilizados, concretamente un IBM RS 6000/SP y un cluster de Pentiums, se puede ver en la Sección 7.2.

Esta memoria se finaliza con dos apéndices. El primero está dedicado a las líneas futuras de trabajo y el segundo a las referencias bibliográficas utilizadas para la elaboración de esta memoria.

Debemos decir, que todos los conceptos y resultados en los que nos basamos para realizar las demostraciones que se presentan en esta memoria, vienen referenciados para la consulta de sus posibles demostraciones.

Capítulo 1

Métodos iterativos secuenciales.

1.1 Introducción

Consideremos el sistema de ecuaciones lineales

$$Ax = b, \tag{1.1}$$

donde A es una matriz invertible de tamaño $n \times n$ y b un vector columna de tamaño n. Para resolver el sistema de ecuaciones lineales (1.1) existen básicamente dos familias de métodos, los *métodos directos* y los *métodos iterativos*.

Los métodos directos permiten obtener la solución exacta del sistema (1.1), salvo errores de redondeo, mediante un número finito de operaciones. Si la matriz es densa el número de operaciones depende del tamaño de la matriz de coeficientes, si por el contrario, la matriz es dispersa el número de operaciones necesarias para resolver el sistema dependerá del número de elementos no nulos

en la matriz de coeficientes y de la estructura del grafo asociado a dicha matriz. En general, los métodos directos se basan en la realización de sucesivas transformaciones algebraicas sobre la matriz inicial A, de forma que la resolución del sistema (1.1) se reduzca a resolver sistemas triangulares. Entre ellos destacamos el método de Gauss y su variante de Gauss-Jordan, la descomposición LU, que consiste simplemente en una generalización del tratamiento matricial del método de Gauss, y la factorización de Choleski, más ventajosa que la descomposición LU y sólo aplicable al caso de sistemas cuya matriz sea hermítica y definida positiva, (ver por ejemplo, [31]).

En los métodos iterativos no se obtiene la solución exacta del sistema, sino una aproximación, pero la importancia de los errores de redondeo disminuye, y aunque éste no es el factor decisivo para la elección de dichos métodos, sin duda, influye en la decisión. Además, cabe destacar, que en el caso de matrices dispersas, en los métodos directos se origina un llenado de la matriz al realizar la factorización, y por tanto el coste es mucho mayor que en los métodos iterativos. Por otra parte, tenemos que decir, que en los métodos iterativos no se conoce el número exacto de operaciones requeridas para obtener una solución y que el tiempo de ejecución dependerá, a veces, de la elección de unos determinados parámetros. De entre todos los métodos iterativos, podemos citar, Jacobi, Gauss-Seidel, SOR y SSOR, ... , (ver por ejemplo, [81] ó [105]). El método del gradiente conjugado (ver [81]), aunque es un método directo y generalmente obtiene una buena aproximación a la solución del sistema en menos de n pasos, debido a su estructura, al número de operaciones que realiza para obtener la solución y a que los errores de redondeo le impiden en muchos casos alcanzar la solución exacta en un número finito de pasos, se trata como un método iterativo.

Atendiendo al contenido de las secciones que aparecen en este capítulo, damos una breve descripción de cada una de ellas. En primer lugar, en la Sección 1.2, introducimos los conceptos de matriz simétrica, hermítica y definida positiva. Además, también damos una breve descripción de normas vectoriales y matriciales y, para finalizar, el concepto de radio espectral que usaremos con bastante frecuencia en esta memoria. En la Sección 1.3, basándonos en el concepto de partición de una matriz, planteamos los métodos iterativos en general y damos resultados de convergencia teniendo en cuenta tanto los distintos tipos de particiones de la matriz A, como las propiedades de dicha matriz de coeficientes. En la Sección 1.4 describimos los métodos iterativos clásicos: Jacobi, Gauss-Seidel, SOR y SSOR, junto con algunos resultados de convergencia más conocidos. En la Sección 1.5 describimos un método iterativo para la resolución de sistemas de ecuaciones lineales, denominado método iterativo en dos etapas (ver [76]). Luego, en la Sección 1.6 describimos el método del gradiente conjugado, cuya generalización mediante la introducción de precondicionadores, dará lugar al método del gradiente conjugado precondicionado que veremos en la Sección 1.7. Por último, en la Sección 1.8 describimos los precondicionadores polinomiales y los precondicionadores basados en la factorización incompleta de Choleski.

1.2 Conceptos y resultados básicos

En esta sección exponemos algunos conceptos y resultados del Álgebra Lineal que utilizaremos en esta memoria. Para ello, denotamos por $\mathbb{K}$ el cuerpo de los números reales $\mathbb{R}$ o el de los complejos $\mathbb{C}$. En primer lugar, introduciremos la notación relativa a vectores y la relativa a matrices, en particular describiremos las matrices simétricas, hermíticas y las definidas positivas. Después daremos resultados relativos a normas vectoriales y matriciales sobre el cuerpo $\mathbb{K}$. Por último, y relacionado con el concepto de norma matricial, introduciremos la definición de radio espectral de una matriz y daremos algunos resultados que lo relacionan con la norma matricial.

Para todo $x \in \mathbb{C}^n$, denotamos por $|x|$ el vector cuyas componentes son los módulos de las correspondientes componentes de x. Los vectores x^T y x^H indican el *traspuesto* y el *conjugado traspuesto* del vector x, respectivamente. Un vector x real es *positivo* (*no negativo*) y escribimos $x > 0$ ($x \geq 0$), si tiene todas sus componentes estrictamente mayores que 0 (mayores o iguales que 0). Diremos $x \leq y$, si $y - x \geq 0$. Análogamente, $x < y$, si $y - x > 0$.

La notación para matrices se define de forma similar. Sea $A \in \mathbb{C}^{n\times n}$, denotamos por $|A|$ la matriz cuyas componentes son los módulos de las correspondientes componentes de A, y por A^T y A^H las matrices traspuesta y conjugada traspuesta de A, respectivamente.

Definición 1.1. Una matriz $A \in \mathbb{C}^{n\times n}$ es *simétrica* si se verifica $A = A^T$.

Definición 1.2. Una matriz $A \in \mathbb{C}^{n\times n}$ es *hermítica* si se verifica $A = A^H$.

Hacemos notar que si A es real se verifica que $A^T = A^H$, por tanto, una matriz real simétrica es un caso especial de una matriz hermítica.

A continuación enunciamos algunas propiedades para las matrices hermíticas que podemos ver, por ejemplo, en [57].

1.- Sea $A \in \mathbb{C}^{n\times n}$, entonces las matrices $A + A^H$, AA^H, A^HA también lo son.

2.- Si $A \in \mathbb{C}^{n\times n}$ es una matriz hermítica, la matriz A^k es hermítica, para todo $k = 1, 2, 3, \ldots$. Si además A es no singular, entonces A^{-1} es hermítica.

3.- Si $A, B \in \mathbb{C}^{n\times n}$ son matrices hermíticas, la matriz $aA + bB$ es hermítica, para todo $a, b \in \mathbb{R}$.

4.- Si $A \in \mathbb{C}^{n\times n}$ es una matriz hermítica entonces, los elementos de la diagonal principal de A son todos reales.

El siguiente teorema da propiedades sobre las matrices hermíticas.

Teorema 1.1. *(Teorema 4.1.3, [57]). Sea $A \in \mathbb{C}^{n\times n}$ una matriz hermítica, entonces*

1) x^HAx es real, para todo $x \in \mathbb{C}^n$.

2) Todos los valores propios de A son reales.

3) S^HAS es una matriz hermítica para toda matriz $S \in \mathbb{C}^{n\times n}$.

Relacionada con los conceptos de matriz simétrica y hermítica está la propiedad de que una matriz sea definida positiva. Las siguientes definiciones se pueden consultar, por ejemplo, en Horn y Johnson [57] o en Ortega [81].

Definición 1.3. Una matriz $A \in \mathbb{C}^{n\times n}$ es *definida positiva* si la parte real de $x^H Ax$ es mayor que cero, para todo vector complejo $x \neq 0$.

Si la desigualdad anterior se debilita tenemos la siguiente definición.

Definición 1.4. Una matriz $A \in \mathbb{C}^{n\times n}$ es *semidefinida positiva* si la parte real de $x^H Ax$ es no negativa, para todo vector complejo $x \neq 0$.

Como podemos observar, claramente si una matriz es definida positiva es también semidefinida positiva.

Enunciamos algunas de las propiedades de las matrices complejas definidas positivas.

1.- Cualquier submatriz principal de una matriz definida positiva es definida positiva.

2.- La suma de dos matrices definidas positivas, es definida positiva. De forma más general, cualquier combinación lineal no negativa de matrices semidefinidas positivas es semidefinida positiva.

3.- Si $A \in \mathbb{C}^{n\times n}$ es una matriz definida positiva, entonces las matrices A^T, A^H y A^{-1} son definidas positivas.

Como podemos observar en la Definición 1.3, para demostrar que una matriz hermítica es definida positiva, es suficiente demostrar que $x^H Ax > 0$, para todo vector $x \neq 0$. Para referirnos a las matrices hermíticas y definidas positivas (semidefinidas positivas), usaremos la notación $A \succ 0$, $(A \succeq 0)$. Además dadas dos matrices hermíticas B y C, escribiremos $B \succ C$, $(B \succeq C)$ si $B - C$ es definida positiva (semidefinida positiva) (ver, por ejemplo, [64] y [74]).

Los dos teoremas que se dan a continuación, dan condiciones necesarias y suficientes para demostrar cuándo una matriz es definida o semidefinida positiva, se pueden ver, por ejemplo, en Horn y Johnson [57].

Teorema 1.2. *(Teorema 2.9, [57]). Una matriz A es definida positiva si, y sólo si, la matriz hermítica $A + A^H$ es definida positiva.*

Teorema 1.3. *(Teorema 7.2.1, [57]). Sea $A \in \mathbb{C}^{n\times n}$ una matriz hermítica, A es semidefinida positiva si, y sólo si, todos sus valores propios son no negativos. A es definida positiva si, y sólo si, todos sus valores propios son positivos.*

A continuación se exponen conceptos relativos a *normas vectoriales* y *normas matriciales*, definidos sobre el cuerpo $\mathbb{K}$. Todos estos resultados se pueden ver, por ejemplo, en [57].

Definición 1.5. Sea $\mathcal{V}$ un espacio vectorial sobre el cuerpo $\mathbb{K}$. Diremos que la funcion $\|\cdot\| : \mathcal{V} \longrightarrow \mathbb{R}$ es una *norma vectorial*, si para todo $x, y \in \mathcal{V}$, se verifican las siguientes propiedades:

1.- $\|x\| \geq 0$.

2.- $\|x\| = 0$, si, y sólo si, $x = 0$.

3.- $\|x\| = |c|\|x\|, \quad \forall c \in \mathbb{K}$.

4.- $\|x + y\| \leq \|x\| + \|y\|$.

Damos a continuación algunos ejemplos conocidos de normas vectoriales.

- Las l_p–normas, que para un vector $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{C}^n$ se definen como

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}, \quad p \geq 1.$$

Nótese que la l_2–norma es la conocida norma euclídea.

- La norma infinito, definida para un vector $x \in \mathbb{C}^n$, de la siguiente forma

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|. \tag{1.2}$$

- Sea $A \in \mathbb{C}^{n \times n}$ una matriz hermítica y definida positiva, dados los vectores $x, y \in \mathbb{C}^n$, la expresión $\langle x, y \rangle = (x^H A y)$ define un producto interno sobre $\mathbb{C}^n$, por lo tanto, $||x||_A = (x^H A x)^{\frac{1}{2}}$ es una norma vectorial en $\mathbb{C}^n$.

Todas estas normas tienen la propiedad de ser *monótonas*, es decir, si v y w son vectores de $\mathbb{C}^n$ tales que $|v| \leq |w|$, entonces $\|v\| \leq \|w\|$.

Designamos por $\mathcal{M}_n$ el espacio vectorial formado por todas las matrices cuadradas de tamaño $n \times n$ definidas sobre un cuerpo $\mathbb{K}$. Definimos en dicho espacio vectorial las normas matriciales.

Definición 1.6. Diremos que la función $\|\cdot\| : \mathcal{M}_n \longrightarrow \mathbb{R}$ es una *norma matricial* si para todo par de matrices $A, B \in \mathcal{M}_n$, se verifican los siguientes axiomas:

1.- $\|A\| \geq 0$.

2.- $\|A\| = 0$, si, y sólo si, $A = O$.

3.- $\|cA\| = |c|\|A\|, \quad \forall c \in \mathbb{K}$.

4.- $\|A+B\| \leq \|A\| + \|B\|$.

5.- $\|AB\| \leq \|A\|\|B\|$.

Hacemos notar que una norma matricial es una norma vectorial que satisface además el axioma (5) o propiedad *submultiplicativa.*

Para estudiar la convergencia de los métodos iterativos paralelos, que planteamos en los Capítulos 4, 5 y 6, se han utilizado en ocasiones normas matriciales que se deducen de las normas vectoriales utilizando la siguiente definición, (ver, por ejemplo, [57]).

Definición 1.7. Sea $\|\cdot\|$ una norma vectorial sobre $\mathbb{K}^n$. Se llama *norma matricial inducida* por dicha norma vectorial a la siguiente función definida sobre $\mathcal{M}_n$.

$$\|A\| = \max_{\|x\|=1} \|Ax\|.$$

La norma introducida en la Definición 1.7 se puede calcular de las siguientes formas equivalentes

$$\|A\| = \max_{\|x\|=1} \|Ax\| = \max_{\|x\|\leq 1} \|Ax\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|}.$$

Toda norma matricial inducida por una norma vectorial es *compatible*, es decir, verifica la siguiente desigualdad

$$\|Ax\| \leq \|A\|\|x\|, \quad x \in \mathbb{K}^n.$$

Además, si denotamos por I a la matriz identidad y $\|\cdot\|$ es una norma compatible, entonces $\|I\| = 1$.

Como ejemplos de normas matriciales inducidas por normas vectoriales, damos las que nos serán de utilidad en el desarrollo de esta memoria.

- La *norma matricial infinito*, inducida por la norma vectorial infinito

$$\|A\|_\infty = \max_{1\le i\le n}\left(\sum_{j=1}^{n}|a_{ij}|\right), \quad \text{si } A=[a_{ij}]_{1\le i,j\le n}.$$

- La norma matricial inducida por la norma vectorial $||.||_A$, donde A es una matriz hermítica y definida positiva.

Otro concepto de gran importancia para el estudio de la convergencia de los métodos iterativos, es el de *radio espectral* de una matriz.

Definición 1.8. Sea A una matriz cuadrada de tamaño $n \times n$, definimos el *radio espectral de* A, y lo denotamos por $\rho(A)$, como el máximo de los módulos de los valores propios de A, es decir

$$\rho(A) = \max_i\{|\lambda_i| \ : \ \lambda_i \text{ es valor propio de } A\}.$$

Los dos teoremas que se dan a continuación se pueden ver, por ejemplo, en Varga [98] y Young [105]. Estos teoremas relacionan el radio espectral de una matriz con su norma. El siguiente teorema nos indica que cualquier norma matricial es una cota superior del radio espectral.

Teorema 1.4. *(Teorema 3.4, [105]). Si* $\|\cdot\|$ *es una norma matricial y* $A \in \mathcal{M}_n$, *entonces se satisface* $\rho(A) \le \|A\|$.

El teorema que se enuncia a continuación, asegura que para toda matriz se puede encontrar una norma matricial de forma que dicha norma esté tan cerca del radio espectral como queramos.

Teorema 1.5. *(Teorema 3.5, [105]). Sea* $A \in \mathcal{M}_n$. *Para todo* $\epsilon > 0$, *existe al menos una norma matricial* $\|\cdot\|$ *tal que* $\rho(A) \le \|A\| \le \rho(A) + \epsilon$.

Además (ver, por ejemplo, Horn y Johnson [57]), dada una norma matricial $\|\cdot\|_\alpha$ existe siempre una norma matricial inducida $\|\cdot\|_\beta$, y por tanto compatible, que verifica

$$\|A\|_\beta \leq \|A\|_\alpha.$$

Por tanto, siempre se puede encontrar una norma matricial inducida satisfaciendo el Teorema 1.5.

1.3 Planteamiento y convergencia

Una vez revisados los conceptos del Álgebra Lineal que necesitaremos en el desarrollo de esta memoria, vamos a pasar a estudiar los conceptos más generales sobre los métodos iterativos secuenciales.

Los métodos iterativos para la resolución de un sistema de ecuaciones lineales $Ax = b$, consisten en generar, a partir de un vector inicial $x^{(0)}$, una sucesión de vectores $\{x^{(l)}\}_{l\geq 1}$ que converja a la solución exacta $\xi = A^{-1}b$ del mismo. Se dice que dicho método es *convergente* si para todo vector inicial $x^{(0)}$, se cumple que $\lim_{l\to\infty} x^{(l)} = \xi$.

El concepto de norma vectorial, dado en la Definición 1.5 (Sección 1.2), se utiliza para el estudio de la convergencia de una sucesión de vectores de la siguiente forma.

Definición 1.9. Sea $\mathcal{V}$ un espacio vectorial sobre el cuerpo $\mathbb{K}$ y sea $\|\cdot\|$ una norma vectorial sobre $\mathcal{V}$. Decimos que la sucesión de vectores $\{x^{(l)}\}_{l\geq 1}$ *converge*

al vector $x \in \mathcal{V}$ con respecto a la norma $\|\cdot\|$, si, y sólo si, la sucesión de números reales $\{\|x^{(l)} - x\|\}_{l\geq 1}$ tiende a 0 cuando l tiende a infinito.

En los espacios de dimensión finita todas las normas son *equivalentes*, en el sentido en que si una sucesión $\{x^{(l)}\}_{l\geq 1}$ converge al vector x respecto a una norma, entonces converge al mismo vector con respecto a cualquier otra norma (ver, por ejemplo, Horn y Johnson [57]). Esto significa que la convergencia de una sucesión de vectores puede ser estudiada con respecto a cualquier norma vectorial.

Para resolver el sistema (1.1) por procedimientos iterativos, consideraremos la matriz A expresada de la forma $A = M - N$, donde M y N son también matrices cuadradas de orden n.

Definición 1.10. Sea la matriz $A \in \mathcal{M}_n$, tal que

$$A = M - N, \quad M, N \in \mathcal{M}_n. \tag{1.3}$$

Si M es no singular, se dice que la expresión (1.3) es una *partición* de la matriz A.

Basándonos en la partición (1.3) de la matriz A, el sistema lineal $Ax = b$ se puede escribir como

$$Mx = Nx + b. \tag{1.4}$$

Para obtener la solución de (1.1), podemos iterar la ecuación (1.4) mediante el esquema

$$Mx^{(l)} = Nx^{(l-1)} + b, \quad l = 1, 2, \ldots, \tag{1.5}$$

o análogamente

$$x^{(l)} = M^{-1}Nx^{(l-1)} + M^{-1}b, \quad l = 1, 2, \ldots, \tag{1.6}$$

y estudiar bajo qué condiciones de las matrices A, M y N, la sucesión de vectores $\{x^{(l)}\}_{l\geq 1}$, generada por el esquema iterativo (1.6), converge a la solución exacta del sistema lineal (1.1). Se llama *matriz de iteración* a la matriz $T = M^{-1}N$.

Claramente, según se elija la partición (1.3) tendremos distintos esquemas iterativos. Los más conocidos son los métodos de *Jacobi, Gauss-Seidel, SOR y SSOR*. En principio, se estudiaron de forma independiente diversas condiciones de convergencia para estos métodos, junto con sus versiones en bloques y los métodos semi-iterativos, (ver, por ejemplo, Varga [98] o Young [105]). Sin embargo, Varga [98] en 1962, introduce el concepto de partición regular de una matriz que en determinados casos engloba a los métodos antes mencionados, e intentó dar condiciones de convergencia sobre este tipo de particiones.

Definición 1.11. Se dice que la partición $A = M - N$ es *regular* si M^{-1} y N son matrices no negativas, es decir, si $M^{-1} \geq O$ y $N \geq O$.

Posteriormente al concepto de partición regular, se definió el concepto de partición débilmente regular, ver por ejemplo, Ortega [79].

Definición 1.12. La partición $A = M - N$ se dice que es *débilmente regular* si $M^{-1} \geq O$ y $M^{-1}N \geq O$.

Como se deduce de las definiciones de ambos tipos de particiones, toda partición regular es débilmente regular ya que el producto de matrices no negativas es otra matriz no negativa.

En la versión original de la definición de partición débilmente regular, dada por Ortega y Rheinboldt [82], se incluía la hipótesis adicional $NM^{-1} \geq O$, sin embargo, nosotros hemos elegido la Definición 1.12 para referirnos a una partición débilmente regular, tal y como hacen muchos autores como Amedjoe [5], Beauwens [8], Berman y Plemmons [10], Elsner [40], Neumann y Plemmons [75], Marek y Szyld [66] y O'Leary y White [77]. Otros autores (ver, por ejemplo, [28], [103] y [104]) denominan a las particiones débilmente regulares, según la definición original, particiones no negativas y según la Definición 1.12 particiones débiles no negativas del primer tipo.

Otro tipo de partición, que está asociada al concepto de matriz definida positiva, es el de partición P-regular que podemos ver, por ejemplo, en [10] y [81].

Definición 1.13. Sea $A \in \mathbb{C}^{n\times n}$, la partición $A = M - N$ es *P-regular* si la matriz $M^H + N$ es definida positiva.

La convergencia del método (1.6) se puede estudiar analizando el error que se comete en cada iteración respecto de la solución exacta ξ. Así, sea $e^{(l)} = x^{(l)} - \xi$ el vector error en la l-ésima iteración. Teniendo en cuenta la expresión (1.4) podemos escribir

$$\xi = T\xi + M^{-1}b,$$

donde $T = M^{-1}N$, y si esta expresión se resta a la expresión (1.6), se obtiene

$$e^{(l)} = Te^{(l-1)}, \quad l = 1, 2, \ldots . \tag{1.7}$$

De donde se obtiene

$$e^{(l)} = Te^{(l-1)} = T^2e^{(l-2)} = \cdots = T^le^{(0)}, \quad l = 1, 2, \ldots .$$

Así pues, la convergencia de la sucesión definida por la ecuación (1.6) a ξ es equivalente a que la sucesión de vectores error $\{e^{(l)}\}_{l\geq 1}$ converja al vector cero. Esto ocurre si, y sólo si, las potencias de la matriz T convergen a la matriz nula, es decir si $\lim_{l\to\infty} T^l = O$.

El resultado teórico básico de convergencia del método iterativo (1.6) está determinado por el *radio espectral* de T y su demostración se basa en la forma canónica de Jordan de esta matriz, tal y como puede verse en Young [105].

Teorema 1.6. *(Teorema 5.1, [105]). Sea T una matriz cuadrada, entonces $\{T^l\}_{l\geq 1}$ converge a la matriz nula si, y sólo si, $\rho(T) < 1$.*

También, para estudiar la convergencia del esquema iterativo (1.6), se utiliza el siguiente lema, que es la versión matricial del Lema de Neumann para series convergentes. Su demostración se puede ver, por ejemplo, en Berman y Plemmons [10].

Lema 1.1. *(Lema 2.1, [10]). La matriz T es convergente, es decir, $\rho(T) < 1$, si, y sólo si $(I-T)^{-1}$ existe y en este caso $(I-T)^{-1} = \sum_{i=0}^{\infty} T^i$.*

Siguiendo el Teorema 1.6, el método iterativo (1.6) converge a la solución del sistema lineal (1.1), para cualquier vector inicial $x^{(0)}$, si, y sólo si, $\rho(M^{-1}N) < 1$.

Sin embargo, cuando tenemos distintas matrices de iteración, no es condición suficiente que su radio espectral sea menor que 1 para asegurar la convergencia del método iterativo, ya que el producto de las distintas matrices de iteración no siempre tiende a cero, (ver Bru y Johnson [17], o Robert, Charnay y Musy [86]). Para ver este resultado, damos un ejemplo que es similar al que presentan Bru y Fuster en [16].

Ejemplo 1.1. *Consideramos la sucesión de matrices*

$$T^{(l)} = \begin{bmatrix} 0,5 & 0 \\ 0 & e^{\frac{-1}{l^2}} \end{bmatrix}, \quad l = 1, 2, \ldots .$$

Claramente podemos observar que para cada l, $\rho(T^{(l)}) < 1$. *Sin embargo, si calculamos el producto de las distintas matrices tenemos*

$$\begin{aligned} T^{(l)}T^{(l-1)}\cdots T^{(1)} &= \begin{bmatrix} 0,5 & 0 \\ 0 & e^{\frac{-1}{l^2}} \end{bmatrix} \begin{bmatrix} 0,5 & 0 \\ 0 & e^{\frac{-1}{(l-1)^2}} \end{bmatrix} \cdots \begin{bmatrix} 0,5 & 0 \\ 0 & e^{\frac{-1}{1}} \end{bmatrix} \\ &= \begin{bmatrix} 0,5^l & 0 \\ 0 & e^{-\left(\frac{1}{l^2}+\frac{1}{(l-1)^2}+\cdots+1\right)} \end{bmatrix}. \end{aligned}$$

Entonces

$$\lim_{l\to\infty} T^{(l)}T^{(l-1)}\cdots T^{(1)} = \begin{bmatrix} 0 & 0 \\ 0 & e^{-\frac{\pi^2}{6}} \end{bmatrix}.$$

Como podemos ver, $\lim_{l\to\infty} T^{(l)}T^{(l-1)}\cdots T^{(1)} \neq O$, *es decir, el producto de las distintas matrices no converge a cero y sin embargo, habíamos visto que* $\rho(T^{(l)}) < 1$; *esto nos indica que cuando se tienen distintas matrices de iteración no es condición suficiente que el radio espectral sea menor que uno para asegurar la convergencia del método iterativo correspondiente.*

Como consecuencia del resultado obtenido en el ejemplo anterior, deducimos que son necesarias otras herramientas para demostrar la convergencia de un método iterativo en el que para cada iteración l se obtiene una matriz de iteración distinta. Una condición suficiente para la convergencia es la que se da en el siguiente lema de Bru y Fuster, (ver [16]).

Lema 1.2. *(Lema 2, [16]). Sea* $T^{(l)}$, $l = 1, 2, \ldots$, *una sucesión de matrices cuadradas complejas. Si existe una norma matricial* $||.||$ *tal que* $||T^{(l)}|| \leq \theta < 1$, $l = 1, 2, \ldots$, *entonces,* $\lim_{l\to\infty} T^{(l)}T^{(l-1)} \cdots T^{(1)} = O$.

Cuando la matriz A es *monótona*, es decir, cuando su inversa es no negativa ($A^{-1} \geq O$), existe un resultado de convergencia que se da en el siguiente teorema que se puede ver, por ejemplo, en Berman y Plemmons [10] o Varga [98].

Teorema 1.7. *(Teorema 3.2, [98] y Teorema 7, [10]). Sea* A *una matriz no singular y* $A = M - N$ *una partición regular o débilmente regular, entonces,*

$$\rho(M^{-1}N) < 1, \quad si,\ y\ sólo\ si, \quad A^{-1} \geq O.$$

Cuando la matriz A es hermítica, es decir $A = A^H$, existe un resultado de convergencia que se da en el siguiente teorema que podemos ver, por ejemplo, en Berman y Plemmons [10] o en Keller [61].

Teorema 1.8. *(Corolario 7.5.44, [10] y Teorema 3, [61]). Sea* $A = M - N$ *una partición* P*-regular de una matriz hermítica* A*, entonces,* $\rho(M^{-1}N) < 1$ *si, y sólo si, la matriz* A *es definida positiva.*

En esta sección hemos visto cómo construir un modelo iterativo basándonos en una partición; esta relación no es casual, ya que es posible ver que todo método iterativo de la forma $x^{(l)} = Tx^{(l-1)} + c$, $l = 1, 2, \ldots$, proviene de una única partición, como pone de manifiesto el siguiente lema cuya demostración aparece en Lanzkron, Rose y Szyld [63].

Lema 1.3. *(Lema 2.3, [63]). Sea* A *una matriz no singular y* T *una matriz de tal forma que existe* $(I - T)^{-1}$, *entonces existe un único par de matrices* P, Q

tal que la matriz P es no singular, $T = P^{-1}Q$ y $A = P - Q$. Las matrices P y Q tienen la forma $P = A(I - T)^{-1}$ y $Q = P - A$.

El siguiente resultado está basado en el lema anterior y fue demostrado por Benzi y Szyld [9] para matrices simétricas y definidas positivas; aquí vamos a enunciarlo en el contexto de las matrices hermíticas y definidas positivas.

Teorema 1.9. *(Teorema 4.6, [9]). Sea A una matriz hermítica y definida positiva. Si las particiones $A = F - G = P - Q$ son P-regulares, entonces la matriz $T = P^{-1}QF^{-1}G$ tiene el radio espectral menor que 1. Además, la única partición $A = B - C$ inducida por la matriz de iteración T, tal que $T = B^{-1}C$, es también P-regular.*

Un resultado que utilizaremos bastante en las demostraciones de algunos de los teoremas que aparecen en esta memoria es el que se da en el siguiente teorema que se puede encontrar, por ejemplo, en [80].

Teorema 1.10. *(Teorema 6.2.3, [80]). Consideramos $A, B \in \mathbb{C}^{n\times n}$ matrices hermíticas. Si cualquiera de ellas es definida positiva, entonces la matriz AB tiene valores propios reales y es diagonizable. Si las matrices A y B son definidas positivas, entonces los valores propios de la matriz AB son positivos. Recíprocamente, si AB tienen valores propios positivos y la matriz A o B es definida positiva, entonces ambas matrices son definidas positivas.*

Definición 1.14. Diremos que una matriz $A = [a_{ij}]_{1\leq i,j\leq n} \in \mathcal{M}_n$ no singular es una M–matriz, si $a_{ij} \leq 0$ para $i \neq j$ y $A^{-1} \geq O$.

El siguiente lema se puede encontrar en Berman y Plemmons [10]. Denotamos por $Z^{n\times n}$ el conjunto de todas las matrices reales de tamaño $n \times n$ que tienen los elementos fuera de la diagonal no positivos.

Lema 1.4. *(Ejercicio 2.6, [10]). Sea $A \in Z^{n\times n}$ una matriz simétrica, A es una M-matriz si, y sólo si, es definida positiva.*

El radio espectral de la matriz de iteración $T = M^{-1}N$ se utiliza para dar una medida de la rapidez de convergencia, ya que cuando menor sea el radio espectral, más rápida será la convergencia. Cuando un esquema iterativo posea una matriz de iteración con menor radio espectral, diremos que el esquema posee un *radio de convergencia* superior.

El siguiente teorema compara el radio espectral de la matriz de iteración de un método iterativo clásico para diferentes particiones en el contexto de matrices hermíticas y definidas positivas. Su demostración se puede ver en Nabben [74].

Teorema 1.11. *(Teorema 2.3, [74]). Sean $A = M_1 - N_1 = M_2 - N_2$ dos particiones de A, con A hermítica y definida positiva.*

Si $O \preceq N_1 \preceq N_2$, entonces

$$\rho(M_1^{-1}N_1) \leq \rho(M_2^{-1}N_2) < 1. \tag{1.8}$$

Si $O \preceq N_1 \prec N_2$, entonces

$$\rho(M_1^{-1}N_1) < \rho(M_2^{-1}N_2) < 1. \tag{1.9}$$

Como podemos observar, bajo las hipótesis de este teorema se verifica que, $N_1 \prec N_2 \Leftrightarrow M_1 \prec M_2 \Leftrightarrow M_1^{-1} \succ M_2^{-1}$, (ver, por ejemplo, [57] y [74]).

Una vez vistos los conceptos básicos sobre los métodos iterativos, pasamos a definir los métodos iterativos clásicos que usaremos en esta memoria.

1.4 Métodos iterativos clásicos

Como hemos visto en la Sección 1.3, cada partición $A = M - N$ de la matriz del sistema lineal $Ax = b$, define un esquema iterativo como el que aparece en la expresión (1.6). Las distintas elecciones de las matrices M y N, dan como resultado los distintos métodos iterativos clásicos.

Supongamos que todos los elementos de la diagonal de $A = [a_{ij}]_{1 \leq i,j \leq n}$ son no nulos y que la matriz A se expresa de la siguiente forma

$$A = D - L - U, \tag{1.10}$$

donde $D = \text{diag}(a_{11}, a_{22}, \ldots, a_{nn})$, y $-L$ y $-U$ son, respectivamente, la parte estrictamente triangular inferior y superior de la matriz A.

Teniendo en cuenta la expresión (1.10), podemos escribir el sistema (1.1) como

$$Dx = (L + U)x + b. \tag{1.11}$$

Como los elementos diagonales a_{ii}, de A son no nulos, podemos establecer el siguiente método iterativo derivado de la expresión (1.11)

$$a_{ii}x_i^{(l+1)} = -\sum_{j=1, j \neq i}^{n} a_{ij}x_j^{(l)} + b_i, \quad 1 \leq i \leq n, \quad l = 0, 1, \ldots, \tag{1.12}$$

donde $x_i^{(0)}$, $1 \leq i \leq n$, son las componentes de un vector inicial $x^{(0)}$.

Claramente, la expresión (1.12) puede ser reescrita como

$$x_i^{(l+1)} = -\sum_{j=1, j \neq i}^{n} \frac{a_{ij}}{a_{ii}}x_j^{(l)} + \frac{b_i}{a_{ii}}, \quad 1 \leq i \leq n, \quad l = 0, 1, \ldots. \tag{1.13}$$

La expresión matricial de (1.12) es

$$Dx^{(l+1)} = (L+U)x^{(l)} + b, \quad l = 0, 1, \ldots,$$

y como D es una matriz no singular, la notación matricial de (1.13) es

$$x^{(l+1)} = D^{-1}(L+U)x^{(l)} + D^{-1}b, \quad l = 0, 1, \ldots.$$

Este método iterativo se denomina *método de Jacobi*. Claramente, las siguientes elecciones de M y N en la partición $A = M - N$, dan lugar a dicho método

$$M = D \quad \text{y} \quad N = L + U. \tag{1.14}$$

La matriz de iteración asociada

$$B = M^{-1}N = D^{-1}(L+U),$$

se denomina usualmente *matriz de Jacobi* asociada a la matriz A.

Como se puede observar en el método de Jacobi, para el cálculo de la i–ésima componente del vector $x^{(l+1)}$, se utilizan los valores de las componentes $x_1^{(l)}, x_2^{(l)}, \ldots, x_{i-1}^{(l)}, x_{i+1}^{(l)}, \ldots, x_n^{(l)}$. Pero en el momento de calcular dicha componente i–ésima ya han sido calculadas las componentes $x_1^{(l)}, x_2^{(l)}, \ldots, x_{i-1}^{(l)}$. Surge entonces, de modo natural, la pregunta siguiente: ¿Por qué no aprovechar la información que se acaba de obtener sobre las primeras $(i-1)$ componentes del vector $x^{(l+1)}$ en el cálculo de la componente i–ésima?.

Una primera ventaja de actuar así sería la de no tener que almacenar dos vectores en el algoritmo de cálculo, pues las componentes del vector $x^{(l+1)}$ se almacenarían en el lugar ocupado por las correspondientes del vector $x^{(l)}$ (ya que

éstas no serían necesarias en los sucesivos cálculos). Además parece probable, que al utilizar datos más cercanos a la solución exacta en el cálculo de las componentes de $x^{(l+1)}$, la velocidad de convergencia mejore.

El *método iterativo de Gauss-Seidel* consiste precisamente en actuar de dicha manera. Con ello, el esquema de cálculo del método resulta ser:

$$x_i^{(l+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(l+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(l)} + \frac{b_i}{a_{ii}}, \quad 1 \leq i \leq n, \quad l = 0, 1, \ldots .$$

Matricialmente este método se puede escribir como sigue

$$(D - L)x^{(l+1)} = Ux^{(l)} + b, \quad l = 0, 1, \ldots , \tag{1.15}$$

y como la matriz triangular inferior $D-L$ es no singular, (1.15) puede ser escrito equivalentemente como

$$x^{(l+1)} = (D - L)^{-1}Ux^{(l)} + (D - L)^{-1}b, \quad l = 0, 1, \ldots .$$

Análogamente al método de Jacobi, el metodo iterativo de Gauss-Seidel puede obtenerse considerando una partición $A = M - N$, en este caso

$$M = D - L \quad \text{y} \quad N = U.$$

Con el fin de acelerar la convergencia, a todo método iterativo de la forma (1.6) se le puede asociar un *método extrapolado o relajado* sustituyendo en cada etapa l, el vector $x^{(l+1)}$, por el vector extrapolado

$$\omega x^{(l+1)} + (1 - \omega)x^{(l)}, \quad \omega \neq 0, \quad \omega \in \mathbb{R},$$

ya que ésto sólo requiere un pequeño esfuerzo computacional adicional. Esta relajación corresponde al siguiente esquema iterativo

$$x^{(l+1)} = H(\omega)x^{(l)} + \omega M^{-1}b, \quad l = 0, 1, \ldots ,$$

con la matriz de iteración

$$H(\omega) = (1-\omega)I + \omega M^{-1}N. \tag{1.16}$$

A este método iterativo se le llama *ω-extrapolación* del esquema (1.6) o también *ω-relajación*, siendo ω el *factor de relajación*. Lo denominaremos simplemente método extrapolado o relajado cuando no exista ambigüedad en la elección de ω.

Si en (1.16) utilizamos la partición de Jacobi, se obtiene el *método de Jacobi relajado (JOR)*. El esquema iterativo es

$$x^{(l+1)} = \left[(1-\omega)I + \omega D^{-1}(L+U)\right] x^{(l)} + \omega D^{-1}b, \quad l = 0, 1, \ldots,$$

o análogamente

$$x_i^{(l+1)} = -\omega \sum_{j=1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(l)} + (1-\omega)x_i^{(l)} + \omega \frac{b_i}{a_{ii}}, \quad 1 \leq i \leq n, \quad l = 0, 1, \ldots.$$

El método de Gauss-Seidel puede ser *sucesivamente* ω-extrapolado, de la siguiente forma. Primero definimos el vector iterado auxiliar $\overline{x}^{(l+1)}$ como

$$\overline{x}_i^{(l+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(l+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(l)} + \frac{b_i}{a_{ii}}, \quad 1 \leq i \leq n, \quad l = 0, 1, \ldots.$$

La componente $x_i^{(l+1)}$ de este método iterativo se define como

$$x_i^{(l+1)} = \omega \overline{x}_i^{(l+1)} + (1-\omega)x_i^{(l)}.$$

Combinando estas dos últimas expresiones obtenemos

$$x_i^{(l+1)} = -\omega \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(l+1)} - \omega \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(l)} + (1-\omega)x_i^{(l)} + \omega \frac{b_i}{a_{ii}}. \tag{1.17}$$

La expresión matricial de este método es

$$(D-\omega L)x^{(l+1)} = [(1-\omega)D+\omega U]\,x^{(l)} + \omega b, \quad l = 0, 1, \ldots,$$

y como la matriz $D-\omega L$ es no singular para cualquier elección de ω, entonces obtenemos

$$x^{(l+1)} = (D-\omega L)^{-1}\,[(1-\omega)D+\omega U]\,x^{(l)} + \omega(D-\omega L)^{-1}b, \quad l = 0, 1, \ldots.$$

El método iterativo que acabamos de contruir recibe el nombre de *método de sobrerrelajaciones sucesivas (SOR)*. La matriz de iteración se denota por

$$\mathcal{L}_\omega = (D-\omega L)^{-1}\,[(1-\omega)D+\omega U]\,.$$

Seleccionando $\omega = 1$, este método iterativo se reduce exactamente al método de Gauss-Seidel.

Las siguientes elecciones de M y N en (1.6) dan lugar respectivamente a los métodos extrapolados JOR y SOR.

Método JOR: $M = \omega^{-1}D$ y $N = \omega^{-1}\,[(1-\omega)D+\omega(L+U)]\,.$
Método SOR: $M = \omega^{-1}(D-\omega L)$ y $N = \omega^{-1}\,[(1-\omega)D+wU],$

donde ω es el factor de relajación.

Hadjidimos [55] en 1978, estudia un nuevo método de resolución de sistemas de ecuaciones lineales. Este método constituye una generalización del método SOR con dos parámetros y su formulación es la siguiente

$$(D-\mu L)x^{(l+1)} = [(1-\omega)D+(\omega-\mu)L+\omega U]\,x^{(l)} + \omega b, \qquad (1.18)$$
$$l = 0, 1, \ldots,$$

donde los números reales μ y $\omega \neq 0$, son dos parámetros fijos. Este método se denomina método de sobrerrelajación acelerada, y se conoce como método AOR. Claramente, este método incluye a los métodos clásicos definidos anteriormente.

Otra importante modificación en el método iterativo SOR es el método *simétrico SOR* ($SSOR$). Es un método de doble paso en donde se aplica un paso del SOR seguido de otro paso del método SOR en orden inverso. Para formalizar dicho método iterativo, consideramos primero el método de Gauss-Seidel para obtener el método Gauss-Seidel simétrico (GSS). En la iteración l, el vector iterado en el método Gauss-Seidel se obtiene mediante la expresión

$$x^{(l+\frac{1}{2})} = (D-L)^{-1}Ux^{(l)} + (D-L)^{-1}b, \tag{1.19}$$

que como se puede apreciar hemos etiquetado mediante el iterado $x^{(l+\frac{1}{2})}$.

Ahora, para obtener el iterado $(l+1)$ usamos el iterado dado en (1.19) de la siguiente forma

$$x_i^{(l+1)} = \frac{1}{a_{ii}}\left(-\sum_{j>i} a_{ii}x_j^{(l+1)} - \sum_{j<i} a_{ij}x_j^{(l+\frac{1}{2})} + b_i\right), \quad i = n, n-1, \ldots, 1.$$

La expresión matricial viene dada por la expresión

$$Dx^{(l+1)} = Ux^{(l+1)} + Lx^{(l+\frac{1}{2})} + b,$$

o

$$x^{(l+1)} = (D-U)^{-1}Lx^{(l+\frac{1}{2})} + (D-U)^{-1}b. \tag{1.20}$$

Entonces, una iteración del método iterativo simétrico de Gauss-Seidel es una combinación de las expresiones (1.19) y de (1.20), que viene expesada por la ecuación

$$x^{(l+1)} = (D-U)^{-1}L(D-L)^{-1}Ux^{(l)} + \overline{d}, \tag{1.21}$$

donde

$$\overline{d} = (D-U)^{-1}L(D-L)^{-1}b + (D-U)^{-1}b.$$

Para obtener el esquema iterativo del método $SSOR$ se introduce un parámetro de relajación ω igual que se hace en el método Gauss-Seidel para obtener el método SOR, en los dos pasos. La expresión matricial de la iteración $SSOR$ es entonces

$$x^{(l+1)} = (D-\omega U)^{-1}[(1-\omega)D + \omega L](D-\omega L)^{-1}[(1-\omega)D + \omega U]x^{(l)} + \overline{d},$$

donde ahora

$$\overline{d} = \omega(D-\omega U)^{-1}\{[(1-\omega)D + \omega L](D-\omega L)^{-1} + I\}b.$$

A continuación se recogen una serie de resultados de convergencia para estos métodos iterativos clásicos que son bastantes conocidos, todos ellos se pueden ver, por ejemplo, en Berman y Plemmons [10], Ortega [81], Varga [98] o en Young [105].

Comenzamos con un teorema que se da en el contexto de las M-matrices no singulares.

Teorema 1.12. *(Teorema A.2.15, [81]). Sea A una matriz no singular. Si A es una M-matriz, los métodos de Jacobi y Gauss-Seidel convergen para cualquier vector inicial $x^{(0)}$.*

También se establece la convergencia de estos métodos clásicos, para el caso de matrices diagonal dominantes e irreducibles.

Definición 1.15. Para $n \geq 2$, una matriz A de tamaño $n \times n$ es *reducible* si existe una matriz de permutación P de tamaño $n \times n$ tal que

$$PAP^t = \begin{bmatrix} A_{11} & A_{12} \\ O & A_{22} \end{bmatrix},$$

donde A_{11} es una submatriz de tamaño $r \times r$ y A_{22} es una submatriz de tamaño $(n-r) \times (n-r)$, con $1 \leq r < n$. Si no existe tal matriz de permutación, entonces A se llama *irreducible*.

Definición 1.16. Sea $A = [a_{ij}]_{1 \leq i,j \leq n}$ una matriz real de tamaño $n \times n$. Se dice que

(i) A es *diagonal dominante* si

$$|a_{ii}| \geq \sum_{j=1, j \neq i}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n. \tag{1.22}$$

(ii) A es *estrictamente diagonal dominante* si las n desigualdades (1.22) son estrictas.

(iii) A es *irreduciblemente diagonal dominante* si A es una matriz irreducible y diagonal dominante con al menos una desigualdad (1.22) estricta.

Si A es una matriz irreduciblemente diagonal dominante, entonces será también invertible con sus elementos diagonales no nulos.

Teorema 1.13. *(Teorema 2.1, [105]). Sea A irreduciblemente diagonal dominante, entonces los métodos de Jacobi, Gauss–Seidel, JOR y SOR (estos dos últimos para $0 < \omega \leq 1$) convergen.*

En el Teorema 1.13 se estudia la convergencia de los métodos relajados cuando $0 < \omega \leq 1$, sin embargo, para el método SOR resulta que el valor óptimo de ω suele ser mayor o igual que 1. Hay que tener en cuenta que este método puede converger cuando el factor de relajación cumple $0 < \omega < 2$, pero nunca en otro caso. Esto queda reflejado en el siguiente resultado (ver, por ejemplo, Varga [98]).

Lema 1.5. *(Lema de Kahan, [98]). El método iterativo SOR no puede converger a menos que* $0 < \omega < 2$.

Muchos son los autores que han estudiado los métodos iterativos clásicos. En particular, existen condiciones de convergencia para los métodos clásicos cuando la matriz A es simétrica y definida positiva. Estos resultados se pueden ver, por ejemplo, en Ortega [81] o en Varga [98].

Teorema 1.14. *(Teorema 3.1.2, [81]). Sea* $A = M - N$, *la partición de Jacobi dada en (1.14). Si* A *es simétrica y definida positiva, entonces el método iterativo de Jacobi converge para cualquier* $x^{(0)}$, *si, y sólo si,* $M + N$ *es definida positiva.*

Teorema 1.15. *(Teorema 3.2.3 Ostrowski-Reich, [81]). Si* A *es simétrica y definida positiva y* $\omega \in]0, 2[$, *entonces el método iterativo SOR converge para cualquier* $x^{(0)}$.

Teorema 1.16. *(Teorema 3.2.4, [81]). Si* A *es simétrica y definida positiva y* $\omega \in]0, 2[$, *entonces el método iterativo SSOR converge para cualquier* $x^{(0)}$.

En 1984, Albrecht y Klein [4] dan un teorema de convergencia para una matriz arbitraria A; la condición necesaria y suficiente que exigen a la matriz

A, con un ω apropiado, para que el metodo JOR converja es que todos sus valores propios pertenezcan exclusivamente a $\mathbb{C}^-$ o $\mathbb{C}^+$ (parte real negativa o positiva respectivamente). Hacen notar además, que el método JOR converge especialmente si A es definida positiva o definida negativa.

En 1991, Dancis en [33] considera la convergencia del sistema de ecuaciones lineales $Ax = b$, cuando la teoría usual del método SOR es aplicable tal y como puede estudiarse en Young [105].

1.5 Métodos iterativos en dos etapas

Los métodos iterativos para resolver sistemas de ecuaciones lineales requieren de la resolución, en cada etapa, de un sistema de ecuaciones más simple. Cuando este sistema se aproxima a su vez por un método iterativo, el método global se llama *método iterativo en dos etapas.* A continuación vamos a describir más detalladamente dicho método.

Consideramos el sistema lineal

$$Ax = b \tag{1.23}$$

y la partición $A = M - N$. Asociado a dicha partición tenemos el esquema iterativo

$$Mx^{(l+1)} = Nx^{(l)} + b, \quad l = 0, 1, 2, \ldots, \tag{1.24}$$

donde $x^{(0)}$ es un vector inicial arbitrario.

Los métodos en dos etapas, también llamados métodos *"inner - outer"*, consisten en aproximar el sistema (1.24) de forma iterativa. Así, si consideramos la partición $M = F - G$ y suponemos que se realizan s iteraciones en el sistema (1.24), que llamaremos *internas*, entonces el esquema que resulta es el siguiente

$$x^{(l+1)} = (F^{-1}G)^s x^{(l)} + \sum_{j=0}^{s-1} (F^{-1}G)^j F^{-1}(Nx^{(l)} + b), \quad l = 0, 1, 2, \ldots . \quad (1.25)$$

Podemos distinguir entre el método *estacionario* en dos etapas, en el que el número de iteraciones internas s permanece fijo en cada una de las etapas *externas*, y el método *no estacionario*, en el que el número de iteraciones internas $s(l)$ puede variar para cada etapa externa l. Hacemos notar que Lanzkron, Rose y Szyld en [63] llaman a los métodos no estacionarios en dos etapas, métodos *dinámicos* en dos etapas. En este caso el esquema iterativo es

$$x^{(l+1)} = (F^{-1}G)^{s(l)} x^{(l)} + \sum_{j=0}^{s(l)-1} (F^{-1}G)^j F^{-1}(Nx^{(l)} + b), \quad l = 0, 1, 2, \ldots . \quad (1.26)$$

En un principio, los procesos iterativos en dos etapas se desarrollaron para la resolución de cierta clase de sistemas de ecuaciones lineales que surgían a partir de la discretización de problemas de valores frontera elípticos. Estos métodos se usan, en particular, para la resolución de aproximaciones a ecuaciones en diferencias parciales simultáneas (ver Smith [93]). Tales procedimientos son de interés puesto que pueden ser frecuentemente extendidos a la resolución de ecuaciones no lineales (Douglas [35], Dupont [38], D'Yakonov [39]). La convergencia de estos procesos se probó para problemas específicos y se estimó el número de operaciones necesarias para reducir la norma del error inicial en cada iteración.

Posteriormente, en 1973, los métodos en dos etapas serían estudiados por Nichols en [76], dando resultados de convergencia bajo condiciones bastante generales. Concretamente, demuestra la convergencia del método no estacionario

bajo las hipótesis de que el número de iteraciones internas $s(l)$, para cada iteración externa l, verifique que $s(l) \geq S$, para un cierto S, y las iteraciones interna y externa sean ambas convergentes, es decir, cumplan las condiciones razonables $\rho(M^{-1}N) < 1$ y $\rho(F^{-1}G) < 1$.

En este trabajo, también se demuestra la convergencia, bajo condiciones más complejas, para el método estacionario en dos etapas, es decir cuando $s(l) = s$. Las condiciones que Nichols plantea son las siguientes:

Los valores propios de $C = M^{-1}N$, $\lambda_i(C)$, y los de la matriz de iteración del proceso interno T_s, $\lambda_i(T_s)$ verifican

$$-1 < \lambda_i(C) < 1, \quad -1 < \lambda_i(T_s) < 1.$$

El valor de s es elegido de forma que $\rho(T_s) < \frac{1-\rho(C)}{1+\rho(C)}$. Y además se ha de cumplir una de las siguientes condiciones:

(i) T_s y C son matrices reales simétricas.

(ii) Existe una matriz P tal que PCP^{-1} y PT_sP^{-1} son ambas reales y simétricas.

(iii) Existe una matriz P tal que PCP^{-1} es simétrica y T_s es simétrica y conmuta con P.

Golub y Overton en [53] y [54], estudian la convergencia de los métodos en dos etapas para el caso en el que la iteración externa se resuelve por el método de Richardson o Chebyshev. Para sistemas de ecuaciones no lineales, han sido ampliamente estudiados los métodos con iteración externa no lineal y con iteración interna lineal, además han sido aplicados a diferentes áreas de la ciencia e ingeniería; por ejemplo, véase Bank y Rose [7], Dembo, Eisenstat y Steihaug [34].

Lanzkron, Rose y Szyld en [63], dan condiciones sobre las particiones externas e internas que aseguran la convergencia, para cualquier número de iteraciones internas, del método en dos etapas, tanto estacionario como no estacionario. Las condiciones dadas hacen referencia a particiones regulares y débilmente regulares, y sus resultados son aplicables a matrices monótonas, que como ya se ha dicho en la Sección 1.3, son matrices no singulares cuya inversa es no negativa. En concreto, exigen que la partición externa sea regular y la partición interna débilmente regular. Los conceptos de partición regular y débilmente regular fueron introducidos en las Definiciones 1.11 y 1.12 respectivamente, de la Sección 1.3.

Teorema 1.17. *Sea $A = M - N$ una partición convergente y regular, y sea $M = F - G$ una partición convergente y débilmente regular, entonces*

1.- *el método en dos etapas estacionario, definido en la expresión (1.25), converge para $s \geq 1$.*

2.- *el método en dos etapas no estacionario,, definido en la expresión (1.26) converge para $s(l) \geq 1,\ l = 0, 1, \ldots$.*

Bajo estas mismas condiciones, demuestran que el radio espectral de la matriz de iteración global decrece cuando el número de iteraciones internas crece, es decir, dicho radio espectral es una función monótona decreciente de s, lo que se puede interpretar, obviamente, como que al aumentar las iteraciones internas mejora la correspondiente aproximación externa. Este resultado es intuitivo, pero sin embargo, si las condiciones del Teorema 1.17 no se satisfacen, el resultado puede no cumplirse, como puede apreciarse en diversos ejemplos que fueron objeto de estudio en [63].

Frommer y Szyld en [47], demuestran que si las particiones interna y externa son convergentes, el método no estacionario en dos etapas converge si

$$\lim_{l\to\infty} s(l) = \infty. \tag{1.27}$$

En particular, comprueban la convergencia para cualquier sucesión $\{s(l)\}_{l=1}^{\infty}$ satisfaciendo (1.27) y analizan la convergencia en el caso en que (1.27) no se cumple pero $s(l)$ es suficientemente grande. Además, dan demostraciones alternativas a los resultados de convergencia vistos en [63], y usan estas demostraciones para extender los resultados de convergencia a particiones de H–matrices.

Definición 1.17. Sea $A = [a_{ij}]_{1\leq i,j\leq n} \in \mathcal{M}_n$. Se define la *matriz comparación* de A y se denota por $\langle A\rangle$, como

$$\langle A\rangle = [b_{ij}] \quad : \quad b_{ij} = \begin{cases} |a_{ij}| & i = j \\ -|a_{ij}| & i \neq j \end{cases} \quad 1 \leq i,j \leq n.$$

Definición 1.18. Diremos que $A = [a_{ij}]_{1\leq i,j\leq n} \in \mathcal{M}_n$ es una H–matriz, si $\langle A\rangle$ es una M–matriz.

Para introducir los resultados de convergencia para el método en dos etapas, demostrados por Frommer y Szyld en [47], damos previamente las siguientes definiciones.

Definición 1.19. Diremos que la partición $A = M - N$ es H–partición si $\langle M\rangle - |N|$ es una M–matriz.

Definición 1.20. Diremos que la partición $A = M - N$ es H–compatible si $\langle A\rangle = \langle M\rangle - |N|$.

Teorema 1.18. *(Teorema 4.5 y 4.6, [47]). Sea $A = M - N$ una H–partición, y sea $M = F - G$ una partición H–compatible, entonces*

1.- El método en dos etapas estacionario, definido en la expresión (1.25), converge para $s \geq 1$.

2.- El método en dos etapas no estacionario, definido en la expresión (1.26), converge para $s(l) \geq 1,\ l = 0, 1, \ldots$.

Es conocido, (ver Frommer y Szyld [47]), que si la partición $A = M - N$ es una H–partición, entonces A y M son H–matrices, por tanto el Teorema 1.18 extiende los resultados de convergencia de los métodos en dos etapas, a una clase de matrices no necesariamente monótonas, a las H–matrices.

Este tipo de matrices forman una clase bastante importante, según se refleja en el siguiente teorema, que puede verse en Frommer y Mayer [43].

Teorema 1.19. *(Corolario 4.3, [43]). Supongamos que la matriz A cumple una de las siguientes condiciones:*

1.- A es una M–matriz.

2.- A es estricta o irreduciblemente diagonal dominante.

3.- $\langle A \rangle$ es simétrica y definida positiva.

Entonces, A es una H–matriz.

Migallón y Penadés, estudian la convergencia de los métodos en dos etapas, para matrices hermíticas y definidas positivas. En [71], demuestran la convergencia del método en dos etapas estacionario para el caso en que se realiza

cualquier número de iteraciones internas $s \geq 1$. El siguiente teorema muestra este resultado.

Teorema 1.20. *(Teorema 2.1, [71]). Sea A una matriz hermítica y definida positiva. Consideramos una partición $A = M - N$ donde M es hermítica y N es semidefinida positiva. Sea $M = F - G$ una partición de M que es P-regular, entonces, para cualquier número de iteraciones internas $s \geq 1$, el método iterativo en dos etapas estacionario (1.25), converge a la solución del sistema lineal (1.23), para cualquier vector inicial $x^{(0)}$.*

Bajo las hipótesis de este teorema y basándose en los resultados del Lema 1.3 (Sección 1.3), demuestran que la matriz de iteración de un método en dos etapas estacionario induce una única partición que es P-regular. Este resultado se da a continuación en el siguiente corolario.

Corolario 1.1. *(Corolario 2.1, [71]). Sea A una matriz hermítica y definida positiva. Consideramos una partición $A = M - N$ donde M es hermítica y N es semidefinida positiva. Sea $M = F - G$ una partición de M que es P-regular, entonces, la matriz de iteración de un método en dos etapas estacionario, induce una única partición que es P-regular.*

Por último y bajo condiciones similares a las dadas para el caso estacionario, dichos autores demuestran la convergencia del modelo no estacionario en dos etapas, dando para ello el siguiente teorema.

Teorema 1.21. *(Teorema 2.2, [71]). Sea A una matriz hermítica y definida positiva. Consideramos una partición $A = M - N$ donde M es hermítica y*

N es semidefinida positiva. Sea $M = F - G$ una partición de M que es P-regular. Suponemos además que la sucesión de iteraciones internas $\{s(l)\}_{l\geq 0}$ está acotada, entonces, el método iterativo en dos etapas no estacionario (1.26), converge a la solución del sistema lineal (1.23), para cualquier vector inicial $x^{(0)}$.

Bajo condiciones similares a las dadas en [71], Migallón y Penadés en [72], demuestran que el radio espectral de la matriz de iteración del método estacionario en dos etapas que resulta de realizar s iteraciones internas, es una función monótona decreciente de s.

Teorema 1.22. *(Teorema 2.1, [72]). Sea la matriz A hermítica y definida positiva. Consideramos las particiones $A = M - N$ y $M = F - G$. Suponemos que las matrices M y F son hermíticas, N es semidefinida positiva y G es definida positiva. Sean s_1 y s_2 enteros positivos. Consideramos dos métodos iterativos en dos etapas, que difieren sólo en el número de iteraciones internas de cada bloque que se realizan para cada iteración externa, donde s_1 es el número de iteraciones internas de un método y s_2 del otro. Si $s_1 > s_2$ entonces $\rho(T_{s_1}) < \rho(T_{s_2}) < 1$.*

1.6 Método del gradiente conjugado

Dado el sistema de ecuaciones lineales

$$Ax = b, \tag{1.28}$$

donde $A \in \mathbb{R}^{n\times n}$ es una matriz simétrica y definida positiva, uno de los métodos más eficientes que existen para resolver el sistema (1.28) cuando la matriz de coeficientes es además grande y dispersa, es el *método del gradiente conjugado.*

Este procedimiento pertenece a una clase de métodos iterativos conocida con el nombre de *métodos de minimización*, cuyo nombre se debe a que resolver simultáneamente un conjunto de n ecuaciones es equivalente a encontrar el mínimo de una función error definida sobre un espacio n-dimensional. En cada paso de un método de este tipo, se usa un conjunto de valores de variables para generar un nuevo conjunto que proporcionan un valor menor en la función error. Veamos de una manera más formal la descripción de este método.

Consideremos la siguiente forma cuadrática

$$Q(x) = \frac{1}{2}x^T Ax - b^T x. \tag{1.29}$$

Si la matriz A es simétrica y definida positiva, el único x que minimiza la forma cuadrática $Q(x)$ es la solución del sistema $Ax = b$, que llamaremos x^*.

La razón de esto es porque el gradiente de la forma cuadrática (1.29) es

$$grad(Q(x)) = \left[\frac{\delta}{\delta x_1}Q(x), \ldots, \frac{\delta}{\delta x_n}Q(x)\right]^T = Ax - b. \tag{1.30}$$

Por tanto, $grad(Q(x^*)) = Ax^* - b = 0$. Por ser la matriz A definida positiva, podemos asegurar que el punto crítico de $Q(x)$ es un mínimo. Luego, para resolver el sistema de ecuaciones (1.28) es suficiente encontrar un vector x^* que minimice la forma cuadrática $Q(x)$.

Hay muchos métodos para minimizar la forma cuadrática $Q(x)$, algunos de ellos se basan en minimizaciones sucesivas sobre rectas. Una iteración de un método de este tipo viene dada por la expresión

$$x^{(l+1)} = x^{(l)} - \alpha_l p^{(l)}, \quad l = 0, 1, \ldots, \tag{1.31}$$

donde $p^{(l)}$ es el vector director de la recta y α_l es el escalar que indica la distancia a la que se encuentra la nueva aproximación de $x^{(l)}$, en la dirección determinada por el vector $p^{(l)}$. Dependiendo de la elección de α_l y de $p^{(l)}$, tenemos distintos métodos, pero quizá la forma más natural de elegir el valor del escalar α_l es tomarlo de tal forma que minimice la función $Q(x)$ en la dirección $p^{(l)}$, es decir,

$$Q(x^{(l)} - \alpha_l p^{(l)}) = \min_{\alpha} Q(x^{(l)} - \alpha p^{(l)}). \tag{1.32}$$

Para $x^{(l)}$ y $p^{(l)}$ fijos, la expresión (1.32) se corresponde con un problema de minimización en una única variable α. Por tanto, puede resolverse explícitamente, concluyendo (ver Ortega [81]), que como A es definida positiva, la forma cuadrática $Q(x)$ en α alcanza el valor mínimo cuando

$$\alpha_l = \frac{(p^{(l)})^T(Ax^{(l)} - b)}{(p^{(l)})^T A p^{(l)}} = -\frac{(p^{(l)})^T r^{(l)}}{(p^{(l)})^T A p^{(l)}}, \tag{1.33}$$

donde $r^{(l)} = b - Ax^{(l)}$ es el vector residual o residuo que se obtiene después de l iteraciones. Este residuo no necesita calcularse explícitamente después de cada iteración ya que puede ser calculado conociendo el residuo de la iteración anterior. Es decir, en la iteración $(l+1)$, el residuo $r^{(l+1)}$ puede ser calculado de la siguiente forma

$$\begin{aligned} r^{(l+1)} &= b - Ax^{(l+1)} \\ &= b - A(x^{(l)} - \alpha_l p^{(l)}) \\ &= b - Ax^{(l)} + \alpha_l A p^{(l)} \\ &= r^{(l)} + \alpha_l A p^{(l)}. \end{aligned}$$

De esta forma, en cada iteración sólo se calcula el producto matriz-vector

$Ap^{(l)}$, el cual ya se ha calculado anteriormente en el cálculo de α_l de la expresión (1.33).

Uno de estos métodos de minimización es el llamado método del *descenso más rápido* o método de *Richardson*, que se caracteriza porque para un vector dado $x^{(l)}$, el vector director $p^{(l)}$ se elige en la dirección opuesta a la del máximo crecimiento de $Q(x^{(l)})$, es decir, $p^{(l)} = -grad(Q(x^{(l)})) = b - Ax^{(l)} = r^{(l)}$. Esta elección natural de los vectores directores $p^{(l)}$, proporciona el siguiente esquema iterativo

$$x^{(l+1)} = x^{(l)} - \alpha_l(b - Ax^{(l)}), \quad l = 0, 1, \dots ,$$

donde α_l viene definida en (1.33).

Podemos observar, que si se normaliza la matriz A para que tenga elementos diagonales igual a 1 y $\alpha_l = 1$, este método se reduce al método iterativo de Jacobi.

Aunque moviéndonos en la dirección opuesta al gradiente se obtiene un decrecimiento máximo local en el valor de la función, el método de Richardson generalmente converge muy lentamente porque los residuos oscilan.

Otra forma de elegir los vectores $p^{(l)}$, es tomar los ejes de coordenadas como direcciones de búsqueda, es decir, los vectores directores $p^{(l)}$ se eligen de forma cíclica como $p^{(0)} = e_1$, $p^{(1)} = e_2, \dots ,$ $p^{(n-1)} = e_n$, $p^{(n)} = e_1$, donde $e_i = [0, \dots , 1, \dots , 0]^T$. En este caso, si α_l se elige como en la expresión (1.33), el vector $x^{(l+1)}$ se puede calcular como

$$x^{(l+1)} = x^{(l)} - \alpha_l e_i = x^{(l)} - \frac{1}{a_{ii}} \left(\sum_{j=1}^{n} a_{ij} x_j^{(l)} - b_i \right) e_i. \tag{1.34}$$

Se puede comprobar (ver Ortega [81]), que n iteraciones del esquema (1.34) son equivalentes a una iteración del método de Gauss-Seidel en el sistema $Ax =$

b. En este contexto, el método de Gauss-Seidel se conoce también como método de *relajación univariante*, ya que en cada iteración del esquema (1.34) sólo se cambia una de las variables.

Dentro de los métodos de minimización, cabe destacar los llamados métodos de direcciones conjugadas, que aparecen cuando elegimos n vectores de dirección no nulos $p^{(0)}, p^{(1)}, \ldots, p^{(n-1)}$ que satisfacen la condición

$$(p^{(i)})^T A p^{(j)} = 0, \quad \text{siempre que } i \neq j. \tag{1.35}$$

Estos vectores son ortogonales con respecto al producto escalar definido por A y se dice que son vectores A-ortogonales o conjugados con respecto a A.

Una de las propiedades básicas que verifican los métodos de direcciones conjugadas viene dada por el siguiente teorema.

Teorema 1.23. *(Teorema 3.3.1, [81]). Si $A \in \mathbb{R}^{n\times n}$ es una matriz simétrica y definida positiva y $p^{(0)}, \ldots, p^{(n-1)}$ son vectores no nulos A-conjugados, entonces para algún $x^{(0)}$ el vector iterado $x^{(l+1)} = x^{(l)} - \alpha_l p^{(l)}$, donde α_l es elegido como en (1.33), converge a la solución exacta del sistema (1.28), en no más de n etapas.*

Como podemos observar, este teorema asegura no sólo que el método converge, sino que en ausencia de errores de redondeo, converge en un número finito de iteraciones menor o igual que n. Por ello y basándonos en esta propiedad, podemos decir que los métodos de direcciones conjugadas son realmente métodos directos. Pero a causa de los errores de redondeo, esta importante propiedad no deja de ser una propiedad teórica, ya que generalmente lo que se obtiene en muchos menos de n pasos es una buena aproximación. Por tanto, se pueden tratar los métodos de direcciones conjugadas como métodos iterativos.

En 1952, Hestenes y Stiefel [56] diseñaron un método de resolución de sistemas de ecuaciones lineales $Ax = b$, siendo A una matriz simétrica y definida positiva, al que llamaron método del gradiente conjugado.

Este método, es un método de direcciones conjugadas donde las direcciones de búsqueda se construyen haciendo A-ortogonales los vectores residuales $\tau^{(l)}$.

El siguiente algoritmo describe este método. Notamos que aquí usamos el producto interno $\langle x, y\rangle = x^T y$.

Algoritmo 1.1. Algoritmo del Gradiente Conjugado.

Dado un vector inicial $x^{(0)}$.

$$p^{(0)} = \tau^{(0)} := b - Ax^{(0)}$$

REPETIR

$$\alpha_l := \frac{\langle \tau^{(l)}, \tau^{(l)}\rangle}{\langle p^{(l)}, Ap^{(l)}\rangle}$$

$$x^{(l+1)} := x^{(l)} - \alpha_l p^{(l)}$$

$$\tau^{(l+1)} := \tau^{(l)} - \alpha_l Ap^{(l)}$$

SI test de convergencia = VERDADERO

entonces FIN

$$\beta_l := -\frac{\langle \tau^{(l+1)}, \tau^{(l+1)}\rangle}{\langle \tau^{(l)}, \tau^{(l)}\rangle}$$

$$p^{(l+1)} := \tau^{(l+1)} - \beta_l p^{(l)}.$$

Como podemos observar en este algoritmo, las direcciones A-conjugadas se van generando a la vez que avanzamos en el cálculo de la solución. Esto es una de las ventajas de este método, ya que no se tienen que calcular de antemano un conjunto de direcciones A-conjugadas.

El criterio de parada que sugiere Ortega en [81], consiste en asegurar que la norma del vector residual $\tau = b - Ax$, sea menor que una cota de error

predeterminada. Si denotamos por $e^{(l)}$ el error que se comete en la iteración l-ésima y x^* es la solución exacta del sistema $Ax = b$, se tiene que

$$e^{(l)} = x^{(l)} - x^* = A^{-1}(Ax^{(l)} - b) = A^{-1}\tau^{(l)}.$$

Por tanto

$$\|e^{(l)}\| \leq \|A^{-1}\|\|\tau^{(l)}\|.$$

Entonces, si se tiene una cota para la norma de A^{-1} exigiendo que $\|\tau^{(l)}\| \leq \epsilon$, podemos asegurar que se verifica $\|e^{(l)}\| \leq \epsilon\|A^{-1}\|$.

Otros criterios de parada que implican otros tipo de cota para el error a priori son:

1. $\|\tau^{(l)}\| \leq \epsilon(\|A\|\|x^{(l)}\| + \|b\|)$.
2. $\|\tau^{(l)}\| \leq \epsilon\|b\|$.
3. $\|\tau^{(l)}\| \leq \epsilon\dfrac{\|x^{(l)}\|}{\|A^{-1}\|}$.
4. $\|\tau^{(l)}\| \leq \epsilon\|\tau^{(0)}\|$.

Existen dos casos importantes en los cuales la solución se alcanza, suponiendo que se trabaja en aritmética exacta, en un número de iteraciones m menor que n. El siguiente teorema muestra estos dos casos.

Teorema 1.24. *(Teorema 9.1.3, [52]). Los vectores iterados por el método del gradiente conjugado convergen a la solución del sistema en no más de m iteraciones si se cumple cualquiera de las condiciones siguientes:*

1.- $p^{(0)} = \tau^{(0)}$ es combinación lineal de m vectores propios de A.

2.- A tiene sólo m valores propios distintos.

Los siguientes teoremas estudian el error que se comete al usar el método del gradiente conjugado. Así, el teorema que enunciamos a continuación muestra que los errores son decrecientes de una iteración a otra.

Teorema 1.25. *(Teorema A.3.5, [81]). Los vectores iterados $x^{(l)}$ del método del gradiente conjugado satisfacen*

$$\|x^* - x^{(l)}\|_2 < \|x^* - x^{(l-1)}\|_2,$$

excepto cuando $x^{(l-1)} = x^*$.

Por último, damos una cota del error en términos del número de condición de la matriz A, utilizando la A-norma y la norma euclídea.

Teorema 1.26. *(Teorema A.3.4, [81]) El vector iterado $x^{(l)}$ del método del gradiente conjugado satisface*

$$\|x^* - x^{(l)}\|_A \leq 2\mu^l \|x^* - x^{(0)}\|_A,$$

y

$$\|x^* - x^{(l)}\|_2 \leq 2\sqrt{c}\mu^l \|x^* - x^{(0)}\|_2,$$

donde $c = cond(A)$ y $\mu = \frac{\sqrt{c}-1}{\sqrt{c}+1}$.

Teniendo en cuenta que A es una matriz simétrica y definida positiva, y suponiendo que los valores propios de A ordenados en orden creciente son $\lambda_n \geq \cdots \geq \lambda_1 > 0$, el número de condición de A puede escribirse

$$c = cond(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\lambda_n}{\lambda_1}$$

Se observa que si $c = 1$ entonces $\mu = 0$, y cuando c tiende a $+\infty$, μ tiende a 1. De esta forma, cuanto más grande sea c, μ se acerca más a 1 y por tanto, la convergencia será más lenta.

1.7 Método del gradiente conjugado precondicionado

El razonamiento hecho al final de la Sección 1.6 sugiere, que si transformamos el sistema de forma que la matriz de coeficientes esté mejor condicionada, es decir, su número de condición sea menor, el método convergerá con un número menor de iteraciones. Esta es la idea básica del método del gradiente conjugado precondicionado.

Dado el sistema $Ax = b$, buscamos una matriz no singular S tal que $cond(\hat{A}) < cond(A)$, siendo $\hat{A} = SAS^T$. Así, el nuevo sistema a resolver es

$$\hat{A}\hat{x} = \hat{b},$$

donde $\hat{x} = S^{-T}x$ y $\hat{b} = Sb$, sistema que al estar mejor condicionado convergerá con mayor rapidez.

Aplicando el método del gradiente conjugado a este nuevo sistema se obtienen los vectores iterados $\hat{x}^{(l)}$, que premultiplicados por la matriz S^T proporcionan los vectores solución $x^{(l)}$.

Para $l = 0$ se tienen las siguientes relaciones entre los vectores de ambos sistemas:

$$\begin{aligned}
\hat{r}^{(0)} &= \hat{b} - \hat{A}\hat{x}^{(0)} = S r^{(0)}, \\
\hat{p}^{(0)} &= S^{-T} p^{(0)}, \\
\langle \hat{p}^{(0)}, \hat{A}\hat{p}^{(0)} \rangle &= \langle p^{(0)}, A p^{(0)} \rangle, \\
\langle \hat{r}^{(0)}, \hat{r}^{(0)} \rangle &= \langle S^T S r^{(0)}, r^{(0)} \rangle.
\end{aligned}$$

Si se define $s^{(0)} = S^T S r^{(0)}$ entonces,

$$\hat{\alpha}_0 = -\frac{\langle s^{(0)}, r^{(0)} \rangle}{\langle p^{(0)}, A p^{(0)} \rangle}.$$

Las fórmulas anteriores dan las bases para escribir el algoritmo del gradiente conjugado precondicionado en términos de los datos originales del problema. Sin embargo, hemos de calcular el vector auxiliar $s^{(0)}$. Si se conoce la matriz S ó $S^T S$, este cálculo se reduce a una multiplicación. Pero este no es generalmente el caso. De hecho, se suele trabajar con una aproximación de la matriz A de forma que la matriz S está definida sólo implícitamente. Se tiene entonces la siguiente definición.

Definición 1.21. Se llama *matriz precondicionante o precondicionador* a la matriz simétrica definida positiva $\mathcal{M} = (S^T S)^{-1}$.

De este modo, el vector $s^{(0)}$ se obtiene resolviendo el sistema lineal $\mathcal{M} s^{(0)} = r^{(0)}$.

Con todas estas consideraciones, el algoritmo para el método del gradiente conjugado precondicionado es el siguiente.

Algoritmo 1.2. Algoritmo del Gradiente Conjugado Precondicionado.

Dado un vector inicial $x^{(0)}$

$r^{(0)} := b - Ax^{(0)}$

Resolver $\mathcal{M}s^{(0)} = r^{(0)}$

$p^{(0)} := s^{(0)}$

REPETIR

$\alpha := -\frac{\langle s^{(l)}, r^{(l)} \rangle}{\langle p^{(l)}, Ap^{(l)} \rangle}$

$x^{(l+1)} := x^{(l)} - \alpha p^{(l)}$

$r^{(l+1)} := r^{(l)} - \alpha Ap^{(l)}$

Resolver $\mathcal{M}s^{(l+1)} = r^{(l+1)}$

SI test de convergencia = VERDADERO

entonces FIN

$\beta := -\frac{\langle s^{(l+1)}, r^{(l+1)} \rangle}{\langle s^{(l)}, r^{(l)} \rangle}$

$p^{(l+1)} := s^{(l+1)} - \beta p^{(l)}$

La diferencia esencial entre el algoritmo del gradiente conjugado y el del gradiente conjugado precondicionado reside en la resolución del sistema auxiliar $\mathcal{M}s = r$, que se realiza en cada iteración de algoritmo del gradiente conjugado precondicionado. Concretamente, ver Ortega [81], el coste por iteración del gradiente conjugado precondicionado es igual al coste por iteración del gradiente conjugado más la resolución del sistema auxiliar. Por tanto, gran parte del coste computacional del método del gradiente conjugado precondicionado corresponde a la resolución de estos sistemas auxiliares, que puede oscilar de $O(n)$ a $O(n^3)$, en función de la estructura de la matriz $\mathcal{M}$.

Esto nos lleva a pensar, que será conveniente encontrar una matriz precondicionante $\mathcal{M}$ que haga que el sistema $\mathcal{M}s = \tau$ sea fácil de resolver. Como precondicionador ideal se podría considerar aquel que consiga un equilibrio entre el número de iteraciones necesarias para encontrar la solución del sistema original y el coste computacional de la resolución del sistema auxiliar en cada iteración, ya que minimizaría el coste computacional del algoritmo.

Existen dos criterios generales para la elección de la matriz $\mathcal{M}$. Uno de ellos se basa en la segunda condición del Teorema 1.24. Puesto que si A tiene sólo m valores propios distintos, el método del gradiente conjugado converge en no más de m iteraciones, aunque éste no es generalmente el caso en problemas prácticos, eligiendo adecuadamente la matriz $\mathcal{M}$, podríamos conseguir que algunos de los valores propios de $\hat{A}$ estuvieran agrupados en torno a un valor, con lo que mejoraría la velocidad de convergencia del algoritmo.

El segundo criterio es elegir $\mathcal{M}$ de forma que la matriz $\hat{A}$ tenga un número de condición mucho menor que A. Se observa que la matriz $\hat{A}$ satisface

$$S^T \hat{A} S^{-T} = S^T S A = \mathcal{M}^{-1} A,$$

y por tanto, las matrices $\hat{A}$ y $\mathcal{M}^{-1}A$ son semejantes y, de aquí,

$$cond(\hat{A}) = \frac{\lambda_{\max}(\mathcal{M}^{-1}A)}{\lambda_{\min}(\mathcal{M}^{-1}A)}.$$

Puesto que el objetivo es reducir al máximo el número de condición de $\hat{A}$, se elige $\mathcal{M}$ de forma que sea simétrica y definida positiva (como se exige en la definición) y tal que la razón entre el mayor y el menor de los valores propios de $\mathcal{M}^{-1}A$ sea lo más próxima a 1. En principio, se podría tomar $\mathcal{M} = A$, con lo que $cond(\hat{A}) = 1$, pero esta elección no es nada práctica ya que el sistema

auxiliar a resolver en cada iteración (en realidad sólo sería una iteración) sería el sistema original.

Intuitivamente, precondicionar puede interpretarse como el intento de hacer que la forma cuadrática $Q(x)$ sea más esférica, de forma que los valores propios se acerquen unos a otros (recordemos que una esfera es una forma cuadrática en la que la matriz tiene todos los valores propios iguales). Tomando $\mathcal{M}= A$ conseguiríamos que la forma cuadrática fuese una esfera, y llegaríamos a la solución con una iteración.

Claramente, los dos requerimientos exigidos a la matriz $\mathcal{M}$ (que sea una buena aproximación de A^{-1} y que el sistema $\mathcal{M}s = \tau$ sea fácil de resolver) son contradictorios, puesto que cuanto más se aproxime $\mathcal{M}$ a la matriz A, el problema de resolver el sistema auxilar $\mathcal{M}s = \tau$, se acerca cada vez más a la dificultad de resolución del problema original $Ax = b$.

Existen diversas técnicas para elegir la matriz precondicionante $\mathcal{M}$. Algunas de ellas se estudiarán más a fondo en la siguiente sección.

1.8 Precondicionadores secuenciales

Existen diversos procedimientos para calcular precondicionadores secuenciales, que se han utilizado para resolver sistemas de ecuaciones lineales en máquinas secuenciales o vectoriales. En esta sección veremos los *precondicionadores polinomiales* y los *precondicionadores basados en la factorización incompleta de Choleski.*

1.8.1 Precondicionadores polinomiales

Este tipo de precondicionadores fue introducido por Stiefel [94] para el cálculo de valores propios. Posteriormente, Rutishauser en [87], adaptó el método dado por Stiefel para sistemas lineales.

Dubois, Greenbaum y Rodrigue en [36], construyen un precondicionador basado en aproximaciones por series truncadas de Neumann, usando el método de Jacobi para resolver el sistema auxiliar $\mathcal{M}s = \tau$.

Johnson, Micchelli y Paul en [59] basados en los resultados de Dubois, Greenbaum y Rodrigue [36], estudian también precondicionadores polinomiales usando el método de Jacobi.

En [1] y [2], Adams extiende los resultados dados por Dubois, Greenbaum y Rodrigue [36] y Johnson, Micchelli y Paul [59], respectivamente, estudiando también precondicionadores mediante series truncadas pero considerando m pasos de un método iterativo subsidiario, incluyendo, en particular, los métodos de Jacobi y SSOR.

Para describir la técnica de los precondicionadores polinomiales, consideramos el siguiente teorema que proporciona una expresión para la matriz A^{-1}, y que se conoce con el nombre de *expansión de Neumann.* Dicho teorema es básicamente el Lema 1.1.

Teorema 1.27. *(Teorema 3.4.1, [81]). Sea A una matriz no singular y $A = M - N$ una partición de A tal que el radio espectral de $M^{-1}N$ es menor que 1. Entonces,*

$$A^{-1} = \left(\sum_{l=0}^{\infty} H^l\right) M^{-1}, \qquad (1.36)$$

siendo $H = M^{-1}N$.

Basándonos en la expresión de A^{-1} en (1.36) podemos considerar, bajo ciertas condiciones de las matrices M y H, las matrices

$$\mathcal{M} = M\left(I + H + \cdots + H^{m-1}\right)^{-1},$$

o bien

$$\mathcal{M}^{-1} = \left(I + H + \cdots + H^{m-1}\right) M^{-1}, \tag{1.37}$$

como aproximaciones de A y de A^{-1}, respectivamente.

Entonces, la solución del sistema auxiliar $\mathcal{M}s = \tau$ que aparece en el algoritmo del gradiente conjugado precondicionado, viene dada por la expresión

$$s = \mathcal{M}^{-1}\tau = \left(I + H + \cdots + H^{m-1}\right) M^{-1}\tau,$$

donde $H = M^{-1}N$.

A la matriz $\mathcal{M}= M\left(I + H + \ldots + H^{m-1}\right)^{-1}$ se le llama *precondicionador polinomial de m pasos.*

Se pueden construir precondicionadores polinomiales más generales introduciendo coeficientes $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ de tal forma, que la matriz $\mathcal{M}$ siga siendo simétrica y definida positiva y además, que la razón entre el mayor y el menor valor propio de $\mathcal{M}^{-1}A$ sea mínima. Entonces, el precondicionador polinomial de *m-pasos generalizado*, viene dado por la expresión

$$\mathcal{M} = M\left(\alpha_0 I + \alpha_1 H + \cdots + \alpha_{m-1}H^{m-1}\right)^{-1},$$

donde H es la matriz de iteración correspondiente a la partición $A = M - N$.

Una de las propiedades más interesantes de estos precondicionadores es que no es necesario tener una expresión explícita de la matriz $\mathcal{M}$, debido al siguiente hecho: si volvemos a considerar el algoritmo del gradiente conjugado precondicionado, una vez obtenida $\mathcal{M}$, tendremos que resolver el sistema auxiliar $\mathcal{M}s = \tau$. Todo esto puede efectuarse de forma equivalente y sin necesidad de calcular $\mathcal{M}$, realizando sobre el sistema $As = \tau$, m iteraciones del esquema

$$Ms^{(l)} = Ns^{(l-1)} + \tau, \quad l = 1, 2, \ldots, m, \tag{1.38}$$

empezando con $s^{(0)} = 0$, de forma que la solución del sistema $\mathcal{M}s = \tau$ vendrá dada por el vector $s^{(m)}$, siendo $\mathcal{M}$ el precondicionador polinomial dado en (1.37). Por tanto, para resolver el sistema auxiliar, sólo es necesario aplicar el esquema iterativo asociado a la partición $A = M - N$ que define el precondicionador $\mathcal{M}$.

Podemos construir distintos precondicionadores polinomiales, estos resultarán de considerar distintos tipos de particiones para la matriz A. Por ejemplo, como caso particular de precondicionador polinomial, se puede elegir M como la diagonal de A, entonces el esquema iterativo (1.38) corresponde al método iterativo de Jacobi y por tanto realizando un numero fijo m de iteraciones del esquema (1.38) se obtiene el método *PCG-Jacobi* de m-pasos. Sin embargo, no podemos usar los métodos iterativos de Gauss-Seidel o SOR, para resolver el sistema auxiliar ya que el precondicionador obtenido con ellos, no siempre es una matriz simétrica y definida positiva. Esta dificultad se soluciona utilizando el método SSOR, que consiste básicamente en realizar un paso del método SOR seguido de otro paso del SOR en orden inverso (ver Sección 1.4).

1.8.2 Precondicionadores basados en la factorización incompleta de Choleski

Otra técnica usual de precondicionamiento es la basada en la factorización incompleta de Choleski. Veamos primero en qué consiste dicha factorización. Si la matriz A es simétrica y definida positiva, existe una matriz triangular inferior L no singular tal que $A = LL^T$. Esta descomposición se conoce como *factorización de Choleski.* Pero cuando la matriz A es dispersa, es decir, el número de elementos no nulos de A es del orden del tamaño de la matriz, al realizar la factorización de Choleski de A generalmente resulta una matriz L mucho menos dispersa que la matriz A. Se produce entonces un *llenado* de la matriz L, es decir, muchos elementos nulos de A son distintos de cero en L. Una forma de suprimir este llenado o parte de él , es mediante la *factorización incompleta de Choleski.*

Supongamos que A es una matriz $n \times n$ y consideremos el conjunto G_n de todos los pares de índices de las entradas de la matriz A, es decir,

$$G_n = \{(i,j) : 1 \leq i,j \leq n\}.$$

Sea G un subconjunto de G_n que verifica:

$$\begin{aligned} &a)\ (i,i) \in G, \quad 1 \leq i \leq n. \\ &b)\ Si\ (i,j) \in G,\ \text{entonces}\ (j,i) \in G, \end{aligned} \tag{1.39}$$

es decir, G contiene los índices de la diagonal de A y además es un conjunto simétrico. Al conjunto G así formado se le llama *conjunto no cero* de la factorización.

Realizaremos la factorización incompleta de Choleski de acuerdo al siguiente principio:

- Si $(i,j) \in G$, calcular l_{ij} mediante la factorización de Choleski.
- Si $(i,j) \notin G$, hacer $l_{ij} = 0$.

Entonces, se tiene que

$$A = M - N = LL^T - N, \tag{1.40}$$

es una partición de A (si M es no singular).

La descomposición de A de la expresión (1.40) es la factoriación incompleta de Choleski de A. Se puede comprobar además que los elementos no nulos de M coinciden con los correspondientes elementos de A (ver [65]).

El algoritmo para la factorización incompleta de Choleski de una matriz A de tamaño $n \times n$, considerando el conjunto no cero G, es el siguiente.

Algoritmo 1.3. Algoritmo de la Factorización Incompleta de Choleski.

Calcular $l_{11} = a_{11}^{1/2}$

Para $i = 2$ hasta n

Para $j = 1$ hasta $i - 1$

Si $(i,j) \notin G$ entonces $l_{ij} = 0$

Si no, calcular $l_{ij} = \dfrac{1}{l_{jj}} \left(a_{ij} - \displaystyle\sum_{k=1}^{j-1} l_{ik} l_{jk} \right)$

Calcular $l_{ii} = \left(a_{ii} - \displaystyle\sum_{k=1}^{i-1} l_{ik}^2 \right)^{1/2}$.

Como ya hemos comentado anteriormente, realizar la factorización incompleta de Choleski soluciona el problema de llenado que se puede producir al hacer la factorización completa; para ilustrar esta situación damos un ejemplo.

Ejemplo 1.2. *Sea la matriz*

$$A = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \end{bmatrix}.$$

Calculamos la matriz L que resulta de realizar la factorización completa de Choleski a la matriz A

$$L = \begin{bmatrix} 2 & 0 & 0 \\ 0,5 & 1,93 & 0 \\ 0,5 & -0,13 & 1,86 \end{bmatrix}.$$

Como podemos observar, en la componente $(3,2)$ de la matriz A que tenía valor cero, se ha realizado un llenado al realizar la factorización, ya que en L pasa a valer $-0,13$. Sin embargo, si realizamos la factorización incompleta de Choleski a la matriz A, tomando como conjunto no cero $G = \{(i,i) : 1 \leq i \leq 3\} \cup \{(3,1)\}$, tenemos

$$L = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0,5 & 0 & 1,93 \end{bmatrix}.$$

Como podemos observar, el elemento $(3,2)$ ahora no se ha llenado debido a que no pertenece al conjunto de elementos no cero, y según el principio anterior no se debe factorizar.

Otro criterio para realizar la factorización incompleta, es el que da Ortega en [81] que dice

> Si $a_{ij} \neq 0$, calcular l_{ij} mediante la factorización de Choleski.
>
> Si $a_{ij} = 0$, hacer $l_{ij} = 0$.

Con este criterio, la matriz del ejemplo anterior tendrá la factorización incompleta de Choleski, dada por la siguiente matriz triangular inferior L.

$$L = \begin{bmatrix} 2 & 0 & 0 \\ 0,5 & 1,93 & 0 \\ 0,5 & 0 & 1,93 \end{bmatrix}.$$

Dentro de una elección aceptable del conjunto G, se puede permitir un cierto nivel de llenado. Por ejemplo, si A es una matriz diagonalmente dispersa, se pueden elegir $h+k$ diagonales por debajo de la diagonal principal para realizar la factorización. Es decir, además de la diagonal principal y la última subdiagonal distinta de cero, se factorizan $h-1$ diagonales desde la diagonal principal, y $k-1$ diagonales desde la última subdiagonal distinta de cero hacia la diagonal principal.

Claramente, según el subconjunto G que se elija se tienen distintas factorizaciones incompletas de una matriz. Desafortunadamente, a diferencia de la factorización completa de Choleski, la factorización incompleta de Choleski no siempre existe aunque la matriz a la que se le realice la factorización sea simétrica y definida positiva. Esto es porque puede ocurrir que al realizar dicha factorización, se necesite realizar una raiz cuadrada de un número negativo.

Por ejemplo si consideramos la matriz A simétrica y definida positiva

$$A = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{bmatrix},$$

y como conjunto $G = \{(i,i) : 1 \leq i \leq 5\} \cup \{(i,5) : 1 \leq i \leq 5\} \cup \{(5,i) : 1 \leq i \leq 5\}$, se puede comprobar que al realizar la factorización incompleta de Choleski de A, el algoritmo falla cuando se tiene que calcular el elemento $(5,5)$ de la matriz L, ya que sería necesario calcular la raiz cuadrada de $-0,5$.

Podemos garantizar dicha factorización, si la matriz A es una M-matriz. Esto queda reflejado en el siguiente teorema, que puede verse en [68]. También en [97] se obtienen resultados similares.

Teorema 1.28. *(Teorema 2.4, [68]). Si A es una M-matriz simétrica, entonces, para cada elección del subconjunto $G \subset G_n$ con las propiedades (1.39), existe una única matriz triangular inferior L tal que, $l_{ij} = 0$ si $(i,j) \notin G$ y una matriz no negativa N con $n_{ij} = 0$ si $(i,j) \in G$, de manera que*

$$A = M - N = LL^T - N \qquad (1.41)$$

es una partición regular de A.

A continuación vamos a considerar el uso de factorizaciones incompletas de Choleski como precondicionadores para el método del gradiente conjugado. Sea $A = LL^T - N$ una factorización incompleta de Choleski de la matriz A, con L

no singular. Entonces elegimos como matriz precondicionante $\mathcal{M}$ del algoritmo del gradiente conjugado precondicionado la matriz

$$\mathcal{M} = LL^T,$$

donde $\mathcal{M}$ es una matriz simétrica y definida positiva. Puesto que L es no singular, la resolución del sistema $\mathcal{M}s = \tau$, se llevará a cabo resolviendo estos dos sistemas triangulares

$$Lz = \tau, \quad L^T s = z.$$

Tenemos que hacer notar, que la factorización incompleta de Choleski se realizará una sola vez al principio del proceso iterativo, y se deberá guardar la matriz L para resolver los sistemas triangulares en cada iteración.

Esta elección de la matriz $\mathcal{M}$, define la clase general de los métodos del gradiente conjugado precondicionado basados en la factorización incompleta de Choleski, denotados por ICCG. Según el número de diagonales que se llenen al hacer la factorización, se obtienen distintos métodos ICCG(k), donde k denota el número de diagonales elegidas para la factorización.

Se puede comprobar, que si A es una matriz en banda, es decir, una matriz tal que $a_{ij} = 0$, si $i - j > \beta$ o bien $j - i < \beta$, para algún β llamado *ancho de banda*, menor que el tamaño de A, entonces la factorización de Choleski de A proporciona una matriz triangular inferior L que cumple esta misma propiedad. Es decir, si A es una matriz en banda, la matriz L también es una matriz en banda con el mismo ancho de banda β. Por tanto, al realizar la factorización incompleta de Choleski de la matriz, deberemos fijarnos en que el llenado se produce dentro de la banda.

Generalmente, si la matriz A es diagonalmente dispersa, es decir, si A tiene pocas diagonales no nulas, la estrategia de llenado suele ser elegir G de forma que la matriz L tenga como máximo alguna diagonal no nula más que A (contando sólo las diagonales no nulas de A por debajo de la diagonal principal, ya que L es una matriz triangular inferior).

Capítulo 2

Arquitecturas paralelas.

2.1 Introducción

Desde que en 1940, Von Neumann presentó el primer computador secuencial de propósito general, con la idea de procesar datos en máquinas secuenciales, el desarrollo tecnológico ha hecho posible el logro de máquinas que son capaces del tratamiento automatizado a gran velocidad.

Ya son seis décadas, en las que gracias a los avances de la tecnología y de la electrónica, se viene produciendo un considerable avance en la evolución de los computadores debido a la demanda que existe, tanto en el campo científico como en el industrial y comercial.

Los avances en la arquitectura de los computadores se han realizado en sus dos vertientes que son: las mejoras en la organización del hardware y los requerimientos del software. En el campo del hardware, las mejoras se dan cuando

aparecen nuevas arquitecturas de computadores con la meta de alcanzar mayor rapidez en la ejecución de las intrucciones, o en la realización de operaciones de entrada/salida (en lo sucesivo E/S), conseguidas mediante la construcción de dispositivos electrónicos más rápidos. Las mejoras para el entorno del software aparecen cuando se realizan nuevas modificaciones para los algoritmos ya existentes, que hacen que éstos se ejecuten con mayor rapidez, también en la construcción de compiladores y lenguajes de programación adaptados a las nuevas máquinas.

La arquitectura de los computadores, en el contexto de la organización hardware, ha avanzado bastante en el transcurso de seis décadas desde sus comienzos con las máquinas secuenciales hasta los procesadores vectoriales y los computadores paralelos.

Las primeras máquinas que se construyeron fueron las máquinas secuenciales de Von Neumann para la ejecución de datos escalares. Éstas se denominaron *computadores de flujo de control*, porque las instrucciones están controladas por el contador de programas. Al ser la ejecución de los programas secuenciales muy lenta, se propusieron otros modelos de computadores. Aparecen entonces las técnicas de "Lookahead" para operaciones I/E (instrucciones de decodificación y ejecución) y para la introducción del paralelismo funcional.

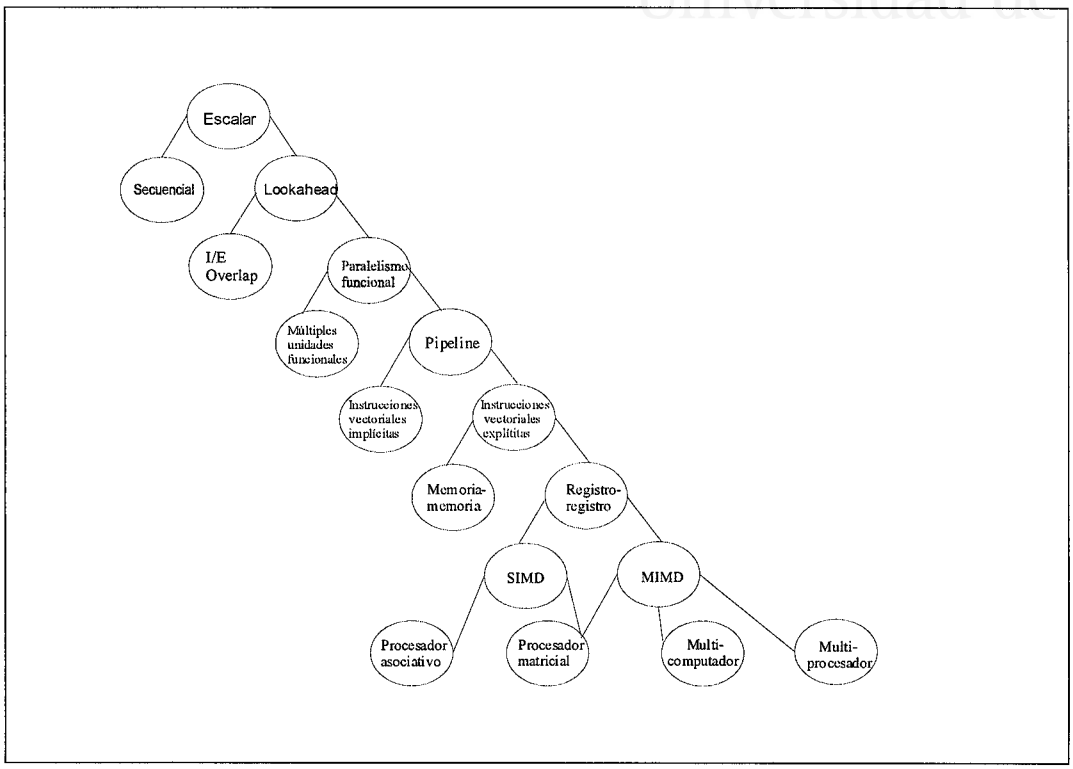
A partir de las construcciones "pipeline" aparecen los procesadores vectoriales, equipados con múltiples vectores "pipelines" para ser usados concurrentemente bajo control hardware. Dentro de los procesadores vectoriales "pipelines", se construyen las arquitecturas *memoria-a-memoria*, que soportan el flujo "pipeline" de operaciones entre vectores, y la arquitectura *registro-a-registro* que usa registros de vectores para hacer de interface entre la memoria y los conductos

funcionales.

Después aparecen los computadores *de flujo de datos*. Su principal característica es que no precisan contadores, ya que la ejecución de una instrucción obedece sólamente a las restricciones impuestas por la dependencia de los datos de todos ellos. El más utilizado es el computador con organización *MIMD* (Multiple Instruction-Multiple Data). Dentro de las arquitecturas *MIMD* y atendiendo a la situación física de la memoria y a la forma en que los procesadores acceden a ella, aparecen los *multiprocesadores de memoria compartida* (shared memory), los *multiprocesadores de memoria distribuida* (distributed memory), y los multicomputadores.

El auge en la arquitectura de los computadores paralelos va desde 1975 hasta nuestros días, donde se produce un avance significativo debido al planteamiento de nuevas tecnologías que surgen tanto en la industria como en el campo de la investigación. En la actualidad, estas máquinas se utilizan en ámbitos donde se aplica la supercomputación, como aplicaciones científicas, Ingeniería (diseño de aviones, simulación de circuitos, síntesis lógica, ...), sistemas de bases de datos (paralelización de bases de datos (queries), procesamiento de lenguaje natural, ...), e Inteligencia Artificial (paralelización de procesos de lógica simbólica, modelado conexionista, ...).

En la siguiente figura, se muestra la evolución de los distintos computadores desde el secuencial hasta los computadores paralelos:



En las siguientes secciones vamos a centrarnos en la computación paralela. En la Sección 2.2, estudiaremos el concepto de paralelismo. En la Sección 2.3, describiremos diferentes arquitecturas paralelas poniendo especial atención a las arquitecturas de tipo $MIMD$, ya que es en este tipo de arquitectura donde realizaremos los experimentos numéricos. En la Sección 2.4, veremos distintos modelos de programación paralela vigentes hoy día. En la Sección 2.5 se estudiarán las medidas de paralelismo que se utilizan para evaluar un algoritmo paralelo y para finalizar este capítulo, en la Sección 2.6 daremos una breve descripción del paquete de software PVM (Parallel Virtual Machine).

2.2 Concepto de paralelismo

El procesamiento paralelo es una técnica que permite incrementar la capacidad computacional de los sistemas informáticos mediante la ejecución concurrente de distintas operaciones. La introducción del paralelismo en los computadores hace que se aumente la capacidad computacional a costa, por lo general, de aumentar las componentes hardware.

La necesidad de introducir el procesamiento paralelo en los computadores viene justificada por la utilización de los mismos en grandes bases de datos, aplicaciones para búsquedas heurísticas, ... , incluso existen aplicaciones cuyo requerimiento computacional es mucho mayor.

Son muchos los campos donde las arquitecturas paralelas se consideran apropiadas bien por el gran tamaño de los problemas que se abordan o por la necesidad de trabajar con problemas en tiempo real. En particular, una de las áreas donde se investiga y se utiliza la programación y por lo tanto, las arquitecturas paralelas, es el Álgebra Lineal Numérica. En ella, por ejemplo, se están desarrollando algoritmos para la resolución de sistemas de ecuaciones lineales implementados para ser ejecutados en máquinas paralelas. Su justificación radica en el uso de vectores y matrices, ya que su utilización implica de forma implícita la utilización del paralelismo.

Por lo tanto, la aparición de los computadores paralelos ha dado lugar no sólo a que se adapten los algoritmos clásicos del Álgebra Lineal para su implementación y ejecución en dichos ordenadores, sino que también se diseñen nuevos métodos que hacen que se obtenga un mejor rendimiento de la máquina,

por ejemplo, cuando se hace un reparto adecuado de la carga de trabajo entre los distintos procesadores.

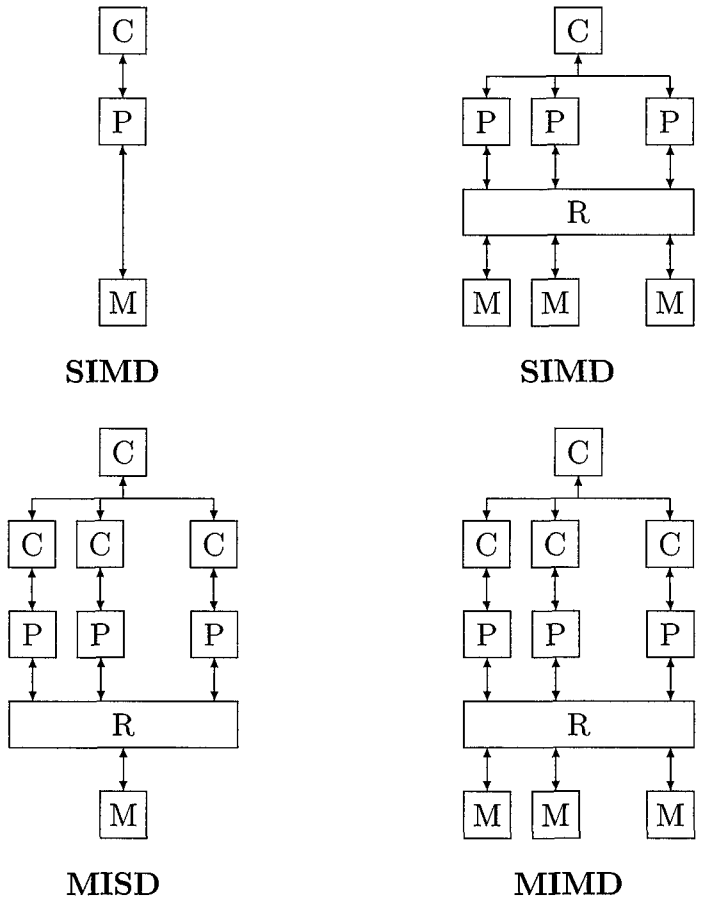
Dentro de un sistema informático el procesamiento paralelo puede plantearse a *nivel de registro* y a *nivel de procesador*.

A nivel de registro. En una máquina secuencial se procesa una palabra cada vez, mientras que en una máquina paralela se procesan varias simultáneamente. Este tipo de paralelismo aparece en los procesadores superescalares que ejecutan a la vez varias instrucciones, y en los computadores matriciales que procesan la misma instrucción sobre distintos datos.

A nivel de procesador. Aquí la unidad de información es el programa. En un sistema paralelo con varios procesadores se ejecutan varias unidades de información en paralelo. Es un paralelismo que aparece en los sistemas multiprocesadores.

2.3 Tipos de arquitecturas paralelas

Flynn [42] presenta una clasificación de los computadores paralelos según se procesen uno o más flujos de instrucciones y/o datos. De forma resumida, esta clasificación queda reflejada en la siguiente figura:



Clasificación de Flynn.
C = Controlador. P = Procesador. R = Red de Interconexión. M = Memoria.

A continuación describimos de forma más detallada esta clasificación:

SISD (Single Instruction-Single Data). Un flujo simple de instrucciones opera sobre un flujo simple de datos. Las instrucciones se ejecutan secuencialmente como en el modelo de Von Neumann, pero pueden estar solapadas

en las etapas de ejecución. Un computador SISD puede tener más de una unidad funcional que están controladas por la unidad de control. Se corresponde con la arquitectura Von Neumann.

SIMD (Single Instruction-Multiple Data). Un flujo simple de instrucciones opera sobre múltiples flujos de datos. Se corresponde con las arquitecturas matriciales, donde múltiples unidades de proceso ejecutan concurrentemente, de forma sincronizada, la misma instrucción sobre distintos datos. Este tipo de arquitectura está formada por varios procesadores pero con una única unidad de control, donde se ejecutan las instrucciones escalares y de control. Estas máquinas son apropiadas para problemas característicos donde haya un alto grado de regularidad, por ejemplo, procesamiento de imágenes y ciertos problemas de simulación. Los algoritmos que están diseñados para multicomputadores no pueden, en general, ser ejecutados eficientemente en computadores SIMD.

MISD (Multiple Instruction-Single Data). Un flujo múltiple de instrucciones opera sobre un flujo simple de datos. Cada vez que un dato es extraído de la memoria se le procesa múltiples veces (por diferentes instrucciones o circuitos) antes de regresar a ella. Es similar a la del tipo SISD. Se corresponde con las arquitecturas segmentadas o sistólicas. Es una arquitectura poco práctica y en la actualidad no se fabrica.

MIMD (Multiple Instruction-Multiple Data). Se ejecutan de forma asíncrona distintas instrucciones en sendas unidades de procesamiento. Todos los procesadores comparten accesos a grupos comunes de módulos de memoria, canales E/S y dispositivos periféricos. El sistema operativo controla

todo el sistema, para facilitar las interacciones entre los procesos y sus programas a diferentes niveles. Además de las memorias y dispositivos de E/S compartidos, cada procesador dispone de su propia memoria local y de dispositivos privados. Disponen de un gran número de procesadores que pueden funcionar en paralelo con un coste relativamente bajo. Con un funcionamiento síncrono, se consideran máquinas a las que tienden las arquitecturas SIMD cuando los elementos de proceso tienden a la simplicidad máxima y la memoria local a desaparecer. Este tipo de arquitectura es útil en problemas cuya velocidad de ejecución viene limitada por el cálculo y no por las E/S. Los sistemas que pertenecen a esta configuración son los sistemas **multiprocesador** y los sistemas **multicomputador**.

(i) **Sistemas multiprocesador**. Es un sistema integrado de computadora que contiene dos o más $CPU's$. Estas cooperan en la ejecución de los programas pudiendo realizar una ejecución simultánea entre dos o más porciones del mismo programa o de programas diferentes, de forma síncrona o asíncrona. Las $CPU's$ comparten una memoria principal común y los subsistemas de E/S se comunican mediante paso de mensajes. Un único sistema operativo lleva el control de todo el proceso. Atendiendo a la topología del sistema, los multiprocesadores se pueden clasificar en **multiprocesadores con memoria compartida** y **multiprocesadores con memoria distribuida**.

- **Multiprocesadores con memoria compartida.** Todos los elementos de proceso tienen acceso a una memoria común, aunque de forma independiente, cada uno de ellos puede tener una

pequeña memoria local (antememoria) donde almacena códigos y resultados intermedios. El mecanismo de comunicación se realiza mediante variables compartidas o paso de mensajes simulado mediante estructuras de datos en memoria compartida. Este tipo de sistema permite cierto grado de sincronización que hace posible un retraso en la comunicación por las interrupciones cruzadas entre procesadores. Contiene una red de interconexión que permite acceder a cualquier módulo de memoria. El acceso a esta red es de forma asíncrona lo que hace que puedan plantearse conflictos y sean necesarios mecanismos de arbitraje. Existen tres tipos de multiprocesadores de memoria compartida: el modelo de *acceso-uniforme-memoria* (UMA), el modelo de *acceso-no-uniforme-memoria* (NUMA) y el modelo de *arquitectura-sólo-memoria-caché* (COMA). En el modelo UMA la memoria está físicamente dividida entre todos los procesadores. Todos los procesadores pueden acceder a la memoria en la misma unidad de tiempo. El acceso a la memoria en el modelo $NUMA$ varía con la posición de la palabra de memoria. La memoria está físicamente distribuida entre todos los procesadores, cada una de ellas, se llama memoria local. La colección de todas ellas forma un espacio de dirección global accesible por todos los procesadores, por ejemplo, el multiprocesador CEDAR construido en la Universidad de Illinois, está construido con esta estructura donde cada cluster es un multiprocesador Alliant FX/80. El modelo COMA es un caso especial del modelo NUMA, en el cual la me-

moria principal está dividida en cachés. En este modelo no hay jerarquías entre cada procesador. Todas las cachés forman un espacio de dirección global. La caché remota está asistida por los directorios de las cachés distribuidas.

- **Multiprocesadores con memoria distribuida.** Cada elemento de proceso tiene su propia memoria donde almacena sus propios datos siendo ésta inaccesible a los demás procesadores. Este tipo de multiprocesador se utiliza en sistemas que están débilmente acoplados donde no se necesita gran cantidad de comunicación. Esta se realiza entre los distintos procesadores mediante paso de mensajes, lo cual conlleva un retraso en la ejecución. Son conocidos también como máquinas de *acceso a memoria no remota* (NORMA).

Una clase de sistema multiprocesador considerado altamente paralelo es el *sistema multicomputador.*

(ii) **Sistema multicomputador.**

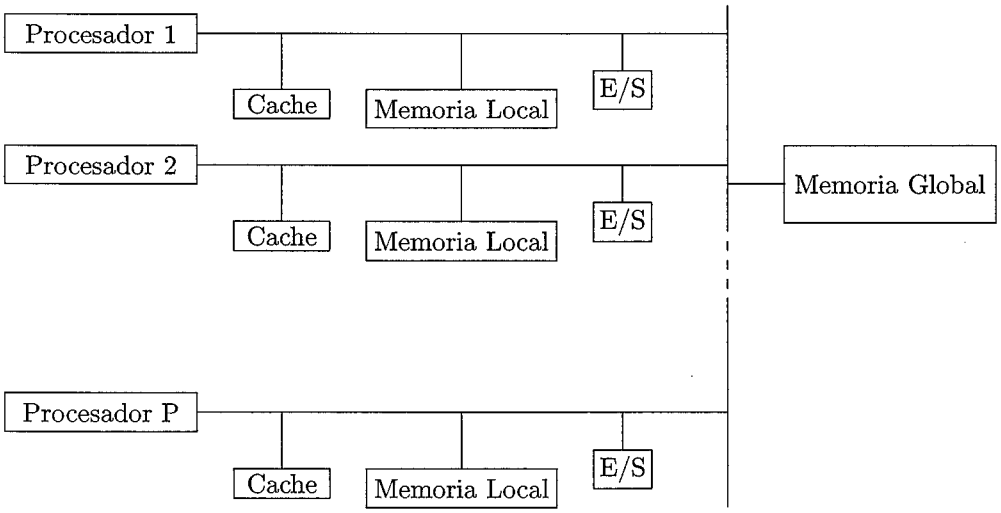
Se organiza como un conjunto de nodos programables conectados mediante una red de paso de mensajes. La red de interconexión puede ser de malla o hipercubo. Cada procesador tiene su propia memoria local, no existiendo una dirección de memoria global sino que cada nodo posee su propio espacio de direcciones. Son sistemas de alto rendimiento en el procesamiento de programas concurrentes, siendo su campo de aplicación el de las operaciones matriciales, resolución de ecuaciones diferenciales, búsquedas heurísticas, análisis de

elementos finitos, ... Algunos multicomputadores comerciales son: Intel iPSC/2, Ametek series 2010, ...

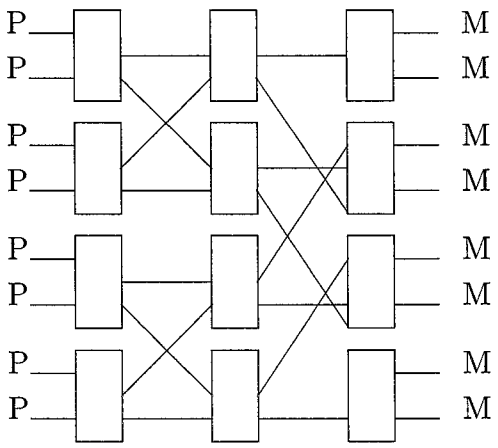
2.3.1 Redes de conexión

En cada sistema multiprocesador como cada procesador comparte con los demás, módulos de memoria y dispositivos E/S, se hace necesaria la existencia de redes de conexión. La función de una red de interconexión en un sistema es la de conectar el mayor número de nodos posibles, con pocos enlaces por nodo y conseguir que haya pocos conflictos de acceso a la red. Existen diferentes tipos de redes de interconexión, vemos algunos de ellos.

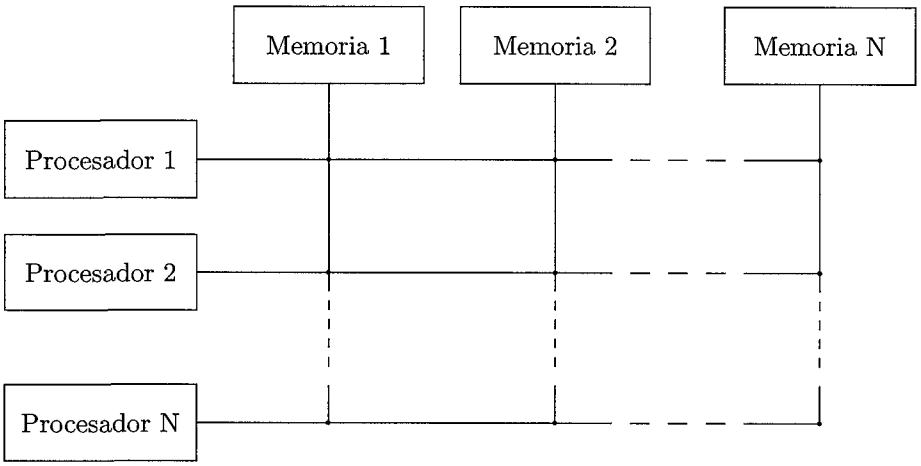
- **Topología de bus:** Esencialmente es una colección de cables y conectores que permiten la transacción de datos, módulos de memoria y periféricos conectados al bus. Generalmente se trata de una red de interconexión pasiva, ya que el control del bus se realiza a través de los dispositivos emisor-receptor o mediante un mecanismo de arbitraje independiente encargado de resolver conflictos.



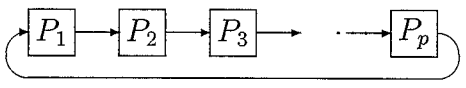
- **Redes multiestado:** Son usadas en computadores SIMD y MIMD. Se conectan $N = 2^n$ procesadores en n fases, cada una de ellas con N conexiones. Este esquema puede utilizarse tanto para sistemas con memoria distribuida, como para sistemas con memoria compartida, según pongamos a la derecha procesadores o memorias.



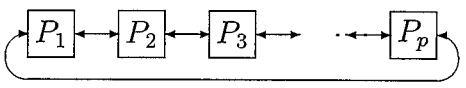
- **Red "Crossbar":** Posee un camino de conexión directo entre cada procesador y cada módulo de memoria. Esto se realiza a través de un bus asociado a cada módulo de memoria. Las ventajas frente a la red bus es que ahora sólo se plantean conflictos cuando se quiere acceder al mismo módulo por dos procesadores.



- **Topología de anillo:** Cada elemento está conectado a los vecinos más próximos por un enlace unidireccional o bidireccional. Un elemento puede leer de un vecino mientras escribe en el otro vecino.

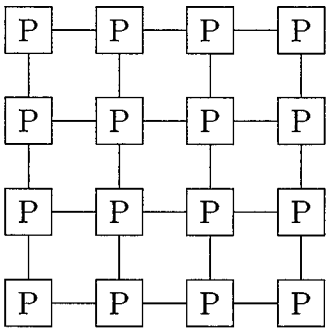


Anillo unidireccional

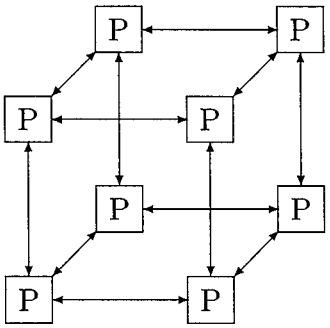


Anillo bidireccional

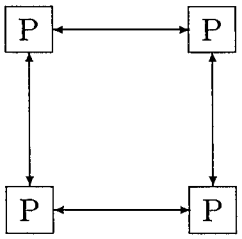
- **Topología de malla:** Cada nodo está conectado a sus cuatro vecinos más próximos. Se dice de malla abierta, cuando además los enlaces de la periferia están anulados.



- **Topología de hipercubo:** Es una arquitectura binaria n-cubo, que puede ser implementada en los sistemas $nCUBE$ y $CM-2$. En general, una topología n-cubo permite conectar un gran número de procesadores, $N = 2^n$, con un diámetro pequeño n y con pocas conexiones. Máquinas construidas con esta topología son Intel iPSC/1, iPSC/2. Una de las ventajas de esta topología es que contiene otras estructuras, en particular la de anillo.



Hipercubo 3-dimensional



Hipercubo 2-dimensional

2.3.2 Modelos de supercomputadores paralelos

A continuación damos una breve descripción de los supercomputadores más utilizados hoy día en computación paralela. Tenemos que destacar que el IBM RS/6000 SP y el *cluster* de Pentiums son los dos multiprocesadores que se han utilizado para obtener los resultados numéricos que se exponen en el Capítulo 7.

IBM RS/6000 SP (SP2)

Es un sistema paralelo escalable de propósito general basado en una arquitectura de memoria distribuida. Proporciona decenas de procesadores dedicados a un único problema. Generalmente está disponible en configuraciones desde 4 hasta 128 procesadores y en algunos casos 512. Las piezas básicas son nodos procesadores que consisten normalmente en un microprocesador POWER, POWER2, POWER3, ó multiprocesador simétrico POWERPC, su memoria, su disco y sus ranuras de expansión microcanal, para la E/S, la conectividad y el adaptador. El IBM RS/6000 SP se caracteriza, entre otros aspectos, por:

- **su escalabilidad:** el sistema puede ampliarse en potencia incorporando más nodos según se necesiten.
- **su compatibilidad:** con todas las versiones existentes para AIX en sistemas inferiores de la familia RS/6000.
- **soportar la computación paralela:** el sistema permite el desarrollo y la ejecución de aplicaciones paralelas.

En su estructura destacan:

- La **estación de control,** desde la cual se instala, administra y monotoriza todo el sistema.

- El **Frame** o armario que contiene los distintos elementos que componen el sistema excepto, la estación de control. Un frame puede contener hasta 16 nodos, pudiendo interconectarse con otros frames para crear sistemas de hasta 128 nodos.

- Los **nodos,** cada uno de los cuales es un ordenador completo con su procesador (POWER2 Super Chip, POWER3 o POWERPC SMP), y su propia memoria, slots de expansión Micro Canal y dispositivos E/S.

- El **High Performance Switch,** que interconecta los nodos de un frame proporcionando un ancho de banda de hasta 110MB/sg entre cada par de nodos.

- Una **LAN Ethernet,** que interconecta los nodos y la estación de control (en este caso con un ancho de banda menor que el proporcionado por el HPS).

- Una **línea serie RS-232,** que conecta la estación de control con cada uno de los frames con el fin de gestionar el hardware. Existe una línea serie por cada frame.

Uno de los lenguajes de datos paralelo que soporta el SP2 es el *High Performance FORTRAN*. Los programas se escriben usando el lenguaje secuencial

FORTRAN (versión FORTRAN 90) donde se especifica mediante grandes arrays (o arreglos) cómo se distribuyen los datos.

Cluster de Pentiums.

Es una red local de computadores heterogéneos que están conectados por cualquier tipo de red (ficra óptica, Ethernet, ...). Los resultados de los tiempos obtenidos en este tipo de máquinas frente al precio que supone un tipo de arquitectura de este tipo pueden considerarse aceptables. El *cluster* del grupo de Computación Paralela de la Universidad de Alicante, está formado por 7 *PC's* conectados mediante un switch con un ancho de banda de hasta 100 Megabytes/sg.

Otros tipos de arquitecturas actuales son:

Silicon Graphics POWER CHALLENGE.

Son multiprocesadores de memoria compartida que tienen procesadores RISC superescalares R8000 y R10000. Generalmente están disponibles desde 2 a 8 nodos como el POWER CHALLENGE L de Silicon Graphics, hasta 288 procesadores como el POWER CHALLENGE array.

CRAY T3E.

Son multiprocesadores de memoria compartida. Los microprocesadores son DC Alpha EV5 con un rendimiento pico de 600 Mflops. Cada procesador tiene su propia memoria local con una capacidad entre 64 Mbytes y 2Gbytes. Están construidos sobre la primera generación de máquinas de CRAY.

CONVEX Exemplar SPP1000.

Es un sistema que consta de un conjunto de hipernodos (de 1 a 16) cada uno de los cuales contiene de 4 a 8 procesadores HP PA-RISC 7100 y una memoria local de 256 Mbytes hasta 2Gbytes. Dentro de cada hipernodo las unidades

funcionales acceden a memoria a través de una red "crossbar", que permite la conexión directa entre todos los nodos. La conexión entre hipernodos se realiza mediante una colección de anillos. Los procesadores se organizan por pares en bloques funcionales.

2.4 Modelos de programación paralela

La construcción de una plataforma hardware para el procesamiento paralelo, debe llevar de forma adicional la construcción de un procesamiento software que soporte dicha arquitectura. Una gran número de sistemas software han sido diseñados para la programación de los computadores paralelos, tales como los sistemas operativos y lenguajes de programación.

Los modelos de programación paralela están específicamente diseñados para sistemas multiprocesadores, multicomputadores o computadores vectoriales SIMD y deben estar provistos de mecanismos de partición del problema principal en tareas separadas para adjudicarlas a cada procesador. Estos mecanismos pueden provenir del *paralelismo implícito*, que divide el problema y adjudica las tareas a cada procesador automáticamente, o del *paralelismo explícito* donde el programador tiene que decidir qué partes deben ser ejecutadas como tareas paralelas independientes.

El paralelismo depende de cómo está implementada la comunicación entre procesadores. Existen cinco modelos que explotan el paralelismo con diferentes paradigmas de ejecución que son: el modelo de *variable compartida*, *paso de*

mensaje, data-parallel, programación orientada a objetos y los modelos *funcionales y lógicos.* Exponemos brevemente cada uno de ellos.

- **Modelos de variable compartida (shared).**

La programación de multiprocesadores está basada en el uso de variables divididas en una memoria común para la comunicación entre procesos. Las variables exigen el uso de memoria compartida y exclusión mutua entre múltiples procesos accediendo al mismo conjunto de variables. Las ventajas de usar este modelo son: atomicidad en las operaciones de memoria, memoria consistente, acceso protegido a las secciones, rápida sincronización, estructuras de datos compartidas y rápido movimiento de datos.

- **Modelos de paso de mensajes.**

Dos procesos que residen en diferentes procesadores pueden comunicarse entre sí mediante el paso de mensajes por una red directa. Los mensajes pueden ser instrucciones, datos, señales, ... Esta comunicación tarda más que la de acceso mediante variables compartidas en memoria común. El paso de mensajes se puede hacer: de forma *síncrona* (sincroniza que los procesos se envíen y reciban en el mismo espacio de tiempo, por lo que no se necesitan buffers) y de forma *asíncrona* (no están sincronizados en el tiempo, por eso necesitan buffers).

- **Modelos data-parallel.**

Son modelos que son fáciles de escribir y de depurar porque el paralelismo es tratado explícitamente por una sincronización en el hardware y por control de flujo de datos. Es aplicado a problemas de grano fino usando mallas regulares,

plantillas (matrices) y conjuntos de datos de señal/imagen. Puede ser implementado en computadores SIMD. Las extensiones que se realizan para datos arrays están representadas por tipos de datos de alto-nivel. Ejemplos de lenguajes que procesan arrays son: el CFD para el Illiac IV; el DAP Fortran para el AMT/Procesador de arrays distribuido; C^* para el TMC/Connection machine y el MPF para la familia de los computadores masivamente paralelos MasPar. Los compiladores soportan las familias de Fortran-77, Fortran-90 y C.

- **Modelos orientados a objetos.**

Los objetos se crean dinámicamente. El proceso se realiza enviando y recibiendo mensajes entre los objetos. El paralelismo se realiza mediante: *concurrencia pipeline* que supone la enumeración de soluciones sucesivas y testea concurrentemente las soluciones que surgen de una evaluación pipeline; *divide-y-vencerás:* supone la evaluación concurrente de diferentes subprogramas y la combinación de sus soluciones es la solución final; *resolución de un problema cooperativo:* todos los objetos interactúan entre sí, los resultados intermedios se almacenan en objetos y memoria compartida mediante paso de mensajes entre ellos.

- **Modelos funcionales y lógicos.**

En los modelos funcionales no se producen efectos al margen de la ejecución. Enfatiza la programación de grano-fino de los MIMD. La mayor parte de los computadores paralelos diseñados para soportar el modelo funcional están orientados al lenguaje *Lisp*. Los modelos de programación lógica están basados en la lógica de predicados para procesamiento de conocimiento mediante grandes bases de datos. Dos lenguajes de programación lógica paralela son el

Prolog Concurrente diseñado por Shapiro [92], y el *Parlog* diseñado por Clark y Gregory [27].

2.5 Evaluación de las prestaciones de un algoritmo paralelo

En esta sección vamos a definir aquellos parámetros que nos van a ser de utilidad para evaluar las prestaciones de los algoritmos en paralelo. Estos son el *incremento de velocidad* (speed-up) y la *eficiencia*. Si suponemos que disponemos de r procesadores definimos,

Definición 2.1. Se llama *incremento de velocidad* (speed-up) de un algoritmo paralelo a

$$S_r = \frac{\text{Tiempo de ejecución en un sólo procesador}}{\text{Tiempo de ejecución en } r \text{ procesadores}}. \tag{2.1}$$

Esta definición compara el mismo algoritmo utilizando uno o r procesadores, sin embargo, esto tiene un inconveniente, ya que dicho algoritmo paralelo no tiene porqué ser más eficiente que cuando se ejecuta en secuencial. Esto nos lleva a la introducción de una nueva medida de paralelismo, que se da en la definición siguiente.

Definición 2.2. El *incremento de velocidad* (speed-up) de un algoritmo paralelo respecto al mejor algoritmo secuencial es

$$S_r^* = \frac{\text{Tiempo de ejecución en un sólo procesador del algoritmo secuencial más rápido}}{\text{Tiempo de ejecución en } r \text{ procesadores del algoritmo paralelo}}. \tag{2.2}$$

Hacemos notar, que no siempre se va a poder determinar el mejor algoritmo secuencial en términos absolutos. Por otro lado, aunque teóricamente se verifica $S_r^* \leq S_r \leq r$, en la práctica se pueden encontrar situaciones en las que los incrementos de velocidad sean superiores al número de procesadores. Así, por ejemplo, podemos encontrarnos con esta situación cuando utilicemos como algoritmo secuencial un algoritmo distinto al paralelo o cuando el algoritmo secuencial necesite acceder más veces a la memoria rápida (caché), lo cual provocará un retraso en la ejecución secuencial y consecuentemente un aumento del speed-up.

Relacionadas con las medidas anteriores están las de eficiencia, que miden el grado de utilización de los procesadores del sistema al ejecutar en él un algoritmo. Para un sistema con r procesadores damos la siguiente definición.

Definición 2.3. Se llama *eficiencia* de un algoritmo paralelo respecto a sí mismo a

$$E_r = \frac{S_r}{r},$$

siendo la eficiencia respeto al mejor algoritmo secuencial,

$$E_r^* = \frac{S_r^*}{r}.$$

Hacemos notar, que para obtener los parámetros definidos en esta sección es necesario realizar el experimento numérico, pues estamos hablando de tiempos reales de ejecución. Por otra parte, no siempre va a ser posible determinar el mejor algoritmo secuencial en términos absolutos.

Concluimos esta sección diciendo que para maximizar la eficiencia de un algoritmo paralelo, hay que tener muy en cuenta el *equilibrio de la carga*, es

decir, la distribución de las tareas entre los distintos procesadores de forma que todos ellos tengan una cantidad de trabajo similar, de hecho, debe procurarse que este reparto equitativo se realice entre todos los puntos de sincronización a fin de evitar que algunos procesadores se mantengan inactivos o *perezosos*. Esto supone que el problema de asignación de tareas a los distintos procesadores sea clave a la hora de mejorar la eficiencia. Esta asignación en los multiprocesadores de memoria distribuida suele ser estática, es decir, que se ha de hacer antes de comenzar con la ejecución del algoritmo, sin embargo, en los de memoria compartida suele ser dinámica, lo que permite que dependiendo de cómo vaya desarrollándose la ejecución, se asignen las tareas a los distintos procesadores.

2.6 PVM

PVM (Parallel Virtual Machine) es un paquete de software que permite a una colección de computadoras heterogéneas (secuenciales, vectoriales o paralelas) conectarse sobre una red cualquiera (Ethernet, fibra óptica, ...) para aparecer como un único recurso computacional concurrente de memoria distribuida. La librería de rutinas está compuesta de primitivas que pueden ser llamadas desde *C* o desde *Fortran*. Estas permiten un funcionamiento completo entre las distintas tareas de una aplicación.

Las computadoras a las que va dirigido este recurso son los multiprocesadores de memoria compartida o distribuida, las supercomputadoras vectoriales y las estaciones de trabajo. En lo siguiente, usaremos los conceptos de *máquina virtual*

para designar el computador distribuido que obtenemos con PVM; *Host* es el término que utilizaremos para denotar cualquiera de los computadores que son usados para crear la maquina virtual, y *tarea* es un término definido como una unidad computacional en PVM, análogamente a un proceso UNIX.

PVM está compuesto de un sistema que es la *librería de rutinas* y de un *daemon*, llamado pvmd3 (pvmd).

El *daemon* pvmd3 reside en todas las computadoras que constituyen la máquina virtual. Cuando un usuario desea comenzar una sesión de PVM, crea primero una máquina virtual poniendo en marcha el PVM. La aplicación PVM puede empezarse entonces desde un *promt* Unix en cualquiera de los *host*. Varios usuarios pueden configurar máquinas virtuales simultáneamente y cada usuario puede ejecutar varias aplicaciones PVM simultáneamente.

PVM es especialmente útil para aplicaciones compuestas de varias partes relacionadas entre sí y con grandes cálculos cada una de ellas. PVM ha sido usado para una gran número de aplicaciones tales como simulación de dinámica molecular, estudios de superconductividad, algoritmos matriciales y también como base para la enseñanza de la programación concurrente o paralela.

Capítulo 3

Métodos iterativos paralelos. Planteamiento y objetivos.

3.1 Introducción

Dado un sistema lineal de ecuaciones

$$Ax = b, \tag{3.1}$$

podemos realizar una partición $A = F - G$, de tal foma que a partir de ella se puede escribir el siguiente esquema iterativo

$$x^{(l+1)} = F^{-1}Gx^{(l)} + F^{-1}b, \quad l = 0, 1, 2, \ldots .$$

Como ya hemos visto en el Capítulo 1, para resolver el sistema (3.1) se pueden utilizar métodos iterativos secuenciales. Pero debido al empuje de las

nuevas tecnologías, la computación paralela, hace posible que estos métodos se puedan resolver e implementar, sin ninguna dificultad, en un multiprocesador como podemos ver en Bertsekas y Tsitsiklis [11].

Para ello, se distribuye la carga de trabajo mediante la asignación de incógnitas entre los distintos procesadores. Esta asignación se puede realizar de cuatro formas que son: *la asignación directa, la reordenación de incógnitas, descomposición en bloques y mediante multiparticiones.* Veamos brevemente en qué consiste cada una de ellas.

La *asignación directa* de incógnitas se puede realizar cuando el método iterativo permite que los procesadores trabajen independientemente, como ocurre por ejemplo, con el método de Jacobi que cuando se ejecuta en paralelo cada procesador actualiza sus incógnitas y después, también en paralelo, envía las variables necesarias a cada procesador. En este método el grado de paralelismo es perfecto, por lo que si no se tuviese en cuenta el tiempo de comunicación, se alcanzaría un *speed-up* teórico igual al número de procesadores.

La *reordenación de incógnitas* permite distribuir la carga entre procesadores de forma independiente. Sin embargo, la distribución de un grupo de incógnitas a cada procesador induce una idea más general, que consiste en dividir la matriz A en bloques y asignar éstos a los distintos procesadores. Es decir, si la matriz A está dividida en bloques cuadrados, invertibles, de tamaño $n_i \times n_i$, de la forma

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1r} \\ A_{21} & A_{22} & \cdots & A_{2r} \\ \vdots & \vdots & & \vdots \\ A_{r1} & A_{r2} & \cdots & A_{rr} \end{bmatrix}, \tag{3.2}$$

se pueden construir los subsistemas siguientes que se asignan cada uno a un

procesador diferente

$$A_{ii}x_i^{(l)} = -\sum_{j=1,j\neq i}^{r} A_{ij}x_j^{(l-1)} + b_i, \quad i = 1,2,\ldots,r, \quad l = 1,2,\ldots, \tag{3.3}$$

donde x_i y b_i son bloques de los vectores x y b respectivamente, y tienen un tamaño de acuerdo al tamaño de los correspondientes bloques de A. La ecuación (3.3) corresponde al esquema iterativo que se conoce como método de Jacobi por bloques (podemos ver un amplio estudio de métodos iterativos por bloques, en Bertsekas y Tsitsiklis [11] o en Young [105]). Sucede que cuando la matriz de coeficientes A es una M-matriz, los métodos por bloques son al menos tan rápidos como los métodos por filas, por lo tanto, a la hora de realizar los subsistemas (3.3) será importante no sólo la elección del método utilizado, sino la elección de los bloques de la matriz A.

Para ver un ejemplo en el que se realizan técnicas de reordenación y descomposición en bloques, podemos dirigirnos a White [102], en el que se da un ejemplo numérico basado en la ecuación de Poisson discretizada mediante diferencias finitas de segundo orden. Por último, tenemos que decir, que la técnica de descomposición por bloques se puede ver como un caso particular de la técnica de multipartición dada por O'Leary y White [77], la cual se expone con mayor detalle en la siguiente sección.

Atendiendo al contenido de las secciones que aparecen en este capítulo, damos una breve descripción de cada una de ellas. En primer lugar, en la Sección 3.2 describiremos la técnica de multipartición dada por O'Leary y White, así como algunos resultados de convergencia de dicho método. En la Sección 3.3, se describen los métodos de multipartición no estacionarios, en los que veremos que cada procesador puede actualizar más de una vez su solución en cada ite-

ración global. En la Sección 3.4, se plantean dos precondicionadores paralelos diseñados para trabajar de forma eficiente en un multiprocesador en paralelo, estos son los precondicionadores polinomiales aditivos y los precondicionadores por bloques.

Por último, en la Sección 3.5 exponemos los motivos que nos han llevado a desarrollar los resultados que aparecen en esta memoria, y describimos los objetivos que se pretenden alcanzar en su desarrollo.

3.2 Métodos iterativos de multipartición

Como ya se ha visto en el Capítulo 1, si tenemos un sistema de ecuaciones $Ax = b$, al realizar una partición de la matriz de coeficientes, por ejemplo, $A = F - G$, el esquema iterativo que resulta de considerar esta partición es

$$x^{(l+1)} = F^{-1}Gx^{(l)} + F^{-1}b, \quad l = 0, 1, 2, \ldots .$$

Claramente, si se realizan distintas particiones de la matriz A, tendremos distintos esquemas iterativos. Bajo estas condiciones, O'Leary y White [77], en 1985, diseñaron la técnica de multipartición creada para la resolución de sistemas de ecuaciones lineales sobre un multiprocesador en paralelo.

El método de multipartición [77], consiste en considerar una colección de particiones

$$A = F_j - G_j, \quad 1 \leq j \leq r, \quad det(F_j) \neq 0, \tag{3.4}$$

y unas matrices de peso $E_j, 1 \leq j \leq r$, diagonales no negativas cuya suma es la identidad. A la colección $\{F_j, G_j, E_j\}_{j=1}^{r}$ la llamaremos multipartición de A. A partir de estas hipótesis, se considera el siguiente algoritmo.

Algoritmo 3.1. Algoritmo de multipartición.

Dado un vector inicial $x^{(0)}$.

Desde $l = 0, 1, 2, \ldots$, hasta convergencia

Desde $j = 1$ hasta r

$$F_j y_j = G_j x^{(l)} + b \tag{3.5}$$

$$x^{(l+1)} = \sum_{j=1}^{r} E_j y_j.$$

La idea clave aquí, es que se supone que se dispone de r procesadores conectados entre sí de tal forma que cada uno de ellos puede ejecutar una sucesión diferente de instrucciones sobre sus datos locales. En cada iteración, el procesador j-ésimo calcula solamente las componentes del vector

$$F_j^{-1} G_j x^{(l)} + F_j^{-1} b,$$

correspondientes a los elementos no nulos de E_j. El procesador pondera estas componentes para que sea capaz de distribuir el vector

$$E_j F_j^{-1} G_j x^{(l)} + E_j F_j^{-1} b$$

a uno de los procesadores, que llamaremos *central*, donde se suman las soluciones de cada uno de los esquemas iterativos asignados a los distintos procesadores, para así actualizar el vector x en la iteración l, es decir para obtener el vector $x^{(l+1)}$.

Hay que observar que el procesador j-ésimo no actualiza una componente que coincida con un elemento nulo en la diagonal de E_j. Es decir, si las matrices E_j son tales que una de ellas tiene el i-ésimo elemento de la diagonal no nulo y todas las demás E_h, $h \neq j$, tienen su elemento diagonal i-ésimo nulo, se puede asignar la fila i de A sólo al procesador j-ésimo. Si este razonamiento se piensa con bloques de filas, sólo se asignaría el correspondiente bloque a un único procesador. De esta forma, con una única partición se puede construir una multipartición y repartir el trabajo entre procesadores, mediante el cálculo de bloques de componentes del vector $x^{(l)}$.

Si denotamos por $T = \sum_{j=1}^{r} E_j F_j^{-1} G_j$, entonces la sucesión de vectores generada por el Algoritmo 3.1 se puede expresar de la siguiente forma

$$x^{(l+1)} = Tx^{(l)} + \sum_{j=1}^{r} E_j F_j^{-1} b, \qquad l = 0, 1, 2, \ldots , \tag{3.6}$$

donde T es la matriz de iteración.

O'Leary y White en [77], investigan también clases de matrices A y tipos de multiparticiones que dan lugar a que el Algoritmo 3.1 converja, demostrando los siguientes resultados.

Teorema 3.1. *(Teorema 1.a y 1.c, [77]).*
a) Sea A una matriz monótona y sea $\{F_j, G_j, E_j\}_{j=1}^{r}$ una multipartición de A, tal que todas las particiones $A = F_j - G_j$ son débilmente regulares, entonces el esquema iterativo (3.6) es convergente.
b) Sea $\{F_j, G_j, E_j\}_{j=1}^{r}$ una multipartición de la matriz A. Si se cumple que $\|F_j^{-1} G_j\|_\infty < 1$, para todo $j = 1, 2, \ldots , r$, entonces el esquema (3.6) es convergente.

En el mismo trabajo dan condiciones sobre las particiones (3.4) y sobre las matrices de peso E_j que aseguran la convergencia del Algoritmo 3.1 cuando la matriz A del sistema es simétrica y definida positiva y las particiones son P-regulares (ver Definición 1.13, Sección 1.3). Para ello, demuestran el siguiente teorema, que nosotros redactamos en el contexto de las matrices hermíticas.

Teorema 3.2. *(Teorema 1.b, [77]). Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares de una matriz hermítica y definida positiva, y sean las matrices de peso E_j de la forma $E_j = \alpha_j I$, $1 \leq j \leq r$, con $\alpha_j > 0$ y $\sum_{j=1}^{r} \alpha_j = 1$, entonces el algoritmo de multipartición (Algoritmo 3.1) es convergente.*

A partir del Teorema 3.2 se siguen los siguientes corolarios, que pueden verse por ejemplo, en [74].

Corolario 3.1. *(Teorema 3.2, [74]). Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares de una matriz hermítica y definida positiva y sean las matrices de peso E_j de la forma $E_j = \alpha_j I$, $1 \leq j \leq r$, con $\alpha_j > 0$ y $\sum_{j=1}^{r} \alpha_j = 1$. Sea F la matriz definida de la forma $F = \sum_{j=1}^{r} E_j F_j^{-1}$ y $T = \sum_{j=1}^{r} E_j F_j^{-1} G_j$, entonces la partición $A = F^{-1} - F^{-1}T$, $l = 0, 1, 2, \ldots$, es P-regular.*

Corolario 3.2. *(Teorema 3.3, [74]). Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones de una matriz hermítica y definida positiva donde las matrices G_j son definidas positivas, y sean las matrices de peso de la forma $E_j = \alpha_j I$. Sean las matrices $T = \sum_{j=1}^{r} E_j F_j^{-1} G_j$ y $F = \sum_{j=1}^{r} E_j F_j^{-1}$, entonces,*

1.- La parte real de los valores propios de T es positiva.

2.- *Si las matrices G_j son hermíticas y definidas positivas, entonces las matrices F^{-1} y $F^{-1}T$ son hermíticas y definidas positivas y todos los valores propios de T son reales y positivos.*

Queremos puntualizar, que si las matrices de peso E_j no son de la forma $E_j = \alpha_j I$, $1 \leq j \leq r$, entonces el Algoritmo 3.1 puede no converger cuando la matriz A es hermítica y definida positiva, incluso aunque las particiones de A sean P-regulares. Para ilustrar esta situación, damos un ejemplo diferente del expuesto por O'Leary y White en [77]. Más tarde, en la Sección 4.2, usaremos este ejemplo para el método de multipartición no estacionario.

Ejemplo 3.1. *Consideramos la matriz hermítica*

$$A = \begin{bmatrix} 0,75 & 0 \\ 0 & 0,75 \end{bmatrix}.$$

Sean $A = F_1 - G_1 = F_2 - G_2$ particiones de A, donde

$$F_1 = \begin{bmatrix} 0,3934 & -2,0660 \\ 2,0660 & 7,6244 \end{bmatrix}, \quad G_1 = \begin{bmatrix} -0,3566 & -2,0660 \\ 2,0660 & 6,8744 \end{bmatrix},$$

$$F_2 = \begin{bmatrix} 7,6244 & 2,0660 \\ -2,0660 & 0,3934 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 6,8744 & 2,0660 \\ -2,0660 & -0,3566 \end{bmatrix}.$$

Dichas particiones son P-regulares, pues las matrices

$$F_1^H + G_1 = \begin{bmatrix} 0,0368 & 0 \\ 0 & 14,498 \end{bmatrix}$$

y

$$F_2^H + G_2 = \begin{bmatrix} 14,498 & 0 \\ 0 & 0,0368 \end{bmatrix},$$

son definidas positivas. Sean las matrices de peso

$$E_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \; E_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

Calculando la matriz de iteración, obtenemos

$$T = E_1 F_1^{-1} G_1 + E_2 F_2^{-1} G_2 = \begin{bmatrix} 0,9594 & 0,2132 \\ 0,2132 & 0,9594 \end{bmatrix}.$$

El radio espectral de T es igual a $1,1726$, que es mayor que 1, por lo que el Algoritmo 3.1 no es convergente. Notamos que las matrices de peso E_1 y E_2 no satisfacen el Teorema 3.2. Sin embargo, si se eligen otras matrices de peso E_1 y E_2, que al igual que las anteriores tampoco satisfacen las condiciones del mencionado teorema, puede ocurrir que el Algoritmo 3.1 converja. Por ejemplo, si las matrices E_1 y E_2 son

$$E_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, E_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$$

la matriz de iteración sería

$$T = E_1 F_1^{-1} G_1 + E_2 F_2^{-1} G_2 = \begin{bmatrix} 0,2132 & -0,2132 \\ -0,2132 & 0,2132 \end{bmatrix},$$

cuyo radio espectral es $0,4264$, que es menor que 1, por lo que el Algoritmo 3.1 sería convergente.

La técnica de multipartición descrita en esta sección ha sido ampliamente estudiada en los últimos años. Así en 1987, White en [100], amplía los resultados obtenidos por O'Leary y él mismo en [77]. De este artículo cabe destacar los resultados numéricos obtenidos de la implementación de este método sobre un multiprocesador de memoria compartida; la conclusión más visible es que disminuye el número de iteraciones cuando aumenta el solapamiento pero sin embargo, el tiempo no disminuye sensiblemente.

Por otro lado, en 1987, Neumann y Plemmons (ver [75]), estudian con detalle la convergencia del método de multipartición cuando la matriz del sistema es monótona ($A^{-1} \geq O$), dándole especial atención al caso en que las particiones están basadas en las clásicas particiones de Jacobi y Gauss-Seidel. Suponen, sin pérdida de generalidad, que la matriz de coeficientes A se puede escribir como

$$A = I - L - U,$$

donde L y U son matrices estrictamente triangular inferior y superior respectivamente. Entonces construyen las siguientes matrices de iteración de los métodos de

$$\text{Jacobi} \quad \mathcal{J} = L + U, \quad \text{y} \tag{3.7}$$

$$\text{Gauss-Seidel} \quad \mathcal{L} = (I - L)^{-1}U. \tag{3.8}$$

Sean L_j y U_j, matrices que satisfacen las condiciones

$$O \leq L_j \leq L, \quad O \leq U_j \leq U, \quad j = 1, 2, \ldots, r,$$

entonces se pueden construir las multiparticiones siguientes

$$\text{Jacobi} \quad \{(I - L_j), (L + U - L_j), E_j\}_{j=1}^{r}, \tag{3.9}$$

$$\text{Gauss-Seidel} \quad \{(I - L_j - U_j), (L + U - L_j - U_j), E_j\}_{j=1}^{r}. \tag{3.10}$$

Con estas multiparticiones concluyen, que si la matriz del sistema es una M-matriz irreducible, los métodos construidos convergen más rápido que el método de Jacobi clásico.

Otros resultados de comparación de radios de convergencia con multiparticiones los dan Climent y Perea en [29], para lo que ellos llaman, multiparticiones débilmente no negativas del primer tipo ($M_j^{-1} \geq O$ y $M_j^{-1}N_j \geq O$, $1 \leq j \leq r$) y débilmente no negativas del segundo tipo ($M_j^{-1} \geq O$ y $N_j M_j^{-1} \geq O$, $1 \leq j \leq r$). Para multiparticiones débilmente no negativas, extienden los resultados dados por Elsner en [40] donde establece los límites superior e inferior del radio espectral del método de multipartición. Bajo esta idea, dan resultados de comparación para multiparticiones no negativas del primer tipo y particiones débilmente no negativas del segundo tipo. También dan resultados de comparación para multiparticiones P-regulares de matrices definidas positivas.

Nabben en [74], basándose en los resultados dados por Neumann y Plemmons [75] y Elsner [40], da también teoremas de comparación para multiparticiones de matrices monótonas, y simétricas y definidas positivas.

Por otra parte, es conocido que una forma de acelerar la convergencia de un método iterativo es usar un factor de relajación, como comentamos en la Sección 1.3. En [77], O'Leary y White construyen algunos ejemplos de multiparticiones de matrices convergentes, discutiendo su uso sobre computadores paralelos, uno de estos ejemplos corresponde a un método de multipartición relajado en el que se utiliza el parámetro de relajación ω, como en el método de Jacobi relajado. Este método está definido por la siguiente iteración

$$x^{(l+1)} = \omega \sum_{j=1}^{r} E_j F_j^{-1} \left(G_j x^{(l)} + b \right) + (1-\omega) x^{(l)}, \quad l = 0, 1, \ldots . \tag{3.11}$$

Claramente, el esquema iterativo (3.11) puede escribirse de la forma

$$x^{(l+1)} = Tx^{(l)} + c, \quad l = 0, 1, \ldots,$$

con

$$T = \omega \sum_{j=1}^{r} E_j F_j^{-1} G_j + (1 - \omega) I, \quad l = 0, 1, \ldots.$$

Frommer y Mayer en [43], estudian la convergencia del método de multipartición relajado cuando A es una H-matriz y el parámetro de relajación ω se encuentra en un determinado intervalo.

White, en [101], da resultados de convergencia para diferentes modelos de matrices de peso, en particular demuestra que los modelos de multipartición basados en matrices de peso llamadas "prewheighting" son en algunos casos mejores que los basados en las tradicionales matrices de peso llamadas "post-wheighting"; en [102], White, da resultados de convergencia, cuando la matriz A del sistema es simétrica y definida positiva y los bloques de la matriz provienen de una matriz que está en forma de disección.

Definición 3.1. Una matriz A está en forma de disección si, y sólo si, es de la siguiente forma

$$A = \begin{bmatrix} A_1 & & & \\ & \ddots & & C \\ & & A_K & \\ & C^T & & A_0 \end{bmatrix}, \tag{3.12}$$

donde A_k son matrices de tamaño $n_k \times n_k$, para $1 \leq k \leq K$, A_0 es una matriz de tamaño $m \times m$ con $n_1 + \cdots + n_k + m = n$, y C es una matriz de tamaño $(n - m) \times m$. Si A_0 tiene también forma de disección, entonces se dice que A tiene forma de disección anidada.

En [44], Frommer y Mayer estudian la convergencia de métodos iterativos basados en la técnica de multipartición, dando diferentes esquemas atendiendo a las matrices de peso y cuando la matriz A del sistema es una M-matriz; en particular muestran que para ciertas multiparticiones de Gauss-Seidel, el método no puede converger más rápidamente que el método iterativo clásico de Gauss-Seidel.

En 1991, Deren [99] presenta una clase de algoritmos relajados basados en la técnica de multipartición y el método AOR. Este tipo de algoritmo es una generalización del esquema iterativo (3.11). La convergencia se discute en el contexto de H-matrices.

En 1992 Szyld y Jones [95] estudiaron la relación existente entre los métodos iterativos en dos etapas (ver Sección 1.5) y los basados en la técnica de multipartición, analizando también los radios de convergencia.

3.3 Otros métodos iterativos basados en la técnica de multipartición

En las secciones anteriores de este capítulo se han introducido diferentes esquemas iterativos para la resolución de sistemas lineales en paralelo.

En general, los diferentes procesos iterativos estudiados resuelven sistemas o subsistemas del sistema de ecuaciones lineales original en cada procesador, actualizando en cada iteración global el vector solución. En estos casos, cada

procesador obtiene solamente una solución de su subsistema asociado.

Puesto que los diferentes procesadores de un multiprocesador MIMD son independientes, se puede pensar de forma natural en producir un esquema iterativo tal que cada procesador pueda actualizar más de una vez su solución en cada iteración global. En este caso, se obtiene una sucesión de vectores diferente de las obtenidas con los métodos de la Sección 3.2 y por lo tanto, se debe realizar el estudio teórico de la convergencia de estas nuevas sucesiones.

Denominaremos a esta clase de esquemas, *métodos no estacionarios*, en el sentido en que cada procesador actualiza su solución un número variable de veces, dependiendo de la iteración global que se está realizando. La iteración global se construye a partir de los vectores de cada uno de los procesadores. La idea de los modelos no estacionarios surge ante el problema de equilibrar la carga de trabajo entre los distintos procesadores. Así, si suponemos que se ha de resolver un sistema en el que hay una gran diferencia entre los tamaños de los distintos bloques de filas o entre la proporción de elementos no nulos que son asignados a distintos procesadores, podrá ocurrir que en un cierto instante algunos procesadores ya hayan actualizado su bloque de vector iterado, mientras que otros continúan trabajando. Parece entonces lógico, que para evitar esta situación los procesadores ociosos actualicen varias veces su bloque de vector. Esto lograría un equilibrio de la carga de trabajo, además de esperar una mejora en la velocidad de convergencia.

Dentro de los métodos no estacionarios distinguiremos dos posibles casos, los *métodos no estacionarios síncronos* y los *métodos no estacionarios asíncronos*. En el primero, cada iteración global se construye a partir de las soluciones de todos los procesadores. Claramente, se necesita una *sincronización* entre todos

los procesadores para construir el vector iterado en cada iteración global. En el caso *asíncrono*, cada solución global se construye a partir de la solución de algún procesador, concretamente el que acaba de actualizar su solución. En este caso, existe una gran libertad entre los procesadores, ya que no tienen que sincronizarse para construir la sucesión de vectores iterados.

En 1988, Bru, Elsner y Neumann, motivados por los trabajos de Ostrowski [83] en 1961 y de Chanzan y Miranker [26] en 1969, estudian en [15] dos modelos iterativos no estacionarios (uno síncrono y otro asíncrono), en paralelo, basados en la técnica de multipartición, para la resolución de grandes sistemas no singulares de ecuaciones lineales. Estos autores utilizan el término "caótico", para referenciar a los modelos no estacionarios, nosotros nos quedamos con este último término ya que, en la literatura clásica, caótico es sinónimo de asíncrono, como se puede ver en Chanzan y Miranker [26]. La convergencia de dichos modelos fue estudiada en [15] en el contexto de las matrices monótonas y particiones débilmente regulares.

Posteriormente Mas, Migallón, Penadés y Szyld [67], estudiaron la convergencia de dichos algoritmos para H-matrices. También Elsner, Koltracht y Neumann en [41] estudian la convergencia de la versión asíncrona cuando las matrices de iteración son paracontractivas, es decir, una matriz $P \in \mathbb{C}^{k,k}$, es paracontractiva cuando dada una norma vectorial $\|.\|$, se verifica que $Px \neq x \Longleftrightarrow \|Px\| < \|x\|$.

Además de los trabajos mencionados, varios han sido los autores que basándose en la técnica de multipartición han introducido y estudiado nuevos algoritmos paralelos, ver por ejemplo, [16], [18], [19], [22], [24], [43], [49] y [50].

Entre otros, tenemos que destacar el trabajo que realizaron en , en 1994, Bru,

Migallón Penadés y Szyld en [20] donde describieron un nuevo modelo iterativo que aúna la técnica de multipartición y el método en dos etapas definido en la Sección 1.5. Este método, denominado de multipartición en dos etapas, engloba a los métodos citados anteriormente, así como a los métodos en dos etapas (ver [18], [19] y [48]). El estudio de la convergencia de estos métodos, al igual que en el caso de los métodos no estacionarios de multipartición, se realizó para matrices monótonas y H-matrices.

Una gran parte de esta memoria va a dedicarse a establecer nuevos resultados de convergencia para los métodos citados en esta sección. Por tanto, y con el fin de facilitar la lectura, la descripción detallada de cada método se realizará en el capítulo correspondiente (ver Capítulos 4 y 5).

3.4 Precondicionadores paralelos

En esta sección veremos precondicionadores que han sido diseñados para que puedan ser implementados, de forma eficiente, en un multiprocesador en paralelo. En particular, destacaremos los *precondicionadores polinomiales aditivos.*

En líneas generales, estos precondicionadores se obtienen al considerar dos particiones de la matriz A, y promediando después los resultados obtenidos en cada iteración.

Para construir precondicionadores polinomiales aditivos de 1-paso, se consideran dos particiones de la matriz A

$$A = F_1 - G_1, \quad A = F_2 - G_2.$$

Después, sobre el sistema $As = \tau$ (ver Sección 1.8.1) se realiza una iteración de los esquemas iterativos

$$F_k s^{(l)} = G_k s^{(l-1)} + \tau, \quad k = 1, 2, \quad l = 0, 1, 2 \ldots, \tag{3.13}$$

partiendo de $s^{(0)} = 0$, obteniéndose

$$s_k^{(1)} = F_k^{-1}\tau, \quad k = 1, 2.$$

A continuación se promedian los resultados obteniendo el valor de $s^{(1)}$

$$s^{(1)} = \frac{1}{2}\left(s_1^{(1)} + s_2^{(1)}\right) = \frac{1}{2}\left(F_1^{-1} + F_2^{-1}\right)\tau, \tag{3.14}$$

y de aquí la matriz

$$\mathcal{M}_1^{-1} = \frac{1}{2}\left(F_1^{-1} + F_2^{-1}\right),$$

será el precondicionador polinomial aditivo de 1-paso. Para construir el precondicionador aditivo de 2-pasos se realiza otra iteración del esquema (3.13), partiendo ahora de $s^{(1)}$ y se promedian los resultados.

Si llamamos $R_k = F_k^{-1}G_k, \quad k = 1, 2$, se obtiene

$$\begin{aligned} s_1^{(2)} &= R_1 s_1^{(1)} + F_1^{-1}\tau, \\ s_2^{(2)} &= R_2 s_2^{(1)} + F_2^{-1}\tau, \end{aligned}$$

luego,

$$s^{(2)} = \frac{1}{2}\sum_{k=1}^{2} s_k^{(2)} = \frac{1}{2}\left[\left(R_1 + R_2\right)s^{(1)} + \left(F_1^{-1} + F_2^{-1}\right)\tau\right]. \tag{3.15}$$

Si denotamos por $H = \frac{1}{2}\left(R_1 + R_2\right)$ y sustituimos en (3.15) el valor obtenido en (3.14) se tiene

$$s^{(2)} = \frac{1}{2}\left(I + H\right)\left(F_1^{-1} + F_2^{-1}\right)\tau. \tag{3.16}$$

de donde obtenemos la matriz

$$\mathcal{M}_2^{-1} = \frac{1}{2}(I+H)\left(F_1^{-1}+F_2^{-1}\right),$$

que es el *precondicionador polinomial aditivo de 2 pasos.*

En general, para m-pasos se obtiene

$$s^{(m)} = \frac{1}{2}\left(I+H+H^2+\cdots+H^{m-1}\right)\left(F_1^{-1}+F_2^{-1}\right)\tau, \tag{3.17}$$

y por tanto, la matriz

$$\mathcal{M}_m^{-1} = \frac{1}{2}\left(I+H+H^2+\cdots+H^{m-1}\right)\left(F_1^{-1}+F_2^{-1}\right),$$

será el *precondicionador polinomial aditivo de m-pasos.*

Si parametrizamos este polinomio queda

$$\mathcal{M}_m^{-1} = \frac{1}{2}\left(\alpha_0 I+\alpha_1 H+\alpha_2 H^2+\cdots+\alpha_{m-1}H^{m-1}\right)\left(F_1^{-1}+F_2^{-1}\right). \tag{3.18}$$

Adams y Ong en [3] dan condiciones que demuestran la validez del precondicionador dado en (3.18), para cualquier valor de m. Consideran para esto dos particiones de la matriz A, de la forma $A = F_k - G_k$, $k = 1, 2$, y demuestran que si se verifican las condiciones de que las matrices F_k, $k = 1, 2$, sean definidas positivas, $F_1 = F_2^T$ y $\alpha_{m-1}H^{m-1} + \alpha_{m-2}H^{m-2} + \cdots + \alpha_1 H^1 + I$ tenga valores propios positivos, entonces el precondicionador dado por la expresión (3.18), es simétrico y definido positivo. Para el caso en que $\alpha_i = 1$ para todo i, la última condición equivale a que $\rho(H) < 1$, cuando se realiza un número par de iteraciones.

También dichas autoras, estudian precondicionadores aditivos construidos mediante la partición SOR y su traspuesta, a este precondicionador lo denominan precondicionador *SOR-aditivo.* Tomando como problema modelo la ecuación de Laplace sobre el cuadrado unidad con condiciones de frontera, llegan a la

conclusión de que el SOR-aditivo es dos veces más rápido que el método SSOR si éstos se usan como métodos iterativos, mientras que la efectividad de ambos es similar cuando se utilizan como precondicionadores. Estos son implementados en un multiprocesador de memoria distribuida y en otro de memoria compartida y las conclusiones que obtienen en ambas implementaciones son similares. Si se utiliza el mismo número de procesadores para ambos métodos, el SSOR es preferible al SOR-aditivo. Sin embargo, el SOR-aditivo con $2r$ procesadores, donde r procesadores están trabajando con una partición y los otros r están trabajando con la partición traspuesta, puede ser más rápido que el SSOR, hasta el punto en que el cálculo puede ser acelerado al menos en un factor de 2.

Siguiendo las ideas básicas de Adams y Ong, Corral en [32] (ver también [14]) construye un precondicionador aditivo para el método del gradiente conjugado usando, a diferencia del construido por Adams y Ong [3], no sólo dos particiones sino en general un número k de particiones arbitrarias de la matriz A de coeficientes del sistema lineal. Es decir, estos precondicionadores están basados en la técnica de multipartición. Corral en [32], también construye otro precondicionador basado en el anterior y la factorización incompleta de Cholesky, estableciendo algunas comparaciones entre ellos. En [32] se dan condiciones para que ambos precondicionadores sean simétricos y definidos positivos.

Otra técnica de precondicionamiento eficaz es la basada en precondicionadores por bloques, es decir, un precondicionador de la forma

$$\mathcal{M} = (L + B)B^{-1}(B + U),$$

en el que L y U son matrices triangulares inferior y superior, respectivamente, y B es una matriz diagonal por bloques, suponiendo que A es una matriz

tridiagonal por bloques. Este tipo de precondicionadores fue introducido independientemente por Concus, Golub y Meurant [30] y por Axelsson, Brinkkemper y Ill'n [6]. La mayoría de estos precondicionadores por bloques se diferencian en la elección de la matriz B. Se trata de precondicionadores totalmente efectivos en problemas bidimensionales donde la matriz A es tridiagonal por bloques, en los que reducen significativamente el número de iteraciones. Sin embargo, en problemas tridimensionales, la experiencia conduce a resultados menos favorables (ver, por ejemplo, [62]).

Se han propuesto varias extensiones y técnicas relacionadas con ésta. Por ejemplo, en [69] Meurant propone una variante del precondicionador por bloques introducido en [30], aproximando las inversas de los bloques diagonales de B por algunas matrices en banda apropiadas.

En [70], Meurant, motivado por la posibilidad de trabajar con multitareas en las máquinas Cray, sugiere una alternativa que consiste en realizar la factorización LU de A no sólo de arriba a abajo, sino también de abajo a arriba al mismo tiempo. La descomposición resultante es la llamada *Factorización Incompleta Entrelazada.*

Reordenando las incógnitas, por ejemplo, mediante reordenación $rojo - negro$ (ver [81]), se puede obtener una estructura de la matriz de forma que permita un paralelismo en los factores triangulares que representan la factorización incompleta. Esta ordenación tiene la dificultad de que a menudo puede conllevar un aumento en el número de iteraciones con respecto al orden estandar. Un estudio sobre el efecto que produce la reordenación sobre el método del gradiente conjugado puede verse en [37].

Si se escriben los factores triangulares de $\mathcal{M}$, es decir, L, en forma bidiagonal

por bloques

$$L = \begin{bmatrix} L_{1,1} & & & & & \\ L_{2,1} & L_{2,2} & & & & \\ & L_{3,2} & L_{3,3} & & & \\ & & & \cdots & & \\ & & & & L_{r,r-1} & L_{r,r} \end{bmatrix},$$

en [91], Seager propone hacer cero algunos bloques no diagonales de L, y por lo tanto, de forma simétrica también hacer cero algunos bloques no diagonales de U, siempre que r no sea demasiado pequeño. Entonces, el proceso de sustitución regresiva se descompone en un conjunto de procesos de sustitución regresiva independientes. La principal desventaja de esta técnica es que el número de iteraciones aumenta, sobre todo, cuando muchos de los bloques no diagonales son descartados.

Por otro lado, los autores Radicati y Robert en [85], proponen dividir la matriz en bloques a lo largo de la diagonal principal, que están solapados en un número pequeño de componentes, calculando en paralelo precondicionadores locales para cada uno de los bloques, es decir, haciendo una factorización incompleta LU a cada bloque por separado, y promediando después los resultados correspondientes a los elementos solapados en procesadores "vecinos". Decir que, un elemento no nulo de A no está necesariamente contenido en uno de estos bloques. Sin embargo, los experimentos muestran que esta técnica ofrece mejores resultados si los bloques cubren todos los elementos no nulos de A.

Una descripción detallada de distintas técnicas de precondicionamiento, puede verse en [90].

3.5 Motivación y objetivos de la memoria

La aparición de los multiprocesadores ha dado lugar no sólo a adaptar los algoritmos clásicos de la computación matricial para su implementación y ejecución en dichos multiprocesadores, sino que además sugiere y motiva la búsqueda de nuevos métodos iterativos que obtengan un buen rendimiento de la máquina.

Suponemos que disponemos de un sistema de ecuaciones lineales

$$Ax = b. \tag{3.19}$$

Como ya se ha dicho en el Capítulo 1, para resolver este sistema se pueden utilizar métodos iterativos secuenciales, pero apoyándonos en la computación paralela, consideraremos la resolución del sistema de ecuaciones lineales (3.19) utilizando métodos iterativos diseñados para su ejecución en paralelo.

En líneas generales, nuestra investigación se ha centrado en dos grandes conjuntos de métodos de resolución de sistemas de ecuaciones lineales:

- Los métodos de multipartición no estacionarios, introducidos por Bru, Elsner y Neumann en [15], y los métodos iterativos paralelos en dos etapas (ver por ejemplo, [18], [19], [20], [48] y [60]). Una de las propiedades de estos métodos, además de su perfecta adecuación al procesamiento paralelo, es que permiten equilibrar la carga de trabajo entre los distintos procesadores al mismo tiempo que mejoran la velocidad de convergencia.

- El uso de precondicionadores paralelos para el método del gradiente conjugado, los cuales están diseñados para la resolución de sistemas de ecuaciones lineales donde la matriz del sistema es grande y dispersa, además

de simétrica y definida positiva.

En el Capítulo 4, *Métodos de multipartición no estacionarios*, motivados por los estudios de convergencia realizados por un lado, por Bru, Elsner y Neumann en [15], para el caso en que la matriz A del sistema es monótona y por otro, por Mas, Migallón, Penadés y Szyld [67], que estudian la convergencia de este método, junto con su versión relajada, cuando la matriz A es una H-matriz, nos planteamos analizar la convergencia del método de multipartición no estacionario síncrono, junto con su versión relajada, cuando la matriz del sistema es hermítica y definida positiva. Además, si queremos reducir el tiempo de comunicación y sincronización entre los distintos procesadores que cooperan en un sistema multiprocesador, tenemos que definir un modelo asíncrono, por lo tanto, también definimos y estudiamos la convergencia del método no estacionario asíncrono junto con su versión relajada, bajo hipótesis similares a las del modelo síncrono.

Los métodos iterativos en dos etapas y el método de multipartición en dos etapas han sido ampliamente estudiados en el contexto de matrices monótonas y H-matrices, podemos citar a Bru, Migallón Penadés y Szyld en [20]. Nosotros, en el Capítulo 5, *Métodos paralelos en dos etapas*, centrándonos en los métodos iterativos tipo Jacobi por bloques, estudiamos la convergencia del método por bloques en dos etapas y la del método de multipartición en dos etapas, para el caso en que la matriz de coeficientes del sistema lineal es hermítica y definida positiva. Además estudiaremos el caso en que el número de iteraciones internas es bastante grande. Se sabe que un parámetro de relajación optimiza el método haciendo que sea más rápido, por ello para ambos métodos se ha considerado también el estudio de la convergencia de su versión relajada, bajo hipótesis si-

milares a la versión anterior pero teniendo en cuenta que se cumplan ciertas restricciones en los parámetros de relajación. Para finalizar este capítulo, se estudian resultados de comparación para el radio espectral de la matriz de iteración que resulta de considerar diferentes particiones; para ello supondremos, que el número de iteraciones internas que debe realizar cada bloque permanece fijo para cada iteración externa.

El método de gradiente conjugado es muy eficiente a la hora de resolver sistemas de ecuaciones lineales donde la matriz de coeficientes es grande y dispersa. Se sabe, que para acelerar la convergencia de este método se buscan matrices precondicionadoras que tengan la propiedad de ser simétricas y definidas positivas, utilizando para ello diversas técnicas (ver por ejemplo [13], [14], [23] y [25]). Entre todas ellas podemos citar el precondicionamiento mediante series truncadas de la matriz A y el de la factorización incompleta de Choleski de dicha matriz de coeficientes.

Desde que Hestenes y Stiefel [56] introdujeron en 1952 precondicionadores secuenciales para el método del gradiente conjugado se ha estudiado mucho sobre ellos, pero no tanto para su aplicación al paralelismo. Motivados por esta necesidad, construimos en el Capítulo 6 *Precondicionadores basados en los métodos en dos etapas*, precondicionadores paralelos mediante series truncadas basados en el método en dos etapas y precondicionadores basados en los métodos en dos etapas con particiones internas basadas en la descomposición incompleta de Choleski, para matrices de coeficientes del sistema divididas en bloques. En ambos casos nos interesa estudiar la validez de dichos precondicionadores.

Los esquemas teóricos presentados en este trabajo han sido implementados en paralelo para su ejecución en dos sistemas de multicomputadores, el IBM

RS/6000 SP y una red de Pentiums de la Universidad de Alicante.

En el Capítulo 7 *Experimentos numéricos*, estudiamos el comportamiento de estos métodos no sólo investigando su eficiencia desde un punto de vista experimental, sino comparándolos entre si.

Capítulo 4

Métodos de multipartición no estacionarios.

4.1 Introducción

En el Capítulo 3, se ha introducido la técnica de multipartición para la resolución de sistemas de ecuaciones lineales

$$Ax = b, \tag{4.1}$$

donde $A \in \mathbb{C}^{n\times n}$ y $x, b \in \mathbb{C}^n$.

Recordamos que dicha técnica consiste en considerar una colección de particiones $A = F_j - G_j$, $1 \leq j \leq r$, con F_j invertibles y unas matrices de peso $O \leq E_j \leq I$, tal que $\sum_{j=1}^{r} E_j = I$. Con estas hipótesis se consideraba el siguiente algoritmo.

Algoritmo 3.1. Algoritmo de multipartición.

Dado un vector inicial $x^{(0)}$.

Desde $l = 0, 1, 2, \ldots$, hasta convergencia

Desde $j = 1$ hasta r

$$F_j y_j = G_j x^{(l)} + b \tag{4.2}$$

$$x^{(l+1)} = \sum_{j=1}^{r} E_j y_j.$$

Como se puede apreciar, el Algoritmo 3.1 corresponde al siguiente esquema iterativo

$$x^{(l+1)} = \sum_{j=1}^{r} E_j P_j x^{(l)}, \qquad l = 0, 1, 2, \ldots, \tag{4.3}$$

donde los operadores $P_j : \mathbb{C}^n \to \mathbb{C}^n$, $1 \leq j \leq r$, se definen como

$$P_j x = F_j^{-1} G_j x + F_j^{-1} b. \tag{4.4}$$

Cuando un algoritmo de multipartición se ejecuta en un sistema multiprocesador, cada operador P_j definido en (4.4), suele representar la tarea asignada a cada uno de los r procesadores para obtener su aproximación local y_j en (4.2). Cada aproximación local, se actualiza exactamente una vez usando el mismo vector $x^{(l)}$ iterado previamente. Sin embargo, es posible actualizar esta aproximación local más de una vez usando diferentes vectores locales iterados anteriormente. Con ello, se conseguiría a priori, mejorar cada aproximación local, lo que supondría que disminuyera el número de iteraciones globales y por tanto el tiempo de comunicación. Además, como veremos más tarde, el número de veces que cada procesador actualiza su aproximación local, podrá variar de

un procesador a otro. Este hecho ayudará a equilibrar la carga asignada a cada procesador en el caso de que no lo estuviera. Esta forma de realizar las iteraciones es la que describe el método de multipartición no estacionario. Podemos decir entonces, que la principal idea de este método es que en la iteración l, cada procesador j, resuelve el sistema definido por sus operadores P_j, $q(l,j)$ veces, actualizando cada vez el término independiente y usando para ello el nuevo vector local calculado. El siguiente algoritmo describe el método de multipartición no estacionario.

Algoritmo 4.1. Algoritmo de multipartición no estacionario.

Dado un vector inicial $x^{(0)}$.

Desde $l = 0, 1, 2, \ldots$, hasta convergencia

En cada procesador j, $j = 1$ hasta r

$$y_j^{(0)} = x^{(l)}$$

Desde $k = 1$ hasta $q(l,j)$

$$F_j y_j^{(k)} = G_j y_j^{(k-1)} + b \qquad (4.5)$$

$$x^{(l+1)} = \sum_{j=1}^{r} E_j y_j^{(q(l,j))}. \qquad (4.6)$$

Este método fue introducido en 1988 por Bru, Elsner y Neumann (ver [15]), los cuales, como ya hemos dicho en el Capítulo 3, Sección 3.2, demostraron la convergencia de este algoritmo cuando la matriz A es monótona ($A^{-1} \geq O$) y las particiones $A = F_j - G_j$ son débilmente regulares ($F_j^{-1} \geq O$ y $F_j^{-1}G_j \geq O$).

En el Algoritmo 4.1, se puede introducir un parámetro de relajación $\omega \in \mathbb{R}$, $\omega \neq 0$, sustituyendo el cálculo del vector iterado $x^{(l+1)}$ en la expresión (4.6) por

la ecuación $x^{(l+1)} = \omega\sum_{j=1}^{r} E_j y^{(q(l,j))} + (1-\omega)x^{(l)}$, $l = 0, 1, 2, \ldots$. De esta forma obtenemos el siguiente algoritmo.

Algoritmo 4.2. Algoritmo de multipartición no estacionario relajado.

Dado un vector inicial $x^{(0)}$.

Desde $l = 0, 1, 2, \ldots$, hasta convergencia.

En cada procesador j, $j = 1$ hasta r

$$y_j^{(0)} = x^{(l)}$$

Desde $k = 1$ hasta $q(l, j)$

$$F_j y_j^{(k)} = G_j y_j^{(k-1)} + b \qquad (4.7)$$

$$x^{(l+1)} = \omega\sum_{j=1}^{r} E_j y_j^{(q(l,j))} + (1-\omega)x^{(l)}. \qquad (4.8)$$

Claramente si $\omega = 1$, el Algoritmo 4.2 se reduce al Algoritmo 4.1.

La convergencia de los Algoritmos 4.1 y 4.2 ha sido estudiada por los autores Mas, Migallón, Penadés y Szyld (ver [67]) cuando A es una H-matriz. Además, en [67] dichos autores dan resultados computacionales de estos algoritmos en un sistema multiprocesador de memoria compartida concluyendo que se consiguen mejores resultados para los algoritmos no estacionarios que para el estacionario correspondiente al Algoritmo 3.1.

En este capítulo, para el estudio de la convergencia de los Algoritmos 4.1 y 4.2, nos centraremos en el caso en que la matriz A del sistema sea hermítica y definida positiva. En la Sección 4.2 estudiaremos la convergencia del Algoritmo 4.1

junto con su versión relajada, dada por el Algoritmo 4.2. En la Sección 4.3 estudiaremos la versión asíncrona de dichos algoritmos, donde las soluciones de los sistemas (4.5) y (4.7), que proceden de cada procesador, no necesitan esperar a las soluciones de los otros procesadores para completar el vector iterado.

4.2 Estudio de la convergencia

Dado un vector inicial $x^{(0)}$, el Algoritmo 4.1 correspondiente al método de multipartición no estacionario produce la siguiente sucesión de vectores

$$x^{(l+1)} = \sum_{j=1}^{r} E_j P_j^{q(l,j)} x^{(l)}, \qquad l = 0, 1, 2, \ldots , \tag{4.9}$$

donde los operadores P_j, $j = 1, 2, \ldots , r$, están definidos en (4.4). Para analizar la convergencia del esquema iterativo (4.9), nos basamos en la forma en que se

han definido los operadores P_j, $1 \leq j \leq r$, y escribimos para $l = 0, 1, 2, \ldots$,

$$\begin{aligned}
x^{(l+1)} &= \sum_{j=1}^{r} E_j P_j^{q(l,j)} x^{(l)} \\
&= \sum_{j=1}^{r} E_j P_j^{q(l,j)-1} \left[F_j^{-1} G_j x^{(l)} + F_j^{-1} b \right] \\
&= \sum_{j=1}^{r} E_j P_j^{q(l,j)-2} \left[F_j^{-1} G_j \left(F_j^{-1} G_j x^{(l)} + F_j^{-1} b \right) + F_j^{-1} b \right] \\
&= \sum_{j=1}^{r} E_j P_j^{q(l,j)-2} \left[(F_j^{-1} G_j)^2 x^{(l)} + (F_j^{-1} G_j) F_j^{-1} b + F_j^{-1} b \right] \\
&= \sum_{j=1}^{r} E_j P_j^{q(l,j)-2} \left[(F_j^{-1} G_j)^2 x^{(l)} + (F_j^{-1} G_j + I) F_j^{-1} b \right] \\
&= \cdots \\
&= \left(\sum_{j=1}^{r} E_j (F_j^{-1} G_j)^{q(l,j)} \right) x^{(l)} + \sum_{j=1}^{r} E_j \left(\sum_{i=0}^{q(l,j)-1} (F_j^{-1} G_j)^i \right) F_j^{-1} b.
\end{aligned}$$

Luego de aquí se deduce que el esquema (4.9) se puede escribir como

$$x^{(l+1)} = T^{(l)} x^{(l)} + c^{(l)}, \qquad l = 0, 1, 2, \ldots, \tag{4.10}$$

donde

$$T^{(l)} = \sum_{j=1}^{r} E_j (F_j^{-1} G_j)^{q(l,j)}, \qquad l = 0, 1, 2, \ldots, \tag{4.11}$$

son las matrices de iteración, y

$$c^{(l)} = \sum_{j=1}^{r} E_j \left(\sum_{i=0}^{q(l,j)-1} (F_j^{-1} G_j)^i \right) F_j^{-1} b, \qquad l = 0, 1, 2, \ldots .$$

Sea ξ la solución exacta del sistema lineal (4.1). Entonces, teniendo en cuenta las particiones $A = F_j - G_j$, $1 \leq j \leq r$, y la definición de los operadores

P_j, $1 \leq j \leq r$, en la expresión (4.4), se tiene que dicha solución es un punto fijo de los operadores P_j, es decir, $P_j(\xi) = \xi$, $1 \leq j \leq r$. Por tanto, siguiendo un razonamiento análogo al realizado previamente para obtener (4.10), podemos escribir $\xi = T^{(l)}\xi + c^{(l)}$ y usando el análisis del error se concluye que $e^{(l+1)} = x^{(l+1)} - \xi = T^{(l)}(x^{(l)} - \xi) = T^{(l)}e^{(l)}$, donde $T^{(l)}$ son las matrices de iteración definidas en (4.11).

Por tanto, el vector error $e^{(l+1)}$ se puede escribir en función del vector error en la iteración inicial, denotado por $e^{(0)}$, de la siguiente forma

$$e^{(l+1)} = T^{(l)}e^{(l)} = \cdots = T^{(l)}T^{(l-1)}\cdots T^{(0)}e^{(0)}, \qquad l = 0, 1, 2, \ldots .$$

A partir de esta expresión, es fácil ver que la sucesión de vectores generada por la iteración (4.9) (o equivalente (4.10)), converge a la solución del sistema lineal (4.1) si, y sólo si, $\lim_{l\to\infty} T^{(l)}T^{(l-1)}\cdots T^{(0)} = O$.

El siguiente lema nos va a ser de gran utilidad para analizar las matrices de iteración $T^{(l)}$, $l = 0, 1, 2, \ldots$, definidas en (4.11) y obtener nuevos resultados de convergencia para los métodos de multipartición no estacionarios.

Lema 4.1. *Sea A una matriz hermítica y definida positiva. Consideremos $A = F - G$ una partición P-regular. Dado $s \geq 1$, existe una única partición $A = M - N$ tal que $(F^{-1}G)^s = M^{-1}N$. Además dicha partición es P-regular.*

Demostración. Como $A = F - G$ es una partición P-regular y la matriz A es hermítica y definida positiva, a partir del Teorema 1.8 (Sección 1.3), se obtiene $\rho(F^{-1}G) < 1$ y por tanto, dado que $s \geq 1$, se tiene $\rho((F^{-1}G)^s) < 1$. Utilizando ahora el Lema 1.1 (Sección 1.3), existe la inversa de la matriz $I - (F^{-1}G)^s$ y entonces el Lema 1.3 (Sección 1.3), nos asegura que existen y además son

únicas, un par de matrices M y N tal que $A = M - N$, con M no singular y $(F^{-1}G)^s = M^{-1}N$.

Para demostrar la P-regularidad de la partición $A = M - N$ procederemos por inducción sobre el entero positivo s. Claramente, para $s = 1$, el resultado se sigue de la unicidad. Supongamos ahora que el resultado es cierto para $s - 1$, es decir, suponemos que la única partición $A = P - Q$ que satisface $(F^{-1}G)^{s-1} = P^{-1}Q$ es P-regular.

Entonces, como las particiones $A = F - G = P - Q$ son particiones P-regulares de una matriz hermítica y definida positiva, por el Teorema 1.9 (Sección 1.3) se sigue que existe una partición P-regular $A = M - N$, tal que $(P^{-1}Q)(F^{-1}G) = M^{-1}N$ y como $M^{-1}N = (P^{-1}Q)(F^{-1}G) = (F^{-1}G)^{s-1}(F^{-1}G) = (F^{-1}G)^s$, queda completada la demostración del lema. ■

Teorema 4.1. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares y $E_j = \alpha_j I$, $1 \leq j \leq r$, con $\alpha_j > 0$ y $\sum_{j=1}^{r} \alpha_j = 1$, entonces, para cada l, existe una única partición $A = R^{(l)} - S^{(l)}$, inducida por cada matriz de iteración $T^{(l)} = \sum_{j=1}^{r} E_j (F_j^{-1}G_j)^{q(l,j)}$, $l = 0, 1, 2, \ldots$, tal que $T^{(l)} = (R^{(l)})^{-1}S^{(l)}$. Además, dicha partición es P-regular.*

Demostración. Como $A = F_j - G_j$, $1 \leq j \leq r$, son particiones P-regulares de una matriz hermítica y definida positiva, entonces del Lema 4.1, se sigue que para cada j y l existe una única partición P-regular $A = M_j^{(l)} - N_j^{(l)}$ tal que $(F_j^{-1}G_j)^{q(l,j)} = (M_j^{(l)})^{-1}N_j^{(l)}$.

Entonces, las matrices de iteración $T^{(l)}$, $l = 0, 1, \ldots$, se pueden escribir como

$$\begin{aligned} T^{(l)} &= \sum_{j=1}^{r} E_j (F_j^{-1} G_j)^{q(l,j)} \\ &= \sum_{j=1}^{r} E_j (M_j^{(l)})^{-1} N_j^{(l)}, \qquad l = 0, 1, 2, \ldots . \end{aligned} \tag{4.12}$$

Por tanto, para cada índice de iteración fijo l, $T^{(l)}$ es la matriz de iteración correspondiente al esquema iterativo definido por el algoritmo de multipartición (Algoritmo 3.1) tomando como particiones $A = M_j^{(l)} - N_j^{(l)}$, $1 \leq j \leq r$.

Como estas particiones son P-regulares, el Teorema 3.2 (Sección 3.2) asegura que, $\rho(T^{(l)}) < 1$. Además, como A es definida positiva, podemos asegurar que para cada l y j, cada matriz $M_j^{(l)}$ es definida positiva, luego también lo será cada matriz $(M_j^{(l)})^{-1}$. Por otro lado, $\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1} = \sum_{j=1}^{r} \alpha_j (M_j^{(l)})^{-1}$ pues $E_j = \alpha_j I$, $1 \leq j \leq r$, y como para cada j y l, la matriz $(M_j^{(l)})^{-1}$ es definida positiva y los escalares α_j, $1 \leq j \leq r$ son positivos, podemos asegurar que la matriz $\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1}$ es definida positiva y por lo tanto, no singular.

Consideremos las matrices $R^{(l)}$ y $S^{(l)}$ definidas de la siguiente forma

$$\begin{aligned} R^{(l)} &= \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1} \right)^{-1}, \\ S^{(l)} &= R^{(l)} T^{(l)}. \end{aligned}$$

Entonces,

$$\begin{aligned}
R^{(l)} - S^{(l)} &= \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1}\right)^{-1} - \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1}\right)^{-1} \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1} N_j^{(l)}\right) \\
&= \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1}\right)^{-1} \left[I - \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1} N_j^{(l)}\right)\right] \\
&= \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1}\right)^{-1} \left[\sum_{j=1}^{r} E_j \left(I - (M_j^{(l)})^{-1} N_j^{(l)}\right)\right] \\
&= \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1}\right)^{-1} \left[\sum_{j=1}^{r} E_j \left((M_j^{(l)})^{-1} (M_j^{(l)} - N_j^{(l)})\right)\right] \\
&= \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1}\right)^{-1} \left(\sum_{j=1}^{r} E_j (M_j^{(l)})^{-1}\right) A.
\end{aligned}$$

Por tanto,

$$A = R^{(l)} - S^{(l)}, \qquad l = 0, 1, 2, \ldots . \tag{4.13}$$

Además, por la propia definición de $S^{(l)}$ claramente se tiene que $T^{(l)} = (R^{(l)})^{-1} S^{(l)}$, $l = 0, 1, 2, \ldots$. Como $\rho(T^{(l)}) < 1$ del Lema 1.3 (Sección 1.3), se sigue que las particiones (4.13) son únicas; además, por el Corolario 3.1 (Sección 3.2), concluimos que dichas particiones son P-regulares, con lo cual queda demostrado el teorema. ■

Del teorema anterior se sigue la convergencia del Algoritmo 4.1, cuando el número de veces $q(l, j)$ que el procesador j-ésimo realiza su trabajo es fijo en cada iteración l, es decir, si $q(l, j) = q(j)$, $l = 0, 1, 2, \ldots, 1 \leq j \leq r$.

Teorema 4.2. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$ particiones P-regulares y $E_j = \alpha_j I$, $1 \leq j \leq r$, con $\alpha_j > 0$ y $\sum_{j=1}^{r} \alpha_j =$*

1. *Suponemos que* $q(l,j) = q(j) \geq 1,\ 1 \leq j \leq r,\ l = 0,1,2,\ldots,$ *entonces, el algoritmo de multipartición no estacionario (Algoritmo 4.1) converge a la solución del sistema lineal (4.1), para cualquier vector inicial* $x^{(0)}$.

Demostración. Si $q(l,j) = q(j) \geq 1,\ 1 \leq j \leq r,\ l = 0,1,2,\ldots,$ podemos escribir las matrices de iteración $T^{(l)}$, definidas en (4.11) de la forma $T^{(l)} = \sum_{j=1}^{r} E_j (F_j^{-1} G_j)^{q(l,j)} = \sum_{j=1}^{r} E_j (F_j^{-1} G_j)^{q(j)},\ l = 0,1,2,\ldots$. Luego en este caso hay una única matriz de iteración $T^{(l)} = T$, y por el Teorema 4.1 tenemos que $\rho(T) < 1$. Por tanto queda demostrada la convergencia. ■

Sin embargo, como ya comentamos en la Sección 1.3, el producto de matrices convergentes puede no tender a cero (ver Bru y Johnson [17] o Robert, Charnay y Musy [86]), y por lo tanto necesitamos de otras herramientas para demostrar la convergencia del Algoritmo 4.1 para cualquier sucesión de enteros $q(l,j) \geq 1$. Así, tenemos el siguiente resultado.

Teorema 4.3. *Sea* A *una matriz hermítica y definida positiva. Sean* $A = F_j - G_j$, *particiones* P*-regulares y* $E_j = \alpha_j I,\ 1 \leq j \leq r,$ *con* $\alpha_j > 0$ *y* $\sum_{j=1}^{r} \alpha_j = 1$. *Suponemos que* $q(l,j) \geq 1,\ 1 \leq j \leq r,\ l = 0,1,2,\ldots,$ *entonces, el algoritmo de multipartición no estacionario (Algoritmo 4.1) converge a la solución del sistema lineal (4.1), para cualquier vector inicial* $x^{(0)}$.

Demostración. Claramente la matriz $A - (F_j^{-1} G_j)^H A (F_j^{-1} G_j)$ es hermítica. Por otro lado, la matriz $A - (F_j^{-1} G_j)^H A (F_j^{-1} G_j)$ puede ser reescrita de la si-

guiente forma

$$\begin{aligned} A-(F_j^{-1}G_j)^H A(F_j^{-1}G_j) &= A-(I-F_j^{-1}A)^H A(I-F_j^{-1}A) \\ &= (F_j^{-1}A)^H A + AF_j^{-1}A - (F_j^{-1}A)^H A(F_j^{-1}A) \\ &= (F_j^{-1}A)^H(F_j+F_j^H-A)(F_j^{-1}A) \\ &= (F_j^{-1}A)^H(F_j^H+G_j)(F_j^{-1}A). \end{aligned}$$

Por ser las particiones $A = F_j - G_j$, P-regulares, se verifica que la matriz $F_j^H + G_j$ es definida positiva y como $F_j^{-1}A$ es no singular, podemos asegurar que la matriz $(F_j^{-1}A)^H(F_j^H + G_j)(F_j^{-1}A)$ es definida positiva. Por tanto, para todo $x \neq 0$, $x^H(A-(F_j^{-1}G_j)^H A(F_j^{-1}G_j))x > 0$. Entonces, si tomamos la norma vectorial $\|\cdot\|_A$ tenemos

$$\begin{aligned} \|F_j^{-1}G_j x\|_A^2 &= (F_j^{-1}G_j x)^H A(F_j^{-1}G_j x) \\ &= x^H(F_j^{-1}G_j)^H A(F_j^{-1}G_j)x < x^H Ax = \|x\|_A^2, \quad \text{para todo } x \neq 0. \end{aligned}$$

Si ahora consideramos la norma matricial inducida por dicha norma vectorial (ver Definición 1.7, Sección 1.2), obtenemos $\|F_j^{-1}G_j\|_A = \max_{x\neq 0}\frac{\|F_j^{-1}G_j x\|_A}{\|x\|_A} < \max_{x\neq 0}\frac{\|x\|_A}{\|x\|_A} = 1$, $1 \leq j \leq r$. De ahí se sigue $\|F_j^{-1}G_j\|_A < 1$, $1 \leq j \leq r$, y entonces existe una constante real $0 \leq \theta < 1$, tal que $\|F_j^{-1}G_j\|_A \leq \theta < 1$, $1 \leq j \leq r$. Por lo tanto, para todo $l = 0, 1, 2, \ldots$, tenemos

$$\begin{aligned} \|T^{(l)}\|_A &= \left\| \sum_{j=1}^{r} E_j \left(F_j^{-1}G_j\right)^{q(l,j)} \right\|_A \\ &\leq \sum_{j=1}^{r} \|E_j\|_A \left\| (F_j^{-1}G_j)^{q(l,j)} \right\|_A \\ &= \sum_{j=1}^{r} \alpha_j \left\| \left(F_j^{-1}G_j\right)^{q(l,j)} \right\|_A \leq \theta < 1. \end{aligned} \tag{4.14}$$

Entonces, por el Lema 1.2 (Sección 1.3), se sigue que $\lim_{l\to\infty} T^{(l)}T^{(l-1)}\cdots T^{(0)} = O$, y la demostración queda finalizada. ■

El Teorema 4.3 generaliza el Teorema 3.2 (Sección 3.2), para el caso no estacionario y además es una prueba alternativa al resultado de convergencia dado por O'Leary y White en [77] para el método de multipartición.

Hacemos notar, que en el Teorema 4.3 hemos considerado las mismas restricciones sobre las matrices de peso E_j que en el Teorema 3.2 (Sección 3.2). Este tipo de restricción desde el punto de vista práctico no es deseable, ya que cada procesador debería actualizar en cada iteración todo el vector. Cuando enunciamos el Teorema 3.2, mostrábamos un ejemplo, (ver Ejemplo 3.1, Sección 3.2), en el que se ponía de manifiesto la posibilidad de que no convergiera el Algoritmo 3.1 si las matrices de peso E_j no eran de la forma $E_j = \alpha_j I$, $1 \leq j \leq r$. Ahora vamos a usar dicho ejemplo para ilustrar lo que sucede con la convergencia del algoritmo de multipartición no estacionario (Algoritmo 4.1), si las matrices de peso E_j no cumplen las condiciones del Teorema 4.3.

Ejemplo 4.1. *Consideramos la matriz hermítica*

$$A = \begin{bmatrix} 0,75 & 0 \\ 0 & 0,75 \end{bmatrix}.$$

Sean $A = F_1 - G_1 = F_2 - G_2$ *particiones P-regulares de A, donde*

$$F_1 = \begin{bmatrix} 0,3934 & -2,0660 \\ 2,0660 & 7,6244 \end{bmatrix}, \quad G_1 = \begin{bmatrix} -0,3566 & -2,0660 \\ 2,0660 & 6,8744 \end{bmatrix},$$

$$F_2 = \begin{bmatrix} 7,6244 & 2,0660 \\ -2,0660 & 0,3934 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 6,8744 & 2,0660 \\ -2,0660 & -0,3566 \end{bmatrix}.$$

Sean las matrices de peso

$$E_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \; E_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

Si calculamos las matrices de iteración correspondientes al Algoritmo 4.1, con $q(l,1) = q(l,2) = 2$, *obtenemos para todo* $l = 0, 1, 2, \ldots,$

$$T^{(l)} = E_1(F_1^{-1}G_1)^2 + E_2(F_2^{-1}G_2)^2 = \begin{bmatrix} 0,87850 & 0,2500 \\ 0,2500 & 0,8750 \end{bmatrix}.$$

Esta matriz tiene un radio espectral igual a $1,1250$, *que es mayor que 1, por lo que el Algoritmo 4.1 no es convergente. De nuevo calculamos las matrices de iteración para* $q(l,1) = q(l,2) = 3$ *y obtenemos para todo* $l = 0, 1, 2, \ldots,$

$$T^{(l)} = E_1(F_1^{-1}G_1)^3 + E_2(F_2^{-1}G_2)^3 = \begin{bmatrix} 0,7862 & 0,2399 \\ 0,2399 & 0,7862 \end{bmatrix}.$$

En este caso el radio espectral es $1,0261$, *que es mayor que 1, por lo que igual que en el caso anterior se deduce que el Algoritmo 4.1 no es convergente. Sin embargo, si se elige* $q(l,1) = q(l,2) = 4$, $l = 0, 1, 2, \ldots,$ *las matrices de iteración tienen la forma*

$$T^{(l)} = E_1(F_1^{-1}G_1)^4 + E_2(F_2^{-1}G_2)^4 = \begin{bmatrix} 0,7031 & 0,2187 \\ 0,2187 & 0,7031 \end{bmatrix},$$

cuyo radio espectral es $0,9218$, *que es menor que 1, por lo que el Algoritmo 4.1 es convergente. De la misma forma, dicho algoritmo es convergente si elegimos como matrices de peso las siguientes*

$$E_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad E_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$$

y la matriz de iteración es

$$T = E_1(F_1^{-1}G_1)^4 + E_2(F_2^{-1}G_2)^4 = \begin{bmatrix} -0,0625 & -0,2187 \\ -0,2187 & -0,0625 \end{bmatrix},$$

cuyo radio espectral es $0,1562$ *que es menor que 1.*

Consecuentemente, este ejemplo nos motiva el estudio de la convergencia del Algoritmo 4.1 sin las hipótesis adicionales de $E_j = \alpha_j I$, $0 \leq \alpha_j \leq 1$, y cuando cada procesador realiza bastantes iteraciones locales (4.5).

Teorema 4.4. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares. Suponemos además que $\lim_{l\to\infty} q(l,j) = \infty$, $1 \leq j \leq r$, entonces, el algoritmo de multipartición no estacionario (Algoritmo 4.1), converge a la solución del sistema lineal (4.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Puesto que $A = F_j - G_j$, $1 \leq j \leq r$, son particiones P-regulares de una matriz hermítica y definida positiva, se tiene por el Teorema 1.8 (Sección 1.3), que $\rho(F_j^{-1}G_j) < 1$, $1 \leq j \leq r$. Luego $\lim_{i\to\infty}(F_j^{-1}G_j)^i = O$, $1 \leq j \leq r$. Entonces, dado un $\epsilon > 0$, existe un entero i_0 tal que dada una norma matricial $\|\cdot\|$, se verifica $\|(F_j^{-1}G_j)^i\| \leq \epsilon$, para todo $i \geq i_0$. En particular, elegimos la norma infinito. Entonces, puesto que $\lim_{l\to\infty} q(l,j) = \infty$, podemos afirmar que existe un l_0 tal que

$$\|(F_j^{-1}G_j)^{q(l,j)}\|_\infty \leq \epsilon, \quad \forall l \geq l_0, \quad 1 \leq j \leq r.$$

De ahí, para $l \geq l_0$,

$$\|T^{(l)}\|_\infty = \left\| \sum_{j=1}^{r} E_j (F_j^{-1}G_j)^{q(l,j)} \right\|_\infty \leq \max_{1\leq j\leq r} \left\| (F_j^{-1}G_j)^{q(l,j)} \right\|_\infty \leq \epsilon.$$

Si tomamos $\epsilon < 1$, por el Lema 1.2 (Sección 1.3), se sigue la convergencia del Algoritmo 4.1. ■

Queremos hacer notar que, recientemente, los autores Migallón, Penadés y Szyld han realizado un estudio en [73], probando resultados de convergencia de método de multipartición no estacionario sin la condición $E_j = \alpha_j I$. La exigencia principal en [73] es que se realicen un número suficientemente grande, pero finito, de iteraciones locales (ver Ejemplo 4.1).

En el Capítulo 1, ya se ha visto cuándo las particiones de Jacobi, Gauss-Seidel y SOR de una matriz hermítica y definida positiva son P-regulares (ver también, por ejemplo, [10] y [81]).

Otra clase de particiones P-regulares de una matrix hermítica y definida positiva la podemos encontrar en la única partición inducida por la matriz de iteración de un método iterativo alternado de la forma

$$\begin{aligned} x^{(l+\frac{1}{2})} &= M^{-1}Nx^{(l)} + M^{-1}b, \\ x^{(l+1)} &= P^{-1}Qx^{(l+\frac{1}{2})} + P^{-1}b, \quad l = 0, 1, 2, \ldots, \end{aligned}$$

donde $A = M - N = P - Q$ son particiones P-regulares (ver [9]).

Como ejemplo importante de este tipo de particiones podemos citar las del método SSOR. Además de este método se pueden considerar, por ejemplo, las iteraciones alternadas basadas en una partición de la forma $A = A_1 + A_2$, donde A_1, A_2 son definidas positivas y $A_2 = A_1^H$. En este caso, tomando $M = \beta I + A_1$, $N = \beta I - A_2$, $P = \beta I + A_2$ y $Q = \beta I - A_1$, con $\beta > 0$, puesto que $M^H + N = P^H + Q = 2\beta I$, las particiones $A = M - N = P - Q$ son claramente P-regulares.

Existen otros caminos para construir r particiones P-regulares de una matriz hermítica y definida positiva, como podemos ver por ejemplo en [77], donde se considera una matriz hermítica y definida positiva A, de tamaño $n \times n$, dividida de la forma, $A = \sum_{j=1}^{r} A_j$, donde cada matriz A_j, $1 \leq j \leq r$ es de menor rango. También se consideran matrices diagonales D_j, $1 \leq j \leq r$, de tal manera que las matrices F_j definidas de la forma $F_j = A_j + D_j$, $1 \leq j \leq r$, son matrices no singulares. Consideramos las particiones $A = F_j - G_j$, $1 \leq j \leq r$. Entonces si las matrices $2(A_j + D_j) - A$, $1 \leq j \leq r$, son definidas positivas, entonces las particiones $A = F_j - G_j$, $1 \leq j \leq r$, son P-regulares y entonces se puede aplicar el resultado de convergencia demostrado anteriormente.

Por otra parte, la demostración del Teorema 4.4 muestra que $\|F_j^{-1}G_j\|_\infty < 1$, $1 \leq j \leq r$, es una condición suficiente para la convergencia del Algoritmo 4.1 para cualquier matriz A, sin hipótesis adicionales sobre las matrices de peso, como se pone de manifiesto en el siguiente corolario.

Corolario 4.1. *Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones de la matriz $A \in \mathbb{C}^{n\times n}$ tal que $\|F_j^{-1}G_j\|_\infty < 1$, $1 \leq j \leq r$. Si $q(l,j) \geq 1$, $1 \leq j \leq r$, $l = 0, 1, 2, \ldots$, entonces, el algoritmo de multipartición no estacionario (Algoritmo 4.1), converge a la solución del sistema lineal (4.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Como $\|F_j^{-1}G_j\|_\infty < 1$, $1 \leq j \leq r$, entonces existe una constante $0 \leq \theta < 1$, tal que $\|F_j^{-1}G_j\|_\infty \leq \theta < 1$, $1 \leq j \leq r$. Calculando la norma infinito de la matriz de iteración $T^{(l)}$, definida en (4.11), obtenemos para $l = 0, 1, 2, \ldots$,

$$\|T^{(l)}\|_\infty = \left\| \sum_{j=1}^{r} E_j (F_j^{-1}G_j)^{q(l,j)} \right\|_\infty \leq \max_{1\leq j\leq r} \left\| (F_j^{-1}G_j)^{q(l,j)} \right\|_\infty \leq \theta < 1.$$

Por el Lema 1.2 (Sección 1.3), se sigue la convergencia del Algoritmo 4.1. ■

Hacemos notar que el Corolario 4.1 se puede demostrar de forma análoga si la norma infinito se reemplaza por cualquier norma matricial que verifique que, dadas las matrices arbitrarias T, T_j y las matrices de peso E_j, $1 \leq j \leq r$, de tal forma que $T = \sum_{j=1}^{r} E_j T_j$, entonces $\|T\| \leq \max_{1 \leq j \leq r} \|T_j\|$, (ver [16]).

Para finalizar esta sección, analizaremos la convergencia de la versión relajada del método de multipartición no estacionario que corresponde al Algoritmo 4.2.

De forma análoga a como se hizo cuando estudiamos la convergencia del método de multipartición no estacionario, para el estudio de la convergencia de su versión relajada, podemos escribir el esquema iterativo correspondiente al Algoritmo 4.2 de la forma

$$x^{(l+1)} = \omega \sum_{j=1}^{r} E_j P_j^{q(l,j)} x^{(l)} + (1-\omega) x^{(l)}, \qquad l = 0, 1, 2, \ldots, \tag{4.15}$$

donde los operadores P_j, $1 \leq j \leq r$, están definidos en (4.4). Razonando como en el caso no relajado, se obtiene

$$x^{(l+1)} = H^{(l)} x^{(l)} + r^{(l)}, \qquad l = 0, 1, 2, \ldots, \tag{4.16}$$

siendo

$$H^{(l)} = \omega T^{(l)} + (1-\omega) I, \qquad l = 0, 1, 2, \ldots, \tag{4.17}$$

con $T^{(l)}$, $l = 0, 1, 2, \ldots$, las matrices definidas en (4.11), es decir, $T^{(l)} = \sum_{j=1}^{r} E_j (F_j^{-1} G_j)^{q(l,j)}$, $l = 0, 1, 2, \ldots$, y

$$r^{(l)} = \omega \sum_{j=1}^{r} E_j \left(\sum_{i=0}^{q(l,j)-1} (F_j^{-1} G_j)^i \right) F_j^{-1} b, \qquad l = 0, 1, 2, \ldots .$$

Siendo ξ la solución del sistema lineal (4.1), podemos escribir $\xi = H^{(l)}\xi + r^{(l)}$. Entonces, usando el análisis de error, se puede escribir ahora

$$e^{(l+1)} = x^{(l+1)} - \xi = H^{(l)}x^{(l)} + r^{(l)} - \left(H^{(l)}\xi + r^{(l)}\right) = H^{(l)}e^{(l)}, \qquad (4.18)$$

donde $H^{(l)}$ son las matrices de iteración definidas en (4.17). Entonces

$$e^{(l+1)} = H^{(l)}e^{(l)} = \cdots = H^{(l)}H^{(l-1)}\cdots H^{(0)}e^{(0)}, \qquad l = 0, 1, 2, \ldots .$$

Por tanto, a partir de esta expresión la sucesión de vectores generada por la iteración (4.15) (o equivalente (4.16)) converge a la solución del sistema lineal (4.1) si, y sólo si, $\lim_{l\to\infty} H^{(l)}H^{(l-1)}\cdots H^{(0)} = O$.

A continuación, utilizando los resultados de convergencia obtenidos para el método de multipartición no estacionario, vamos a dar resultados de convergencia para el método de multipartición no estacionario relajado (Algoritmo 4.2).

Comenzamos con un teorema que asegura la convergencia del Algoritmo 4.2 para valores del factor de relajación ω entre 0 y $\frac{2}{1+\max\limits_{1\le j\le r}\|F_j^{-1}G_j\|_A}$, bajo las mismas hipótesis que en el Teorema 4.3.

Teorema 4.5. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$, particiones P-regulares y $E_j = \alpha_j I$, $1 \le j \le r$, con $\alpha_j > 0$ y $\sum_{j=1}^{r} \alpha_j = 1$. Suponemos que $q(l,j) \ge 1$, $1 \le j \le r$, $l = 0, 1, 2, \ldots$, entonces, si $0 < \omega < \frac{2}{1+\max\limits_{1\le j\le r}\|F_j^{-1}G_j\|_A}$, el algoritmo de multipartición no estacionario relajado (Algoritmo 4.2) converge a la solución del sistema lineal (4.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Por (4.17) se sabe que las matrices de iteración del método de multipartición no estacionario relajado tienen la forma

$$H^{(l)} = \omega T^{(l)} + (1-\omega)I, \qquad l = 0, 1, 2, \ldots , \qquad (4.19)$$

donde $T^{(l)}$, $l = 0, 1, 2, \ldots$, son las matrices de iteración definidas en (4.11).

Por otra parte, en el Teorema 4.3 demostramos que existe una constante real θ, con $0 \leq \theta < 1$, tal que $\|T^{(l)}\|_A \leq \theta < 1$, $1 \leq j \leq r$. Luego, para todo $l = 0, 1, 2, \ldots$, tenemos

$$\begin{aligned} \|H^{(l)}\|_A &= \|\omega T^{(l)} + (1-\omega)I\|_A \\ &\leq \omega\|T^{(l)}\|_A + |1-\omega|\|I\|_A \\ &= \omega\|T^{(l)}\|_A + |1-\omega|. \end{aligned}$$

Atendiendo al valor de ω distinguimos dos casos:
a) $0 < \omega < 1$.

$$\omega\|T^{(l)}\|_A + |1-\omega| = \omega\|T^{(l)}\|_A + 1 - \omega \leq \omega\theta + 1 - \omega = \omega(\theta - 1) + 1 < 1.$$

b) $1 \leq \omega < \dfrac{2}{1 + \max\limits_{1 \leq j \leq r} \|F_j^{-1}G_j\|_A}$.

Teniendo en cuenta la expresión (4.14) de la demostración del Teorema 4.3 y además, que $\|F_j^{-1}G_j\|_A \leq \theta < 1$, para $1 \leq j \leq r$, se obtienen las siguientes

desigualdades

$$\begin{aligned}
\omega\|T^{(l)}\|_A + |1-\omega| &= \omega\|T^{(l)}\|_A + \omega - 1 \\
&= \omega\left\|\sum_{j=1}^{r} E_j \left(F_j^{-1}G_j\right)^{q(l,j)}\right\|_A + \omega - 1 \\
&\leq \omega\sum_{j=1}^{r}\alpha_j \left\|(F_j^{-1}G_j)^{q(l,j)}\right\|_A + \omega - 1 \\
&\leq \omega\sum_{j=1}^{r}\alpha_j \left\|(F_j^{-1}G_j)\right\|_A + \omega - 1 \\
&\leq \omega\max_{1\leq j\leq r}\|F_j^{-1}G_j\|_A + \omega - 1 \\
&= \omega\left(1 + \max_{1\leq j\leq r}\|F_j^{-1}G_j\|_A\right) - 1 \\
&< \frac{2}{1+\max_{1\leq j\leq r}\|F_j^{-1}G_j\|_A}\left(1 + \max_{1\leq j\leq r}\|F_j^{-1}G_j\|_A\right) - 1 = 1.
\end{aligned}$$

Por lo tanto, por el Lema 1.2 (Sección 1.3), para $0 < \omega < \frac{2}{1+\max_{1\leq j\leq r}\|F_j^{-1}G_j\|_A}$, se obtiene la convergencia del Algoritmo 4.2. ■

La convergencia del Algoritmo 4.2 también se puede asegurar con las mismas hipótesis que las que se dieron para el método de multipartición no estacionario en el Teorema 4.4, tomando ω en un intervalo $]0, \omega_0[$, con $\omega_0 > 1$.

Teorema 4.6. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares. Suponemos además que $\lim_{l\to\infty} q(l,j) = \infty$, $1 \leq j \leq r$, entonces, si $0 < \omega < \frac{2}{1+\epsilon}$, con $0 < \epsilon < 1$, el algoritmo de multipartición no estacionario relajado (Algoritmo 4.2), converge a la solución del sistema lineal (4.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Sean $H^{(l)}$, $l = 0, 1, 2, \ldots$, las matrices de iteración del método de multipartición no estacionario relajado, dadas en (4.17). Entonces

$$\|H^{(l)}\|_\infty = \|\omega T^{(l)} + (1-\omega)I\|_\infty,$$

donde $T^{(l)}$ son las matrices de iteración del esquema iterativo no estacionario, definidas en (4.11). Sea $0 < \epsilon < 1$, de la demostración del Teorema 4.4, se sigue que existe un entero l_0, tal que para $l \geq l_0$

$$\|T^{(l)}\|_\infty \leq \epsilon < 1.$$

Por tanto

$$\begin{aligned} \|H^{(l)}\|_\infty &= \omega\|T^{(l)}\|_\infty + |1-\omega| \\ &\leq \omega\epsilon + |1-\omega|. \end{aligned}$$

Como en el teorema anterior, distinguimos dos casos :

a) $0 < \omega < 1$.

$$\omega\epsilon + |1-\omega| = \omega\epsilon + 1 - \omega = \omega(\epsilon - 1) + 1 < 1.$$

b) $1 \leq \omega < \dfrac{2}{1+\epsilon}$.

$$\omega\epsilon + |1-\omega| = \omega\epsilon + \omega - 1 = \omega(\epsilon+1) - 1 < \frac{2}{1+\epsilon}(1+\epsilon) - 1 = 1.$$

Luego por el Lema 1.2 (Sección 1.3), queda demostrada la convergencia del Algoritmo 4.2. ■

En el corolario siguiente, demostraremos la convergencia del método de multipartición no estacionario relajado, bajo las mismas condiciones que en el Corolario 4.1, tomando ω en el intervalo $\left]0, \frac{2}{1+\max\limits_{1\leq j\leq r}\|F_j^{-1}G_j\|_\infty}\right[$.

Corolario 4.2. *Sean* $A = F_j - G_j$, $1 \leq j \leq r$, *particiones de la matriz* $A \in \mathbb{C}^{n\times n}$ *tal que* $\|F_j^{-1}G_j\|_\infty < 1$, $1 \leq j \leq r$. *Suponemos* $q(l,j) \geq 1$, $1 \leq j \leq r$, $l = 0,1,2,\ldots$, *entonces, si* $0 < \omega < \frac{2}{1+\max\limits_{1\leq j\leq r}\|F_j^{-1}G_j\|_\infty}$, *el algoritmo de multipartición no estacionario relajado (Algoritmo 4.2), converge a la solución del sistema lineal (4.1), para cualquier vector inicial* $x^{(0)}$.

Demostración. Sean $H^{(l)}$, $l = 0,1,2,\ldots$, las matrices de iteración del método de multipartición no estacionario relajado dadas en (4.17), es decir, $H^{(l)} = \omega T^{(l)} + (1-\omega)I$, con $T^{(l)}$ definidas en (4.11).

Entonces

$$\|H^{(l)}\|_\infty \leq \omega\|T^{(l)}\|_\infty + |1-\omega|.$$

En el Corolario 4.1, demostramos que existe una constante real θ con $0 \leq \theta < 1$, tal que $\|T^{(l)}\|_\infty \leq \theta < 1$. Luego, para todo $l = 0,1,2,\ldots$, tenemos

a) $0 < \omega < 1$.

$$\omega\|T^{(l)}\|_\infty + |1-\omega| = \omega\|T^{(l)}\|_\infty + 1 - \omega \leq \omega\theta + 1 - \omega = \omega(\theta-1) + 1 < 1.$$

b) $1 \leq \omega < \dfrac{2}{1+\max\limits_{1\leq j\leq r}\|F_j^{-1}G_j\|_\infty}$.

De la demostración del Corolario 4.1, tenemos que, para $1 \leq j \leq r$, se satisface $\|F_j^{-1}G_j\|_\infty \leq \theta < 1$; además, teniendo en cuenta la forma en que están

definidas las matrices $T^{(l)}$ en (4.11), obtenemos las siguientes desigualdades

$$\begin{aligned}
\omega\|T^{(l)}\|_\infty + |1-\omega| &= \omega\|T^{(l)}\|_\infty + \omega - 1 \\
&= \omega \left\| \sum_{j=1}^{r} E_j \left(F_j^{-1}G_j\right)^{q(l,j)} \right\|_\infty + \omega - 1 \\
&\leq \omega \sum_{j=1}^{r} \alpha_j \left\| (F_j^{-1}G_j)^{q(l,j)} \right\|_\infty + \omega - 1 \\
&\leq \omega \sum_{j=1}^{r} \alpha_j \left\| F_j^{-1}G_j \right\|_\infty + \omega - 1 \\
&\leq \omega \max_{1\leq j\leq r} \|F_j^{-1}G_j\|_\infty + \omega - 1 \\
&\leq \omega \left(1 + \max_{1\leq j\leq r} \|F_j^{-1}G_j\|_\infty\right) - 1 \\
&< \frac{2}{1 + \max\limits_{1\leq j\leq r} \|F_j^{-1}G_j\|_\infty} \left(1 + \max_{1\leq j\leq r} \|F_j^{-1}G_j\|_\infty\right) - 1 = 1.
\end{aligned}$$

Luego por el Lema 1.2 (Sección 1.3), para $0 < \omega < \frac{2}{1+\max\limits_{1\leq j\leq r}\|F_j^{-1}G_j\|_\infty}$, se obtiene la convergencia del Algoritmo 4.2. ■

4.3 Versión asíncrona

En esta sección vamos a estudiar la convergencia de las versiones asíncronas de los métodos iterativos de multipartición no estacionarios descritos en la Sección 4.1.

El interés que nos lleva a definir un modelo asíncrono, está en reducir el tiempo de comunicación y sincronización entre los distintos procesadores que

cooperan en un sistema multiprocesador. Estos modelos aplicados sobre procedimientos iterativos de resolución de sistemas de ecuaciones lineales, permiten que cada procesador pueda actualizar o recuperar algunas componentes del vector aproximación a la solución, en cualquier instante. Por lo tanto, en teoría, ningún procesador se mantiene ocioso.

Para ilustrar esto, consideramos de nuevo el Algoritmo 4.1, que reproducimos aquí.

Algoritmo 4.1. Algoritmo de multipartición no estacionario.
Dado un vector inicial $x^{(0)}$.

Desde $l = 0, 1, 2, \ldots$, hasta convergencia

En cada procesador j, $j = 1$ hasta r

$$y_j^{(0)} = x^{(l)}$$

Desde $k = 1$ hasta $q(l, j)$

$$F_j y_j^{(k)} = G_j y_j^{(k-1)} + b \qquad (4.20)$$

$$x^{(l+1)} = \sum_{j=1}^{r} E_j y_j^{(q(l,j))}. \qquad (4.21)$$

La sincronización de este algoritmo radica en que en cada iteración l, la etapa $(l+1)$ se realiza sólo después de que todos los procesadores hayan calculado sus iterados locales $y_j^{(q(l,j))}$, $1 \leq j \leq r$.

Para construir la versión asíncrona de dicho algoritmo, consideramos un esquema donde todos los procesadores están siempre trabajando sin necesidad de esperar a que otros procesadores finalicen sus cálculos.

Cada iteración global se construye a partir de la solución de algún procesador, precisamente el que acaba de actualizar su solución.

Concretamente, vamos a considerar una implementación paralela del método de multipartición no estacionario en la que cada parte del vector $x^{(l+1)}$, es decir, $E_j y_j^{(q(l,j))}$, pueda ser actualizada sin esperar a que las otras partes de $x^{(l+1)}$ lo sean.

En la construcción de dicho modelo asíncrono (ver [15]), necesitaremos una sucesión de enteros positivos de la forma $\{j_l\}_{l=0}^{\infty}$, $1 \leq j_l \leq r$, donde j_l representa el procesador que actualiza la aproximación de la solución en la l-ésima iteración.

También definimos un escalar $r_l - 1$, que indica el número de veces que se ha actualizado el vector de la iteración global por otros procesadores distintos del procesador j_l-ésimo durante el tiempo que éste ha necesitado para realizar sus cálculos. Esto implica que r_l es el menor entero positivo tal que, $j_l = j_{l+r_l}$.

Suponemos además, que existe un entero positivo K tal que $0 \leq r_l - 1 < K$, es decir, en el cálculo del iterado l–ésimo, un proceso no puede utilizar ningún valor de las componentes de un iterado j si $j < l - K$.

Teniendo en cuenta lo descrito previamente, podemos plantear el método de multipartición no estacionario asíncrono mediante el siguiente algoritmo.

Notemos que debido a los posibles retrasos limitados por el entero positivo K, la definición formal del esquema asíncrono, exige considerar $K + 1$ vectores iniciales $x^{(-K)} = x^{(-K+1)} = \cdots = x^{(0)}$.

Algoritmo 4.3. Algoritmo de multipartición no estacionario asíncrono.

Dados los vectores iniciales $x^{(-K)} = x^{(-K+1)} = \cdots = x^{(0)}$.

En cada procesador j_l, $l = 0, 1, 2, \ldots$, hasta convergencia

$$y_{j_l}^{(0)} = x^{(l)}$$

Para $k = 1$ hasta $q(l, j_l)$

$$F_{j_l} y_{j_l}^{(k)} = G_{j_l} y_{j_l}^{(k-1)} + b$$

$$x^{(l+r_l)} = (I - E_{j_l}) x^{(l+r_l-1)} + E_{j_l} y_{j_l}^{(q(l,j_l))}. \qquad (4.22)$$

A continuación pasamos a formalizar la versión relajada de este modelo asíncrono de la misma forma que se hizo en el caso síncrono. Es decir, vamos a introducir en el Algoritmo 4.3 un parámetro de relajación $\omega \in \mathbb{R}$, $\omega \neq 0$, sustituyendo el vector iterado $y_{j_l}^{(q(l,j_l))}$ en la ecuación (4.22), por la expresión $\omega y_{j_l}^{(q(l,j_l))} + (1 - \omega) x^{(l)}$. De esta forma el método de multipartición no estacionario relajado asíncrono, viene descrito por el siguiente algoritmo.

Algoritmo 4.4. Algoritmo de multipartición no estacionario relajado asíncrono.

Dados los vectores iniciales $x^{(-K)} = x^{(-K+1)} = \cdots = x^{(0)}$.

En cada procesador j_l, $l = 0, 1, 2, \ldots$, hasta convergencia

$$y_{j_l}^{(0)} = x^{(l)}$$

Para $k = 1$ hasta $q(l, j_l)$

$$F_{j_l} y_{j_l}^{(k)} = G_{j_l} y_{j_l}^{(k-1)} + b$$

$$x^{(l+r_l)} = (I - E_{j_l}) x^{(l+r_l-1)} + E_{j_l} \left(\omega y_{j_l}^{(q(l,j_l))} + (1-\omega) I \right). \tag{4.23}$$

Hacemos notar que claramente si $\omega = 1$ el Algoritmo 4.4 se reduce al Algoritmo 4.3.

A continuación vamos a estudiar la convergencia de los Algoritmos 4.3 y 4.4.

Para tal propósito, observamos en primer lugar, que el Algoritmo 4.3 correspondiente al método de multipartición no estacionario asíncrono produce la siguiente sucesión de vectores

$$x^{(l+r_l)} = (I - E_{j_l}) x^{(l+r_l-1)} + E_{j_l} P_{j_l}^{q(l,j_l)} x^{(l)}, \quad l = 0, 1, 2, \ldots, \tag{4.24}$$

donde los operadores P_j, $1 \leq j \leq r$, están definidos en (4.4) y el escalar r_l es el menor entero positivo tal que $j_l = j_{l+r_j}$. Para analizar la convergencia del esquema asíncrono (4.24), construiremos un procedimiento iterativo en $\mathbb{C}^{nK}$, donde K es el entero positivo que satisface $0 \leq r_l - 1 < K$.

Sea ξ la única solución del sistema lineal (4.1) y sea $e^{(l)} = x^{(l)} - \xi$ el vector error en la iteración l-ésima del esquema asíncrono definido en (4.24). Definimos el vector

$$\overline{e}_l = \begin{bmatrix} e^{(l)} \\ e^{(l-1)} \\ \vdots \\ e^{(l-K+1)} \end{bmatrix} \in \mathbb{C}^{nK}. \tag{4.25}$$

Como K satisface, $0 \leq r_l - 1 < K$, $l = 0, 1, 2, \ldots, r$, podemos asegurar que $l + r_l - K \leq l \leq l + r_l - 1$, $l = 0, 1, 2, \ldots$. Estas desigualdades nos permiten asegurar que el vector $e^{(l)}$ de dimensión n, es un bloque del vector $\overline{e}_{l+r_l-1}$, nK-dimensional , es decir,

$$\overline{e}_{l+r_l-1} = \begin{bmatrix} e^{(l+r_l-1)} \\ \vdots \\ e^{(l)} \\ \vdots \\ e^{(l+r_l-K)} \end{bmatrix}. \tag{4.26}$$

Como podemos observar, el vector $e^{(l)}$ ocupa la posición r_l en el vector anterior.

Denotamos por S_l, $l = 0, 1, 2, \ldots$, la matriz de tamaño $n \times nK$ que está dividida en bloques de tamaño n y que tiene un bloque identidad de tamaño n en la posición r_l y el resto de los $K - 1$ bloques son cero. Luego, podemos escribir

$$e^{(l)} = S_l \overline{e}_{l+r_l-1}. \tag{4.27}$$

Vamos a analizar seguidamente la expresión del error

$$\xi^{(l+r_l)} = x^{(l+r_l)} - \xi.$$

Puesto que ξ es un punto fijo de los operadores P_j definidos en (4.4), la expresión del error se puede escribir para $l = 0, 1, 2, \ldots$,

$$\begin{aligned} e^{(l+r_l)} &= x^{(l+r_l)} - \xi \\ &= (I - E_{j_l})x^{(l+r_l-1)} + E_{j_l}P_{j_l}^{q(l,j_l)}x^{(l)} - (I - E_{j_l})\xi - E_{j_l}P_{j_l}^{q(l,j_l)}\xi \\ &= (I - E_{j_l})e^{(l+r_l-1)} + E_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}e^{(l)}. \end{aligned} \tag{4.28}$$

Por tanto,

$$\overline{e}_{l+r_l} = B_{l+r_l}\overline{e}_{l+r_l-1}, \tag{4.29}$$

donde la matriz $B_{l+r_l} \in \mathbb{C}^{nK\times nK}$, está definida como

$$B_{l+r_l} = \begin{bmatrix} (I - E_{j_l}) & O & \ldots & O & O \\ I & O & \ldots & O & O \\ \vdots & \vdots & & \vdots & \vdots \\ O & O & \ldots & I & O \end{bmatrix} + \begin{bmatrix} E_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}S_l \\ O \\ \vdots \\ O \end{bmatrix}. \tag{4.30}$$

A continuación y basándonos en este desarrollo, vamos a dar condiciones suficientes de convergencia para el Algoritmo 4.3. El siguiente teorema demuestra la convergencia de dicho algoritmo bajo hipótesis similares a las del Teorema 4.3 para el caso síncrono.

Teorema 4.7. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares y $E_j = \alpha_j I$, $1 \leq j \leq r$, con $0 < \alpha_j \leq 1$. Suponemos que existe un entero positivo K, tal que, $0 \leq r_l - 1 < K$. Si $q(l,j) \geq 1$, $1 \leq j \leq r$, $l = 0, 1, 2, \ldots$, entonces, el algoritmo de multipartición no estacionario asíncrono (Algoritmo 4.3), converge a la solución del sistema lineal (4.1), para cualquier vector inicial.*

Demostración. Teniendo en cuenta la expresión (4.29), tenemos que $\overline{e}_{l+K} = B_{l+K}B_{l+K-1}\cdots B_{l+1}\overline{e}_l$, con $\overline{e_l}$ definido en (4.25).

Sea μ un vector arbitrario distinto de cero de dimensión nK, dividido como en (4.25), es decir, $\mu = (\mu^i)_{i=1}^K \in \mathbb{C}^{nK}$, con $\mu^i \in \mathbb{C}^n$. Definimos la siguiente norma vectorial en $\mathbb{C}^{nK}$

$$\|\mu\| = \max_{1\leq i\leq K} \|\mu^i\|_A,$$

y su norma matricial inducida en $\mathbb{C}^{nK\times nK}$. Para demostrar que $\lim_{l\to\infty}\overline{e}_l = 0$, es suficiente demostrar que existe una constante real $0 \leq \gamma < 1$, tal que $\|B_{l+K}B_{l+K-1}\cdots B_{l+1}\| \leq \gamma < 1, \quad l = 0, 1, 2, \ldots$.

Para tal propósito, consideramos el vector v_ν definido por

$$v_\nu = B_{l+\nu}B_{l+\nu-1}\cdots B_{l+1}\mu := \begin{bmatrix} v_\nu^1 \\ v_\nu^2 \\ \vdots \\ v_\nu^K \end{bmatrix},$$

$$v_\nu^i \in \mathbb{C}^n,\ i = 1, 2, \ldots, K,\ \nu \geq 1, \tag{4.31}$$

donde $B_{l+\nu}$, $l = 0, 1, 2, \ldots$, son las matrices definidas en (4.30).

Demostraremos que para todo vector $\mu \in \mathbb{C}^{nK} - \{0\}$, existe una constante real $0 \leq \gamma < 1$ tal que se verifica $\|v_K\| \leq \gamma\|\mu\|$.

Por la forma en que están definidas las matrices B_{l+r_l} en (4.30), la matriz S_l

en (4.27) y teniendo en cuenta que $E_j = \alpha_j I,\ 1 \leq j \leq r$, podemos escribir

$$
\begin{aligned}
v_1 &= B_{l+1}\mu \\
&= \left\{ \begin{bmatrix} (1-\alpha_{j_l})I & O & \dots & O & O \\ I & O & \dots & O & O \\ \vdots & \vdots & & \vdots & \vdots \\ O & O & \dots & I & O \end{bmatrix} + \begin{bmatrix} \alpha_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}S_l \\ O \\ \vdots \\ O \end{bmatrix} \right\} \begin{bmatrix} \mu^1 \\ \mu^2 \\ \vdots \\ \mu^K \end{bmatrix} \\
&= \begin{bmatrix} (1-\alpha_{j_l})\mu^1 + \alpha_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}S_l\mu \\ \mu^1 \\ \vdots \\ \mu^{K-1} \end{bmatrix} \\
&= \begin{bmatrix} (1-\alpha_{j_l})\mu^1 + \alpha_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}\mu^t \\ \mu^1 \\ \vdots \\ \mu^{K-1} \end{bmatrix} = \begin{bmatrix} v_1^1 \\ v_1^2 \\ \vdots \\ v_1^K \end{bmatrix},\ 1 \leq t \leq K. \qquad (4.32)
\end{aligned}
$$

Como podemos observar en la expresión anterior, la primera componente del vector v_1 es $v_1^1 = (1-\alpha_{j_l})\mu^1 + \alpha_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}\mu^t,\ 1 \leq t \leq K$.

Por otro lado, como $A = F_j - G_j,\ 1 \leq j \leq r$, son particiones P-regulares y la matriz A es hermítica y definida positiva, por el Teorema 4.3 sabemos que existe una constante real $0 \leq \theta < 1$ tal que $\|F_j^{-1}G_j\|_A \leq \theta$, y entonces $\|(F_j^{-1}G_j)^s\|_A \leq \theta < 1$, para todo $s \geq 1,\ 1 \leq j \leq r$. Teniendo en cuenta estos

resultados podemos escribir

$$\begin{aligned}
\|v_1^1\|_A &\leq (1-\alpha_{j_l})\|\mu^1\|_A + \alpha_{j_l}\|(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}\|_A\|\mu^t\|_A \\
&\leq (1-\alpha_{j_l})\|\mu^1\|_A + \alpha_{j_l}\theta\|\mu^t\|_A \\
&\leq (1-\alpha_{j_l})\max_{1\leq i\leq K}\|\mu^i\|_A + \alpha_{j_l}\theta\max_{1\leq i\leq K}\|\mu^i\|_A \\
&= [(1-\alpha_{j_l})+\alpha_{j_l}\theta]\,\|\mu\|_A \leq \gamma\|\mu\|,
\end{aligned}$$

donde $\gamma = \max_{1\leq j\leq r}[(1-\alpha_j)+\alpha_j\theta] < 1$.

También en la expresión (4.32) podemos observar que se verifica, $v_1^2 = \mu^1$, $v_1^3 = \mu^2, \ldots, v_1^K = \mu^{K-1}$, es decir, $v_1^i = \mu^{i-1}, i = 2,3,\ldots,K$. Tomando la norma vectorial $\|.\|_A$, podemos asegurar

$$\|v_1^i\|_A = \|\mu^{i-1}\|_A \leq \max_{1\leq j\leq K}\|\mu^j\|_A = \|\mu\|, \quad i = 2,3,\ldots,K. \tag{4.33}$$

Luego, para v_1 se verifica

$$\|v_1^i\|_A \leq \begin{cases} \gamma\|\mu\| & i = 1 \\ \|\mu\| & i = 2,\ldots,K. \end{cases} \tag{4.34}$$

A continuación demostraremos que

$$\|v_\nu^i\|_A \leq \begin{cases} \gamma\|\mu\| & i = 1,2,\ldots,\nu \\ \|\mu\| & i = \nu+1,\ldots,K. \end{cases} \tag{4.35}$$

La prueba de (4.35) se hará por inducción sobre ν. Para $\nu = 1$ el resultado se cumple por (4.34). Supongamos que se cumple para el entero positivo $\nu - 1$, es decir,

$$\|v_{\nu-1}^i\|_A \leq \begin{cases} \gamma\|\mu\| & i = 1,2,\ldots,\nu-1 \\ \|\mu\| & i = \nu,\nu+1,\ldots,K. \end{cases} \tag{4.36}$$

Teniendo en cuenta que para cualquier $\nu \geq 1$, se cumple

$$
\begin{aligned}
v_\nu &= B_{l+\nu} v_{\nu-1} \\
&= \left\{ \begin{bmatrix} (1-\alpha_{j_l})I & O & \dots & O & O \\ I & O & \dots & O & O \\ \vdots & \vdots & & \vdots & \vdots \\ O & O & \dots & I & O \end{bmatrix} + \begin{bmatrix} \alpha_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)} S_l \\ O \\ \vdots \\ O \end{bmatrix} \right\} \begin{bmatrix} v_{\nu-1}^1 \\ v_{\nu-1}^2 \\ \vdots \\ v_{\nu-1}^K \end{bmatrix} \\
&= \begin{bmatrix} (1-\alpha_{j_l})v_{\nu-1}^1 + \alpha_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)} S_l v_{\nu-1} \\ v_{\nu-1}^1 \\ \vdots \\ v_{\nu-1}^{K-1} \end{bmatrix} \\
&= \begin{bmatrix} (1-\alpha_{j_l})v_{\nu-1}^1 + \alpha_{j_l}(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)} v_{\nu-1}^t \\ v_{\nu-1}^1 \\ \vdots \\ v_{\nu-1}^{K-1} \end{bmatrix} = \begin{bmatrix} v_\nu^1 \\ v_\nu^2 \\ \vdots \\ v_\nu^K \end{bmatrix}, \quad 1 \leq t \leq K,
\end{aligned}
$$

se obtiene la siguiente igualdad

$$v_\nu^{1+i} = v_{\nu-1}^i, \ i = 1, 2, \dots, K-1, \tag{4.37}$$

y además, como por hipótesis de inducción se verifica que $\|v_{\nu-1}^i\|_A \leq \|\mu\|$, $i = 1, 2, \dots, K$, se tiene que

$$
\begin{aligned}
\|v_\nu^1\|_A &\leq (1-\alpha_{j_l})\|v_{\nu-1}^1\|_A + \alpha_{j_l}\|(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}\|_A \|v_{\nu-1}^t\|_A \\
&\leq (1-\alpha_{j_l})\|\mu\|_A + \alpha_{j_l}\theta\|\mu\|_A \\
&= [(1-\alpha_{j_l}) + \alpha_{j_l}\theta]\,\|\mu\|_A \leq \gamma\|\mu\|.
\end{aligned} \tag{4.38}
$$

Ahora, utilizando la hipótesis de inducción (4.36), la igualdad (4.37) y la desigualdad (4.38), obtenemos (4.35). De ahí que, $\|v_K\| = \|B_{l+K}B_{l+K-1}\cdots B_{l+1}\mu\| \leq \gamma\|\mu\|$, para todo vector no nulo, $\mu \in \mathbb{C}^{nK}$. Por lo tanto, por definición de norma matricial inducida, tenemos que $\|B_{l+K}B_{l+K-1}\cdots B_{l+1}\| \leq \gamma < 1, \quad l = 0, 1, 2, \ldots$. Así queda demostrada la convergencia del Algoritmo 4.3. ■

En el siguiente teorema, estudiamos la convergencia del Algoritmo 4.3 bajo hipótesis similares a las del Teorema 4.4 correspondiente al modelo síncrono. Además, como las matrices E_j ya no serán de la forma $\alpha_j I$, necesitamos tener la seguridad de que toda componente del vector iterado se actualiza con cierta regularidad. Esto se cumple, exigiendo que la sucesión $\{j_l\}_{l=0}^{\infty}$ sea regulada (ver por ejemplo, [15]).

Definición 4.1. Una sucesión de enteros positivos $\{j_l\}_{l=0}^{\infty}$, $1 \leq j_l \leq r$, con r fijo, se llama regulada, si existe un entero positivo K, tal que cada uno de los enteros $1, 2, \ldots, r$, aparece por lo menos una vez cada K elementos consecutivos de la sucesión.

Queremos hacer notar que, si la sucesión de enteros $\{j_l\}_{l=0}^{\infty}$ está regulada por el entero positivo K, entonces los escalares r_l verifican que $0 \leq r_{l-1} < K$.

Teorema 4.8. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares. Suponemos además que $\lim_{l\to\infty} q(l, j) = \infty$, $1 \leq j \leq r$ y que la sucesión de enteros $\{j_l\}_{l=0}^{\infty}$, $1 \leq j_l \leq r$ es regulada, entonces, el algoritmo de multipartición no estacionario asíncrono (Algoritmo 4.3), converge a la solución del sistema lineal (4.1), para cualquier vector inicial.*

Demostración. Sea v_ν el vector definido de la siguiente forma

$$v_\nu = |B_{l+\nu}B_{l+\nu-1}\cdots B_{l+1}|\overline{z} = \begin{bmatrix} v_\nu^1 \\ v_\nu^2 \\ \vdots \\ v_\nu^K \end{bmatrix}, \quad v_\nu^i \in \mathbb{C}^n,\ i = 1, 2, \ldots, K,\ \nu \geq 1,$$

y $\overline{z} = ((z)^T, (z)^T, \ldots, (z)^T)^T \in \mathbb{R}^{nK}$, donde $z = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$.

Por otro lado, por la expresión (4.29), claramente se verifica, $\overline{e}_{l+2K-1} = B_{l+2K-1}B_{l+2K-2}\cdots B_{l+1}\overline{e}_l$. Luego, para demostrar la convergencia del Algoritmo 4.3, comprobaremos que existe una constante real $0 \leq \gamma < 1$, tal que

$$v_{2K-1} = |B_{l+2K-1}B_{l+2K-2}\cdots B_{l+1}|\overline{z} \leq \gamma\overline{z},$$

por lo que la norma infinito verificará

$$\|B_{l+2K-1}B_{l+2K-2}\cdots B_{l+1}\|_\infty \leq \gamma < 1.$$

En el Teorema 4.4, comprobamos que dado un $\theta > 0$, existe un i_0 tal que dada una norma matricial $\|.\|$, se verifica que $\|(F_j^{-1}G_j)^i\| \leq \theta$, para todo $i \geq i_0$. Como $\lim_{l\to\infty} q(l, j) = \infty$, $1 \leq j \leq r$, tomando la norma infinito, existe un l_0 y una constante real θ, tal que se verifica

$$\|(F_j^{-1}G_j)^{q(l,j)}\|_\infty \leq \theta < 1, \quad \forall l \geq l_0. \tag{4.39}$$

Luego, por definición de la norma infinito, si tomamos $z = (1, 1, \ldots, 1)^T$, obtenemos

$$|(F_j^{-1}G_j)^{q(l,j)}|z \leq \theta z, \quad \forall l \geq l_0. \tag{4.40}$$

De la definición de B_ν en (4.30), de la forma de las matrices S_l en (4.27), teniendo en cuenta la desigualdad anterior y el vector $\overline{z}$ definido anteriormente, podemos escribir

$$
\begin{aligned}
|B_{l+\nu}|\overline{z} &\leq \left\{ \begin{bmatrix} (I-E_{j_l}) & O & \dots & O & O \\ I & O & \dots & O & O \\ \vdots & \vdots & & \vdots & \vdots \\ O & O & \dots & I & O \end{bmatrix} + \begin{bmatrix} E_{j_l}|(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}|S_l \\ O \\ \vdots \\ O \end{bmatrix} \right\} \begin{bmatrix} z \\ z \\ \vdots \\ z \end{bmatrix} \\
&= \begin{bmatrix} (I-E_{j_l})z + E_{j_l}|(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}|S_l\overline{z} \\ z \\ \vdots \\ z \end{bmatrix} \\
&= \begin{bmatrix} (I-E_{j_l})z + E_{j_l}|(F_{j_l}^{-1}G_{j_l})^{q(l,j_l)}|z \\ z \\ \vdots \\ z \end{bmatrix} \\
&\leq \begin{bmatrix} (I-E_{j_l})z + E_{j_l}\theta z \\ z \\ \vdots \\ z \end{bmatrix}, \quad \forall l \geq l_0.
\end{aligned}
$$

Como $0 \leq \theta < 1$, se cumple $(I-E_{j_l}) + E_{j_l}\theta \leq (I-E_{j_l}) + E_{j_l} = I$, entonces

tenemos

$$|B_{l+\nu}|\overline{z} \leq \begin{bmatrix} z \\ z \\ \vdots \\ z \end{bmatrix} = \overline{z}, \quad l \geq l_0, \quad \nu \geq 1.$$

Como $v_\nu \leq |B_{l+\nu}||B_{l+\nu-1}| \ldots |B_{l+1}|\overline{z}$, $\nu \geq 1$, claramente, $v_\nu \leq \overline{z}$.

Puesto que $v_\nu \leq |B_{l+\nu}|v_{\nu-1}$, al igual que se hizo en el teorema anterior, analizamos cada componente del vector v_ν,

$$\begin{aligned} v_\nu^1 &\leq (I - E_{j_l})v_{\nu-1}^1 + E_{j_l}|(F_{j_l}^{-1}G_{j_l})^{q(l,j)}|S_l v_{\nu-1} \qquad (4.41) \\ &\leq (I - E_{j_l})v_{\nu-1}^1 + E_{j_l}|(F_{j_l}^{-1}G_{j_l})^{q(l,j)}|S_l\overline{z} \\ &= (I - E_{j_l})v_{\nu-1}^1 + E_{j_l}|(F_{j_l}^{-1}G_{j_l})^{q(l,j)}|z \\ &\leq (I - E_{j_l})z + E_{j_l}\theta z \\ &= z + (\theta - 1)E_{j_l}z. \qquad (4.42) \end{aligned}$$

Dependiendo del valor que tienen los elementos diagonales en la matriz E_j, podemos asegurar para cada una de las componentes del vector v_ν^1 las siguientes acotaciones. Sea $h \in \{1, 2, \ldots, n\}$:

1.- Si el elemento que ocupa la posición h de E_{j_l} es nulo, entonces por la expresión (4.41) obtenemos

$$(v_\nu^1)_h \leq (v_{\nu-1}^1)_h.$$

2.- Si el elemento que ocupa la posición h de E_{j_l} tiene un valor α, con $0 < \alpha \leq 1$, entonces por la ecuación (4.42) se sigue

$$(v_\nu^1)_h \leq z_h + (\theta - 1)\alpha z_h = (1 + (\theta - 1)\alpha)z_h.$$

Si tomamos $\gamma = 1 + (\theta - 1)\alpha < 1$, entonces

$$(v_\nu^1)_h \leq \gamma z_h.$$

Por otro lado, como la sucesión $\{j_l\}_{l=0}^{\infty}$ es regulada, entonces todos los enteros $1, 2, \ldots, r$, aparecen en el subconjunto $\{j_l+1, \ldots, j_l+\nu\}$, $\nu \geq K$. Luego teniendo en cuenta los dos casos anteriores, y que las matrices E_j son no negativas y satisfacen $\sum_{j=1}^{r} E_j = I$, para todo $h = 1, 2, \ldots, n$, obtenemos $(v_\nu^1)_h \leq \gamma z_h$, $\nu \geq K$, es decir

$$v_\nu^1 \leq \gamma z \quad \text{si} \quad \nu \geq K. \tag{4.43}$$

Además, teniendo en cuenta que

$$v_{\nu+1} = \begin{bmatrix} v_{\nu+1}^1 \\ v_{\nu+1}^2 \\ \vdots \\ v_{\nu+1}^K \end{bmatrix} \leq |B_{l+\nu+1}| v_\nu = \begin{bmatrix} * \\ v_\nu^1 \\ \vdots \\ v_\nu^{K-1} \end{bmatrix},$$

se tiene para todo $\nu \geq 1$, $v_{\nu+1}^{1+i} \leq v_\nu^i$, $i = 1, 2, \ldots, K-1$, de donde

$$v_{2K-1}^{1+i} \leq v_{2K-i-1}^1, \quad i = 1, 2, \ldots, K-1. \tag{4.44}$$

Como $2K - i - 1 \geq K$, para $i = 1, 2, \ldots, K-1$, por las ecuaciones (4.43) y (4.44) obtenemos

$$v_{2K-1}^{1+i} \leq \gamma z, \quad i = 0, 1, 2, \ldots, K-1.$$

Luego

$$v_{2K-1} = |B_{l+2K-1} B_{l+2K-2} \cdots B_{l+1}| \overline{z} \leq \gamma \overline{z},$$

con lo que queda demostrada la convergencia del Algoritmo 4.3. ■

A continuación demostraremos la convergencia del Algoritmo 4.3, bajo condiciones similares a las dadas para el caso síncrono en el Corolario 4.1.

Corolario 4.3. *Sean* $A = F_j - G_j$, $1 \leq j \leq r$, *particiones de la matriz* $A \in \mathbb{C}^{n\times n}$ *tal que* $\|(F_j^{-1}G_j)\|_\infty < 1$, $1 \leq j \leq r$. *Suponemos* $q(l,j) \geq 1$, $1 \leq j \leq r$, $l = 0, 1, 2, \ldots$, *y una sucesión de enteros* $\{j_l\}_{l=0}^{\infty}$, $1 \leq j_l \leq r$, *regulada entonces, el algoritmo de multipartición no estacionario asíncrono (Algoritmo 4.3), converge a la solución del sistema lineal (4.1), para cualquier vector inicial.*

Demostración. Como $\|F_j^{-1}G_j\|_\infty < 1, \quad 1 \leq j \leq r$, entonces existe una constante $0 \leq \theta < 1$, tal que $\|F_j^{-1}G_j\|_\infty \leq \theta < 1, \quad 1 \leq j \leq r$, luego podemos asegurar que $\|(F_j^{-1}G_j)^i\|_\infty \leq \theta < 1$ para todo $i \geq 1$. Entonces como se satisface la acotación (4.39), la demostración se sigue de la misma forma que en el Teorema 4.8. ■

Queremos hacer notar que en el Corolario 4.3, la norma infinito puede sustituirse por cualquier norma matricial que proporcione una desigualdad análoga a la de la expresión (4.40).

Para finalizar esta sección, analizamos la convergencia de la versión relajada del método de multipartición no estacionario asíncrono correspondiente al Algoritmo 4.4. Dicho algoritmo, produce la siguiente sucesión de vectores

$$x^{(l+r_l)} = (I - E_{j_l})x^{(l+r_l-1)} + E_{j_l}\left(\omega P_{j_l}^{q(l,j_l)} + (1-\omega)I\right)x^{(l)}, \tag{4.45}$$

donde los operadores P_j están definidos en (4.4) y el escalar r_l es el menor entero positivo tal que $j_l = j_{l+r_j}$. Para analizar la convergencia del esquema asíncrono relajado (4.45), construiremos de nuevo un procedimiento en $\mathbb{C}^{nK}$, donde K es el entero positivo que satisface $0 \leq r_l - 1 < K$.

Puesto que ξ es un punto fijo de los operadores P_j definidos en (4.4), si $e^{(l)} = x^{(l)} - \xi$ es el vector error en la iteración l-ésima del esquema asíncrono definido en (4.45), entonces la expresión del error se puede escribir para $l = 0, 1, 2, \ldots$,

$$\begin{aligned} e^{(l+r_l)} &= x^{(l+r_l)} - \xi \\ &= (I - E_{j_l})x^{(l+r_l-1)} + E_{j_l}\left(\omega P_{j_l}^{q(l,j_l)} + (1-\omega)I\right)x^{(l)} \\ &- (I - E_{j_l})\xi - E_{j_l}\left(\omega P_{j_l}^{q(l,j_l)} + (1-\omega)I\right)\xi \\ &= (I - E_{j_l})e^{l+r_l-1} + E_{j_l}\left(\omega (F_{j_l}^{-1}G_{j_l})^{q(l,j_l)} + (1-\omega)I\right)e^{(l)}. \quad (4.46) \end{aligned}$$

Por tanto

$$\overline{e}_{l+r_l} = B_{l+r_l}\overline{e}_{l+r_l-1}, \quad (4.47)$$

donde la matriz $B_{l+r_l} \in \mathbb{C}^{nK\times nK}$, está definida como

$$B_{l+r_l} = \begin{bmatrix} (I - E_{j_l}) & O & \ldots & O & O \\ I & O & \ldots & O & O \\ \vdots & \vdots & & \vdots & \vdots \\ O & O & \ldots & I & O \end{bmatrix} + \begin{bmatrix} E_{j_l}\left(\omega (F_{j_l}^{-1}G_{j_l})^{q(l,j_l)} + (1-\omega)I\right)S_l \\ O \\ \vdots \\ O \end{bmatrix} \quad (4.48)$$

Comenzamos con un teorema que asegura la convergencia del Algorimo 4.4 para valores del factor de relajación ω entre 0 y $\frac{2}{1+\max\limits_{1\le j\le r}\|F_j^{-1}G_j\|_A}$, usando las mismas hipótesis que en el Teorema 4.7.

Teorema 4.9. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$, $1 \le j \le r$, particiones P-regulares y $E_j = \alpha_j I$, $1 \le j \le r$, con $0 < \alpha_j \le 1$. Suponemos que existe un entero positivo K, tal que, $0 \le r_l - 1 < K$. Si $q(l,j) \ge 1$, $1 \le j \le r$, $l = 0, 1, 2, \ldots$, y $0 < \omega < \frac{2}{1+\max\limits_{1\le j\le r}\|F_j^{-1}G_j\|_A}$, entonces, el*

algoritmo de multipartición no estacionario relajado asíncrono (Algoritmo 4.4), converge a la solución del sistema lineal (4.1), para cualquier vector inicial.

Demostración. Un análisis en la demostración del Teorema 4.7 nos lleva a la conclusión de que si demostramos que existe una constante real θ, con $0 \leq \theta < 1$, tal que se verifica

$$\|\omega(F_j^{-1}Q_j)^{q(l,j)} + (1-\omega)I\|_A \leq \theta, \quad 1 \leq j \leq r, \tag{4.49}$$

entonces la demostración se sigue de forma totalmente análoga a la del Teorema 4.7. Para $0 < \omega < \frac{2}{1+\max\limits_{1\leq j\leq r}\|F_j^{-1}G_j\|_A}$, la acotación (4.49) es cierta por el Teorema 4.5, por lo que queda demostrada la convergencia del Algoritmo 4.4. ■

La convergencia del Algoritmo 4.4 también se puede asegurar con las mismas hipótesis que las que se dieron para el método de multipartición no estacionario asíncrono, en el Teorema 4.8, tomando ω en el intervalo $]0, \omega_0[$, con $\omega_0 > 1$.

Teorema 4.10. *Sea A una matriz hermítica y definida positiva. Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares. Suponemos además que $\lim\limits_{l\to\infty} q(l,j) = \infty$, $1 \leq j \leq r$, que la sucesión de enteros $\{j_l\}_{l=0}^{\infty}$, $1 \leq j_l \leq r$ es regulada, y que $0 < \omega < \frac{2}{1+\epsilon}$, con $0 < \epsilon < 1$, entonces, el algoritmo de multipartición no estacionario relajado asíncrono (Algoritmo 4.4), converge a la solución del sistema lineal (4.1), para cualquier vector inicial.*

Demostración. Razonando de la misma forma que en el teorema anterior, vemos que el Teorema 4.8 nos lleva a la conclusión de que si demostramos que existe una constante real θ, con $0 \leq \theta < 1$, tal que se verifica

$$\|\omega(F_j^{-1}Q_j)^{q(l,j)} + (1-\omega)I\|_\infty \leq \theta, \quad 1 \leq j \leq r, \tag{4.50}$$

entonces, la demostración también se sigue de la misma forma que la del Teorema 4.8. Además, para $0 < \omega < \frac{2}{1+\epsilon}$, la acotación (4.50) es cierta por el Teorema 4.6, por lo que queda demostrada la convergencia del Algoritmo 4.4. ■

Por último, el siguiente corolario demuestra la convergencia del método de multipartición no estacionario relajado asíncrono, bajo las mismas condiciones que en el Corolario 4.3, tomando ω en el intervalo $\left]0, \frac{2}{1+\max\limits_{1\leq j\leq r}\|F_j^{-1}G_j\|_\infty}\right[$.

Corolario 4.4. *Sean $A = F_j - G_j$, $1 \leq j \leq r$, particiones de la matriz $A \in \mathbb{C}^{n\times n}$ tal que $\|(F_j^{-1}G_j)\|_\infty < 1$, $1 \leq j \leq r$. Suponemos $q(l,j) \geq 1$, $1 \leq j \leq r$, $l = 0, 1, 2, \ldots$, $\{j_l\}_{l=0}^{\infty}$, $1 \leq j_l \leq r$ es una sucesión de enteros regulada, y $0 < \omega < \frac{2}{1+\max\limits_{1\leq j\leq r}\|F_j^{-1}G_j\|_\infty}$, entonces, el algoritmo de multipartición no estacionario relajado asíncrono (Algoritmo 4.4), converge a la solución del sistema lineal (4.1), para cualquier vector inicial.*

Demostración. A partir de la demostración del Corolario 4.3 llegamos a la conclusión de que si existe una constante real θ, con $0 \leq \theta < 1$, tal que se verifica

$$\|\omega(F_j^{-1}Q_j)^{q(l,j)} + (1-\omega)I\|_\infty \leq \theta, \quad 1 \leq j \leq r, \tag{4.51}$$

entonces, la demostración se sigue de forma totalmente análoga a la del Corolario 4.3. Como $0 < \omega < \frac{2}{1+\max\limits_{1\leq j\leq r}\|F_j^{-1}G_j\|_\infty}$, entonces la acotación (4.51) es cierta por el Corolario 4.2. De esta forma queda demostrada la convergencia del Algoritmo 4.4. ■

4.4 Conclusiones

Hemos dedicado este capítulo al estudio de dos esquemas iterativos basados en el método de multipartición no estacionario.

El primero de dichos esquemas, dado por la expresión (4.9), corresponde al método de multipartición no estacionario síncrono, siendo el esquema (4.15) el esquema iterativo de dicho método correspondiente a su versión relajada. Para este modelo síncrono hemos estudiado su convergencia considerando que la matriz de coeficientes A del sistema lineal (4.1) es hermítica y definida positiva.

Los principales resultados de convergencia para el modelo síncrono (4.9) pueden verse en los Teoremas 4.2, 4.3 y 4.4. En ellos se demuestra la convergencia del método de multipartición no estacionario síncrono considerando r particiones P-regulares. Teniendo en cuenta unas determinadas condiciones para las matrices de peso E_j, es decir, $E_j = \alpha_j I$, $1 \leq j \leq r$, con $0 < \alpha_j \leq 1$, en los Teoremas 4.2 y 4.3 se considera que cada procesador pueda realizar un número variable de veces su trabajo en cada iteración. Por tanto estos teoremas han de verse más como resultados teóricos que como una herramienta computacional. Sin embargo, en el Teorema 4.4 no aparece dicha restricción en las matrices de peso exigiendo que cada procesador realice bastantes iteraciones internas.

Para el esquema (4.15), correspondiente al método de multipartición no estacionario síncrono relajado, los principales resultados de convergencia se dan en los Teoremas 4.5 y 4.6. En ellos, se demuestra la convergencia bajo hipótesis similares a las dadas en los teoremas para el estudio del esquema (4.9), pero teniendo en cuenta además un factor de relajación ω en determinados intervalos.

El segundo esquema para el que hemos realizado estudios de convergencia es el modelo de multipartición no estacionario asíncrono, dado por la expresión (4.24), junto con su versión relajada dada por la expresión (4.45). Los principales resultados de convergencia para el modelo de multipartición no estacionario asíncrono, se pueden ver en los Teoremas 4.7, y 4.8, y para su versión relajada, la convergencia se estudia en los Teoremas 4.9 y 4.10. En todos ellos se ha demostrado la convergencia del método en cuestión, bajo hipótesis similares a las de los correpondientes teoremas para la versión síncrona.

Para las versiones relajadas se ha probado la convergencia para factores de relajación entre 0 y ω_0, con $\omega_0 > 1$.

Capítulo 5

Métodos paralelos en dos etapas.

5.1 Introducción

En este capítulo nos centraremos en la resolución en paralelo de sistemas de ecuaciones lineales mediante métodos iterativos en dos etapas. Consideramos un sistema lineal no singular de la forma

$$Ax = b, \tag{5.1}$$

donde la matriz $A \in \mathbb{C}^{n\times n}$ y los vectores $x,\ b \in \mathbb{C}^n$.

Supongamos, sin pérdida de generalidad, que la matriz A está dividida en $r \times r$ bloques, con los bloques diagonales cuadrados de tamaño n_j, $\sum_{j=1}^{r} n_j = n$.

De esta forma, el sistema (5.1) se puede escribir como

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1r} \\ A_{21} & A_{22} & \cdots & A_{2r} \\ \vdots & \vdots & & \vdots \\ A_{r1} & A_{r2} & \cdots & A_{rr} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_r \end{bmatrix}, \tag{5.2}$$

donde x y b están divididos de acuerdo al tamaño de los bloques de A.

Para resolver el sistema (5.2), se pueden utilizar métodos iterativos por bloques clásicos. Las ventajas de estos métodos frente a los métodos iterativos puntuales consisten en que, generalmente, aceleran la convergencia y son de fácil adaptación para su procesamiento en paralelo. Podemos encontrar una descripción detallada de estos métodos, por ejemplo, en Berman y Plemmons [10], Ortega [80], [81] o Varga [98].

En particular, nos vamos a centrar en los métodos iterativos tipo Jacobi por bloques. En estos métodos, se considera una partición de la matriz A, de la forma $A = M - N$, donde M es una matriz no singular, diagonal por bloques, denotada por

$$M = \text{diag}(M_1, \ldots, M_j, \ldots, M_r), \tag{5.3}$$

y cada bloque M_j es de orden n_j, $1 \leq j \leq r$. A partir de estas hipótesis se considera el siguiente algoritmo.

Algoritmo 5.1. Algoritmo tipo Jacobi por bloques.

Dado un vector inicial $x^{(0)} = \left((x_1^{(0)})^T, \ldots, (x_r^{(0)})^T\right)^T$.

Desde $l = 0, 1, 2, \ldots$, hasta convergencia

Desde $j = 1$ hasta r

$$M_j x_j^{(l+1)} = (Nx^{(l)} + b)_j. \tag{5.4}$$

Hacemos notar que en el método clásico de Jacobi por bloques (ver Sección 3.1), la matriz diagonal M definida en (5.3), está formada por los bloques diagonales de la matriz A dada en (5.2), o sea, $M = \text{diag}(A_{11}, \ldots, A_{jj}, \ldots, A_{rr})$.

En el Algoritmo 5.1, se puede ver que para cada iteración l, $l = 0, 1, 2, \ldots$, se resuelven r sistemas lineales independientes de la forma (5.4). Por lo tanto, cada sistema lineal puede ser resuelto por un procesador diferente. Sin embargo, si el tamaño de las matrices M_j, $1 \leq j \leq r$, es grande, es conveniente aproximar la solución del sistema mediante otro método iterativo. A estos métodos se les denomina *métodos iterativos por bloques en dos etapas.*

Para describir dichos métodos, consideramos el sistema lineal (5.2) y la partición $A = M - N$, con M definida en (5.3). Asociado a dicha partición tenemos el esquema iterativo

$$Mx^{(l+1)} = Nx^{(l)} + b, \qquad l = 0, 1, 2, \ldots, \tag{5.5}$$

donde $x^{(0)}$ es un vector inicial arbitrario. Para cada matriz M_j consideramos una partición de la forma

$$M_j = F_j - G_j, \qquad 1 \leq j \leq r. \tag{5.6}$$

Entonces, para aproximar la solución de (5.4), en cada iteración *externa* l se realizan para cada j, $1 \leq j \leq r$, $q(l, j)$ iteraciones *internas* del método iterativo

definido por la partición (5.6). Es decir, este método produce una sucesión de vectores

$$x^{(l+1)} = \begin{bmatrix} x_1^{(l+1)} \\ x_2^{(l+1)} \\ \vdots \\ x_r^{(l+1)} \end{bmatrix}, \qquad l = 0, 1, 2, \ldots, \tag{5.7}$$

donde cada vector $x_j^{(l+1)}$, $1 \leq j \leq r$, es una aproximación a la solución del sistema lineal

$$M_j y_j = (Nx^{(l)} + b)_j, \qquad l = 0, 1, 2, \ldots, \tag{5.8}$$

que se ha obtenido realizando $q(l, j)$ etapas internas de un método iterativo basado en las particiones dadas en (5.6).

Así pues, el cálculo de los vectores $x_j^{(l+1)}$, $1 \leq j \leq r$, en cada iteración externa l, se realiza ejecutando las siguientes $q(l, j)$ etapas internas

$$\begin{aligned} y_j^{(k+1)} &= F_j^{-1} G_j y_j^{(k)} + F_j^{-1} (Nx^{(l)} + b)_j, \qquad (5.9) \\ k &= 0, 1, \ldots, q(l, j) - 1, \\ y_j^{(0)} &= x_j^{(l)}. \end{aligned}$$

Finalizadas estas $q(l, j)$ etapas, se define el vector x_j en la iteración externa $l + 1$ como

$$x_j^{(l+1)} = y_j^{(q(l,j))}. \tag{5.10}$$

Claramente, si se dispone de r procesadores, cada cálculo de los vectores $x_j^{(l+1)}$, $1 \leq j \leq r$, de la expresión (5.10), puede asignarse a un procesador realizándose así todos los cálculos en paralelo.

El modelo que se ha descrito corresponde a un esquema *síncrono*, ya que cada procesador j, $j = 1, 2, \ldots, r$, realiza un número arbitrario de iteraciones locales para calcular el vector $x_j^{(l+1)}$, antes de que se forme la nueva aproximación a la solución global. Dicho método se puede expresar mediante el algoritmo que se da a continuación.

Algoritmo 5.2. Algoritmo por bloques en dos etapas.

Dado un vector inicial $x^{(0)} = \left((x_1^{(0)})^T, \ldots, (x_r^{(0)})^T\right)^T$.

Desde $l = 0, 1, 2, \ldots,$ hasta convergencia

Desde $j = 1$ hasta r

$y_j^{(0)} = x_j^{(l)}$

Desde $k = 1$ hasta $q(l, j)$

$$F_j y_j^{(k)} = G_j y_j^{(k-1)} + (Nx^{(l)} + b)_j \qquad (5.11)$$

$x^{(l+1)} = \left((y_1^{(q(l,1))})^T, (y_2^{(q(l,2))})^T, \ldots, (y_r^{(q(l,r))})^T\right)^T$.

Al igual que se hizo con los métodos en dos etapas secuenciales (ver Sección 1.5), podemos distinguir entre los métodos por bloques en dos etapas estacionarios y los no estacionarios. En los métodos estacionarios, el número de iteraciones internas $q(l, j)$ es el mismo para cada j, $1 \leq j \leq r$, y para cada una de las etapas externas l; sin embargo, en los modelos no estacionarios el número de iteraciones internas $q(l, j)$, puede variar en cada bloque j y para cada etapa externa l.

Como se ha dicho en la Sección 3.3, los autores Bru, Migallón, Penadés

y Szyld (ver [20]) han generalizado los métodos por bloques en dos etapas, obteniendo los métodos de multipartición en dos etapas, que como su nombre indica aúnan los métodos en dos etapas y la técnica de multipartición.

Recordamos que la técnica de multipartición (ver Sección 3.1), se basa en considerar una colección de particiones

$$A = P_j - Q_j, \quad 1 \leq j \leq r, \quad det(P_j) \neq 0, \tag{5.12}$$

y unas matrices de peso $E_j, 1 \leq j \leq r$, diagonales no negativas cuya sumà es la identidad. Entonces, se plantea el siguiente esquema iterativo

$$x^{(l+1)} = \sum_{j=1}^{r} E_j P_j^{-1} Q_j x^{(l)} + \sum_{j=1}^{r} E_j P_j^{-1} b, \quad l = 0, 1, 2, \ldots, \tag{5.13}$$

donde $x^{(0)}$ es un vector inicial cualquiera.

Como podemos observar, el Algoritmo 5.1 se puede ver como un caso particular del esquema iterativo (5.13) cuando todas las particiones (5.12) son la misma, es decir, tomando $P_j = M = \text{diag}(M_1, \ldots, M_r)$ y las matrices diagonales E_j tienen unos en las entradas correspondientes al bloque diagonal M_j y cero en los demás casos.

Al igual que en los métodos tipo Jacobi por bloques, en cada iteración l del esquema iterativo (5.13), se necesitan resolver r sistemas lineales independientes, pero cuando estos sistemas lineales no se resuelven exactamente sino que se obtienen aproximaciones de la solución usando métodos iterativos, obtenemos el método de multipartición en dos etapas. Es decir, consideramos las particiones

$$P_j = B_j - C_j, \quad 1 \leq j \leq r. \tag{5.14}$$

Entonces, dado un vector inicial $x^{(0)}$, construimos una sucesión de vectores $\{x^{(l)}\}_{l=0}^{\infty}$, de la siguiente forma. Sea $x_j^{(l+1)}$ una aproximación a la solución del

sistema lineal

$$P_j y = Q_j x^{(l)} + b, \quad l = 0, 1, 2, \ldots , \tag{5.15}$$

que se ha obtenido realizando $q(l, j)$ iteraciones internas de un método iterativo basado en la partición $P_j = B_j - C_j$. Entonces, una vez calculados, a partir del vector $x^{(l)}$, todos los vectores $x_j^{(l+1)}$, $1 \leq j \leq r$, el siguiente elemento de la sucesión se construye como

$$x^{(l+1)} = \sum_{j=1}^{r} E_j x_j^{(l+1)}. \tag{5.16}$$

Luego el cálculo de los vectores $x_j^{(l+1)}$, $j = 1, 2, \ldots , r$, en cada iteración externa l, implica la realización de $q(l, j)$ etapas internas de la forma

$$y_j^{(k+1)} = B_j^{-1} C_j y_j^{(k)} + B_j^{-1}(C_j x^{(l)} + b_j), \tag{5.17}$$

$$k = 0, 1, \ldots , q(l, j) - 1,$$

$$y_j^{(0)} = x^{(l)}.$$

Una vez finalizadas estas $q(l, j)$ etapas internas, se define el vector $x_j^{(l+1)}$ como

$$x_j^{(l+1)} = y_j^{(q(l,j))}. \tag{5.18}$$

En el cálculo de las componentes de $x_j^{(l+1)}$, el procesador j, calcula sólo aquellas en las que el correspondiente elemento de la diagonal de la matriz E_j es no nulo. Sólo estas componentes serán relevantes para el cálculo del vector $x^{(l+1)}$ en (5.16).

En este modelo al igual que el anterior, cada procesador j puede realizar un número arbitrario de iteraciones locales para calcular el vector $x_j^{(l+1)}$ antes de

que se forme la nueva aproximación a la solución global (5.16), por lo tanto, este es un esquema no estacionario *síncrono*, que viene descrito por el siguiente algoritmo.

Algoritmo 5.3. Algoritmo de multipartición en dos etapas.

Dado un vector inicial $x^{(0)}$.

Desde $l = 0, 1, 2, \ldots,$ hasta convergencia

Desde $j = 1$ hasta r

$$y_j^{(0)} = x^{(l)}$$

Desde $k = 1$ hasta $q(l, j)$

$$B_j y_j^{(k)} = C_j y_j^{(k-1)} + (Q_j x^{(l)} + b) \quad (5.19)$$

$$x^{(l+1)} = \sum_{j=1}^{r} E_j y_j^{(q(l,j))} .$$

Podemos observar que el Algoritmo 5.2, se puede ver como un caso particular del Algoritmo 5.3, tomando $P_j = M,\ 1 \leq j \leq r$, donde M es la matriz diagonal por bloques definida en (5.3), es decir, $M = \text{diag}(M_1, \ldots, M_r)$, $B_j = \text{diag}(F_1, \ldots, F_r)$, con $F_j,\ 1 \leq j \leq r$, definida en (5.6) y $E_j,\ 1 \leq j \leq r$, son matrices por bloques divididas de acuerdo al tamaño de los bloques de A, y que tienen el bloque diagonal j igual a la identidad y ceros el resto.

También podemos decir, que el Algoritmo 5.3 se reduce al método de multipartición (5.13), cuando las particiones internas son de la forma $P_j = P_j - O$ y el número de iteraciones internas $q(l, j) = 1,\ 1 \leq j \leq r,\ l = 0, 1, 2, \ldots$.

Por otra parte, el método de multipartición no estacionario (Algoritmo 3.1,

Sección 3.2) es un caso especial del Algoritmo 5.3 cuando las particiones externas (5.12) son todas de la forma $A = A - O$.

Al igual que en los métodos por bloques en dos etapas, para los métodos de multipartición en dos etapas tenemos los modelos no estacionarios y estacionarios, según si las iteraciones internas $q(l, j)$, varían o no, para cada j y/o para cada iteración externa l.

Como ya se ha comentado en el Capítulo 3, el estudio realizado hasta el momento sobre la convergencia de los Algoritmos 5.2 y 5.3 se ha centrado en el contexto de las matrices monótonas y H-matrices. Para el Algoritmo 5.2, podemos ver en [48] un análisis de la convergencia para el caso en que la matriz A es monótona, la partición externa es regular y las particiones internas débilmente regulares; también se estudia la convergencia para el caso en que A es una H-matriz, la partición externa es una H-partición y las particiones internas son H-compatibles. En [19], también se estudia la convergencia del Algoritmo 5.2, incluyendo la versión relajada del mismo, para el caso en que la matriz A es monótona, y las particiones interna y externas satisfacen las mismas hipótesis que se dan en [48]. Es en este artículo donde se obtienen por primera vez resultados numéricos sobre estos métodos. Dichos resultados se realizaron sobre un multiprocesador de memoria compartida (ALLIANT FX/80).

En el estudio de la convergencia del Algoritmo 5.3, debemos citar a Bru, Migallón, Penadés y Szyld, que en [20], platean y estudian la convergencia del algoritmo para los casos en que la matriz A es monótona y H-matriz. Para el caso en que la matriz A es monótona, suponen las particiones externas regulares y las particiones internas débilmente regulares y, para el caso en que la matriz A es una H-matriz las particiones externas e internas son H-compatibles.

En este capítulo, vamos a analizar la convergencia de los Algoritmos 5.2 y 5.3, cuando la matriz del sistema A es hermítica y definida positiva. Además, estudiaremos el caso en que el número de iteraciones internas es suficientemente grande. Concretamente, en la Sección 5.2, estudiaremos la convergencia del Algoritmo 5.2 junto con su generalización, o Algoritmo 5.3 y probaremos que los resultados de convergencia del Algoritmo 5.2 no siempre se pueden extender al Algoritmo 5.3. En la Sección 5.3 daremos resultados de monotonicidad para dichos métodos.

5.2 Convergencia

Dado un vector inicial $x^{(0)}$, el Algoritmo 5.3 correspondiente al método de multipartición en dos etapas, produce la sucesión de vectores (ver, por ejemplo [20] y [84]),

$$x^{(l+1)} = \sum_{j=1}^{r} E_j R_{lj}^{q(l,j)} x^{(l)}, \quad l = 0, 1, 2, \ldots, \tag{5.20}$$

donde los operadores $R_{lj} : \mathbb{C}^n \to \mathbb{C}^n$, se definen para $l = 0, 1, 2, \ldots$, y $1 \leq j \leq r$ como

$$R_{lj}x = B_j^{-1}C_j x + B_j^{-1}\left(Q_j x^{(l)} + b\right). \tag{5.21}$$

Para analizar la convergencia del esquema iterativo (5.20), nos basamos en la forma en que están definidos los operadores R_{lj}, $l = 0, 1, 2, \ldots$, $1 \leq j \leq r$, y

escribimos dicho esquema de la siguiente forma

$$
\begin{aligned}
& x^{(l+1)} \\
&= \sum_{j=1}^{r} E_j R_{lj}^{q(l,j)} x^{(l)} \\
&= \sum_{j=1}^{r} E_j R_{lj}^{q(l,j)-1} \left[B_j^{-1} C_j x^{(l)} + B_j^{-1} \left(Q_j x^{(l)} + b \right) \right] \\
&= \sum_{j=1}^{r} E_j R_{lj}^{q(l,j)-2} \left[B_j^{-1} C_j \left(B_j^{-1} C_j x^{(l)} + B_j^{-1} \left(Q_j x^{(l)} + b \right) \right) + B_j^{-1} \left(Q_j x^{(l)} + b \right) \right] \\
&= \sum_{j=1}^{r} E_j R_{lj}^{q(l,j)-2} \left[(B_j^{-1} C_j)^2 x^{(l)} + (B_j^{-1} C_j + I) B_j^{-1} \left(Q_j x^{(l)} + b \right) \right] \\
&= \cdots \\
&= \sum_{j=1}^{r} E_j \left[(B_j^{-1} C_j)^{q(l,j)} x^{(l)} + \left(\sum_{i=0}^{q(l,j)-1} (B_j^{-1} C_j)^i \right) B_j^{-1} (Q_j x^{(l)} + b) \right]. \qquad (5.22)
\end{aligned}
$$

Luego el Algoritmo 5.3 se puede escribir mediante el siguiente esquema iterativo

$$x^{(l+1)} = T^{(l)} x^{(l)} + c^{(l)}, \quad l = 0, 1, 2, \ldots, \qquad (5.23)$$

donde $T^{(l)}$ son las matrices de iteración

$$T^{(l)} = \sum_{j=1}^{r} E_j \left[(B_j^{-1} C_j)^{q(l,j)} + \sum_{i=0}^{q(l,j)-1} (B_j^{-1} C_j)^i B_j^{-1} Q_j \right],$$

o equivalentemente

$$T^{(l)} = \sum_{j=1}^{r} E_j \left[(B_j^{-1} C_j)^{q(l,j)} + \left(I - (B_j^{-1} C_j)^{q(l,j)} \right) P_j^{-1} Q_j \right], \qquad (5.24)$$

y

$$c^{(l)} = \sum_{j=1}^{r} E_j \sum_{i=0}^{q(l,j)-1} (B_j^{-1} C_j)^i B_j^{-1} b.$$

Sea ξ la solución exacta del sistema lineal (5.1). Es fácil ver que dicha solución es un punto fijo del esquema iterativo (5.23). Entonces $\xi = T^{(l)}\xi + c^{(l)}$, donde $T^{(l)}$ son las matrices de iteración definidas en (5.24). Sea $e^{(l+1)} = x^{(l+1)} - \xi$ el vector error en la iteración $l+1$. Entonces, el vector $e^{(l+1)}$ se puede escribir en función del vector error en la iteración inicial, denotado por $e^{(0)}$, de la siguiente forma

$$e^{(l+1)} = T^{(l)}e^{(l)} = \cdots = T^{(l)}T^{(l-1)}\cdots T^{(0)}e^{(0)}, \quad l = 0, 1, 2, \ldots .$$

A partir de esta expresión, se tiene que la sucesión de vectores generada por la iteración (5.20) (o equivalente (5.23)) converge a la solución del sistema lineal (5.1) si, y sólo si $\lim_{l\to\infty} T^{(l)}T^{(l-1)}\cdots T^{(0)} = O$.

Como ya mencionamos en la Sección 1.3, el producto de matrices convergentes no siempre es convergente, por lo tanto, para demostrar la convergencia del Algoritmo 5.3 necesitamos de otras herramientas. En primer lugar, analizaremos la convergencia del Algoritmo 5.3 exigiendo que tanto las particiones externas e internas sean convergentes. Comenzamos con un teorema que demuestra dicha convergencia en el caso particular en el que todas las particiones externas son la misma. En este caso, es necesario exigir que se realicen suficientes iteraciones internas.

Teorema 5.1. *Consideramos el sistema lineal no singular (5.1). Suponemos que las particiones externas (5.12) son todas $A = P - Q$, tal que $\rho(P^{-1}Q) < 1$. Sean $P = B_j - C_j$, $1 \leq j \leq r$, particiones convergentes. Si $\lim_{l\to\infty} q(l, j) = \infty$, $1 \leq j \leq r$, entonces, el algoritmo de multipartición en dos etapas (Algoritmo 5.3), converge a la solución del sistema lineal (5.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Como $A = P - Q$ es una partición convergente, es decir, verifica $\rho(P^{-1}Q) < 1$, por el Teorema 1.5 (Sección1.2), existe una norma matricial inducida $\|\cdot\|$, tal que $\rho(P^{-1}Q) \leq \|P^{-1}Q\| < 1$.

Como también las particiones $P = B_j - C_j$, $1 \leq j \leq r$ son convergentes, se verifica $\rho(B_j^{-1}C_j) < 1$, lo que implica por el Teorema 1.6 (Sección 1.3), que $\lim_{s\to\infty}(B_j^{-1}C_j)^s = O$, con $s \geq 1$, $1 \leq j \leq r$. Como por hipótesis $\lim_{l\to\infty} q(l,j) = \infty$, claramente se obtiene $\lim_{l\to\infty}(B_j^{-1}C_j)^{q(l,j)} = O$, $1 \leq j \leq r$. Por tanto, $\lim_{l\to\infty}\sum_{j=1}^{r} E_j(B_j^{-1}C_j)^{q(l,j)} = O$.

Luego, dado un $\epsilon > 0$ existe un índice l_0 tal que $\left\|\sum_{j=1}^{r} E_j(B_j^{-1}C_j)^{q(l,j)}\right\| \leq \epsilon$, para todo $l \geq l_0$. Entonces para $l \geq l_0$, de (5.24) obtenemos

$$\begin{aligned}
\left\|T^{(l)}\right\| &= \left\|\sum_{j=1}^{r} E_j\left[(B_j^{-1}C_j)^{q(l,j)} + \left(I - (B_j^{-1}C_j)^{q(l,j)}\right)P^{-1}Q\right]\right\| \\
&= \left\|\sum_{j=1}^{r} E_j(B_j^{-1}C_j)^{q(l,j)} + \sum_{j=1}^{r} E_j\left((I - (B_j^{-1}C_j)^{q(l,j)})P^{-1}Q\right)\right\| \\
&\leq \left\|\sum_{j=1}^{r} E_j(B_j^{-1}C_j)^{q(l,j)}\right\| + \left\|\left(I - \sum_{j=1}^{r} E_j(B_j^{-1}C_j)^{q(l,j)}\right)P^{-1}Q\right\| \\
&\leq \left\|\sum_{j=1}^{r} E_j(B_j^{-1}C_j)^{q(l,j)}\right\| + \left(1 + \left\|\sum_{j=1}^{r} E_j(B_j^{-1}C_j)^{q(l,j)}\right\|\right)\|P^{-1}Q\| \\
&\leq \epsilon + (1+\epsilon)\|P^{-1}Q\| = \alpha_\epsilon.
\end{aligned}$$

Si tomamos $\epsilon < \frac{1-\|P^{-1}Q\|}{1+\|P^{-1}Q\|}$, tenemos

$$\begin{aligned}
\alpha_\epsilon &= \epsilon\left(1 + \|P^{-1}Q\|\right) + \|P^{-1}Q\| \\
&< 1 - \|P^{-1}Q\| + \|P^{-1}Q\| = 1.
\end{aligned}$$

Luego, por el Lema 1.2 (Sección 1.3), se sigue la convergencia del Algoritmo 5.3. ■

Claramente el método en dos etapas clásico de Nichols [76], se puede ver como un caso particular del Algoritmo 5.3 cuando en éste se considera una única partición interna y una única partición externa. Frommer y Szyld en [47], estudian la convergencia del método en dos etapas clásico introducido por Nichols demostrando en su Teorema 2.4, la convergencia del método cuando tanto la partición externa como interna son convergentes y se realizan suficientes iteraciones internas. Por tanto, el Teorema 5.1 se puede ver como una extensión del Teorema 2.4 de [47].

Por otro lado, en el Teorema 1 de Bru, Migallón y Penadés [19], se estudia la convergencia del Algoritmo 5.3 para el caso en que se considera una única partición externa $A = P - Q$, con $P = \text{diag}(A_{11}, \ldots, A_{rr})$, que verifica $\|P^{-1}Q\|_\infty < 1$ y las particiones internas son convergentes. También Bru, Migallón, Penadés y Szyld en su Teorema 3.1 de [20], estudian la convergencia de dicho algoritmo para el caso en que las particiones externas $A = P_j - Q_j$, satisfacen $\|P_j^{-1}Q_j\|_\infty < 1$, y las particiones internas son convergentes; por lo tanto, cuando se considera una única partición externa, el Teorema 5.1 se puede considerar una generalización de los teoremas anteriores, en el sentido en que debilitamos la condición $\|P^{-1}Q\|_\infty < 1$, dando otra más general, $\rho(P^{-1}Q) < 1$.

Hacemos notar, que cuando consideramos en el Algoritmo 5.3 que todas las particiones externas sean la misma, incluimos en los resultados de convergencia vistos anteriormente, no sólo a los métodos por bloques en dos etapas (Algoritmo 5.2), sino que también incluimos a los métodos solapados por bloques (ver, por ejemplo, [21], [45], [46] y [60]).

Como podemos observar en el Teorema 5.1, una de las condiciones que aseguraba la convergencia del Algoritmo 5.3 era que todas las particiones externas fueran la misma, sin embargo, de no ser así, dicho algoritmo puede no converger como se pone de manifiesto en el siguiente ejemplo.

Ejemplo 5.1. Consideramos la matriz

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}.$$

Sean $A = P_1 - Q_1 = P_2 - Q_2$ particiones de A, donde

$$P_1 = \begin{bmatrix} 2 & 1 \\ -1 & 1 \end{bmatrix}, \quad Q_1 = \begin{bmatrix} 0 & 2 \\ 0 & -1 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} -1 & 0 \\ 2 & 0 \end{bmatrix}.$$

Dichas particiones son convergentes, pues las matrices

$$P_1^{-1}Q_1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad P_2^{-1}Q_2 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix},$$

tienen un radio espectral igual a 0.

Consideramos las particiones internas triviales $P_1 = P_1 - O$ y $P_2 = P_2 - O$, y las matrices de peso

$$E_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad E_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Si calculamos las matrices de iteración correspondientes al Algoritmo 5.3 para cualquier número de iteraciones internas, todas ellas son

$$T^{(l)} = E_1 P_1^{-1} Q_1 + E_2 P_2^{-1} Q_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad l = 0, 1, 2, \ldots .$$

Esta matriz tiene un radio espectral igual a 1, por lo que el Algoritmo 5.3 no es convergente.

Como podemos ver en [20], para asegurar la convergencia del Algoritmo 5.3 cuando se consideran diferentes particiones externas, se debe verificar $\|P_j^{-1}Q_j\| < 1$, donde $\|.\|$ es cualquier norma ponderada asociada a un vector positivo, por ejemplo, la norma infinito.

Como consecuencia inmediata del Teorema 5.1, se sigue la convergencia del Algoritmo 5.3 y por lo tanto del Algoritmo 5.2, exigiendo la realización de bastantes iteraciones internas, cuando la matriz A es hermítica y definida positiva y cuando todas las particiones externas son la misma. Esto queda reflejado en el siguiente corolario.

Corolario 5.1. *Consideramos el sistema lineal (5.1), con A hermítica y definida positiva. Sea $A = P - Q$ una partición P-regular. Suponemos que la matriz P es hermítica y que las particiones $P = B_j - C_j$, $1 \leq j \leq r$, son P-regulares. Si $\lim_{l\to\infty} q(l,j) = \infty$, $1 \leq j \leq r$, entonces, el método de multipartición en dos etapas (Algoritmo 5.3), converge a la solución del sistema lineal (5.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Por ser la partición $A = P - Q$, P-regular y A hermítica y definida positiva, por el Teorema 1.8 (Sección 1.3), se verifica $\rho(P^{-1}Q) < 1$. Además P es hermítica y definida positiva, y como las particiones $P = B_j - C_j$, $1 \leq j \leq r$, son P-regulares, se obtiene de nuevo por el Teorema 1.8 (Sección 1.3), que $\rho(B_j^{-1}C_j) < 1$, $1 \leq j \leq r$. Luego, por el Teorema 5.1 se sigue la convergencia del Algoritmo 5.3. ■

A continuación vamos a analizar la convergencia de los Algoritmos 5.2 y 5.3, para cualquier número de iteraciones internas. Hacemos notar, que aunque las particiones de A sean P-regulares y la matriz A sea hermítica y definida positiva, puede no darse la convergencia de los Algoritmos 5.2 y 5.3. El siguiente ejemplo muestra esta situación.

Ejemplo 5.2. Consideramos la matriz hermítica y definida positiva

$$A = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}.$$

Sea $A = P - Q$ una partición de A, donde

$$P = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Dicha partición es P-regular pues la matriz

$$P^H + Q = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix},$$

es definida positiva. Se observa además que la matriz Q no es semidefinida positiva. Sean $P = B_1 - C_1 = B_2 - C_2$, particiones de P tal que

$$B_1 = B_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad C_1 = C_2 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Dichas particiones son P-regulares, ya que

$$B_1^H + C_1 = B_2^H + C_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

es definida positiva.

Sean las matrices de peso

$$E_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad E_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Si calculamos la matrices de iteración correspondientes al Algoritmo 5.3 con $q(l,j) = 1$, todas son

$$\begin{aligned} T^{(l)} &= (B_1^{-1}C_1) + (I - (B_1^{-1}C_1))P^{-1}Q \\ &= \begin{bmatrix} -0.5 & 0 \\ 0 & -0.5 \end{bmatrix} + \begin{bmatrix} 3/2 & 0 \\ 0 & 3/2 \end{bmatrix} \begin{bmatrix} 0 & 1/3 \\ 1/3 & 0 \end{bmatrix} \\ &= \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}, \quad l = 0, 1, 2, \ldots . \end{aligned}$$

Esta matriz tiene un radio espectral igual a 1, por lo que el Algoritmo 5.3 no es convergente.

En el siguiente teorema damos un resultado de convergencia para el Algoritmo 5.2 realizando un número acotado de iteraciones internas $q(l,j)$. Esta hipótesis es muy realista, ya que en la práctica siempre se realiza un número máximo de iteraciones internas en cada bloque.

Teorema 5.2. *Consideramos el sistema lineal (5.2), donde la matriz A es hermítica y definida positiva. Sea $A = M - N$, donde $M = \mathrm{diag}(M_1, \ldots, M_r)$ es la matriz diagonal por bloques definida en (5.3). Suponemos que M es hermítica y N semidefinida positiva. Sean $M_j = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares. Si la sucesión de iteraciones internas $q(l,j)$, $l = 0, 1, 2, \ldots$, $1 \leq j \leq$*

r, está acotada, entonces, el método por bloques en dos etapas (Algoritmo 5.2), converge a la solución del sistema lineal (5.2), para cualquier vector inicial $x^{(0)}$.

Demostración. Sea la matriz $H^{(l)}$ definida de la forma

$$H^{(l)} = \text{diag}\left((F_1^{-1}G_1)^{q(l,1)}, \ldots, (F_r^{-1}G_r)^{q(l,r)}\right), \quad l = 0, 1, 2, \ldots . \tag{5.25}$$

Como el Algoritmo 5.2 es un caso particular del Algoritmo 5.3, la matriz de iteración definida en (5.24) para el Algoritmo 5.3, se puede escribir ahora para el Algoritmo 5.2 como

$$T^{(l)} = H^{(l)} + (I - H^{(l)})M^{-1}N. \tag{5.26}$$

Por hipótesis, $M_j = F_j - G_j$, $1 \leq j \leq r$, es una partición P-regular de la matriz hermítica y definida positiva M_j. Entonces por el Teorema 1.8 (Sección 1.3), se verifica $\rho(F_j^{-1}G_j) < 1$, $1 \leq j \leq r$, y por el Lema 1.1 (Sección 1.3), para todo j y l, tal que, $1 \leq j \leq r$, $l = 0, 1, 2, \ldots$, la matriz $I - (F_j^{-1}G_j)^{q(l,j)}$ es no singular.

Luego, como existe la matriz $(I - (F_j^{-1}G_j)^{q(l,j)})^{-1}$, por el Lema 1.3 (Sección 1.3), para cada j y l, existe un único par de matrices $P_j^{(l)}$, $Q_j^{(l)}$, que son de la forma $P_j^{(l)} = M_j(I - (F_j^{-1}G_j)^{q(l,j)})^{-1}$ y $Q_j^{(l)} = P_j^{(l)} - M_j$, tal que $(F_j^{-1}G_j)^{q(l,j)} = (P_j^{(l)})^{-1}Q_j^{(l)}$. Además, por el Lema 4.1 (Sección 4.2), la partición $M_j = P_j^{(l)} - Q_j^{(l)}$ es también P-regular.

Para cada l, $l = 0, 1, 2, \ldots$, consideramos las matrices

$$P^{(l)} = \text{diag}\left(P_1^{(l)}, \ldots, P_r^{(l)}\right), \quad Q^{(l)} = \text{diag}\left(Q_1^{(l)}, \ldots, Q_r^{(l)}\right).$$

Claramente, la partición $M = P^{(l)} - Q^{(l)}$ es P-regular y $P^{(l)} = M(I - H^{(l)})^{-1}$.

Luego, la matriz de iteración (5.26), se puede escribir

$$\begin{aligned} T^{(l)} &= H^{(l)} + M^{-1}N - H^{(l)}M^{-1}N \\ &= I - I + H^{(l)} + M^{-1}N - H^{(l)}M^{-1}N \\ &= I - (I - H^{(l)})(I - M^{-1}N) \\ &= I - (I - H^{(l)})M^{-1}A = I - (P^{(l)})^{-1}A. \end{aligned} \tag{5.27}$$

Claramente, la matriz $A - (T^{(l)})^H A T^{(l)}$ es hermítica. Además

$$\begin{aligned} & A - (T^{(l)})^H A T^{(l)} \\ &= A - (I - (P^{(l)})^{-1}A)^H A (I - (P^{(l)})^{-1}A) \\ &= \left((P^{(l)})^{-1}A\right)^H A + A(P^{(l)})^{-1}A - \left((P^{(l)})^{-1}A\right)^H A \left((P^{(l)})^{-1}A\right) \\ &= \left((P^{(l)})^{-1}A\right)^H \left[(P^{(l)})^H + P^{(l)} - A\right] \left((P^{(l)})^{-1}A\right) \\ &= \left((P^{(l)})^{-1}A\right)^H \left[(P^{(l)})^H + Q^{(l)} + N\right] \left((P^{(l)})^{-1}A\right). \end{aligned} \tag{5.28}$$

Entonces como la partición $M = P^{(l)} - Q^{(l)}$ es P-regular, se verifica que la matriz $(P^{(l)})^H + Q^{(l)}$ es definida positiva y como N es semidefinida positiva, de (5.28) se sigue que la matriz hermítica $A - (T^{(l)})^H A T^{(l)}$ es definida positiva. Entonces, tomando la norma vectorial $\|\cdot\|_A$ tenemos, para todo $x \neq 0$ y $l = 0, 1, 2, \ldots,$

$$\|T^{(l)}x\|_A^2 = (T^{(l)}x)^H A (T^{(l)}x) = x^H (T^{(l)})^H A T^{(l)} x < x^H A x = \|x\|_A^2.$$

Ahora consideramos la norma matricial inducida por dicha norma vectorial (ver Definición 1.7, Sección 1.2) y obtenemos $\|T^{(l)}\|_A = \max_{x \neq 0} \frac{\|T^{(l)}x\|_A}{\|x\|_A} <$

$\max_{x\neq 0}\frac{\|x\|_A}{\|x\|_A} = 1,\ 1 \leq j \leq r$, de donde se sigue que $\|T^{(l)}\|_A < 1,\ 1 \leq j \leq r$. Por tanto, como las sucesiones $\{q(l,j)\}_{l=0}^{\infty},\ 1 \leq j \leq r$, están acotadas, hay un número finito de matrices de iteración. Luego, para $l = 0, 1, 2, \ldots$, existe una constante real $0 \leq \theta < 1$, tal que $\|T^{(l)}\|_A \leq \theta < 1,\ l = 0, 1, 2, \ldots$. Entonces, por el Lema 1.2 (Sección 1.3), la demostración queda finalizada. ■

Queremos hacer notar que las hipótesis sobre la partición externa $A = M - N$ en el Teorema 5.2, implican que esta partición es P-regular. Sin embargo, como hemos visto en el Ejemplo 5.2, el hecho de que la partición sea P-regular, no garantiza la convergencia del método por bloques en dos etapas cuando la matriz N no es semidefinida positiva. Además, la condición sobre el número de iteraciones internas que se realizan en cada bloque, puede ser debilitada suponiendo que existe una subsucesión $\{l_k\}_{k=0}^{\infty}$ tal que las sucesiones $\{q(l_k,j)\}_{k=0}^{\infty},\ 1 \leq j \leq r$, están acotadas (ver por ejemplo, [16]).

Por otra parte, de la demostración del Teorema 5.2, se sigue que cada matriz de iteración del Algoritmo 5.2, induce una única partición que es P-regular. Este resultado se demuestra en el siguiente corolario.

Corolario 5.2. *Consideramos el sistema lineal (5.2), donde la matriz A es hermítica y definida positiva. Sea $A = M - N$, donde $M = \mathrm{diag}(M_1, \ldots, M_r)$ es la matriz diagonal por bloques definida en (5.3). Suponemos que M es hermítica y N es semidefinida positiva. Sean $M_j = F_j - G_j,\ 1 \leq j \leq r$, particiones P-regulares, entonces, la única partición inducida por cada matriz de iteración $T^{(l)},\ l = 0, 1, 2, \ldots$, del método por bloques en dos etapas (Algoritmo 5.2), es P-regular.*

Demostración. De la demostración del Teorema 5.2, tenemos $\rho(T^{(l)}) <$

1, $l = 0, 1, 2, \ldots$, donde $T^{(l)}$ es la matriz de iteración definida en (5.26). Entonces, para cada l, de los Lemas 1.1 y 1.3 (Sección 1.3), se sigue que existe un único par de matrices B_l y C_l, tal que $T^{(l)} = B_l^{-1}C_l$, donde $B_l = A(I - T^{(l)})^{-1}$ y $C_l = B_l - A$.

En la demostración del Teorema 5.2 hemos visto que $T^{(l)} = I-(P^{(l)})^{-1}A$, $l = 0, 1, 2, \ldots$, con $P^{(l)} = M(I - H^{(l)})^{-1}$ y $H^{(l)}$ definida en (5.25).

Entonces podemos escribir la matriz B_l como

$$B_l = A(I - T^{(l)})^{-1} = A(I - I + (P^{(l)})^{-1}A)^{-1} = P^{(l)}.$$

Ahora, siguiendo la demostración del Teorema 5.2, como $B_l = P^{(l)}$, se verifica que la matriz $B_l^H + C_l = B_l^H + B_l - A$ es definida positiva y por lo tanto la partición $A = B_l - C_l$ es P-regular, y la demostración queda finalizada. ■

Como podemos apreciar, los resultados de convergencia del Corolario 5.1 y del Teorema 5.2 están basados en particiones P-regulares de una matriz hermítica. Sin embargo, en el Teorema 5.2 exigimos que además la partición externa $A = M - N$, satisfaga que N sea semidefinida positiva.

Una forma sencilla de asegurar las hipótesis sobre la partición externa en el Teorema 5.2, es la que damos a continuación a partir de la partición de Jacobi por bloques. Sea la partición $A = \tilde{M} - \tilde{N}$ donde $\tilde{M} = \mathrm{diag}(A_{11}, \ldots, A_{rr})$ y se consideran matrices cuadradas diagonales no negativas D_j, de tamaño n_j, $1 \leq j \leq r$, tal que la matriz $\tilde{N}+\mathrm{diag}(D_1, \ldots, D_r)$ es semidefinida positiva. Entonces la partición externa

$$A = M - N, \tag{5.29}$$

donde la matriz M es

$$M = \mathrm{diag}(M_1, \ldots, M_r),$$

con

$$M_j = A_{jj} + D_j,$$

y la matriz N está definida como

$$N = \tilde{N} + \mathrm{diag}(D_1, \ldots, D_r),$$

satisface las hipótesis del Teorema 5.2.

Otra forma de asegurar que N sea semidefinida positiva en el Teorema 5.2 es mediante el uso de un parámetro de relajación $\omega \in \mathbb{R}$, $\omega \neq 0$, en la iteración externa. El uso de este parámetro equivale a considerar la partición $A = \frac{1}{\omega}M - \langle \frac{1-\omega}{\omega}M + N \rangle$. Si la partición $A = M - N$ es P-regular, entonces para $0 < \omega \leq 0,5$ la matriz $\frac{1-\omega}{\omega}M + N$ es semidefinida positiva.

Teorema 5.3. *Consideramos el sistema lineal (5.1), donde la matriz A es hermítica y definida positiva. Sean $A = P_j - Q_j$, $1 \leq j \leq r$, particiones de la matriz A, tal que P_j es hermítica y Q_j es semidefinida positiva. Sean $P_j = B_j - C_j$, $1 \leq j \leq r$, particiones P-regulares y $E_j = \alpha_j I$, $1 \leq j \leq r$, con $\alpha_j > 0$, $\sum_{j=1}^{r} \alpha_j = 1$. Suponemos además que la sucesión de números de iteraciones internas $q(l,j)$, $l = 0, 1, 2, \ldots$, $1 \leq j \leq r$, está acotada, entonces, el método de multipartición en dos etapas (Algoritmo 5.3), converge a la solución del sistema lineal (5.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Sean $T^{(l)}$ las matrices de iteración del esquema iterativo correspondiente al método de multipartición en dos etapas, definidas en (5.24). Si consideramos las matrices $T_j^{(l)}$ definidas de la forma

$$T_j^{(l)} = (B_j^{-1}C_j)^{q(l,j)} + \left(I - (B_j^{-1}C_j)^{q(l,j)}\right) P_j^{-1}Q_j, \qquad (5.30)$$

$$l = 0, 1, 2, \ldots, \quad 1 \leq j \leq r,$$

entonces, las matrices de iteración $T^{(l)}$, con $l = 0, 1, 2, \ldots$, pueden escribirse

$$T^{(l)} = \sum_{j=1}^{r} E_j T_j^{(l)}. \tag{5.31}$$

Para cada j y l, $T_j^{(l)}$ es la matriz de iteración correspondiente a un método iterativo en dos etapas, donde la partición externa es $A = P_j - Q_j$, la partición interna es de la forma $P_j = B_j - C_j$, y $q(l, j)$ es el número de iteraciones internas (ver Capítulo 1, Sección 1.5).

Por el Corolario 1.1 (Sección 1.5), sabemos que cada matriz de iteración $T_j^{(l)}$ induce una única partición $A = R_j^{(l)} - S_j^{(l)}$, que es P-regular. Luego, las matrices de iteración $T^{(l)}$ de (5.31), se pueden escribir

$$T^{(l)} = \sum_{j=1}^{r} E_j T_j^{(l)} = \sum_{j=1}^{r} E_j (R_j^{(l)})^{-1} S_j^{(l)}, \quad l = 0, 1, 2, \ldots .$$

Para cada l, $l = 0, 1, 2, \ldots$, la matriz $T^{(l)}$ se puede ver como la matriz de iteración de un método de multipartición, en el que se han considerado para cada j, $1 \leq j \leq r$, la partición P-regular, $A = R_j^{(l)} - S_j^{(l)}$ y las matrices de peso E_j. Por el Teorema 4.1 (Sección 4.2), sabemos que si las matrices de peso E_j son de la forma $E_j = \alpha_j I$, entonces la única partición $A = U^{(l)} - V^{(l)}$ inducida por cada matriz de iteración $T^{(l)}$, es P-regular y entonces por el Lema 1.8 (Sección 1.3), se verifica $\rho(T^{(l)}) < 1$, $l = 0, 1, 2, \ldots$.

Como $T^{(l)} = (U^{(l)})^{-1} V^{(l)} = (U^{(l)})^{-1}(U^{(l)} - A) = I - (U^{(l)})^{-1} A$, $l = 0, 1, 2, \ldots$,

podemos escribir

$$\begin{aligned}
&A - (T^{(l)})^H A T^{(l)} \\
&= A - (I - (U^{(l)})^{-1}A)^H A (I - (U^{(l)})^{-1}A) \\
&= \left((U^{(l)})^{-1}A\right)^H A + A(U^{(l)})^{-1}A - \left((U^{(l)})^{-1}A\right)^H A \left((U^{(l)})^{-1}A\right) \\
&= \left((U^{(l)})^{-1}A\right)^H \left[(U^{(l)})^H + U^{(l)} - A\right] \left((U^{(l)})^{-1}A\right) \\
&= \left((U^{(l)})^{-1}A\right)^H \left[(U^{(l)})^H + V^{(l)}\right] \left((U^{(l)})^{-1}A\right).
\end{aligned}$$

Al ser las particiones $A = U^{(l)} - V^{(l)}$, P-regulares, se verifica que la matriz $(U^{(l)})^H + V^{(l)}$ es definida positiva y como $(U^{(l)})^{-1}A$ es no singular, podemos afirmar que las matrices $A - (T^{(l)})^H A T^{(l)}$ son definidas positivas, es decir, para todo $x \neq 0$, $x^H \left(A - (T^{(l)})^H A T^{(l)}\right) x > 0$.

Por lo tanto, para todo $x \neq 0$, se verifica

$$\|T^{(l)}x\|_A < \|x\|_A.$$

Si consideramos la norma matricial inducida por dicha norma vectorial (ver Definición 1.7, Sección 1.2), obtenemos $\|T^{(l)}\|_A = \max_{x\neq 0} \frac{\|T^{(l)}x\|_A}{\|x\|_A} < \max_{x\neq 0} \frac{\|x\|_A}{\|x\|_A} = 1$, $1 \leq j \leq r$. De donde $\|T^{(l)}\|_A < 1$, $1 \leq j \leq r$, y como el número de iteraciones internas $q(l,j)$, $l = 0, 1, 2, \ldots$, $1 \leq j \leq r$, está acotado, entonces podemos asegurar que hay un número finito de matrices de iteración $T^{(l)}$. Luego, existe una constante real $0 \leq \theta < 1$, tal que $\|T^{(l)}\|_A \leq \theta < 1$, $l = 0, 1, 2, \ldots$. Entonces, por el Lema 1.2 (Sección 1.3), la demostración queda finalizada. ■

Del teorema anterior se deduce, que cada matriz de iteración del Algoritmo 5.3 está inducida por una única partición que es P-regular. Este resultado se demuestra en el siguiente corolario.

Corolario 5.3. *Consideramos el sistema lineal (5.1), donde la matriz A es hermítica y definida positiva. Sean $A = P_j - Q_j$, $1 \leq j \leq r$, particiones de la matriz A, tal que P_j es hermítica y Q_j es semidefinida positiva. Sean $P_j = B_j - C_j$, $1 \leq j \leq r$, particiones P-regulares y $E_j = \alpha_j I$, $1 \leq j \leq r$, con $\alpha_j > 0$, $\sum_{j=1}^{r} \alpha_j = 1$. Suponemos además que la sucesión de números de iteraciones internas $q(l,j)$, $l = 0,1,2,\ldots$, $1 \leq j \leq r$, está acotada, entonces, la única partición inducida por cada matriz de iteración $T^{(l)}$, $l = 0,1,2,\ldots$, del método de multipartición en dos etapas (Algoritmo 5.3), es P-regular.*

Demostración. De la demostración del Teorema 5.3, tenemos que $\rho(T^{(l)}) < 1$, $l = 0,1,2,\ldots$, donde $T^{(l)}$ es la matriz de iteración del esquema iterativo correspondiente al método de multipartición en dos etapas, definida en (5.24). Entonces, para cada l, de los Lemas 1.1 y 1.3 (Sección 1.3), se sigue que existe un único par de matrices $F^{(l)}$ y $G^{(l)}$, tal que $T^{(l)} = (F^{(l)})^{-1}G^{(l)}$, donde $F^{(l)} = A(I - T^{(l)})^{-1}$ y $G^{(l)} = F^{(l)} - A$.

Como por hipótesis $E_j = \alpha_j I$, $1 \leq j \leq r$, de la demostración del Teorema 5.3, se sigue que $F^{(l)} = U^{(l)}$ y $G^{(l)} = V^{(l)}$, y como se demuestra en el Teorema 4.1 (Sección 4.2), dicha partición es P-regular, con lo que queda completada la demostración. ■

Aunque las hipótesis sobre las matrices de peso, en el Teorema 5.3 y en el Corolario 5.3 son muy restrictivas en la práctica, a continuación damos un ejemplo en el que mostramos que si las matrices de peso E_j no son de la forma $E_j = \alpha_j I$, $1 \leq j \leq r$, entonces el Algoritmo 5.3 puede no converger aunque las particiones $P_j = B_j - C_j$, $1 \leq j \leq r$, sean P-regulares.

Ejemplo 5.3. Consideramos la matriz simétrica y definida positiva

$$A = \begin{bmatrix} 0,5 & 0,25 \\ 0,25 & 0,5 \end{bmatrix}.$$

Sean $A = P_1 - Q_1 = P_2 - Q_2$, particiones de A, tal que

$$P_1 = P_2 = P, \quad Q_1 = Q_2 = Q,$$

donde

$$P = \begin{bmatrix} 0,75 & 0 \\ 0 & 0,75 \end{bmatrix}, \quad Q = \begin{bmatrix} 0,25 & -0,25 \\ -0,25 & 0,25 \end{bmatrix}.$$

Notamos que la matriz P es hermítica y la matriz Q es semidefinida positiva.

Sean $P_1 = B_1 - C_1$ y $P_2 = B_2 - C_2$, particiones de P_1 y P_2 respectivamente, tal que

$$B_1 = \begin{bmatrix} 4 & 1 \\ -1 & 0,5 \end{bmatrix}, \quad C_1 = \begin{bmatrix} 3,25 & 1 \\ -1 & -0,25 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} 0,5 & -1 \\ 1 & 4 \end{bmatrix}, \quad C_2 = \begin{bmatrix} -0,25 & -1 \\ 1 & 3,25 \end{bmatrix}.$$

Dichas particiones son P-regulares, pues las matrices

$$B_1^H + C_1 = \begin{bmatrix} 7,25 & 0 \\ 0 & 0,25 \end{bmatrix}, \quad B_2^H + C_2 = \begin{bmatrix} 0,25 & 0 \\ 0 & 7,25 \end{bmatrix},$$

son definidas positivas.

Sean las matrices de peso

$$E_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad E_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Notamos que las matrices E_j, $j = 1, 2$, no satisfacen el Teorema 5.3. Para todo $l = 0, 1, 2, \ldots$, si calculamos las matrices de iteración correspondientes al Algoritmo 5.3, para $q(l, 1) = q(l, 2) = 1$, todas son

$$T^{(l)} = \sum_{j=1}^{2} E_j \left[B_j^{-1} C_j + \left(I - B_j^{-1} C_j \right) P^{-1} Q \right] = \begin{bmatrix} 1 & 0,125 \\ 0,125 & 1 \end{bmatrix}.$$

Esta matriz tiene un radio espectral igual a 1,125 que es mayor que 1, por lo que el Algoritmo 5.3 no es convergente.

Este ejemplo nos ha mostrado por tanto, que las hipótesis sobre las matrices de peso en el Teorema 5.3 no se pueden aligerar como se hizo en el Teorema 5.2 para el Algoritmo 5.2.

Para finalizar esta sección, analizaremos la convergencia de la versión relajada de los Algoritmos 5.2 y 5.3.

Para obtener la versión relajada del Algoritmo 5.2, se introduce un parámetro de relajación $\omega \neq 0$ de tal forma que la computación del vector $y_j^{(k)}$ en (5.11) viene dada ahora por la ecuación (ver por ejemplo, [18]).

$$F_j y_j^{(k)} = \omega \left(G_j y_j^{(k-1)} + (N x^{(l)} + b)_j \right) + (1 - \omega) F_j y_j^{(k-1)}.$$

Con esta notación, el método por bloques en dos etapas relajado, viene expresado por el siguiente algoritmo.

Algoritmo 5.4. Algoritmo por bloques en dos etapas relajado.
Dado un vector inicial $x^{(0)} = \left((x_1^{(0)})^T, \ldots, (x_r^{(0)})^T\right)^T$.

Desde $l = 0, 1, 2, \ldots$, hasta convergencia

Desde $j = 1$ hasta r

$$y_j^{(0)} = x_j^{(l)}$$

Desde $k = 1$ hasta $q(l, j)$

$$F_j y_j^{(k)} = \omega \left(G_j y_j^{(k-1)} + (Nx^{(l)} + b)_j\right) + (1 - \omega) F_j y_j^{(k-1)} \tag{5.32}$$

$$x^{(l+1)} = \left((y_1^{(q(l,1))})^T, (y_2^{(q(l,2))})^T, \ldots, (y_r^{(q(l,r))})^T\right)^T.$$

Claramente, el Algoritmo 5.4, puede considerarse un caso especial del Algoritmo 5.2 en el que se han sustituido las particiones internas $M_j = F_j - G_j$ por las siguientes

$$M_j = \frac{1}{\omega} F_j - \left(\frac{1 - \omega}{\omega} F_j + G_j\right), \quad 1 \leq j \leq r, \quad \omega \neq 0.$$

Por tanto, si denotamos por $H_j = (1 - \omega) I + \omega F_j^{-1} G_j$, $1 \leq j \leq r$, de (5.25) y (5.26) se sigue que la matriz de iteración para el esquema iterativo producido por el Algoritmo 5.4 se puede escribir como

$$T^{(l)} = \overline{H}^{(l)} + (I - \overline{H}^{(l)}) M^{-1} N, \quad l = 0, 1, 2, \ldots, \tag{5.33}$$

con

$$\overline{H}^{(l)} = \operatorname{diag}\left(H_1^{q(l,1)}, \ldots, H_r^{q(l,r)}\right). \tag{5.34}$$

La versión relajada del Algoritmo 5.3, también se obtiene introduciendo un parámetro de relajación $\omega \neq 0$, de tal forma que la computación del vector $y_j^{(k)}$ en (5.19) viene dada ahora por la expresión

$$B_j y_j^{(k)} = \omega \left(C_j y_j^{(k-1)} + Q_j x^{(l)} + b \right) + (1-\omega) B_j y_j^{(k-1)}.$$

Teniendo en cuenta esta notación, el siguiente algoritmo describe el método de multipartición en dos etapas relajado.

Algoritmo 5.5. Algoritmo de multipartición en dos etapas relajado.

Dado un vector inicial $x^{(0)}$.

Desde $l = 0, 1, 2, \ldots,$ hasta convergencia

Desde $j = 1$ hasta r

$y_j^{(0)} = x^{(l)}$

Desde $k = 1$ hasta $q(l,j)$

$$B_j y_j^{(k)} = \omega \left(C_j y_j^{(k-1)} + Q_j x^{(l)} + b \right) + (1-\omega) B_j y_j^{(k-1)}. \tag{5.35}$$

$$x^{(l+1)} = \sum_{j=1}^{r} E_j y_j^{(q(l,j))}.$$

A partir de (5.24) y siguiendo un razonamiento análogo al que se ha realizado para el Algoritmo 5.4, obtenemos que las matrices de iteración para el esquema iterativo definido por el Algoritmo 5.5 están definidas por la expresión

$$T^{(l)} = \sum_{j=1}^{r} E_j \left[R_j^{q(l,j)} + (I - R_j^{q(l,j)}) P_j^{-1} Q_j \right], \quad l = 0, 1, 2, \ldots, \tag{5.36}$$

donde

$$R_j = (1-\omega)I + \omega B_j^{-1} C_j, \quad j = 1, 2, \ldots, r. \tag{5.37}$$

Es decir, el Algoritmo 5.5 puede verse como un caso especial del Algoritmo 5.3, en el que las particiones internas son de la forma

$$P_j = \frac{1}{\omega} B_j - \left(\frac{1-\omega}{\omega} B_j + C_j \right).$$

A continuación damos resultados de convergencia para los Algoritmos 5.4 y 5.5.

Comenzamos con un teorema, que bajo las mismas hipótesis que el Teorema 5.1, asegura la convergencia del Algoritmo 5.5 para valores del factor de relajación entre $0 < \omega \leq \frac{2}{1+\rho}$, $\rho = \max\limits_{1 \leq j \leq r} (B_j^{-1} C_j)$.

Teorema 5.4. *Consideramos el sistema lineal no singular (5.1). Suponemos que las particiones externas (5.12) son todas $A = P - Q$, tal que $\rho(P^{-1}Q) < 1$. Sean $P = B_j - C_j$, $1 \leq j \leq r$, particiones convergentes. Si $\lim\limits_{l \to \infty} q(l, j) = \infty$, $1 \leq j \leq r$ y $0 < \omega < \frac{2}{1+\rho}$, con $\rho = \max\limits_{1 \leq j \leq r} \rho(B_j^{-1} C_j)$, entonces, el algoritmo de multipartición en dos etapas relajado (Algoritmo 5.5), converge a la solución del sistema lineal (5.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Siguiendo la demostración del Teorema 5.1 y teniendo en cuenta la expresión (5.36), para probar la convergencia del Algoritmo 5.5 será suficiente ver que las matrices R_j definidas en (5.37) como

$$R_j = (1-\omega)I + \omega B_j^{-1} C_j,$$

satisfacen que su radio espectral es menor que 1.

Por las propiedades del radio espectral, podemos asegurar que

$$\begin{aligned}\rho(R_j) &= \rho\left((1-\omega)I + \omega B_j^{-1}C_j\right)\\ &\leq |1-\omega| + \omega\rho(B_j^{-1}C_j)\\ &\leq |1-\omega| + \omega \max_{1\leq j\leq r}\rho(B_j^{-1}C_j).\end{aligned}$$

Además, como las particiones $P = B_j - C_j$ son convergentes, entonces se verifica $\max_{1\leq j\leq r}\rho(B_j^{-1}C_j) = \rho < 1$.

Luego, para $1 \leq j \leq r$, $l = 0, 1, 2, \ldots$, y según el valor del factor ω, distinguimos dos casos:

a) $0 < \omega \leq 1$.

$$\begin{aligned}\rho(R_j) &\leq |1-\omega| + \omega\rho\\ &= 1 - \omega + \omega\rho\\ &< 1.\end{aligned}$$

b) $1 < \omega < \dfrac{2}{1+\rho}$.

$$\begin{aligned}\rho(R_j) &\leq |1-\omega| + \omega\rho\\ &= \omega - 1 + \omega\rho\\ &= \omega(1+\rho) - 1\\ &< \frac{2}{(1+\rho)}(1+\rho) - 1\\ &= 1.\end{aligned}$$

Luego así queda demostrada la convergencia del Algoritmo 5.5. ■

Como consecuencia de este teorema se sigue la convergencia del Algoritmo 5.5 bajo las mismas hipótesis que el Corolario 5.1, para valores del factor de relajación ω, $0 < \omega < \frac{2}{1+\rho}$, con $\rho = \max_{1\leq j\leq r}\rho(B_j^{-1}C_j)$.

Corolario 5.4. *Consideramos el sistema lineal (5.1), donde la matriz A es hermítica y definida positiva. Sea $A = P - Q$ una partición P-regular. Suponemos que la matriz P es hermítica y que las particiones $P = B_j - C_j$, $1 \leq j \leq r$, son P-regulares. Si $\lim_{l\to\infty} q(l,j) = \infty$, $1 \leq j \leq r$ y $0 < \omega < \frac{2}{(1+\rho)}$, con $\rho = \max_{1\leq j\leq r}\rho(B_j^{-1}C_j)$, entonces, el método de multipartición en dos etapas relajado (Algoritmo 5.5), converge a la solución del sistema lineal (5.1), para cualquier vector inicial $x^{(0)}$.*

Demostración. Consideramos para cada j, las matrices $R_j = (1-\omega)I + \omega B_j^{-1}C_j$. En el Corolario 5.1 ya vimos que por ser las particiones $P = B_j - C_j$, P-regulares, se verificaba que $\rho(B_j^{-1}C_j) < 1$. Por tanto, siguiendo la demostración del Teorema 5.4, si $0 < \omega < \frac{2}{1+\rho}$ se tiene que $\rho(R_j) < 1$. Así la demostración queda finalizada. ■

El siguiente teorema, bajo las mismas hipótesis que el Teorema 5.2, asegura la convergencia del Algoritmo 5.4 para valores del factor de relajación entre $0 < \omega \leq 1$.

Teorema 5.5. *Consideramos el sistema lineal (5.2), donde la matriz A es hermítica y definida positiva. Sea $A = M - N$, donde $M = \mathrm{diag}(M_1, \ldots, M_r)$ es la matriz diagonal por bloques definida en (5.3). Suponemos que M es hermítica y N es semidefinida positiva. Sean $M_j = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares. Suponemos que las sucesiones de iteraciones internas $q(l,j)$, $l =$*

$0, 1, 2, \ldots$, $1 \leq j \leq r$, *están acotadas y que* $0 < \omega \leq 1$, *entonces, el método por bloques en dos etapas relajado (Algoritmo 5.4), converge a la solución del sistema lineal (5.2), para cualquier vector inicial* $x^{(0)}$.

Demostración. Como se ha indicado anteriormente el Algoritmo 5.4, puede considerarse como un caso particular del Algoritmo 5.2, en el que las particiones internas son de la forma

$$M_j = \frac{1}{\omega} F_j - \left(\frac{1-\omega}{\omega} F_j + G_j \right), \quad 1 \leq j \leq r. \tag{5.38}$$

Para demostrar la convergencia del Algoritmo 5.4, será suficiente comprobar que la matriz

$$\left(\frac{1}{\omega} F_j \right) + \left(\frac{1-\omega}{\omega} F_j + G_j \right), \quad 1 \leq j \leq r, \tag{5.39}$$

es definida positiva.

La expresión (5.39) se puede escribir como

$$\begin{aligned} \frac{1}{\omega} F_j + \left(\frac{1-\omega}{\omega} F_j + G_j \right) &= \left(\frac{1}{\omega} + \frac{1-\omega}{\omega} \right) F_j + G_j \\ &= \left(\frac{2-\omega}{\omega} \right) F_j + G_j \\ &= \left(1 + \frac{2-2\omega}{\omega} \right) F_j + G_j \\ &= \left(\frac{2-2\omega}{\omega} \right) F_j + (F_j + G_j), \quad 1 \leq j \leq r. \end{aligned}$$

Por hipótesis, al ser la partición $M_j = F_j - G_j$, P-regular, tenemos que la matriz $F_j + G_j$ es definida positiva. Por otro lado, si $0 < \omega < 1$, se verifica que $\frac{2-2\omega}{\omega} > 0$, entonces, podemos asegurar que la matriz $\left(\frac{2-2\omega}{\omega} \right) F_j + (F_j + G_j)$, es definida positiva y, por lo tanto las particiones (5.38) son P-regulares. A partir de aquí, por el Teorema 5.2 la demostración queda finalizada. ■

Al igual que vimos para el Teorema 5.2, un corolario en el que demostrábamos que cada matriz de iteración del Algoritmo 5.2 estaba inducida por una única partición P-regular, ahora para la versión relajada, damos también un corolario que demuestra este resultado cuando $0 < \omega \leq 1$.

Corolario 5.5. *Consideramos el sistema lineal (5.2), donde la matriz A es hermítica y definida positiva. Sea $A = M - N$, donde $M = \mathrm{diag}(M_1, \ldots, M_r)$ es la matriz diagonal por bloques definida en (5.3). Suponemos que M es hermítica y N es semidefinida positiva. Sean $M_j = F_j - G_j$, $1 \leq j \leq r$, particiones P-regulares y sea ω tal que $0 < \omega \leq 1$, entonces, la única partición inducida por cada matriz de iteración $T^{(l)}$, $l = 0, 1, 2, \ldots$, del método por bloques en dos etapas relajado (Algoritmo 5.4), es P-regular.*

Demostración. De la demostración del Teorema 5.5, tenemos que para todo $l = 0, 1, 2, \ldots$, se verifica $\rho(T^{(l)}) < 1$, donde $T^{(l)}$ es la matriz de iteración de un método por bloques en dos etapas relajado, definida por la expresión (5.33). A partir de aquí, la demostración se sigue de forma análoga a como se hizo en el Corolario 5.2, y la prueba queda finalizada. ■

A continuación damos resultados de convergencia para el Algoritmo 5.5. Comenzamos con un teorema que asegura la convergencia de dicho algoritmo para valores del factor de relajación ω en el intervalo $]0, 1]$, usando las mismas hipótesis que en el Teorema 5.3.

Teorema 5.6. *Consideramos el sistema lineal (5.1), donde la matriz A es hermítica y definida positiva. Sean $A = P_j - Q_j$, $1 \leq j \leq r$, particiones de la matriz A, tal que P_j es hermítica y Q_j es semidefinida positiva. Sean $P_j =$*

$B_j - C_j$, $1 \leq j \leq r$, particiones P-regulares y $E_j = \alpha_j I$, $1 \leq j \leq r$, con $\alpha_j > 0$, $\sum_{j=1}^{r} \alpha_j = 1$. Suponemos además, que la sucesión de números de iteraciones internas $q(l,j)$, $l = 0,1,2,\ldots,\ 1 \leq j \leq r$, está acotada y que $0 < \omega \leq 1$, entonces, el método de multipartición en dos etapas relajado (Algoritmo 5.5), converge a la solución del sistema lineal (5.1), para cualquier vector inicial $x^{(0)}$.

Demostración. Consideramos las matrices $T_j^{(l)}$ definidas mediante la expresión

$$T_j^{(l)} = R_j^{q(l,j)} + (I - R_j^{q(l,j)})P_j^{-1}Q_j, \qquad (5.40)$$
$$l = 0,1,2,\ldots,\ \ 1 \leq j \leq r,$$

donde las matrices R_j, están definidas como

$$R_j = (1-\omega)I + \omega B_j^{-1}C_j, \quad l = 0,1,2,\ldots,\ \ 1 \leq j \leq r.$$

Entonces, teniendo en cuenta (5.36), las matrices de iteración correspondientes al método de multipartición en dos etapas relajado se pueden escribir como

$$T^{(l)} = \sum_{j=1}^{r} E_j T_j^{(l)}. \qquad (5.41)$$

Para cada j y l, $T_j^{(l)}$ es la matriz de iteración de un método en dos etapas relajado, en el que la partición externa es $A = P_j - Q_j$, la partición interna es

$$P_j = \frac{1}{\omega}B_j - \left(\frac{1-\omega}{\omega}B_j + C_j\right), \qquad (5.42)$$

y $q(l,j)$ es el número de iteraciones internas.

Para comprobar la convergencia del Algoritmo 5.5, será suficiente demostrar que las particiones internas (5.42) son P-regulares.

Como $0 < \omega \leq 1$ y $P_j = B_j - C_j,\ 1 \leq j \leq r$ son particiones P-regulares, al igual que en la demostración del Teorema 5.5, podemos asegurar que la matriz $\left(\frac{2-2\omega}{\omega}\right) B_j + (B_j + C_j)$, es definida positiva y, por lo tanto las particiones (5.42) son P-regulares. A partir de aquí, por el Teorema 5.3 la demostración queda finalizada. ■

Del teorema anterior se deduce que cada matriz de iteración del Algoritmo 5.5 está inducida por una única partición que es P-regular. Este resultado se demuestra en el siguiente corolario.

Corolario 5.6. *Consideramos el sistema lineal (5.1), donde la matriz A es hermítica y definida positiva. Sean $A = P_j - Q_j,\ 1 \leq j \leq r$, particiones de la matriz A, tal que P_j es hermítica y Q_j es semidefinida positiva. Sean $P_j = B_j - C_j,\ 1 \leq j \leq r$, particiones P-regulares y $E_j = \alpha_j I,\ 1 \leq j \leq r$, con $\alpha_j > 0,\ \sum_{j=1}^{r} \alpha_j = 1$. Suponemos además que la sucesión de números de iteraciones internas $q(l,j),\ l = 0,1,2,\ldots,\ 1 \leq j \leq r$, está acotada, entonces, la única partición inducida por cada matriz de iteración $T^{(l)},\ l = 0,1,2,\ldots,$ del método de multipartición en dos etapas relajado (Algoritmo 5.5), es P-regular.*

Demostración. De la demostración del Teorema 5.6 tenemos que para todo $l = 0,1,2\ldots,$ se verifica $\rho(T^{(l)}) < 1$, donde $T^{(l)}$ es la matriz de iteración de un método de multipartición en dos etapas relajado definida por la expresión (5.41). A partir de aquí la demostración se sigue de forma similar a la del Corolario 5.3 y así la demostración queda finalizada. ■

5.3 Monotonicidad

Como ya se ha comentado anteriormente, para comparar la velocidad de convergencia de un método iterativo, es usual comparar el radio espectral de la matriz de iteración para diferentes particiones.

A continuación damos un teorema de comparación para las matrices de iteración del Algoritmo 5.2 cuando el número de iteraciones de cada bloque permanece fijo en cada iteración externa.

Teorema 5.7. *Sea A una matriz hermítica y definida positiva. Consideramos la partición $A = M - N$, donde $M = \mathrm{diag}(M_1, \ldots, M_r)$ es la matriz diagonal por bloques definida en (5.3). Suponemos que M es hermítica y N es semidefinida positiva. Sean $M_j = F_j - G_j$, $1 \leq j \leq r$, tal que F_j es hermítica y G_j es definida positiva. Sean $q_1(j)$, $q_2(j)$, $1 \leq j \leq r$ enteros positivos. Consideramos dos métodos en dos etapas que difieren sólo en el número de iteraciones internas de cada bloque, fijo para cada iteración externa, $q_1(j)$, $1 \leq j \leq r$, en un caso y $q_2(j)$, $1 \leq j \leq r$, en el otro. Sean T_1, T_2 las matrices de iteración del Algoritmo 5.2 con $q_1(j)$ y $q_2(j)$, $1 \leq j \leq r$, iteraciones internas respectivamente. Si $q_1(j) > q_2(j)$, $1 \leq j \leq r$, entonces $\rho(T_1) < \rho(T_2) < 1$.*

Demostración. Para $k = 1, 2$, consideramos la matriz T_k definida como

$$T_k = H_k + (I - H_k)M^{-1}N, \tag{5.43}$$

donde la matriz H_k está definida como

$$H_k = \mathrm{diag}\left((F_1^{-1}G_1)^{q_k(1)}, \ldots, (F_r^{-1}G_r)^{q_k(r)}\right). \tag{5.44}$$

Claramente, la matriz T_k es la matriz de iteración de un método iterativo en dos etapas, definido por el Algoritmo 5.2, en el que el número de iteraciones internas en cada bloque j, $1 \leq j \leq r$, es $q_k(j)$.

Por el Corolario 5.2 (Sección 5.2), podemos asegurar que cada matriz de iteración T_k induce una única partición $A = M_{T_k} - N_{T_k}$, con $M_{T_k} = M(I - H_k)^{-1}$ y $N_{T_k} = M_{T_k} - M$ que además es P-regular. Como además A es definida positiva podemos asegurar para cada j y l que la matriz M_{T_k} es definida positiva, luego también lo será cada matriz $M_{T_k}^{-1}$.

Teniendo en cuenta la definición de H_k en (5.44), y que

$$M_j^{-1} = (I - F_j^{-1}G_j)^{-1}F_j^{-1} = \sum_{i=0}^{\infty}(F_j^{-1}G_j)^i F_j^{-1},$$

podemos escribir la matriz $M_{T_k}^{-1}$ como

$$\begin{aligned} M_{T_k}^{-1} &= (I - H_k)M^{-1} \\ &= \left(I - \operatorname{diag}\left((F_1^{-1}G_1)^{q_k(1)}, \ldots, (F_r^{-1}G_r)^{q_k(r)}\right)\right) M^{-1} \\ &= \left(I - \operatorname{diag}\left((F_1^{-1}G_1)^{q_k(1)}, \ldots, (F_r^{-1}G_r)^{q_k(r)}\right)\right) \operatorname{diag}(M_1^{-1}, \ldots, M_r^{-1}) \\ &= \operatorname{diag}\left(\sum_{i=0}^{q_k(1)-1}(F_1^{-1}G_1)^i F_1^{-1}, \ldots, \sum_{i=0}^{q_k(r)-1}(F_r^{-1}G_r)^i F_r^{-1}\right). \end{aligned} \tag{5.45}$$

Como para cada j, $1 \leq j \leq r$, M_j y F_j son matrices hermíticas, también la matriz $G_j = F_j - M_j$ es hermítica. Luego, la matriz $\sum_{i=0}^{q_k(j)-1}(F_j^{-1}G_j)^i F_j^{-1}$ es hermítica y por la forma en que están definidas las matrices $M_{T_k}^{-1}$ en (5.45), podemos decir que la matriz $M_{T_k}^{-1}$ es hermítica. Por lo tanto, también son hermíticas las matrices M_{T_k} y $N_{T_k} = M_{T_k} - A$.

Por la forma en que están definidas las matrices T_k en (5.43) y $M_{T_k} =$

$M(I - H_k)^{-1}$, podemos escribir la matriz hermítica N_{T_k} de la forma

$$\begin{aligned} N_{T_k} &= M_{T_k} T_k \\ &= M_{T_k}(H_k + (I - H_k)M^{-1}N) \\ &= M_{T_k} H_k + M_{T_k}(I - H_k)M^{-1}N \\ &= M_{T_k} H_k + M_{T_k} M_{T_k}^{-1} N \\ &= M_{T_k} H_k + N. \end{aligned} \tag{5.46}$$

Como las matrices N y N_{T_k} son hermíticas, de (5.46) se sigue que $M_{T_k}H_k$ es hermítica. Por otro lado, para cada j, como F_j y G_j son hermíticas y definidas positivas, entonces por el Teorema 1.10 (Sección 1.3), la matriz $F_j^{-1}G_j$ tiene valores propios positivos y por lo tanto también son positivos los valores propios de la matriz H_k, $k = 1, 2$. Como además M_{T_k} es hermítica y definida positiva, y $H_k = M_{T_k}^{-1}(M_{T_k}H_k)$, de nuevo por el Teorema 1.10 (Sección 1.3), la matriz $M_{T_k}H_k$ es definida positiva. Por tanto, teniendo en cuenta que N es semidefinida positiva, de (5.46) podemos asegurar que la matriz N_{T_k} es definida positiva.

Para cada j, consideramos dos enteros positivos $q_1(j)$, $q_2(j)$, tal que $q_1(j) > q_2(j)$ y la matriz $M_{T_1}^{-1} - M_{T_2}^{-1}$, que claramente es hermítica. Teniendo en cuenta el Teorema 1.11 (Sección 1.3), para completar la demostración será suficiente ver que la matriz $M_{T_1}^{-1} - M_{T_2}^{-1}$ es definida positiva, ya que como se explicó en la Sección 1.3 esto es equivalente a decir que, $N_{T_1} \prec N_{T_2}$.

A partir de (5.45), podemos escribir

$$\begin{aligned} M_{T_1}^{-1} - M_{T_2}^{-1} &= \operatorname{diag}\left(\sum_{i=0}^{q_1(1)-1} (F_1^{-1}G_1)^i F_1^{-1}, \ldots, \sum_{i=0}^{q_1(r)-1} (F_r^{-1}G_r)^i F_r^{-1} \right) \\ &- \operatorname{diag}\left(\sum_{i=0}^{q_2(1)-1} (F_1^{-1}G_1)^i F_1^{-1}, \ldots, \sum_{i=0}^{q_2(r)-1} (F_r^{-1}G_r)^i F_r^{-1} \right) \\ &= \operatorname{diag}\left(\sum_{i=q_2(1)}^{q_1(1)-1} (F_1^{-1}G_1)^i F_1^{-1}, \ldots, \sum_{i=q_2(r)}^{q_1(r)-1} (F_r^{-1}G_r)^i F_r^{-1} \right). \end{aligned}$$

Puesto que F_j es hermítica y definida positiva, y $\sum_{i=q_2(j)}^{q_1(j)-1} (F_j^{-1}G_j)^i$, $1 \leq j \leq r$ tiene valores propios positivos, entonces del Teorema 1.10 (Sección 1.3), se sigue que la matriz diagonal por bloques $M_{T_1}^{-1} - M_{T_2}^{-1}$ es definida positiva, y así la demostración queda finalizada. ■

El resultado obtenido en el Teorema 5.7 parece intuitivo, pero como ilustramos a continuación, si las condiciones impuestas en el mismo no se satisfacen, dicho teorema puede no ser cierto.

Ejemplo 5.4. Consideramos la matriz simétrica y definida positiva

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

Sea $A = M - N$ una partición de la matriz A, donde

$$M = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}, \quad N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Como se puede observar, la matriz M es hermítica y la matriz N es semidefinida positiva. Sea $M = F - G$, una partición de M, tal que

$$F = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad G = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Dicha partición es P-regular, pues la matriz

$$F^H + G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

es definida positiva. Según el Teorema 2.1 de [71], el método en dos etapas converge para estas particiones. Sin embargo, si tomamos $q_1 = 2$ y $q_2 = 1$, se observa que calculando las matrices de iteración correspondientes, tenemos

$$T_{q_1} = (F^{-1}G)^2 + (I - (F^{-1}G)^2)M^{-1}N = \begin{bmatrix} 0,5 & 0 \\ 0 & 0,5 \end{bmatrix},$$

y

$$T_{q_2} = (F^{-1}G)^1 + (I - (F^{-1}G)^1)M^{-1}N = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Si calculamos el radio espectral de cada una de las matrices de iteración tenemos $\rho(T_{q_1}) = 0,5$ y $\rho(T_{q_2}) = 0$, que son menores que 1, pero sin embargo,

$$\rho(T_{q_2}) < \rho(T_{q_1}),$$

siendo $q_2 < q_1$. Esto es debido a que la matriz G no es definida positiva, como exigía el Teorema 5.7.

A continuación se da un teorema de comparación para el Algoritmo 5.3.

Teorema 5.8. *Sea A una matriz hermítica y definida positiva. Consideramos las particiones $A = P_j - Q_j$, $1 \leq j \leq r$, tal que P_j es hermítica y Q_j es semidefinida positiva. Sean $P_j = B_j - C_j$, $1 \leq j \leq r$, tal que la matriz B_j es hermítica y C_j definida positiva. Consideramos $E_j = \alpha_j I$, $1 \leq j \leq r$, con $\alpha_j > 0$, $\sum_{j=1}^{r} \alpha_j = 1$. Consideramos además, dos métodos de multipartición en dos*

etapas que difieren sólo en el número de iteraciones internas que se realizan para cada partición externa fija, con $q_1(j)$, $1 \leq j \leq r$ *número de iteraciones internas en un caso y* $q_2(j)$, $1 \leq j \leq r$, *en el otro. Sean* T_1, T_2 *las matrices de iteración correspondientes al Algoritmo 5.3 con* $q_1(j)$ *y* $q_2(j)$, $1 \leq j \leq r$ *el número de iteraciones internas que realiza cada método. Si* $q_1(j) > q_2(j)$, $1 \leq j \leq r$, *entonces* $\rho(T_1) < \rho(T_2) < 1$.

Demostración. Para $k = 1, 2$ y para cada j, $1 \leq j \leq r$, definimos la matriz $T_{k,j}$ como

$$T_{k,j} = (B_j^{-1}C_j)^{q_k(j)} + \left(I - (B_j^{-1}C_j)^{q_k(j)}\right) P_j^{-1}Q_j, \quad 1 \leq j \leq r.$$

La matriz $T_{k,j}$ es la matriz de iteración correspondiente a un método en dos etapas donde las particiones externas son $A = P_j - Q_j$, las particiones internas son $P_j = B_j - C_j$, y $q_k(j)$ es el número de iteraciones internas. Por el Corolario 5.2 (Sección 5.2), se deduce que la única partición $A = R_{k,j} - S_{k,j}$ inducida por cada matriz de iteración $T_{k,j}$ es P-regular.

Entonces, para $k = 1, 2$, cada matriz de iteración T_k del Algoritmo 5.3 se puede escribir como

$$T_k = \sum_{j=1}^{r} E_j T_{k,j}, \quad \text{con} \quad T_{k,j} = R_{k,j}^{-1} S_{k,j}. \tag{5.47}$$

Además, la matriz $R_{k,j}$ se puede escribir como

$$R_{k,j} = P_j \left(I - (B_j^{-1}C_j)^{q_k(j)}\right)^{-1}, \quad k = 1, 2, \quad 1 \leq j \leq r.$$

Luego

$$\begin{aligned} R_{k,j}^{-1} &= \left(I - (B_j^{-1}C_j)^{q_k(j)}\right) P_j^{-1} \\ &= \left(I - (B_j^{-1}C_j)^{q_k(j)}\right) (I - B_j^{-1}C_j)^{-1} B_j^{-1} \\ &= \sum_{i=0}^{q_k(j)-1} (B_j^{-1}C_j)^i B_j^{-1}. \end{aligned} \tag{5.48}$$

Como P_j y B_j son hermíticas entonces la matriz $C_j = P_j - B_j$ es hermítica, luego la matriz $\sum_{i=0}^{q_k(j)-1} (B_j^{-1}C_j)^i B_j^{-1}$ es hermítica y por la forma en que están definidas las matrices $R_{k,j}^{-1}$ en (5.48), podemos asegurar que también dichas matrices son hermíticas. Claramente, también son hermíticas las matrices $R_{k,j}$ y $S_{k,j} = R_{k,j} - A$.

Por otro lado, para cada j, $1 \leq j \leq r$, la matriz $S_{k,j}$ se puede escribir para $k = 1, 2$, como

$$S_{k,j} = R_{k,j}T_{k,j} = R_{k,j}(B_j^{-1}C_j)^{q_k(j)} + Q_j.$$

Como $S_{k,j}$ y Q_j son matrices hermíticas entonces la matriz $R_{k,j}(B_j^{-1}C_j)^{q_k(j)}$ es hermítica. Por otro lado, como B_j y C_j son hermíticas y definidas positivas, entonces por el Teorema 1.10 (Sección 1.3), la matriz $B_j^{-1}C_j$ tiene valores propios positivos y por lo tanto también son positivos los valores propios de $(B_j^{-1}C_j)^{q_k(j)}$. Como además $R_{k,j}$ es hermítica y definida positiva entonces, por el mismo teorema, la matriz $R_{k,j}(B_j^{-1}C_j)^{q_k(j)}$ es definida positiva y como Q_j es semidefinida positiva, $S_{k,j}$ es definida positiva.

Si ahora suponemos que $q_1(j) > q_2(j)$ para cada $j = 1, 2 \ldots, r$, entonces de forma análoga a como se hizo en el Teorema 5.7 , podemos escribir la matriz $R_{1,j}^{-1} - R_{2,j}^{-1} = \sum_{i=q_2(j)}^{q_1(j)-1} (B_j^{-1}C_j)^i B_j^{-1}$ que claramente es hermítica. Como B_j es

hermítica y definida positiva y $\sum_{i=q_2(j)}^{q_1(j)-1} (B_j^{-1}C_j)^i$ tiene valores propios positivos, entonces del Teorema 1.10 (Sección 1.3), se sigue que $R_{1,j}^{-1} - R_{2,j}^{-1}$ es definida positiva, y de aquí $S_{1,j} \prec S_{2,j}$.

Por otro lado, de (5.47) cada matriz T_k se puede ver como la matriz de iteración de un método de multipartición correspondiente a las particiones P-regulares $A = R_{k,j} - S_{k,j}$, $k = 1, 2$, $1 \leq j \leq r$, y las matrices de peso $E_j = \alpha_j I$, $1 \leq j \leq r$.

Del Teorema 4.1 (Sección 4.2) si $E_j = \alpha_j I$, para $k = 1, 2$, la única partición $A = U_k - V_k$ inducida por cada matriz de iteración T_k es de la forma $U_k^{-1} = \sum_{j=1}^{r} E_j R_{k,j}^{-1}$ y $V_k = U_k - A$. Por ser $R_{k,j}^{-1}$ hermítica, la matriz U_k^{-1} es hermítica, luego también lo son U_k y V_k. También, por la forma en que se definió la matriz U_k podemos asegurar, que es definida positiva.

Además como $R_{1j}^{-1} \succ R_{2j}^{-1}$, entonces $\sum_{j=1}^{r} E_j R_{1,j}^{-1} \succ \sum_{j=1}^{r} E_j R_{2,j}^{-1}$ y por lo tanto $U_1^{-1} \succ U_2^{-1}$. Entonces, $O \preceq V_1 \prec V_2$, y entonces por el Teorema 1.11 (Sección 1.3) $\rho(T_1) < \rho(T_2) < 1$, y así la demostración queda finalizada. ■

5.4 Conclusiones

En este capítulo nos hemos centrado en el estudio de la convergencia de los métodos iterativos en dos etapas, para el caso en que la matriz de coeficientes A es hermítica y definida positiva.

Considerando los métodos iterativos tipo Jacobi por bloques, se describe el

método iterativo por bloques en dos etapas, dado por el Algoritmo 5.2.

Después, considerando la técnica de multipartición y los métodos en dos etapas, se describe el método de multipartición en dos etapas, que viene dado por el Algoritmo 5.3. Este algoritmo se puede ver como un caso especial del Algoritmo 5.2.

En primer lugar se ha analizado la convergencia del Algoritmo 5.3 cuando tanto las particiones externas como las internas son convergentes y se realizan bastantes iteraciones internas. En este caso los principales resultados vienen dados por el Teorema 5.1 y su Corolario 5.1. Los resultados de convergencia se basan en considerar todas las particiones externas iguales. Como se pone de manifiesto en este capítulo, si eso no es así, el Algoritmo 5.3 puede no converger.

A continuación se analiza la convergencia de los Algoritmos 5.2 y 5.3 para cualquier número de iteraciones internas cuando la matriz A es hermítica y definida positiva. Los principales resultados vienen dados en los Teoremas 5.2 y 5.3 y sus respectivos corolarios. El Teorema 5.2 demuestra la convergencia del Algoritmo 5.2 suponiendo que las particiones internas son P-regulares. El Teorema 5.3 demuestra la convergencia del Algoritmo 5.3 bajo condiciones similares a las del Teorema 5.2 pero exigiendo además que las matrices de peso sean de la forma $E_j = \alpha_j I$. Esto, desde el punto de vista de la paralelización del método no es bueno, pero como se pone de manifiesto en este capítulo, si no es así, el Algoritmo 5.3 puede no converger. Por tanto, este resultado ha de verse más como un resultado teórico que como un resultado para su uso experimental.

Para las versiones relajadas se ha probado la convergencia para factores de relajación entre 0 y ω_0, con $\omega_0 > 1$, bajo condiciones similares a las dadas en el caso no relajado. Los resultados obtenidos se pueden ver en los Teoremas 5.4,

5.5 y 5.6 y sus respectivos corolarios.

Por último, en los Teoremas 5.7 y 5.8, damos resultados de comparación para el radio espectral de la matriz de iteración que resulta de considerar diferentes particiones, correspondientes a los Algoritmo 5.2 y Algoritmo 5.3 respectivamente. Para ello, suponemos que el número de iteraciones internas que realiza cada bloque permanece fijo para cada iteración externa.

Capítulo 6

Precondicionadores basados en los métodos en dos etapas.

6.1 Introducción

Dado un sistema de ecuaciones lineales

$$Ax = b, \tag{6.1}$$

donde $A \in \mathbb{R}^{n \times n}$ es una matriz simétrica y definida positiva y x y b son vectores de tamaño n, uno de los métodos más eficientes para resolverlo es el *método del gradiente conjugado precondicionado.* Como ya vimos en la Sección 1.7, la idea del método del gradiente conjugado precondicionado consiste en aplicar el método del gradiente conjugado a un sistema lineal

$$\hat{A}\hat{x} = \hat{b},$$

donde $\hat{A} = SAS^T$, $\hat{x} = (S^{-1})^T x$ y $\hat{b} = Sb$. Este sistema está mejor condicionado que el sistema (6.1).

El método del gradiente conjugado se puede aplicar sin calcular explícitamente $\hat{A}$, sino resolviendo el sistema auxiliar $\mathcal{M}s = \tau$ en cada iteración, donde $\mathcal{M} = S^T S$ y $\tau = b - Ax$ es el residuo en la iteración correspondiente.

Como ya vimos en la Sección 1.7, dicho método viene dado por el siguiente algoritmo.

Algoritmo 1.2. Algoritmo del Gradiente Conjugado Precondicionado.

Dado un vector inicial $x^{(0)}$.

$\tau^{(0)} := b - Ax^{(0)}$

Resolver $\mathcal{M}s^{(0)} = \tau^{(0)}$

$p^{(0)} := s^{(0)}$

REPETIR

$\alpha := -\frac{\langle s^{(l)}, \tau^{(l)} \rangle}{\langle p^{(l)}, Ap^{(l)} \rangle}$

$x^{(l+1)} := x^{(l)} - \alpha p^{(l)}$

$\tau^{(l+1)} := \tau^{(l)} - \alpha Ap^{(l)}$

Resolver $\mathcal{M}s^{(l+1)} = \tau^{(l+1)}$

SI test de convergencia = VERDADERO

entonces FIN

$\beta := -\frac{\langle s^{(l+1)}, \tau^{(l+1)} \rangle}{\langle s^{(l)}, \tau^{(l)} \rangle}$

$p^{(l+1)} := s^{(l+1)} - \beta p^{(l)}$

Hacemos notar que las distintas técnicas de precondicionamiento permiten evitar el cálculo explícito de la matriz $\mathcal{M}$.

Una de esas técnicas, es la basada en las series truncadas de la matriz A. Esta técnica, como ya vimos en la Sección 1.8.1, consiste en considerar una

partición de la matriz A, $A = P - Q$, de tal forma que para resolver el sistema $As = \tau$, se realizan m-iteraciones del procedimiento iterativo definido por la partición anterior, empezando por $s^{(0)} = 0$. Entonces, la solución del sistema $\mathcal{M}s = \tau$ viene dada por la expresión

$$s = (I + R + R^2 + \cdots + R^{m-1})P^{-1}\tau,$$

donde $R = P^{-1}Q$, y la matriz precondicionante viene dada por la expresión

$$\mathcal{M}_m = P(I + R + R^2 + \cdots + R^{m-1})^{-1}.$$

Como vimos en el Capítulo 1, son muchos los autores que han estudiado distintos precondicionadores para el método del gradiente conjugado, desde que fue introducido por Hestenes y Stiefel [56] en 1952. Sin embargo, la mayoría de ellos se han dedicado al estudio de precondicionadores secuenciales, o bien, a la vectorización de dichos precondicionadores.

Este capítulo se centra en la parte menos estudiada de este área, los precondicionadores en paralelo. Concretamente, en la Sección 6.2, siguiendo la idea básica del precondicionamiento por series truncadas, construiremos un precondicionador basado en los métodos por bloques en dos etapas. En la Sección 6.3, se construye un precondicionador uniendo la técnica de precondicionamiento basada en la factorización incompleta de Choleski y los precondicionadores vistos en la Sección 6.2. En ambas secciones, se presentan condiciones suficientes para que dichos precondicionadores sean simétricos y definidos positivos.

6.2 Construcción del precondicionador

Sea A una matriz simétrica y definida positiva, consideramos una partición de dicha matriz de la forma

$$A = P - Q, \tag{6.2}$$

donde P es una matriz diagonal por bloques de la forma

$$P = \text{diag}(P_1, \ldots, P_r), \tag{6.3}$$

y cada P_j, $1 \leq j \leq r$, es una matriz no singular, de tamaño n_j, $\sum_{j=1}^{r} n_j = n$.

Como podemos observar, realizar m pasos del procedimiento iterativo definido por esta partición para aproximar la solución del sistema $As = \tau$, corresponde a ejecutar m pasos del método *tipo Jacobi por bloques.*

Esto es, en cada iteración l, $l = 1, 2, \ldots, m$, se necesitan resolver r sistemas linealmente independientes de la forma

$$P_j s_j^{(l)} = (Q s^{(l-1)} + \tau)_j, \quad 1 \leq j \leq r, \tag{6.4}$$

empezando con el vector $s_j^{(0)} = 0$. Claramente, cada sistema lineal (6.4) puede ser resuelto por un procesador diferente.

Cuando el tamaño de los bloques diagonales P_j, $1 \leq j \leq r$, es grande, parece lógico aproximar la solución de los sistemas (6.4), mediante otro método iterativo, entonces estamos en presencia de un *método iterativo en dos etapas* (ver Sección 1.5).

Teniendo en cuenta la descripción teórica de este método, consideramos para cada bloque P_j una partición de la forma

$$P_j = B_j - C_j, \quad 1 \leq j \leq r, \tag{6.5}$$

de tal forma que en cada iteración l, para cada j, $1 \leq j \leq r$, realizamos $q(j)$ iteraciones, que llamaremos *internas*, del procedimiento iterativo definido por esta partición, para aproximar la solución del sistema lineal (6.4).

Luego (ver Capítulo 5), precondicionar mediante la técnica de dos etapas equivale a realizar m-pasos del esquema iterativo

$$s^{(l)} = Ts^{(l-1)} + W^{-1}\tau, \quad l = 1, 2, \ldots, m, \tag{6.6}$$

eligiendo $s^{(0)} = 0$, donde

$$T = H + (I - H)P^{-1}Q, \tag{6.7}$$

$$W = P(I - H)^{-1}, \tag{6.8}$$

donde P es la matriz diagonal por bloques definida en (6.3) y

$$H = \text{diag}((B_1^{-1}C_1)^{q(1)}, \ldots, (B_r^{-1}C_r)^{q(r)}). \tag{6.9}$$

Por tanto, a partir de (6.6) podemos escribir

$$\begin{aligned} s^{(m)} &= Ts^{(m-1)} + W^{-1}\tau \\ &= T(Ts^{(m-2)} + W^{-1}\tau) + W^{-1}\tau \\ &= T^2 s^{(m-2)} + (I + T)W^{-1}\tau \\ &= T^3 s^{(m-3)} + (I + T + T^2)W^{-1}\tau \\ &= \cdots \\ &= T^m s^{(0)} + (I + T + T^2 + \cdots + T^{m-1})W^{-1}\tau. \end{aligned}$$

Luego después de m pasos, eligiendo $s^{(0)} = 0$, el vector aproximación a la solución de $As = \tau$ viene dado por la expresión

$$s^{(m)} = (I + T + T^2 + \cdots + T^{m-1})W^{-1}\tau,$$

y entonces el precondicionador que se obtiene usando esta técnica viene dado por la expresión

$$\mathcal{M}_m = W(I + T + T^2 + \cdots + T^{m-1})^{-1}, \tag{6.10}$$

con T y W matrices definidas en (6.7) y (6.8) respectivamente.

A continuación estudiamos la validez del precondicionador (6.10), dando unas determinadas condiciones a la partición externa (6.2) y a las particiones internas (6.5), para que la matriz $\mathcal{M}_m$, definida en (6.10), sea simétrica y definida positiva.

Teorema 6.1. *Sea A una matriz simétrica y definida positiva. Sea $A = P - Q$ una partición de A, donde $P = \mathrm{diag}(P_1, \ldots, P_r)$ es la matriz diagonal por bloques definida en (6.3). Supongamos que P es simétrica y Q es semidefinida positiva. Sean $P_j = B_j - C_j$, $1 \leq j \leq r$, particiones P-regulares tal que las matrices B_j son simétricas, entonces la matriz precondicionadora $\mathcal{M}_m$, definida en (6.10), es simétrica.*

Demostración. En (6.8) se definía la matriz W mediante la expresión $W = P(I - H)^{-1}$, donde la matriz H se definía en (6.9), y la matriz P, definida en (6.3), era de la forma $P = \mathrm{diag}(P_1, P_2, \ldots, P_r)$. Teniendo en cuenta la forma en que están definidas las matrices H y P, podemos escribir la matriz W^{-1} como

$$\begin{aligned} & W^{-1} \\ & = (I - H)P^{-1} \\ & = \left(I - \mathrm{diag}((B_1^{-1}C_1)^{q(1)}, \ldots, (B_r^{-1}C_r)^{q(r)})\right) P^{-1} \\ & = \mathrm{diag}\left((I - (B_1^{-1}C_1)^{q(1)})P_1^{-1}, \ldots, (I - (B_r^{-1}C_r)^{q(r)})P_r^{-1}\right) \end{aligned} \tag{6.11}$$

Ahora bien, las matrices B_j, $1 \leq j \leq r$ son no singulares. Además, puesto que las matrices P_j son simétricas y definidas positivas y las particiones $P_j = B_j - C_j$, $1 \leq j \leq r$, son P-regulares por el Teorema 1.7 (Sección 1.3) se cumple que $\rho(B_j^{-1}C_j) < 1$, y del Lema 1.1 (Sección 1.3), se sigue que la matriz $(I - B_j^{-1}C_j)$, $1 \leq j \leq r$, es no singular. Luego podemos escribir $P_j^{-1} = (B_j - C_j)^{-1} = (B_j(I - B_j^{-1}C_j))^{-1} = (I - B_j^{-1}C_j)^{-1}B_j^{-1}$, $1 \leq j \leq r$. Por tanto, a partir de la expresión (6.11) obtenemos

$$
\begin{aligned}
& W^{-1} \\
& = \operatorname{diag}\left((I - (B_1^{-1}C_1)^{q(1)})(I - B_1^{-1}C_1)^{-1}B_1^{-1}, \ldots, (I - (B_r^{-1}C_r)^{q(r)})(I - B_r^{-1}C_r)^{-1}B_r^{-1}\right) \\
& = \operatorname{diag}\left((I - (B_1^{-1}C_1)^{q(1)})\sum_{i=0}^{\infty}(B_1^{-1}C_1)^i B_1^{-1}, \ldots, (I - (B_r^{-1}C_r)^{q(r)})\sum_{i=0}^{\infty}(B_r^{-1}C_r)^i B_r^{-1}\right) \\
& = \operatorname{diag}\left(\sum_{i=0}^{q(1)-1}(B_1^{-1}C_1)^i B_1^{-1}, \ldots, \sum_{i=0}^{q(r)-1}(B_r^{-1}C_r)^i B_r^{-1}\right).
\end{aligned}
$$

Por comodidad escribimos

$$W^{-1} = \operatorname{diag}(R_1, \ldots, R_r),$$

donde

$$R_j = \sum_{i=0}^{q(j)-1}(B_j^{-1}C_j)^i B_j^{-1}, \quad 1 \leq j \leq r.$$

Por hipótesis, las matrices P_j y B_j, $1 \leq j \leq r$, son simétricas, luego podemos asegurar que la matriz $C_j = B_j - P_j$ es también simétrica. Entonces para $1 \leq j \leq r$, cada matriz $(B_j^{-1}C_j)^i B_j^{-1}$, $i = 0, 1, \ldots, q(j) - 1$, es simétrica, por tanto lo serán las matrices R_j. Luego por la forma en que está definida la matriz W^{-1}, podemos asegurar que también es simétrica.

Por otro lado, la matriz T correspondiente a la matriz de iteración de un

método por bloques en dos etapas, definida en (6.7), se puede escribir como

$$\begin{aligned} T &= H + (I - H)P^{-1}Q \\ &= H + (I - H)(I - P^{-1}A) \\ &= H + (I - P^{-1}A) - H(I - P^{-1}A) \\ &= I - (I - H)P^{-1}A. \end{aligned}$$

Teniendo en cuenta la forma en que está definida la matriz W en (6.8) obtenemos

$$T = I - W^{-1}A. \tag{6.12}$$

Luego teniendo en cuenta la expresión (6.10) y sustituyendo T por su expresión en (6.12) obtenemos

$$\begin{aligned} \mathcal{M}_m^{-1} &= (I + T + T^2 + \cdots + T^{m-1})W^{-1} \\ &= \left(I + (I - W^{-1}A) + (I - W^{-1}A)^2 + \cdots + (I - W^{-1}A)^{m-1}\right) W^{-1} \\ &= \sum_{i=0}^{m-1} (I - W^{-1}A)^i W^{-1}. \end{aligned}$$

Como podemos observar, la matriz $\mathcal{M}_m^{-1}$ es una combinación lineal de términos de la forma $(W^{-1}A)^i W^{-1}$, $i = 0, 1, \ldots, m-1$, que son matrices simétricas. Luego la matriz $\mathcal{M}_m^{-1}$ y por tanto, la matriz $\mathcal{M}_m$ son simétricas y la demostración queda finalizada. ■

El siguiente teorema demuestra que el precondicionador $\mathcal{M}_m$, definido en (6.10), es definido positivo.

Teorema 6.2. *Sea A una matriz simétrica y definida positiva. Sea $A = P - Q$ una partición de A, donde $P = \mathrm{diag}(P_1, \ldots, P_r)$ es la matriz diagonal por bloques definida en (6.3). Supongamos que P es simétrica y Q es semidefinida positiva. Sean $P_j = B_j - C_j$, $1 \leq j \leq r$, particiones P-regulares tal que B_j es simétrica, entonces la matriz precondicionadora $\mathcal{M}_m$, definida en (6.10), es definida positiva.*

Demostración. Puesto que P_j, $1 \leq j \leq r$ es simétrica y las particiones $P_j = B_j - C_j$, $1 \leq j \leq r$ son P-regulares, de la demostración del Corolario 5.2 se sigue que la matriz $W = P(I - H)^{-1}$ es definida positiva, donde la matriz H está definida por $H = \mathrm{diag}((B_1^{-1}C_1)^{q(1)}, \ldots, (B_r^{-1}C_r)^{q(r)})$. Por otro lado, de (6.10) se sigue

$$\mathcal{M}_m^{-1}W = (I + T + T^2 + \cdots + T^{m-1}), \tag{6.13}$$

con $T = I - W^{-1}A$.

Puesto que A y W^{-1} son simétricas y definidas positivas, por el Teorema 1.10 (Sección 1.3), los valores propios de $W^{-1}A$ son positivos y por lo tanto los de T son reales. Si se denota por λ un valor propio cualquiera de T, entonces los valores propios de $\mathcal{M}_m^{-1}W$ son de la forma

$$1 + \lambda + \cdots + \lambda^{m-1} = \frac{1 - \lambda^m}{1 - \lambda}. \tag{6.14}$$

Del Teorema 5.2 (Sección 5.2) y con las hipótesis de este teorema, se sigue que $\rho(T) < 1$, luego claramente la expresión (6.14) es positiva, y esto nos indica que los valores propios de $\mathcal{M}_m^{-1}W$ son positivos. Entonces por el Teorema 1.10 (Sección 1.3), como W es definida positiva podemos asegurar que la matriz simétrica $\mathcal{M}_m^{-1}$ y por lo tanto, la matriz simétrica $\mathcal{M}_m$, son definidas positivas y de esta forma la demostración queda finalizada. ■

6.3 Un precondicionador en dos etapas usando la FIC

Consideramos, sin pérdida de generalidad, que la matriz A está dividida en $r \times r$ bloques con los bloques diagonales de tamaño n_j, $\sum_{j=1}^{r} n_j = n$, de tal forma que el sistema (6.1) puede ser escrito de la forma

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1r} \\ A_{21} & A_{22} & \cdots & A_{2r} \\ \vdots & \vdots & & \vdots \\ A_{r1} & A_{r2} & \cdots & A_{rr} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_r \end{bmatrix}, \tag{6.15}$$

donde x y b son vectores que están divididos de acuerdo al tamaño de los bloques de A. Esta forma de la matriz A puede ser que se obtenga de forma natural, debido a la estructura del problema, o se puede obtener usando algún algoritmo de partición en bloques como $MARCA$, $TPABLO$, ... , (ver [78]).

Teniendo en cuenta la partición $A = P - Q$, definida en (6.2), consideramos ahora el caso particular en el que la matriz P, está formada por los bloques diagonales de la matriz A dada en (6.15), es decir, $P = \text{diag}(A_{11}, \ldots, A_{rr})$. Esta partición es conocida como *partición de Jacobi por bloques.*

Para construir el precondicionador de esta sección, consideramos para cada bloque A_{jj} una partición basada en la factorización incompleta de Choleski de una matriz simétrica y definida positiva.

Es decir, las particiones internas definidas en (6.5), se definen ahora como

$$A_{jj} = B_j - C_j \quad \text{con} \quad B_j = L_j L_j^T \quad \text{y} \quad C_j \neq O, \tag{6.16}$$

donde L_j es una matriz triangular inferior.

Sucede que no a todas las matrices simétricas y definidas positivas se les puede realizar la factorización incompleta de Choleski, por ello, para asegurar su existencia trabajaremos con unas determinadas matrices.

Sea $Z^{n\times n}$ el conjunto de todas las matrices reales de tamaño $n \times n$ que tienen los elementos fuera de la diagonal no positivos. Teniendo en cuenta estas consideraciones, el siguiente teorema da condiciones suficientes que aseguran la existencia de las particiones obtenidas mediante la factorización incompleta de Choleski y además asegura que estas particiones sean regulares. Su demostración se puede ver en [68] (Teorema 2.4).

Teorema 6.3. *Si $A \in Z^{n\times n}$ es una matriz simétrica y definida positiva, existe para cada subconjunto G de $G_n = \{(i,j) : 1 \leq i,j \leq n\}$ que tiene la propiedad de que si $(i,j) \in G$ implica que $(j,i) \in G$, una única matriz triangular inferior $L = (l_{ij})$ y una matriz simétrica no negativa $N = (n_{ij})$, con $l_{ij} = 0$ si $(i,j) \notin G$ y $n_{ij} = 0$ si $(i,j) \in G$, tal que la partición $A = LL^T - N$ es regular.*

Queremos hacer notar que el método iterativo en dos etapas que resulta de tomar como partición externa $A = P - Q$ con $P = \mathrm{diag}(A_{11}, \ldots, A_{rr})$ y $P = B_j - C_j$, obtenida como se indica en el Teorema 6.3, converge a la solución del sistema (6.15) (ver por ejemplo [19], [47] y [67]).

A continuación vamos a estudiar la validez del precondicionador en dos etapas basado en la factorización incompleta de Choleski.

Teorema 6.4. *Sea $A \in Z^{n\times n}$ una matriz simétrica y definida positiva. Sea $A = P - Q$ una partición de A, donde $P = \mathrm{diag}(A_{11}, A_{22}, \ldots, A_{rr})$. Consideramos para cada matriz A_{jj} la factorización incompleta de Choleski $A_{jj} =$*

$B_j - C_j$, $1 \leq j \leq r$, obtenida como se indica en el Teorema 6.3, entonces, la matriz precondicionadora $\mathcal{M}_m$ definida en (6.10) es simétrica.

Demostración. Como la matriz A es simétrica, claramente la matriz $P = \text{diag}(A_{11}, \ldots, A_{rr})$, también es simétrica.

Como A es definida positiva, también cada bloque diagonal $A_{jj}, j = 1, 2, \ldots, r$ es definido positivo, luego por el Lema 1.4 (Sección 1.3), las matrices A_{jj} son M-matrices no singulares, y entonces $A_{jj}^{-1} \geq O$, es decir, son monótonas.

Por otro lado, el Teorema 6.3 indica que las particiones $A_{jj} = B_j - C_j$, $1 \leq j \leq r$, son regulares y como $A_{jj}^{-1} \geq O$, $1 \leq j \leq r$, entonces por el Teorema 1.7 (Sección 1.3), se verifica que $\rho(B_j^{-1}C_j) < 1$, $1 \leq j \leq r$.

De esta forma, para todo j tal que, $1 \leq j \leq r$, podemos escribir

$$A_{jj}^{-1} = (I - B_j^{-1}C_j)^{-1}B_j^{-1}.$$

Entonces, si $H = \text{diag}((B_1^{-1}C_1)^{q(1)}, \ldots, (B_r^{-1}C_r)^{q(r)})$, escribimos

$$\begin{aligned}
W^{-1} &= (I - H)P^{-1} \\
&= \text{diag}((I - (B_1^{-1}C_1)^{q(1)})A_{11}^{-1}, \ldots, (I - (B_r^{-1}C_r)^{q(r)})A_{rr}^{-1}) \\
&= \text{diag}\left((I - (B_1^{-1}C_1)^{q(1)})(I - B_1^{-1}C_1)^{-1}B_1^{-1}, \ldots, (I - (B_r^{-1}C_r)^{q(r)})(I - B_r^{-1}C_r)^{-1}B_r^{-1}\right) \\
&= \text{diag}\left(\sum_{i=0}^{q(1)-1}(B_1^{-1}C_1)^i B_1^{-1}, \ldots, \sum_{i=0}^{q(r)-1}(B_r^{-1}C_r)^i B_r^{-1}\right).
\end{aligned}$$

Al igual que hicimos en el Teorema 6.1, escribimos la matriz W^{-1} de la forma

$$W^{-1} = \text{diag}\,(R_1, \ldots, R_r),$$

$$R_j = \sum_{i=0}^{q(j)-1}(B_j^{-1}C_j)^i B_j^{-1}, \quad 1 \leq j \leq r.$$

Por la forma en que se ha construido la partición $A_{jj} = B_j - C_j$, según el Teorema 6.3, podemos asegurar que las matrices B_j y C_j son simétricas, luego la matriz W^{-1} es simétrica.

De la misma forma que se hizo en el Teorema 6.1, la matriz $\mathcal{M}_m^{-1}$ y por lo tanto $\mathcal{M}_m$ son simétricas y así la demostración queda finalizada. ■

Teorema 6.5. *Sea $A \in Z^{n\times n}$ una matriz simétrica y definida positiva. Sea $A = P - Q$ una partición de A, donde $P = \text{diag}(A_{11}, A_{22}, \ldots, A_{rr})$. Consideramos la factorización incompleta de Choleski que resulta de considerar las particiones $A_{jj} = B_j - C_j$, $1 \leq j \leq r$, obtenidas como en el Teorema 6.3, entonces la matriz precondicionadora $\mathcal{M}_m$, definida en (6.10), es definida positiva.*

Demostración. Las matrices A_{jj} y B_j son simétricas y definidas positivas. Por el Teorema 6.3, sabemos que las particiones $A_{jj} = B_j - C_j$, $1 \leq j \leq r$, son regulares. Además, podemos asegurar por el Lema 1.4 y el Teorema 1.7 que $\rho(B_j^{-1}C_j) < 1$ y por tanto $B_j^T + C_j$, $1 \leq j \leq r$, es definida positiva.

Por tanto, razonando de forma similar a la demostración del Teorema 6.2, se obtiene que $\mathcal{M}_m$ es definida positiva y así la demostración queda finalizada. ■

6.4 Conclusiones

En este capítulo hemos construido y demostrado la validez de dos precondicionadores paralelos basados en el método en dos etapas para el método del

gradiente conjugado. Dichos precondicionadores siguen la idea del precondicionamiento por series truncadas.

Para construir el primer precondicionador, basándonos en el método en dos etapas, consideramos un partición externa de tipo Jacobi por bloques y exigimos que se realicen $q(j)$ iteraciones internas en cada bloque j, $1 \leq j \leq r$, entonces el precondicionador resultante viene dado por la expresión (6.10). Para demostrar su validez se debe demostrar que dicho precondicionador es simétrico y definido positivo. La prueba de que es simétrico se da en el Teorema 6.1 y la propiedad de ser definido positivo se da en el Teorema 6.2. En ambos teoremas se consideran que las particiones internas son P-regulares.

Para el segundo precondicionador, nos basamos también en la técnica en dos etapas. Para ello se considera la matriz de coeficientes A dividida en bloques y una partición externa Jacobi por bloques. Para obtener las particiones internas, cada procesador realiza la factorización incompleta de Choleski a su bloque diagonal correspondiente. Para asegurarnos la existencia de dicha factorización, trabajamos sólo con matrices de tamaño $n \times n$ que tienen los elementos fuera de la diagonal no positivos. En los Teoremas 6.4 y 6.5, se demuestra que el precondicionador así construido es simétrico y definido positivo.

Capítulo 7

Experimentos numéricos.

7.1 Introducción

En los Capítulos 4, 5 y 6, se han estudiado desde el punto de vista teórico, distintos métodos iterativos para la resolución de sistemas de ecuaciones lineales en paralelo. En este capítulo, nos planteamos el estudio experimental de dichos métodos para su ejecución en multiprocesadores, dando resultados numéricos que muestren su comportamiento.

Dicho estudio se va a centrar en los métodos expuestos en los Capítulos 5 y 6. Más concretamente, presentaremos resultados para los métodos por bloques en dos etapas y analizaremos el uso de éstos como precondicionadores para el método del gradiente conjugado. Hacemos notar, que en el contexto de las matrices hermíticas y definidas positivas, los métodos vistos en el Capítulo 4 tienen más interés teórico que práctico, debido a las condiciones que se exigen a

las matrices de peso. Es por esto que desde el punto de vista experimental nos centramos en los métodos de los Capítulos 5 y 6.

Previamente a la presentación de los resultados, en la Sección 7.2 daremos una breve descripción de las dos plataformas de multiprocesadores que utilizaremos para la obtención de los resultados experimentales e indicaremos las librerías y el paquete de software para el paso de mensajes que utilizamos en la ejecución. En la Sección 7.3 se presentan las distintas matrices con las que vamos a obtener los resultados numéricos. Utilizaremos principalmente una de las más conocidas matrices por banda: la matriz de Laplace. También daremos resultados usando las matrices, que hemos denominado Biarmónicas, que provienen de la discretización de una ecuación diferencial parcial elíptica que relaciona el operador Biarmónico con una función determinada y que verifica determinadas condiciones de contorno (ver, por ejemplo [88]). Todas ellas son matrices simétricas y definidas positivas. En la Sección 7.4 se realizará el estudio experimental de los métodos iterativos por bloques en dos etapas paralelos, correspondientes al Capítulo 5. En la Sección 7.5, se analizarán experimentalmente los precondicionadores paralelos desarrollados en el Capítulo 6. En primer lugar, se realizará un estudio experimental de los precondicionadores por bloques basados en la técnica en dos etapas (Subsección 7.5.2) y después se presentarán resultados para los precondicionadores por bloques en dos etapas que se basan en la factorización incompleta de Choleski para la obtención de la partición interna (Subsección 7.5.3). Por último, en la Sección 7.6 compararemos los distintos métodos que han sido motivo de estudio experimental en esta memoria, y daremos conclusiones sobre su comportamiento en general.

7.2 Multiprocesadores utilizados.

Todos los resultados experimentales que se exponen en este capítulo, se han obtenido, básicamente, en un multiprocesador de memoria distribuida: el **IBM RS/6000 SP** de la Universidad de Alicante. También daremos algunos resultados sobre un **Cluster** de Pentiums, que pertenece al grupo de Computación Paralela del Departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante.

Las caraterísticas generales de cada una de las plataformas son:

El IBM RS/6000 SP está formado por 8 nodos *thin*, conectados mediante un switch con un ancho de banda de 30-35 Mgbytes/seg., con la configuración siguiente:

- Procesador: POWER2 Super Chip 120 MHz.
- L1 cache: 128 Kilobytes instrucciones.
- RAM: 256 Mgbytes.
- Anchura del bus de memoria: 256 bits.
- Tiempo de latencia: 40 microseg.
- Velocidad del bus: 160 Mgbytes/seg.
- Sistema operativo: AIX 4.1.5.

El Cluster está formado por 6 $PC's$ conectados mediante un switch con un rendimiento pico de 100 Mgbytes/seg., con la configuración siguiente:

- Procesador: Pentium 120 MHz.
- RAM: 64 Mgbytes.
- Ancho de banda: 6.5 Mgbytes/seg.
- Sistema operativo: LINUX 2.2.5.

Como lenguaje de programación se ha utilizado el Fortran. En el IBM RS/6000 SP el XL Fortran para el sistema operativo AIX y en el cluster, el GNU Fortran.

Utilizaremos como paquete de software para paso de mensajes el PVM (ver [51] y [58]). Este paquete de software permite a una colección de ordenadores heterogéneos (secuenciales, vectoriales o paralelos) conectarse sobre una red cualquiera, para aparecer como un único recurso computacional concurrente de memoria distribuida. Podemos ver su descripción en el Capítulo 2 (Sección 2.6). Las librerías que utilizaremos para trabajar con vectores, serán las de la colección BLAS [12], mientras que para el manejo de matrices dispersas se han utilizado distintas rutinas de la librería SPARSKIT [89].

7.3 Problema modelo

En esta sección se presentarán las matrices de coeficientes A, con las que vamos a trabajar para la obtención de los resultados numéricos.

Tomaremos como problemas modelos dos tipos de matrices, la matriz que proviene de la discretización de la ecuación de Laplace y la matriz que proviene de la discretización de la ecuación Biarmónica. Todas estas matrices aparecen en multitud de problemas físicos y técnicos.

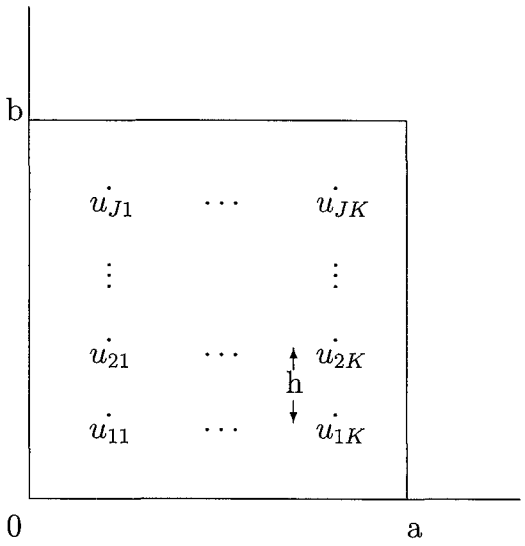
La ecuación de Laplace relaciona las parciales de segundo orden de una función de dos variables $u(x, y)$ de la forma

$$\nabla^2 u = u_{xx} + u_{yy} = 0. \tag{7.1}$$

A continuación se busca una solución en la región o dominio $\Omega = [0, a] \times [0, b]$ satisfaciendo las condiciones de contorno de *Dirichlet*, correspondientes a los valores que toma la función u en la frontera de Ω, éstas vienen dadas por las expresiones

$$\begin{aligned} u(s,0) &= u(0,t) = u(s,b) = 0, \\ u(a,t) &= 100, \qquad 0 \leq s \leq a, \quad 0 \leq t \leq b. \end{aligned} \tag{7.2}$$

Para obtener una aproximación a la ecuación (7.1) se puede utilizar el método de diferencias finitas. En este método se discretiza el rectángulo Ω con una malla de puntos separados a la misma distancia h de la forma



Reemplazando las derivadas por cocientes de diferencias en cada nodo (x_i, y_i) en que se divide la región, se obtiene

$$\begin{aligned} \nabla^2 u(x_i, y_i) &= \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} \\ &+ \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{h^2}. \end{aligned} \tag{7.3}$$

Si llamamos $u(x_i, y_j) = u_{ij}$ obtenemos

$$\nabla^2 u(x_i, y_i) = \frac{1}{h^2}[u_{i+1,j} + u_{i-1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}] = 0.$$

Entonces se obtiene el sistema lineal de ecuaciones

$$Ax = b,$$

donde

$$x = (u_{11}, \ldots, u_{1K}, u_{21}, \ldots, u_{2K}, \ldots, u_{J1}, \ldots, u_{JK})^T,$$

A es una matriz tridiagonal por bloques que tiene $J \times J$ bloques de tamaño $K \times K$, es decir, A tiene la estructura

$$A = \begin{bmatrix} B & -I & & & \\ -I & B & -I & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & B \end{bmatrix}, \tag{7.4}$$

con los bloques identidad I de tamaño $K \times K$, y la matriz tridiagonal B de tamaño $K \times K$ viene dada de la forma

$$B = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{bmatrix}. \tag{7.5}$$

Como podemos observar, claramente la matriz A es hermítica. Además A es definida positiva.

El vector b contiene los valores conocidos de la función u en la frontera del rectángulo Ω.

Para el problema de la matriz de Laplace, elegimos matrices A definidas en (7.4) y B (bloques diagonales de A, definidos en (7.5)) de distintos tamaños. En la Tabla 1, se pueden ver los distintos tamaños de la matriz A, el orden de las matrices diagonales B, el número de bloques B, el número de procesadores con el que resolveremos el problema en cuestión y el número de filas con las que trabaja cada procesador, es decir, el tamaño de cada matriz A_{jj}, $j = 1, 2, \ldots, r$,

donde r es el número de procesadores. Notemos que, en esta tabla, realmente estamos indicando tres matrices distintas de tamaño 4096, en función del orden y número de bloques B. Para distinguir a estas matrices utilizaremos la siguiente notación:

Notación	Orden B	Número bloques B
4096-N	64	64
4096-S	32	128
4096-B	128	32

Notemos que la matriz 4096-S tiene un ancho de banda pequeño cuando se compara con la matriz 4096-N, mientras que la matriz 4096-B podríamos decir que posee un ancho de banda mayor.

En la Tabla 1, en la columna *Filas/procesador*, las expresiones que aparecen de tipo exponencial, indican en el exponente el número de procesadores que trabajan con el número de filas que se da en la base; por ejemplo, 1344^2, expresa que dos procesadores trabajan con 1344 filas cada uno. Utilizaremos una notación similar en el resto de este capítulo.

El siguiente problema modelo utilizado, puede ser relacionado con el cálculo de la desviación o deformación $u(x, y)$, de una placa cuadrada, sujetada por sus cuatro lados, cuando se le ejerce una presión $p(x, y)$ en cada punto (x, y). Esta desviación puede ser modelada mediante una ecuación diferencial parcial elíptica que relaciona el operador Biarmónico

$$\nabla^4 = \frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4},$$

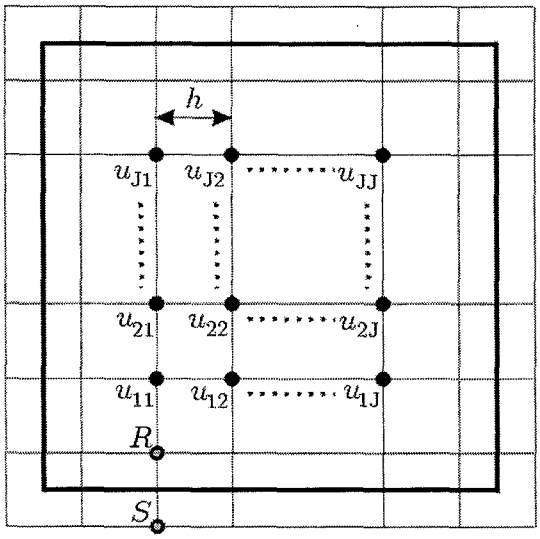
con la función que determina la presión p, satisfaciendo condiciones de contorno de *Dirichlet* y de *Neumann*. Más concretamente la desviación $u(x, y)$ verifica:

$$\begin{aligned} \nabla^4 u &= p, && \text{en el interior,} \\ u = 0,\ \frac{\partial u}{\partial \vec{v}} &= 0, && \text{en la frontera.} \end{aligned} \tag{7.6}$$

Notemos que $\vec{v}$ usualmente hace referencia a un vector unitario perpendicular a la frontera del dominio y dirigido hacia el exterior. Recordemos que la derivada direccional $\dfrac{\partial u}{\partial \vec{v}}$ se define como:

$$\frac{\partial u}{\partial \vec{v}}(x) = \lim_{h \to 0} \frac{u(x + h\vec{v}) - u(x)}{h}.$$

Para obtener una solución aproximada de la ecuación (7.6) usaremos, como en el caso de la ecuación de Laplace, el método de diferencias finitas. En este caso, para explotar las condiciones de contorno, se construye una malla de manera que la frontera del cuadrado no coincide con ninguna línea de la malla sino que las bisecta, tal y como se puede ver en la siguiente figura:



Si consideramos R y S dos puntos genéricos de la malla, vecinos de la frontera del cuadrado, es fácil ver que la condición de contorno $u = 0$ en la frontera, implica $u(R) + u(S) = 0$; además la condición de Neumann $\dfrac{\partial u}{\partial \vec{v}} = 0$ nos lleva a la conclusión de que $u(R) - u(S) = 0$. Por tanto, $u(R) = u(S) = 0$, es decir, la función u toma valor nulo en todos los puntos de la malla más próximos a la frontera del dominio. Por tanto, obtenemos $J \times J$ puntos (x_i, y_j) en donde debemos aplicar la ecuación diferencial (7.6) para obtener los valores $u_{ij} = u(x_i, x_j),\ i, j = 1, 2, \ldots, J$.

Si aplicamos dos veces la aproximación utilizada para el operador de Laplace en (7.3), obtenemos que

$$\begin{aligned} h^4\nabla^4 u_{ij} &= u_{i-2,j} \\ &+ 2u_{i-1,j-1} - 8u_{i-1,j} + 2u_{i-1,j+1} \\ &+ u_{i,j-2} - 8u_{i,j-1} + 20u_{i,j} - 8u_{i,j+1} + u_{i,j+2} \\ &+ 2u_{i+1,j-1} - 8u_{i+1,j} + 2u_{i+1,j+1} \\ &+ u_{i+2,j} \\ &= h^4 p_{ij} \end{aligned} \tag{7.7}$$

donde p_{ij} corresponde al valor de p en los puntos (x_i, y_j).

Las ecuaciones (7.7) dan lugar a un sistema de $J \times J$ ecuaciones lineales

$$Ax = b,$$

donde

$$\begin{aligned} x &= (u_{11}, u_{12}, \ldots, u_{1J}, u_{21}, u_{22}, \ldots, u_{2J}, \ldots, u_{J1}, u_{J2}, \ldots, u_{JJ})^T, \\ b &= h^4(p_{11}, p_{12}, \ldots, p_{1J}, p_{21}, p_{22}, \ldots, p_{2J}, \ldots, p_{J1}, p_{J2}, \ldots, p_{JJ})^T, \end{aligned}$$

y A es una matriz pentadiagonal por bloques de la forma

$$A = \begin{bmatrix} C & D & I & & & & & \\ D & C & D & I & & & & \\ I & D & C & D & I & & & \\ & I & D & C & D & I & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & I & D & C & D \\ & & & & & I & D & C \end{bmatrix},$$

con I la matriz identidad y C, D matrices de tamaño $J \times J$ definidas como

$$C = \begin{bmatrix} 20 & -8 & 1 & & & & & \\ -8 & 20 & -8 & 1 & & & & \\ 1 & -8 & 20 & -8 & 1 & & & \\ & 1 & -8 & 20 & -8 & 1 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & 1 & -8 & 20 & -8 \\ & & & & & 1 & -8 & 20 \end{bmatrix},$$

$$D = \begin{bmatrix} -8 & 2 & & & & & & \\ 2 & -8 & 2 & & & & & \\ & 2 & -8 & 2 & & & & \\ & & 2 & -8 & 2 & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & 2 & -8 & 2 \\ & & & & & & 2 & -8 \end{bmatrix}.$$

En la Tabla 2, se presentan las características de las matrices Biarmónicas que usaremos en la exposición de los distintos resultados numéricos, es decir, los distintos tamaños de la matriz A, el orden de las matrices diagonales C, el número de procesadores utilizado y el número de filas con las que trabaja cada procesador.

Orden A	Orden B	Número bloques B	Número procesad.	Filas/ procesador
1024	32	32	2	512^2
			4	256^4
4096	64	64	2	2048^2
	32	128	3	1344^2, 1408
	128	32	4	1024^4
10000	100	100	2	5000^2
			3	3300^2, 3400
			4	2500^4
			5	2000^5
16384	128	128	2	8192^2
			3	5376^2, 5632
			4	4096^4
			4	6400, 3328^3
			5	3584, 3200^4
40000	200	200	2	20000^2
			3	13200^2, 13600
			4	10000^4
			5	8000^5
50000	500	100	2	25000^2
			4	12500^4
262144	512	512	2	131072^2
			3	87040^2, 88064

Tabla 1: *Matrices de prueba de Laplace.*

Orden A	Orden C, D	Número bloques C	Número procesad.	Filas/ procesador
1024	32	32	2	512^2
			4	256^4
4096	64	64	2	2048^2
			4	1024^4
10000	100	100	2	5000^2
			4	2500^4
16384	128	128	2	8192^2
			4	4096^4
40000	200	200	2	20000^2

Tabla 2: *Matrices de prueba de la ecuación Biarmónica.*

7.4 Implementación de los métodos iterativos paralelos en dos etapas

7.4.1 Introducción

En esta sección vamos a describir el algoritmo paralelo con el que obtendremos resultados numéricos que nos muestren el comportamiento del método por bloques en dos etapas visto en el Capítulo 5.

Para describir las particiones que utilizaremos, consideramos que la matriz A está dividida en bloques, con los bloques diagonales de tamaño n_j, $\sum_{j=1}^{r} n_j = n$, tal que el sistema $Ax = b$ se puede escribir como

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1r} \\ A_{21} & A_{22} & \cdots & A_{2r} \\ \vdots & \vdots & & \vdots \\ A_{r1} & A_{r2} & \cdots & A_{rr} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_r \end{bmatrix}, \tag{7.8}$$

donde x y b son vectores que están divididos de acuerdo al tamaño de los bloques de A.

En el Teorema 5.2 se supone que en la partición $A = M - N$, la matriz N es semidefinida positiva. Para asegurar esta condición en la práctica, podemos elegir la siguiente partición de A:

- Sea $A = \tilde{M} - \tilde{N}$ la partición de Jacobi por bloques de la matriz A, es decir, $\tilde{M}$ es de la forma

$$\tilde{M} = \mathrm{diag}(A_{11}, \ldots, A_{rr}). \tag{7.9}$$

- Consideramos una matriz cuadrada diagonal no negativa D, de tamaño n, definida de la forma

$$D = \text{diag}\left(\sum_{j=1, j\neq 1}^{n} |\tilde{n}_{1j}|, \ldots , \sum_{j=1, j\neq n}^{n} |\tilde{n}_{nj}| \right).$$

- Construimos la partición $A = M - N$, donde

$$M = \text{diag}(A_{11}, \ldots , A_{rr}) + D = \text{diag}(M_1, \ldots , M_r), \qquad (7.10)$$

$$N = \tilde{N} + D. \qquad (7.11)$$

La partición así construida satisface las hipótesis del Teorema 5.2 (Sección 5.2), pues la matriz M es hermítica y la matriz N es semidefinida positiva.

A continuación se define el algoritmo síncrono para la ejecución en paralelo del método por bloques en dos etapas.

Algoritmo 7.1. Algoritmo paralelo síncrono del método por bloques en dos etapas.

Matriz de coeficientes:	A.
Término independiente:	b.
Factores no estacionarios	$q(l, j),\ 1 \leq j \leq r,\ l = 0, 1, 2 \ldots .$

Etapa 1 $\rightsquigarrow$ Elegir el vector inicial $x^{(0)}$ y $l = 0$.

Etapa 2 $\rightsquigarrow$ Etapa interna:

Para cada $j = 1, 2, \ldots, r$ hacer

$$z_j = (Nx^{(l)} + b)_j.$$

Tomar $y_j^{(0)} = x_j^{(l)}$.

Realizar $q(l, j)$ etapas de un método iterativo (ver Nota-1) para obtener una solución aproximada del sistema

$$M_j y_j = z_j.$$

Etapa 3 $\rightsquigarrow$ Etapa externa:

$$\begin{aligned} x_j^{(l+1)} &= y_j^{(q(l,j))}. \\ x^{(l+1)} &= (x_1^{(l+1)}, \ldots, x_r^{(l+1)}). \end{aligned}$$

Etapa 4 $\rightsquigarrow$ Criterio de convergencia:

$$\|x^{(l+1)} - x^{(l)}\|_1 < \epsilon$$

- Si test de convergencia = VERDADERO, entonces FIN.
- Si test de convergencia = FALSO, entonces

  $l \longleftarrow l + 1$.

  Volver a la etapa 2.

Nota-1: En el cálculo de las iteraciones internas, es decir, para obtener la aproximación a la solución del sistema $M_j y_j = z_j$ utilizaremos distintos métodos iterativos: Gauss-Seidel, SOR y SSOR.

La parte principal del algoritmo la constituye la Etapa 2, que la denominaremos *etapa no estacionaria*, donde se realizan $q(l, j)$, $1 \leq j \leq r$, etapas de un método iterativo para aproximar la solución del sistema $M_j y_j = z_j$, para obtener

el vector $x_j^{(l+1)}$. Una vez que todos los procesadores han terminado sus cálculos, se combinan en un mismo instante (fase síncrona), para formar el nuevo vector iterado global $x^{(l+1)}$.

Los factores $q(l,j),\ 1 \leq j \leq r,\ l = 0,1,2\ldots$, indican el número de iteraciones internas mediante las que el bloque j-ésimo de vector iterado, es actualizado en una iteración externa l. Generalmente, ejecutaremos para cada bloque un número fijo de iteraciones internas, es decir, $q(l,j) = q,\ j = 1,2,\ldots,r,\ l = 0,1,2\ldots$, aunque también mostraremos resultados teniendo en cuenta otras consideraciones en los factores no estacionarios.

Enumeramos a continuación, las matrices con las que trabajaremos en el resto de esta sección, el vector inicial $x^{(0)}$, el término independiente considerado y el criterio de parada usado:

- Matriz de coeficientes A:

  Laplace : tamaños 4096, 10000, 16384, 40000 y 50000 (ver Tabla 1).

  Biarmónica: tamaños 4096 y 10000 (ver Tabla 2).

- Vector inicial:

  Laplace: $x^{(0)} = (0,\ldots,0)^T$.

  Biarmónica: $x^{(0)} = (0,\ldots,0)^T$.

- Término independiente:

  Laplace: $b = (b_1^T, b_2^T, \ldots, b_J^T)^T,\ b_i \in \mathbb{R}^K,\ b_i = (0,\ldots,100)^T$.

  Biarmónica: $b = (1,\ldots,1)^T$.

- Criterio de parada:

Laplace: $\|x^{(l+1)} - x^{(l)}\|_1 < \epsilon,\ \epsilon = 10^{-7}, 10^{-4}, 10^{-2}$.

Biarmónica: $\|x^{(l+1)} - x^{(l)}\|_1 < \epsilon,\ \epsilon = 10^{-2}$.

En el resto de esta sección realizaremos un análisis exhaustivo de los métodos por bloques en dos etapas, centrándonos en los resultados obtenidos para las matrices de Laplace. Finalizaremos dando algunos resultados para las matrices Biarmónicas. En estos últimos resultados se observará que los tiempos de ejecución y número de iteraciones necesarios para alcanzar convergencia son comparativamente elevados, motivo por el cual no se realiza un estudio exhaustivo del comportamiento de estas matrices sobre los métodos por bloques en dos etapas. Por el contrario, cuando estudiemos el uso de los métodos en dos etapas como precondicionadores paralelos, sí mostraremos resultados para las matrices Biarmónicas, ya que, como veremos, en este caso, el comportamiento es más eficiente.

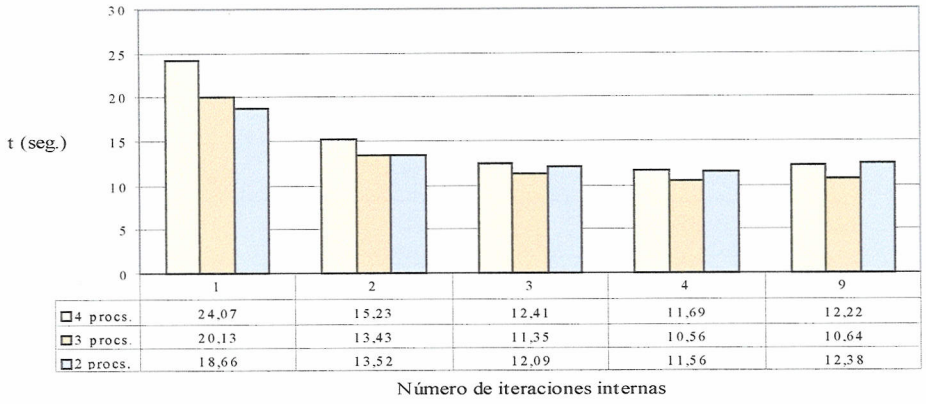
7.4.2 Comportamiento de los métodos usando iteraciones internas tipo Gauss-Seidel

A continuación, presentaremos los resultados numéricos obtenidos en el IBM RS/6000 SP, del método por bloques en dos etapas para matrices de Laplace. En concreto, para las tres matrices de tamaño 4096, para la matriz de tamaño 10000 y para la matriz de tamaño 16384. La estructura de dichas matrices se ha explicado en la Sección 7.3.

En esta subsección se ha utilizado para aproximar los sistemas internos el método iterativo de Gauss-Seidel, es decir, el método SOR con $\omega = 1$.

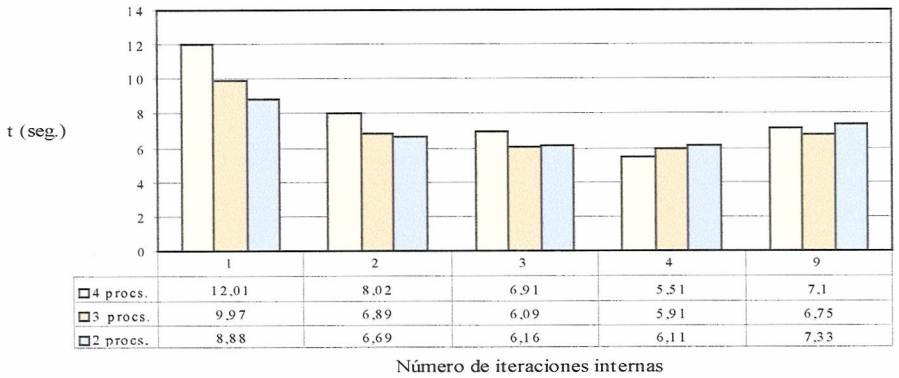
Concretamente, en la Tabla 3, se presentan resultados del Algoritmo 7.1 para las matrices de tamaños 10000 y 16384, y para la matriz 4096-N, tomando como cota de parada $\epsilon = 10^{-2}$, y usando $r = 2$, 3 y 4 procesadores. Recordamos que esta última matriz de tamaño 4096 tiene 64 bloques B de tamaño 64. En las Figuras 1(a), 1(b) y 2 se comparan los tiempos paralelos obtenidos al aplicar el método a las tres matrices de Laplace de tamaño 4096, con distintos anchos de banda. La Figura 1(a) muestra los resultados numéricos para la matriz 4096-N, la Figura 1(b) muestra los resultados para la matriz 4096-B (32 bloques B de tamaño 128) y la Figura 2 corresponde a la matriz 4096-S (que tiene 128 bloques B de tamaño 32). Para cada una de las matrices se han obtenido los resultados para 2, 3 y 4 procesadores y la cota de parada elegida en este caso ha sido $\epsilon = 10^{-7}$.

Como se puede observar en la Tabla 3, el método tiene un comportamiento similar para las distintas matrices, en el sentido de que el número de iteraciones globales decrece conforme aumentamos las iteraciones internas en cada bloque. Por tanto, siempre que esta reducción de iteraciones globales compense la realización de más iteraciones internas en cada bloque, se observará un menor tiempo de ejecución. Este hecho es independiente del número de procesadores usados. Así vemos, en la Tabla 3, que los tiempos se reducen al aumentar el número de iteraciones internas q, pero sólo hasta un cierto valor, en el cual los tiempos comienzan a subir. Es a partir de este valor crítico en donde la reducción en el número de iteraciones globales no compensa la realización de más iteraciones internas. En esta tabla, se observa también, un ligero aumento del valor del q



	1	2	3	4	9
4 procs.	24,07	15,23	12,41	11,69	12,22
3 procs.	20,13	13,43	11,35	10,56	10,64
2 procs.	18,66	13,52	12,09	11,56	12,38

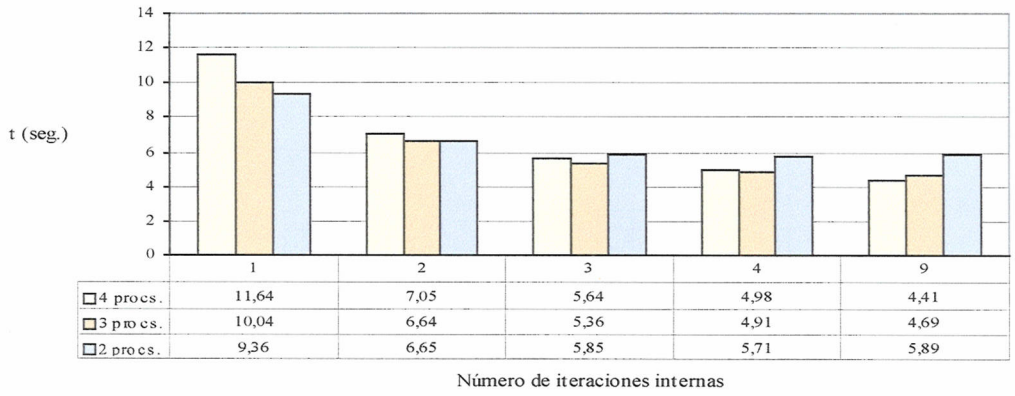
(a) 64 bloques B de tamaño 64 (4096-N).



	1	2	3	4	9
4 procs.	12,01	8,02	6,91	5,51	7,1
3 procs.	9,97	6,89	6,09	5,91	6,75
2 procs.	8,88	6,69	6,16	6,11	7,33

(b) 32 bloques B de tamaño 128 (4096-B).

Figura 1: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matrices de Laplace 4096, usando 2, 3 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-7}$.



	1	2	3	4	9
4 procs.	11,64	7,05	5,64	4,98	4,41
3 procs.	10,04	6,64	5,36	4,91	4,69
2 procs.	9,36	6,65	5,85	5,71	5,89

Figura 2: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 4096-S (128 bloques B de tamaño 32), usando 2, 3 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-7}$.

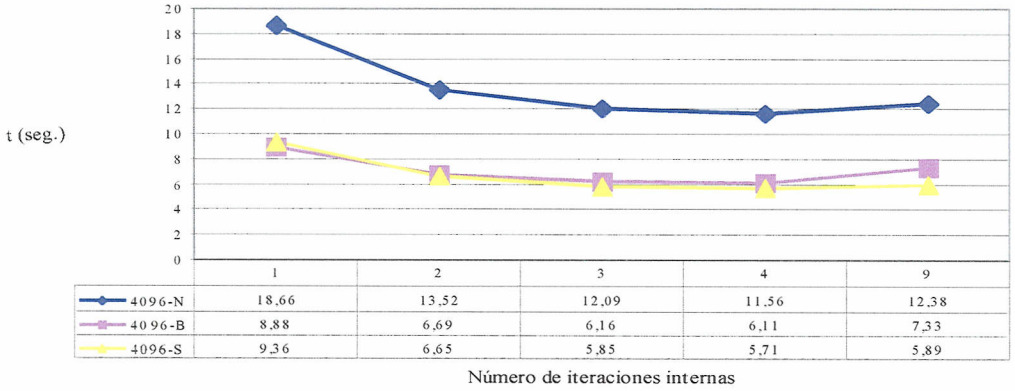
óptimo conforme aumenta el número de procesadores. Podemos pensar, que esto es debido a que generalmente, cuando aumentamos el número de procesadores, los bloques que se asignan a cada procesador son más pequeños y por tanto cada iteración interna requiere de menos cálculo, lo que compensará la realización de más iteraciones internas y disminuirá el número de comunicaciones.

Si ahora nos fijamos en los resultados obtenidos para la matriz 4096-N en la Tabla 3, y en la Figura 1(a), en la cual se ha exigido una cota de parada más fuerte, observamos que la disminución de los tiempos al reducir la exigencia del criterio de parada es considerable. Además con ambas cotas de parada, cuando $q > 1$, los tiempos disminuyen generalmente de considerar 2 a 3 procesadores para estas matrices, sin embargo los tiempos empiezan a aumentar cuando consideramos 4 procesadores. Esto es debido a que podemos considerar las matrices

de Laplace de tamaño 4096 pequeñas frente a un uso de 4 procesadores. En este caso, el tiempo de comunicación predomina comparado con el tiempo de cálculo. Debemos sin embargo observar, como ilustran las Figuras 1(b) y 2, que para matrices relativamente pequeñas, este hecho, en determinados casos, se invierte a medida que aumentamos el número de iteraciones internas, consiguiendo que el mejor tiempo se obtenga usando 4 procesadores. Así, por ejemplo, para la matriz 4096-S el mejor tiempo, 4,41 segundos, se consigue al considerar 4 procesadores con $q = 9$, frente a los 4,69 segundos al usar 3 procesadores y 5,89 segundos al usar 2 procesadores. Para la matriz 4096-B el mejor tiempo, 5,51 segundos, se consigue al considerar 4 procesadores con $q = 4$. Por el contrario, si $q = 1$, el tiempo aumenta con el número de procesadores utilizados al considerar matrices de este tamaño.

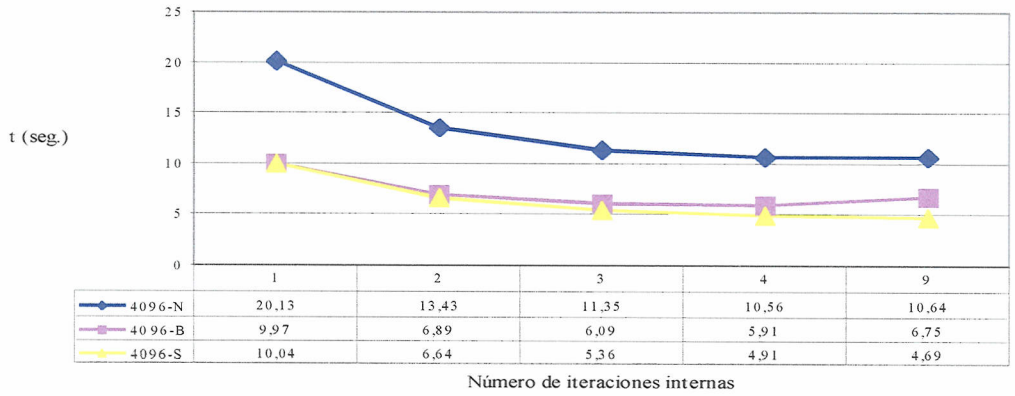
Por otro lado, atendiendo al ancho de banda de las matrices, si comparamos los tiempos obtenidos para cada una de las matrices de tamaño 4096 elegidas, los mejores tiempos se consiguen cuando se toman 128 bloques de tamaño 32, es decir, cuando tenemos muchos bloques B pequeños. En este caso los bloques M_j definidos en (7.10) poseen más información que cuando el ancho de banda es mayor. Así, es de esperar que las aproximaciones a las soluciones de los sistemas $M_j y_j = z_j$ aporten más información y, por lo tanto, se obtengan mejores resultados al aumentar el número de iteraciones internas. Esto queda reflejado, de forma clara, en la Figura 3.

En el resto de apartados de esta sección, realizaremos un estudio más exhaustivo de los métodos por bloques en dos etapas, atendiendo a los distintos factores que afectan al problema, pero antes de esto realizaremos ciertas consideraciones sobre la elección de la cota de parada en el resto de resultados numéricos.



	1	2	3	4	9
4096-N	18,66	13,52	12,09	11,56	12,38
4096-B	8,88	6,69	6,16	6,11	7,33
4096-S	9,36	6,65	5,85	5,71	5,89

(a) 2 procesadores.



	1	2	3	4	9
4096-N	20,13	13,43	11,35	10,56	10,64
4096-B	9,97	6,89	6,09	5,91	6,75
4096-S	10,04	6,64	5,36	4,91	4,69

(b) 3 procesadores.

Figura 3: *Método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Relación de tiempos entre matrices de Laplace 4096, con distintos anchos de banda, usando 2 y 3 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-7}$.

		Matriz de Laplace 4096-N		Matriz de Laplace 10000		Matriz de Laplace 16384	
r	q	Iter.	T.Par.	Iter.	T.Par.	Iter.	T.Par.
2	1	4337	8,52	10309	39,96	16714	99,88
	2	2412	6,39	5658	31,78	9126	80,56
	3	1741	5,81	4027	29,65	6457	75,33
	4	1400	**5,62**	3190	29,05	5085	73,8
	5	1194	5,63	2680	**29,03**	4246	**73,67**
	6	1056	5,72	2338	29,39	3681	74,17
	9	828	6,16	1762	31,31	2727	78,21
3	1	4428	9,13	10449	38,26	16894	89,88
	2	2506	6,31	5803	27,84	9312	67,11
	3	1840	5,43	4179	24,86	6654	60,9
	4	1503	5,13	3350	23,83	5291	58,1
	5	1300	**5,01**	2846	**23,55**	4461	**57,52**
	6	1164	5,01	2508	23,63	3902	57,59
	9	941	5,27	1941	25,03	2960	60,54
4	1	4513	10,75	10581	40,82	17062	92,39
	2	2593	7,01	5939	28,02	9485	64,21
	3	1932	5,85	4324	24,11	6837	56,21
	4	1599	5,41	3501	22,53	5484	52,58
	5	1399	5,14	3003	21,87	4661	51,39
	6	1266	**5,07**	2670	**21,72**	4108	**51,08**
	9	1048	5,25	2113	22,83	3180	53,16

Tabla 3: *Método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Resultados para distintas matrices de Laplace usando 2, 3 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

7.4.3 Sobre la elección de la cota de parada

En todos los experimentos presentados hasta el momento en la ejecución del Algoritmo 7.1, se ha utilizado como criterios de parada $\|x^{(l+1)} - x^{(l)}\|_1 < \epsilon$, con $\epsilon = 10^{-2}$ y $\epsilon = 10^{-7}$. Como comentamos en la Subsección 7.4.2, al debilitar la cota de parada se produce una reducción importante en el tiempo de ejecución. Nos interesa entonces, estudiar cómo se comporta el residuo conforme se debilita dicha cota. La Tabla 4 ilustra dicho comportamiento para la matriz 4096-S usando 3 procesadores. En esta tabla se comparan los tiempos con diferentes cotas de parada en el IBM RS/6000 SP y se calcula en cada caso la norma euclidea del residuo $\tau = Ax - b$, en la iteración de parada. Como se puede apreciar, los tiempos se reducen de considerar la cota $\epsilon = 10^{-7}$ a considerar la cota $\epsilon = 10^{-4}$ en un 24, 12% de media, y en un 50, 24% de media, si consideramos la cota $\epsilon = 10^{-2}$, y los errores pueden seguir considerándose aceptables.

	$\epsilon = 10^{-7}$		$\epsilon = 10^{-4}$		$\epsilon = 10^{-2}$		% Dif.	% Dif.
q	T.Par.	$\|\|\tau\|\|_2$	T.Par.	$\|\|\tau\|\|_2$	T.Par.	$\|\|\tau\|\|_2$	$10^{-7}/10^{-4}$	$10^{-7}/10^{-2}$
1	10,04	4E-9	8,42	3E-6	4,97	3E-4	16,13%	50,49%
2	6,64	2E-9	5,29	2E-6	3,28	2E-4	20,33%	50,6%
3	5,36	1E-9	4,31	1E-6	2,74	1E-3	19,58%	48,88%
9	4,69	1E-9	3,58	1E-6	2,45	1E-4	23,66%	47,76%
10	6,21	1E-9	3,67	1E-6	2,89	1E-4	40,91%	53,46%

Tabla 4: *Método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Comparando distintas cotas de parada para una matriz de Laplace 4096-S, usando 3 procesadores.*

Como uno de nuestros objetivos es comparar este método con el método del gradiente conjugado precondicionado, hacemos la siguiente reflexión: en el método del gradiente conjugado precondicionado, vamos a utilizar como criterio

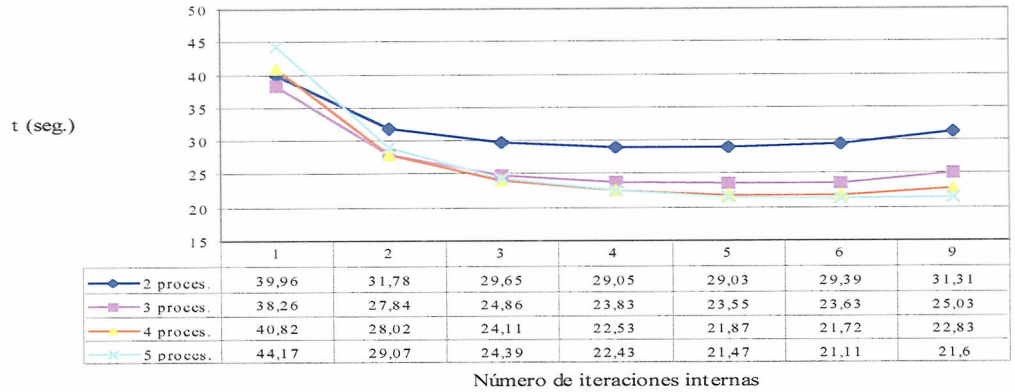
	$\epsilon = 10^{-7}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-2}$
q	Error	Error	Error
1	2E-8	1E-6	1E-4
2	3E-9	3E-7	7E-5
3	9E-9	4E-7	5E-5
9	5E-9	2E-7	4E-5
10	6E-9	3E-7	5E-5

Tabla 5: *Comparación de distintas cotas de parada atendiendo al residuo para el método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 16384, usando 2 procesadores.*

de parada $\tau^T\tau < \epsilon$, con $\epsilon = 10^{-7}$, donde $\tau = Ax - b$, corresponde al residuo que se produce en la iteración de parada. Por otro lado, como puede observarse en las Tablas 4 y 5, si $\epsilon = 10^{-7}$, se produce un residuo τ tal que $\tau^T\tau$, es del orden de 10^{-17}, independientemente del tamaño de la matriz. Para poder realizar la comparación de los dos métodos tendremos que debilitar, por tanto, la cota de parada actual hasta conseguir $\tau^T\tau < 10^{-7}$. Esto se consigue tomando como cota de parada en el método por bloques en dos etapas $\epsilon = 10^{-2}$.

7.4.4 Evolución de los tiempos atendiendo al número de procesadores y al tamaño de la matriz

En la Subsección 7.4.2 se ha visto que para matrices de tamaños pequeños (digamos, por ejemplo, de tamaño 4096), generalmente, no se consigue acelerar demasiado el método de usar 2, a 3 ó 4 procesadores. Nuestro objetivo en este apartado es ver cómo evolucionan los tiempos, para una matriz de un determina-

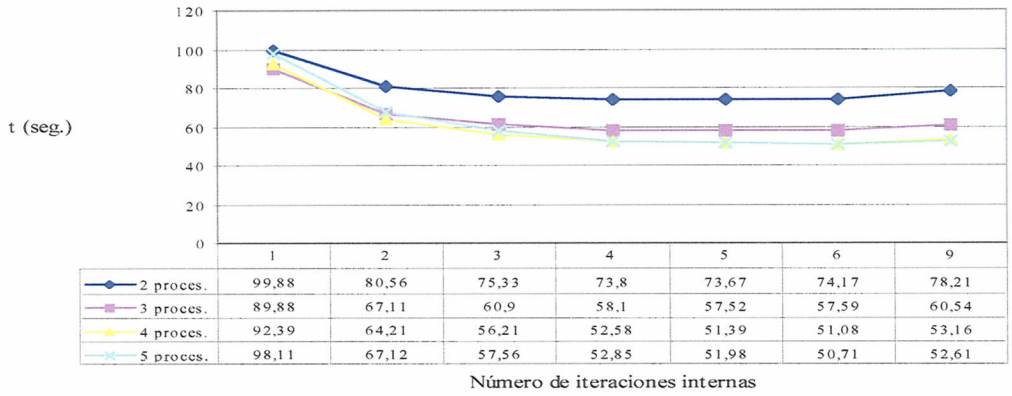


	1	2	3	4	5	6	9
2 proces.	39,96	31,78	29,65	29,05	29,03	29,39	31,31
3 proces.	38,26	27,84	24,86	23,83	23,55	23,63	25,03
4 proces.	40,82	28,02	24,11	22,53	21,87	21,72	22,83
5 proces.	44,17	29,07	24,39	22,43	21,47	21,11	21,6

Figura 4: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 10000, usando 2, 3, 4 y 5 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

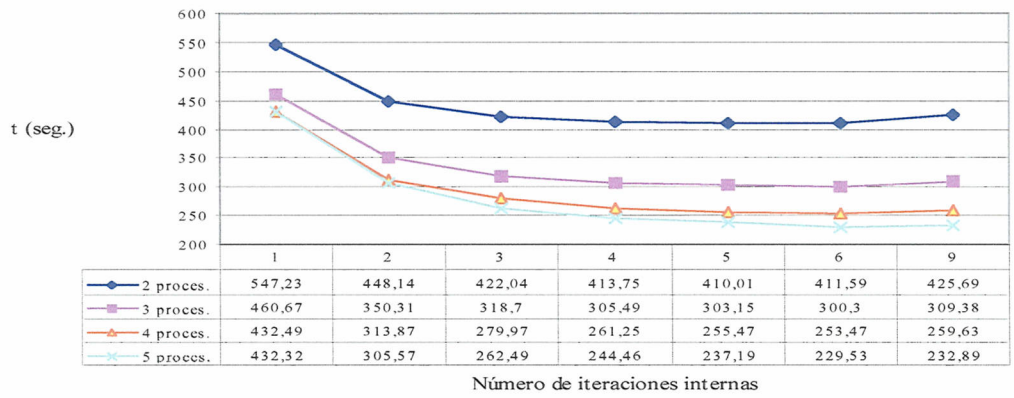
do tamaño, atendiendo al número de procesadores. Para esto, en las Figuras 4, 5(a) y 5(b) presentamos los tiempos de convergencia obtenidos para distintos números de iteraciones internas, para las matrices de tamaño 10000, 16384 y 40000, respectivamente, atendiendo al número de procesadores utilizado.

Como se puede observar, si $q = 1$, el mejor tiempo se obtiene utilizando 3 procesadores para las matrices de tamaño 10000 y 16384, y para la matriz 40000 usando 5 procesadores. Conforme aumenta el número de iteraciones internas, el tamaño de la matriz influye en la elección del número de procesadores a utilizar. Así, para las matrices de tamaño 10000 y 16384, los mejores tiempos son 21,11 segundos y 50,71 segundos respectivamente, que se obtienen con 5 procesadores y $q = 6$, pero la reducción de tiempo frente al uso de 4 procesadores es prácticamente inapreciable (0,8%). La reducción en tiempo frente al uso de



	1	2	3	4	5	6	9
2 proces.	99,88	80,56	75,33	73,8	73,67	74,17	78,21
3 proces.	89,88	67,11	60,9	58,1	57,52	57,59	60,54
4 proces.	92,39	64,21	56,21	52,58	51,39	51,08	53,16
5 proces.	98,11	67,12	57,56	52,85	51,98	50,71	52,61

(a) Tamaño de la matriz: 16384.



	1	2	3	4	5	6	9
2 proces.	547,23	448,14	422,04	413,75	410,01	411,59	425,69
3 proces.	460,67	350,31	318,7	305,49	303,15	300,3	309,38
4 proces.	432,49	313,87	279,97	261,25	255,47	253,47	259,63
5 proces.	432,32	305,57	262,49	244,46	237,19	229,53	232,89

(b) Tamaño de la matriz: 40000.

Figura 5: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matrices de Laplace 16384 y 40000, usando 2, 3, 4 y 5 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

2 procesadores está en torno al 27% para la matriz de tamaño 10000 y al 30% para la matriz de tamaño 16384. Sin embargo, para la matriz de tamaño 40000 el mejor tiempo, 229,53 segundos obtenido también con 5 procesadores y $q = 6$, consigue rebajar el tiempo en un 10% respecto al uso de 4 procesadores y en un 45% respecto al uso de 2 procesadores. Además se observa que a medida que el número de iteraciones internas crece, se hace más significativa la reducción en tiempo a medida que aumentamos el número de procesadores utilizados; la razón es obvia, un incremento en el número de iteraciones internas hace que el tiempo de cálculo crezca frente al de comunicación, que disminuye.

De las figuras anteriores también se deduce que conforme aumentamos el tamaño de la matriz el método iterativo se hace más lento. La Figura 6 ilustra en qué medida los tiempos crecen al aumentar el tamaño de la matriz manteniendo el mismo número de procesadores.

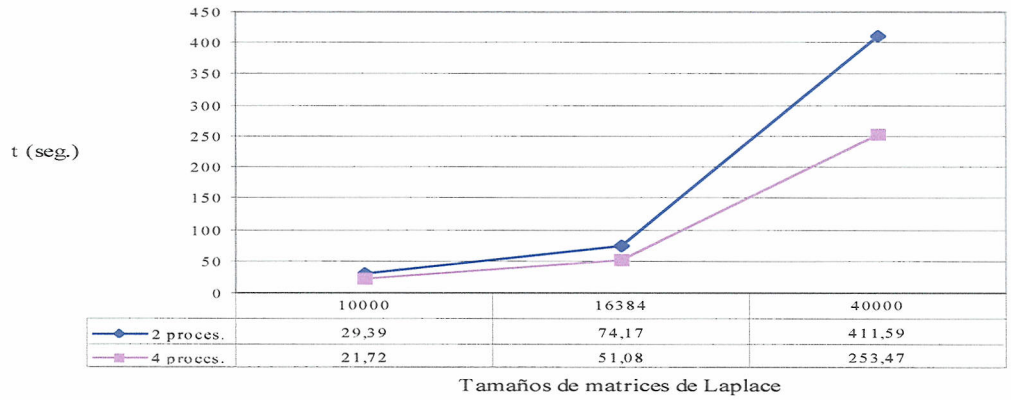


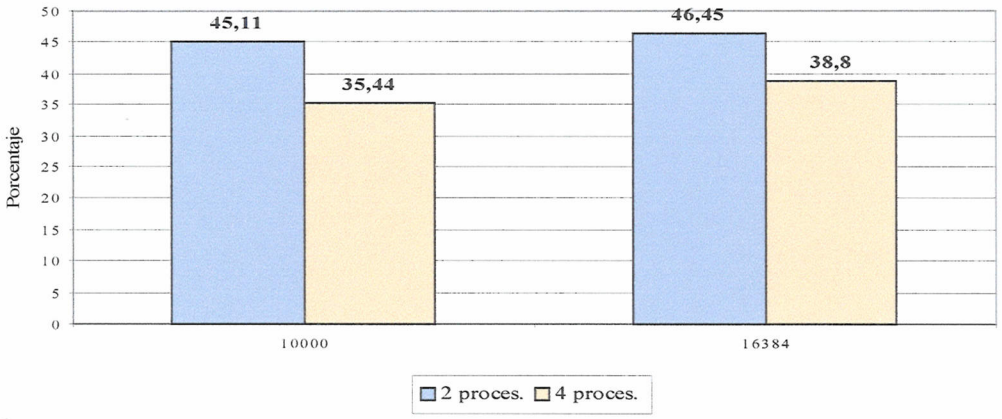
Figura 6: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matrices de Laplace 10000, 16384 y 40000, usando 2 y 4 procesadores y* $q = 6$. *IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

7.4.5 Comparación de los métodos en dos etapas con el método de Jacobi por bloques

A continuación, vamos a comparar los resultados del método por bloques en dos etapas con el método de Jacobi por bloques, que en principio posee un buen paralelismo inherente. En este caso, los subsistemas diagonales se resuelven (no se aproximan) utilizando la factorización completa de Choleski (FCC). Para ello, presentamos en la Figura 7 los tiempos obtenidos con el método de Jacobi por bloques, para las matrices de tamaño 10000 y 16384, usando 2 y 4 procesadores. En dicha figura presentamos también el tiempo que se ha necesitado para calcular la factorización, y en la gráfica, se da el porcentaje que este tiempo representa frente a la resolución completa del problema. Si comparamos estos datos con los obtenidos con el método por bloques en dos etapas usando iteraciones internas Gauss-Seidel que se expusieron en la Tabla 3, se observa claramente que este método es más rápido que el método de Jacobi por bloques, incluso descontando el tiempo de la factorización. La Figura 8 resume los tiempos obtenidos para distintos métodos por bloques en dos etapas de la Tabla 3 y los compara con los obtenidos con el método de Jacobi por bloques, para 2 y 4 procesadores y para las matrices de Laplace de tamaños 10000 y 16384. Notemos que además se presenta en esta figura, el tiempo correspondiente al método de Jacobi por bloques sin considerar el tiempo necesario para obtener la factorización completa de Choleski (Bl/Jacobi − FCC).

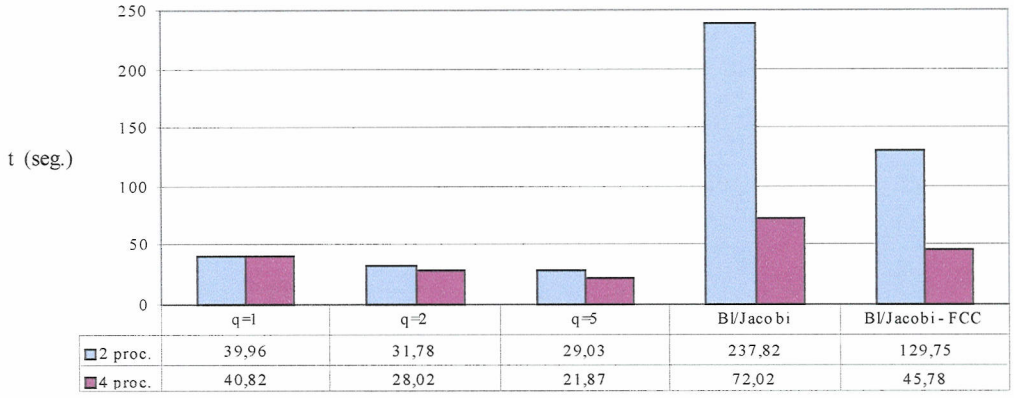
	Matriz de Laplace 10000			Matriz de Laplace 16384		
Procs.	Iter.	T. Par.	T. FCC	Iter.	T. Par.	T. FCC
2	378	237,82	108,07	495	805,03	374,01
4	580	72,02	26,24	754	235,5	91,58

(a)



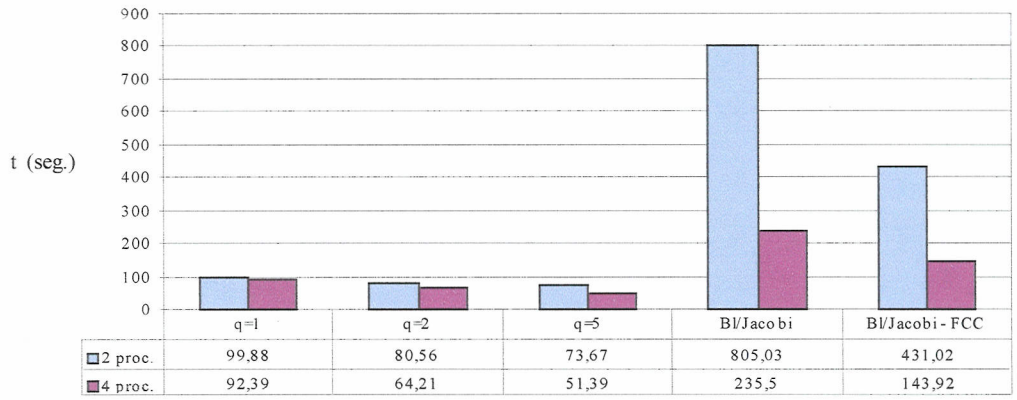
(b)

Figura 7: *Resultados del método Jacobi por bloques. Matrices de Laplace 10000 y 16384 usando 2 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.



	q=1	q=2	q=5	Bl/Jacobi	Bl/Jacobi-FCC
2 proc.	39,96	31,78	29,03	237,82	129,75
4 proc.	40,82	28,02	21,87	72,02	45,78

(a) Matriz Laplace: 10000.



	q=1	q=2	q=5	Bl/Jacobi	Bl/Jacobi-FCC
2 proc.	99,88	80,56	73,67	805,03	431,02
4 proc.	92,39	64,21	51,39	235,5	143,92

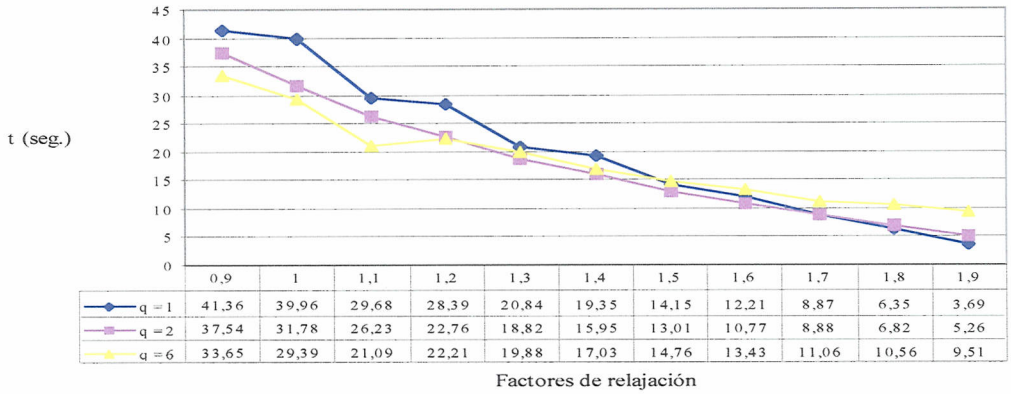
(b) Matriz Laplace: 16384.

Figura 8: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas Gauss-Seidel y del método de Jacobi por bloques. Matrices de Laplace 10000 y 16384, usando 2 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

7.4.6 Influencia del parámetro de relajación

En las secciones anteriores, hemos estudiado el comportamiento de los métodos por bloques en dos etapas usando para la aproximación de los sistemas internos el método iterativo Gauss-Seidel, es decir, usando el método SOR para el caso particular en que $\omega = 1$. En esta sección vamos a estudiar el comportamiento del Algoritmo 7.1 con iteraciones internas SOR para distintos valores del parámetro de relajación ω. Las Tablas 6 y 7 muestran los tiempos de ejecución de dicho algoritmo para las matrices de Laplace de tamaño 10000 y 16384, usando 3 y 4 procesadores, respectivamente. Como se puede apreciar, para cada factor de relajación ω, las conclusiones son similares a las obtenidas cuando se utilizan iteraciones internas Gauss-Seidel ($\omega = 1$). En particular, el número de iteraciones globales decrece conforme aumentamos el número de iteraciones internas. Este decrecimiento en el número de iteraciones globales produce un menor tiempo de ejecución cuando compensa el hecho de realizar un número de iteraciones internas determinado en cada bloque. Sin embargo, cabe mencionar que para valores del parámetro de relajación próximos a 2, la realización de más iteraciones internas no consigue reducir el tiempo de ejecución.

Las Figuras 9, 10, 11, 12, 13 y 14, ilustran cómo influye la elección del parámetro de relajación ω en los tiempos de ejecución de los métodos por bloques en dos etapas usando iteraciones internas SOR. En las Figuras 9 y 10 hemos considerado, para la matriz de Laplace de tamaño 10000, varios métodos por bloques en dos etapas que difieren en el número de iteraciones internas realizadas en cada bloque. Para cada uno de ellas se ha calculado el tiempo de ejecución para distintos valores de ω. La Figura 9 presenta los resultados para

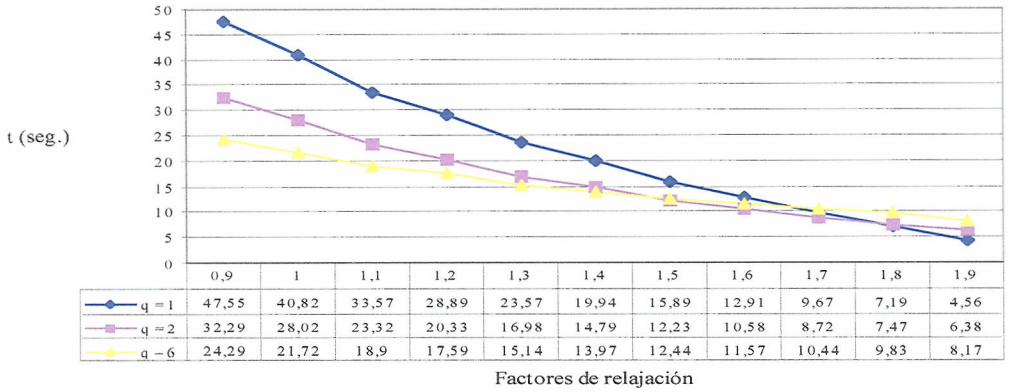


	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	41,36	39,96	29,68	28,39	20,84	19,35	14,15	12,21	8,87	6,35	3,69
q = 2	37,54	31,78	26,23	22,76	18,82	15,95	13,01	10,77	8,88	6,82	5,26
q = 6	33,65	29,39	21,09	22,21	19,88	17,03	14,76	13,43	11,06	10,56	9,51

Figura 9: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR. Matriz de Laplace 10000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

2 procesadores y la Figura 10, para 4 procesadores. De forma análoga, las Figuras 11 y 12 muestran los resultados obtenidos para la matriz de Laplace de tamaño 16384 para 2 y 4 procesadores, respectivamente, y las Figuras 13 y 14 para la matriz de Laplace de tamaño 40000 para 2 y 5 procesadores.

De estas figuras se deduce que para las matrices de menor tamaño los tiempos bajan a medida que se aumenta el número de iteraciones internas q, para cualquier factor de relajación hasta cierto valor crítico. A partir de este valor la situación empieza a invertirse hasta llegar a valores del factor de relajación muy próximos a $1,9$, donde se obtiene el mejor tiempo de ejecución del método con $q = 1$. Es decir, el mejor tiempo se obtiene, independientemente del número de procesadores para ω en torno a 1,9 y $q = 1$. Sin embargo, queremos hacer notar, que se aprecia un aumento de dicho valor crítico atendiendo al número

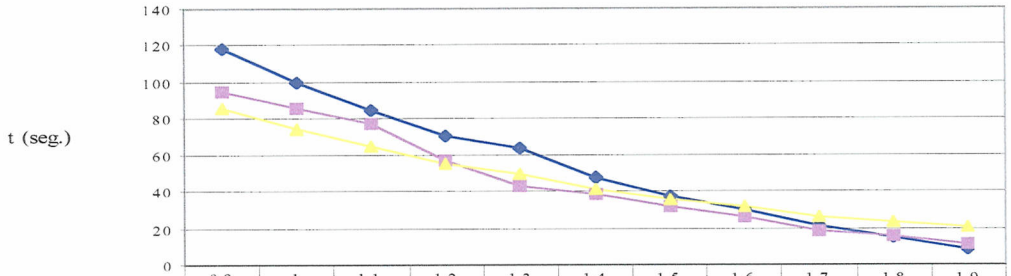


	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	47,55	40,82	33,57	28,89	23,57	19,94	15,89	12,91	9,67	7,19	4,56
q = 2	32,29	28,02	23,32	20,33	16,98	14,79	12,23	10,58	8,72	7,47	6,38
q = 6	24,29	21,72	18,9	17,59	15,14	13,97	12,44	11,57	10,44	9,83	8,17

Factores de relajación

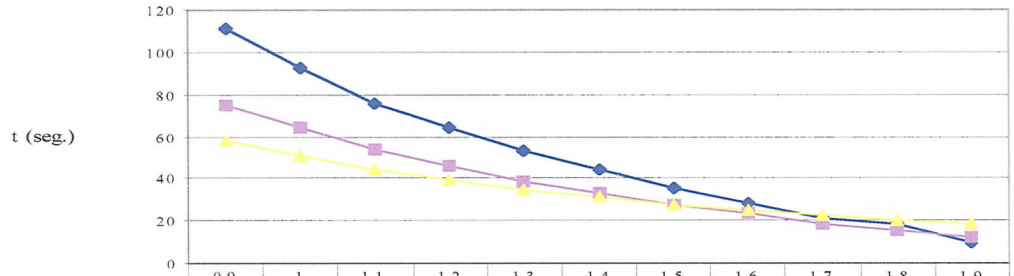
Figura 10: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR. Matriz de Laplace 10000, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

de procesadores usado, que se hace más patente cuando aumenta el tamaño de la matriz. Así, de las Figuras 13 y 14 se deduce que, para la matriz de tamaño 40000, el mejor tiempo, 29,09 segundos, se ha obtenido con 5 procesadores, para $\omega = 1,9$ y $q = 6$, consiguiendo una reducción en el tiempo frente al uso de 2 procesadores de un 33%.



	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	118,15	99,88	84,54	70,43	64,23	47,83	37,87	30,08	22,05	15,35	8,72
q = 2	94,65	85,6	77,32	57,24	43,12	38,12	32,04	26,34	18,34	15,83	11,62
q = 6	85,4	74,17	65,09	55,22	49,33	41,28	35,98	31,98	26,34	23,4	20,52

Figura 11: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR. Matriz de Laplace 16384, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.



	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	110,83	92,39	75,85	64,38	52,91	44,46	35,38	28,38	21,24	18,44	9,39
q = 2	74,99	64,21	53,71	46,28	38,61	33,07	27,34	23,11	18,72	15,41	12,27
q = 6	57,73	51,08	44,05	39,57	34,49	31,26	27,61	25,1	22,27	20,46	18,67

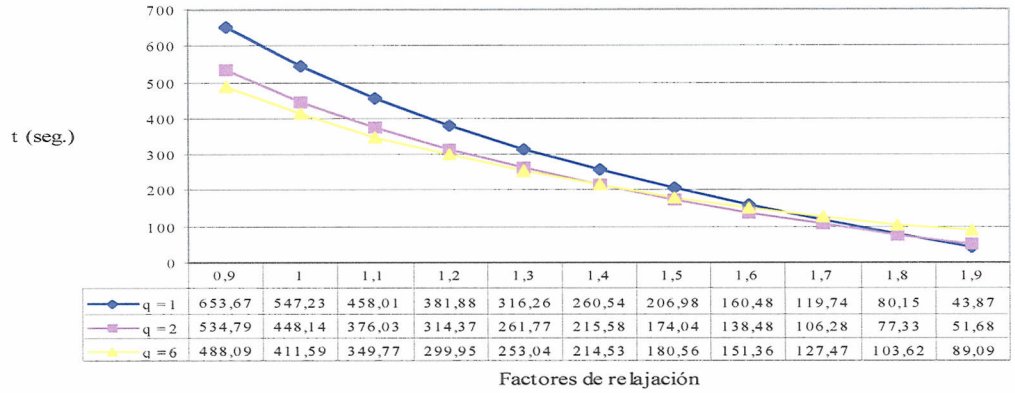
Figura 12: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR. Matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

		Matriz de Laplace 10000		Matriz de Laplace 16384	
q	ω	Iter.	T.Par.	Iter.	T.Par.
1	1	10449	38,26	16894	89,88
2		5803	27,84	9312	67,11
5		4179	23,55	6654	57,52
9		3350	25,03	5291	60,54
1	1,2	7360	26,96	11865	63,12
2		4172	20,09	6645	47,79
5		2167	17,96	3337	43,05
9		1835	19,32	2329	49,92
1	1,4	5053	18,51	8108	43,21
2		2981	14,35	4688	33,78
5		1681	13,93	2526	32,53
9		1143	15,21	1881	38,44
1	1,6	3237	11,84	5150	27,41
2		2081	10,02	3195	22,9
5		1321	10,95	1918	24,79
9		980	11,26	1552	31,68
1	1,8	1740	6,54	2709	14,41
2		1395	6,73	2037	14,64
5		1051	8,79	1457	18,78
9		829	8,99	1310	26,71
1	1,9	1069	3,91	1613	8,55
2		1130	5,55	1567	11,28
5		945	7,87	1272	16,34
9		773	9,43	1216	24,86

Tabla 6: *Comparando el método por bloques en dos etapas con iteraciones internas SOR, para distintas matrices de Laplace, usando 3 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

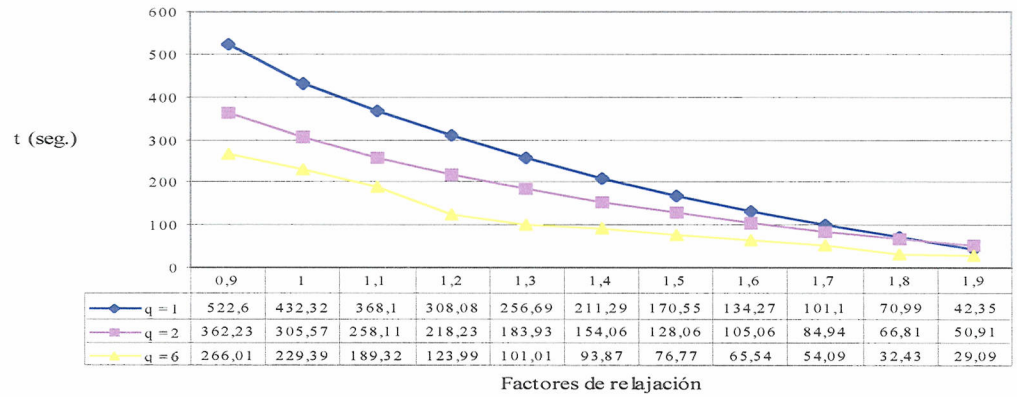
		Matriz de Laplace 10000		Matriz de Laplace 16384	
q	ω	Iter.	T.Par.	Iter.	T.Par.
1	1	10581	40,82	17062	92,39
2		5939	28,02	9485	64,21
5		3003	21,87	4661	51,39
9		2113	22,83	3180	53,16
1	1,2	7483	28,89	12021	64,38
2		4314	20,33	6825	46,28
5		2334	16,99	3551	39,04
9		2113	22,67	3180	52,92
1	1,4	5170	19,94	8256	44,46
2		3133	14,79	4881	33,07
5		1857	13,51	2751	30,36
9		1484	15,94	2119	35,40
1	1,6	3351	12,91	5294	28,38
2		2248	10,58	3408	23,11
5		1504	10,97	2154	23,76
9		1296	13,86	1796	29,98
1	1,8	1854	7,19	2854	18,44
2		1584	7,47	2277	15,41
5		1239	9,02	1701	18,72
9		1160	12,41	1558	26,03
1	1,9	1186	4,56	1761	9,39
2		1351	6,38	1819	12,27
5		1123	8,15	1522	16,92
9		1108	11,91	1468	24,46

Tabla 7: *Comparando el método por bloques en dos etapas con iteraciones internas SOR, para distintas matrices de Laplace, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.



	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	653,67	547,23	458,01	381,88	316,26	260,54	206,98	160,48	119,74	80,15	43,87
q = 2	534,79	448,14	376,03	314,37	261,77	215,58	174,04	138,48	106,28	77,33	51,68
q = 6	488,09	411,59	349,77	299,95	253,04	214,53	180,56	151,36	127,47	103,62	89,09

Figura 13: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR. Matriz de Laplace 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.



	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	522,6	432,32	368,1	308,08	256,69	211,29	170,55	134,27	101,1	70,99	42,35
q = 2	362,23	305,57	258,11	218,23	183,93	154,06	128,06	105,06	84,94	66,81	50,91
q = 6	266,01	229,39	189,32	123,99	101,01	93,87	76,77	65,54	54,09	32,43	29,09

Figura 14: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR. Matriz de Laplace 40000, usando 5 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

7.4.7 Comparación de los métodos usando iteraciones internas SOR y SSOR

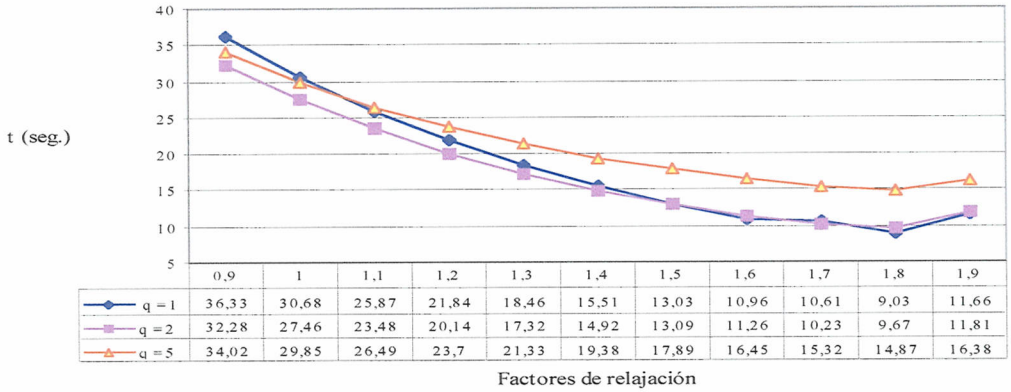
En esta sección presentamos resultados ilustrativos del método por bloques en dos etapas utilizando para la aproximación del sistema interno el método SSOR y comparamos dichos resultados con los obtenidos utilizando como aproximación de dicho sistema la obtenida por el método SOR.

Estudiaremos el comportamiento considerando las matrices de Laplace de tamaños 10000, 16384 y 40000. En las Tablas 8 y 9 podemos observar los resultados numéricos, atendiendo al factor de relajación, al utilizar 2 procesadores y en las Tablas 10 y 11 cuando utilizamos 4 procesadores. Como se puede observar, en dichas tablas, conforme se aumenta el factor de relajación, el número de iteraciones internas óptimas disminuye. Por otro lado, para cualquier q, los tiempos bajan al aumentar el factor de relajación ω, hasta un ω óptimo en el que los tiempos comienzan a subir. Este ω óptimo ha resultado en nuestros experimentos en torno a 1,7 y 1,8, algo más bajos que utilizando el método SOR en las iteraciones internas. Esto es independiente del tamaño de la matriz y del número de procesadores elegido. Esto queda reflejado, de forma más clara, en las Figuras 15, 16 y 17. Sin embargo, el número de iteraciones internas óptimo parece aumentar con el número de procesadores utilizados, para un tamaño de matriz fija. Esto es debido a que conforme aumenta el número de procesadores, los tamaños de los bloques asignados a estos son más pequeños, y por tanto la realización de más iteraciones internas se compensa con la reducción en el número de iteraciones globales. Así por ejemplo, para la matriz de tamaño 16384 usando 2 procesadores, se obtiene el mejor tiempo (20,13 segundos) con

$q = 1$ y con 4 procesadores (20,99) para $q = 2$.

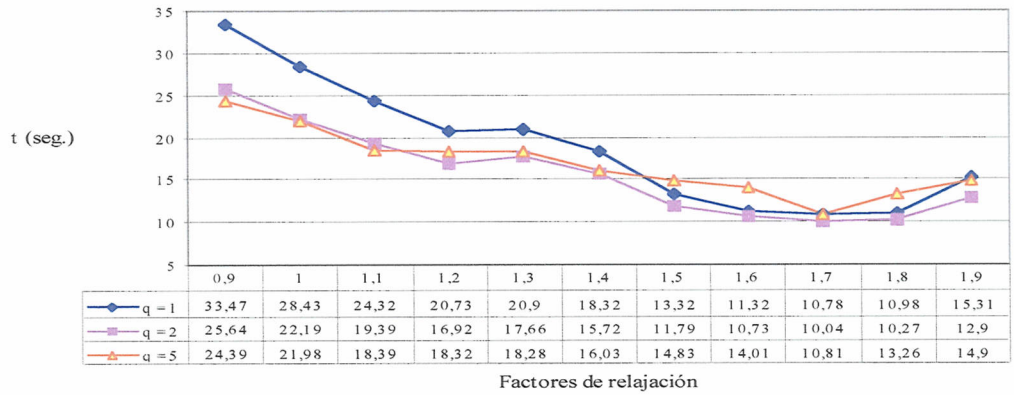
Por otra parte, aunque generalmente el tiempo de ejecución baja al aumentar, de forma racional, el número de procesadores, se ha observado que esto se invierte en determinados casos cuando trabajamos con el factor de relajación óptimo.

Para finalizar esta sección, comparamos los resultado de este método con el que resulta de aplicar el método SOR para el cálculo de las iteraciones internas. En las Figuras 18 y 19 comparamos ambos métodos para la matriz de Laplace de tamaño 16384, para distintas elecciones del factor de relajación ω. En las Figuras 20 y 21 aparecen los resultados para la matriz de Laplace de tamaño 40000. Como podemos observar, para valores pequeños de ω próximos a 1 se obtienen mejores resultados con el método SSOR, pero conforme nos vamos acercando al ω óptimo usando iteraciones internas SSOR, esto se invierte y resulta ser mejor el método SOR. Tengamos en cuenta que aunque para $\omega = 1$, se consigue la resolución del método por bloques en dos etapas con iteraciones internas SOR, con más iteraciones de las necesarias cuando se utiliza el método SSOR, a medida que aumentamos el factor de relajación ω se va igualando el número de iteraciones, llegando incluso a invertirse la situación (ver Figuras 22 y 23). Si tenemos en cuenta que el factor de relajación óptimo suele ser relativamente grande y que cada iteración del método SSOR corresponde a dos iteraciones SOR, es fácil deducir, como se refleja en las figuras y tablas anteriores que el mejor algoritmo paralelo se obtiene con el método SOR, para cualquier tamaño de matriz.



	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	36,33	30,68	25,87	21,84	18,46	15,51	13,03	10,96	10,61	9,03	11,66
q = 2	32,28	27,46	23,48	20,14	17,32	14,92	13,09	11,26	10,23	9,67	11,81
q = 5	34,02	29,85	26,49	23,7	21,33	19,38	17,89	16,45	15,32	14,87	16,38

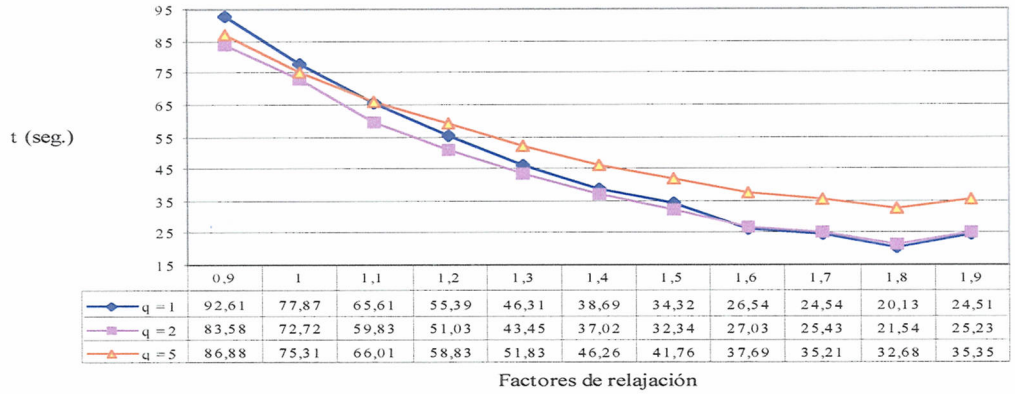
(a) 2 procesadores.



	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	33,47	28,43	24,32	20,73	20,9	18,32	13,32	11,32	10,78	10,98	15,31
q = 2	25,64	22,19	19,39	16,92	17,66	15,72	11,79	10,73	10,04	10,27	12,9
q = 5	24,39	21,98	18,39	18,32	18,28	16,03	14,83	14,01	10,81	13,26	14,9

(b) 4 procesadores.

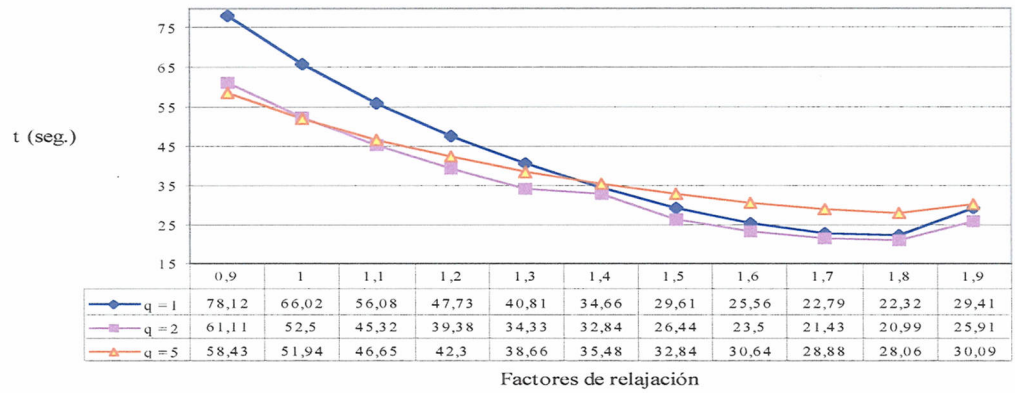
Figura 15: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SSOR. Matriz de Laplace 10000, usando 2 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.



	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	92,61	77,87	65,61	55,39	46,31	38,69	34,32	26,54	24,54	20,13	24,51
q = 2	83,58	72,72	59,83	51,03	43,45	37,02	32,34	27,03	25,43	21,54	25,23
q = 5	86,88	75,31	66,01	58,83	51,83	46,26	41,76	37,69	35,21	32,68	35,35

Factores de relajación

(a) 2 procesadores.

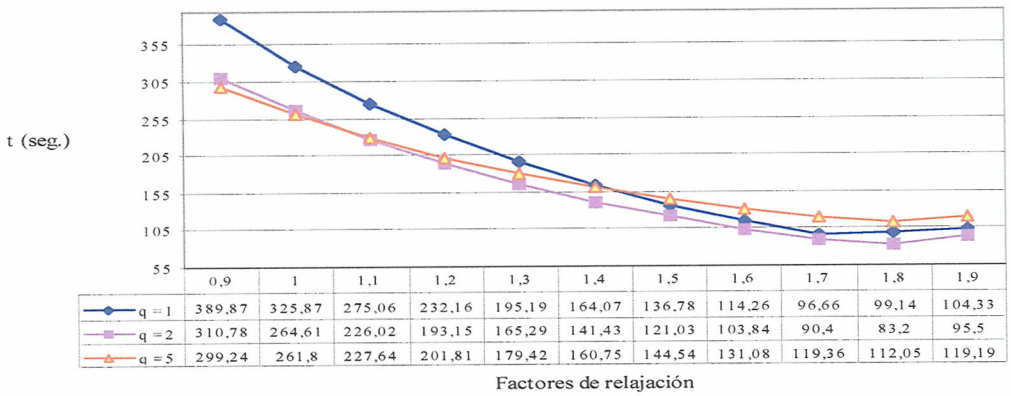


	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	78,12	66,02	56,08	47,73	40,81	34,66	29,61	25,56	22,79	22,32	29,41
q = 2	61,11	52,5	45,32	39,38	34,33	32,84	26,44	23,5	21,43	20,99	25,91
q = 5	58,43	51,94	46,65	42,3	38,66	35,48	32,84	30,64	28,88	28,06	30,09

Factores de relajación

(b) 4 procesadores.

Figura 16: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SSOR. Matriz de Laplace 16384, usando 2 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.



	0,9	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	389,87	325,87	275,06	232,16	195,19	164,07	136,78	114,26	96,66	99,14	104,33
q = 2	310,78	264,61	226,02	193,15	165,29	141,43	121,03	103,84	90,4	83,2	95,5
q = 5	299,24	261,8	227,64	201,81	179,42	160,75	144,54	131,08	119,36	112,05	119,19

Figura 17: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SSOR. Matriz de Laplace 40000, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

		Matriz de Laplace 10000		Matriz de Laplace 16384		Matriz de Laplace 40000	
q	ω	Iter.	T.Par.	Iter.	T.Par.	Iter.	T.Par.
1	0,9	6721	36,33	10860	92,61	26045	513,02
2		3749	32,28	6002	83,58	14219	466,35
3		2717	32,03	4308	82,49	10073	461,97
4		2193	32,9	3442	84,22	7943	467,84
5		1876	34,02	2917	86,88	6644	477,66
6		1665	35,57	2565	90	5767	490,22
9		1316	40,59	1978	100,77	4291	534,39
1	1	5661	30,68	9128	77,87	21833	430,33
2		3190	27,46	5085	72,72	11976	392,70
3		2337	27,54	3680	70,39	8529	390,81
4		1906	28,56	2966	72,39	6763	397,95
5		1646	29,85	2534	75,31	5688	409,17
6		1474	31,45	2245	78,68	4965	421,86
9		1191	36,8	1767	89,84	3751	465,56
1	1,2	4037	21,84	6469	55,39	15347	302,54
2		2341	20,14	3685	51,03	8535	279,75
3		1763	20,82	2728	52,27	6170	282,85
4		1475	22,11	2247	54,98	4967	293,20
5		1304	23,7	1958	58,83	4239	311,27
6		1191	25,42	1768	62,06	3752	321,16
9		1009	31,15	1455	74,16	2943	365,00
1	1,4	2862	15,51	3332	38,69	10576	208,50
2		1732	14,92	2671	37,02	6017	197,23
3		1357	16,02	2046	39,23	4454	204,07
4		1174	17,6	1736	42,47	3669	216,47
5		1067	19,38	1554	46,26	3199	230,34
6		997	21,27	1435	50,35	2888	247,09
9		876	23,09	1242	63,26	2378	294,84

Tabla 8: *Comparación de tiempos del método por bloques en dos etapas con iteraciones internas SSOR, para distintas matrices de Laplace, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

		Matriz de Laplace 10000		Matriz de Laplace 16384		Matriz de Laplace 40000	
q	ω	Iter.	T.Par.	Iter.	T.Par.	Iter.	T.Par.
1	1,6	2024	10,96	3112	26,54	6979	137,55
2		1303	11,26	1937	27,03	4131	135,43
3		1071	12,71	1554	29,75	3176	145,60
4		962	14,93	1371	33,5	2709	159,51
5		901	16,45	1266	37,69	2435	175,21
6		862	18,53	1199	42,03	2258	191,71
1	1,8	1659	9,03	2356	20,13	4636	91,37
2		1120	9,67	1556	21,54	2930	95,98
3		944	11,15	1294	24,78	2365	108,44
4		861	12,92	1169	28,59	2092	123,22
5		815	14,87	1099	32,68	1937	139,20
6		787	16,89	1056	37,05	1839	156,21
9		747	23,32	994	50,65	1692	209,85
1	1,9	2146	11,66	2870	24,51	4967	97,9
2		1369	11,81	1822	25,23	3113	102,47
3		1105	13,11	1466	28,06	2483	113,75
4		975	14,7	1290	31,71	2169	127,7
5		898	16,38	1186	35,35	1985	142,95
6		849	18,18	1119	39,24	1865	158,34
9		773	24,01	1016	51,78	1679	210,35

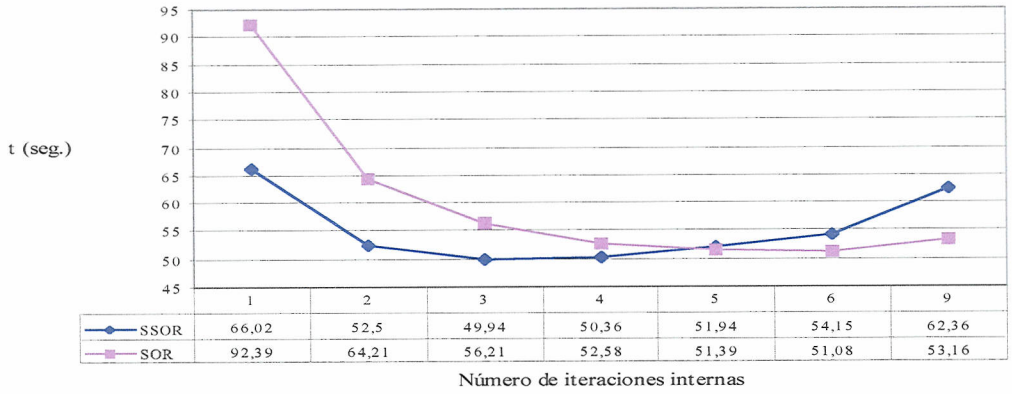
Tabla 9: *Comparación de tiempos del método por bloques en dos etapas con iteraciones internas SSOR, para distintas matrices de Laplace, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

		Matriz de Laplace 10000		Matriz de Laplace 16384		Matriz de Laplace 40000	
q	ω	Iter.	T.Par.	Iter.	T.Par.	Iter.	T.Par.
1	0,9	7010	33,47	11229	78,12	26619	386,87
2		4056	25,64	6395	61,11	14833	310,75
3		3042	23,9	4725	57,47	10726	294,43
4		2531	23,83	3878	57,52	8628	292,3
5		2224	24,39	3366	58,43	7353	299,24
6		2020	25,46	3024	60,38	6495	303,6
9		1682	28,82	2455	68,2	5055	334,04
1	1	5953	28,43	9502	66,28	22415	325,87
2		3507	22,19	5490	60,13	12610	264,61
3		2672	21,04	4111	49,94	9206	253,2
4		2253	21,25	3414	50,36	7470	253,56
5		2002	21,98	2994	64,81	6418	261,8
6		1756	22,32	2714	54,15	5433	267,54
9		1561	26,78	2250	73,5	4529	299,55
1	1,2	4348	20,73	6847	47,73	15968	232,16
2		2678	16,92	4117	39,38	9216	193,15
3		2116	16,65	3184	40,13	6892	189,67
4		1837	17,3	2716	42,3	5715	193,82
5		1671	18,32	2437	44,53	5005	201,81
6		1562	19,55	2251	44,95	4530	214,99
9		1561	23,79	2022	53,9	3741	247,27
1	1,4	3213	18,32	4981	34,66	11282	164,07
2		2092	15,72	3137	30,06	6755	141,43
3		1982	18,54	2524	30,73	5220	145,76
4		1655	18,11	2222	32,84	4452	150,55
5		1441	16,03	2044	35,48	3992	160,75
6		1323	19,09	1926	38,48	3688	172,27
9		1262	19,99	1735	48,14	3185	210,26

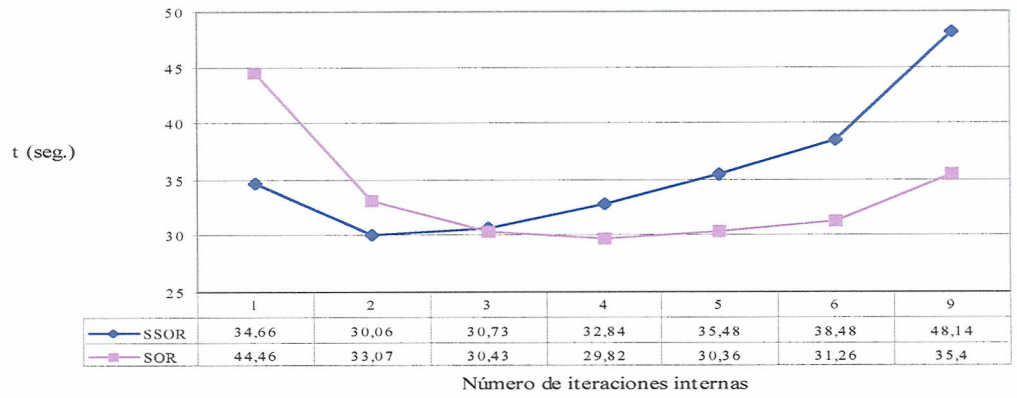
Tabla 10: *Comparación de tiempos del método por bloques en dos etapas con iteraciones internas SSOR, para distintas matrices de Laplace, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

		Matriz de Laplace 10000		Matriz de Laplace 16384		Matriz de Laplace 40000	
q	ω	Iter.	T.Par.	Iter.	T.Par.	Iter.	T.Par.
1	1,5	2786	13,32	4255	29,61	9426	136,78
2		1870	11,79	2758	26,44	5778	121,03
3		1574	12,43	2267	27,56	4555	124,64
4		1433	13,48	2029	29,99	3950	133,74
5		1352	14,83	1891	32,84	3591	144,54
6		1299	16,27	1800	35,98	3354	156,67
9		1215	20,85	1654	46,02	2967	195,88
1	1,6	2456	11,32	3672	25,56	7868	114,26
2		1696	10,73	2451	23,5	4956	103,84
3		1454	11,44	2056	25,04	3991	109,25
4		1342	12,66	1869	27,6	3521	119,13
5		1278	14,01	1762	30,64	3247	131,08
6		1238	15,5	1693	33,84	3068	143,33
9		1176	20,16	1585	44,06	2779	183,5
1	1,8	2306	10,98	3205	22,32	6023	99,14
2		1603	10,27	2191	20,99	3971	83,2
3		1375	10,82	1859	22,66	3292	90,17
4		1268	11,95	1702	25,16	2966	100,76
5		1209	13,26	1615	28,06	2781	112,05
6		1173	14,72	1562	31,24	2665	127,48
9		1126	19,32	1488	41,4	2496	165,08
1	1,9	3177	15,31	3345	29,41	7185	104,33
2		2045	12,9	2001	25,91	4557	95,50
3		1660	13,06	2191	26,69	3663	100,39
4		1470	13,86	1768	29,87	3220	108,94
5		1358	14,9	1532	30,09	2958	119,19
6		1287	16,12	1344	43,54	2789	130,27
9		1177	20,58	1244	50,31	2528	167,02

Tabla 11: *Comparación de tiempos del método por bloques en dos etapas con iteraciones internas SSOR, para distintas matrices de Laplace, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

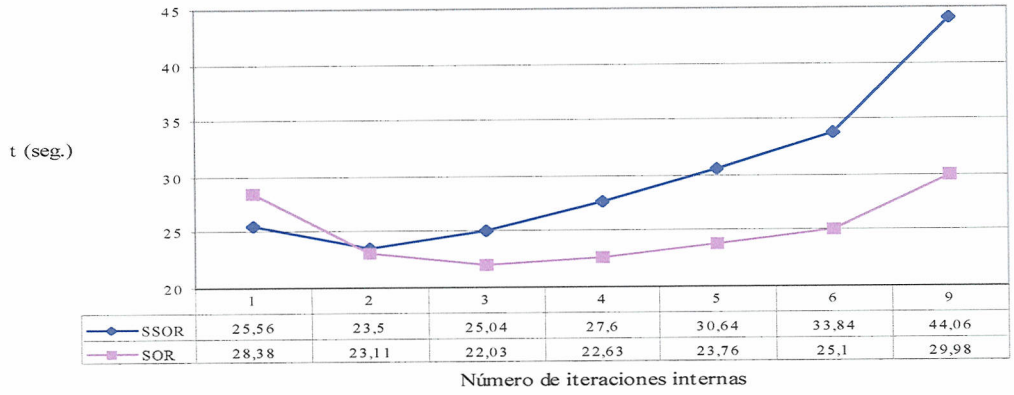


(a) $\omega = 1$

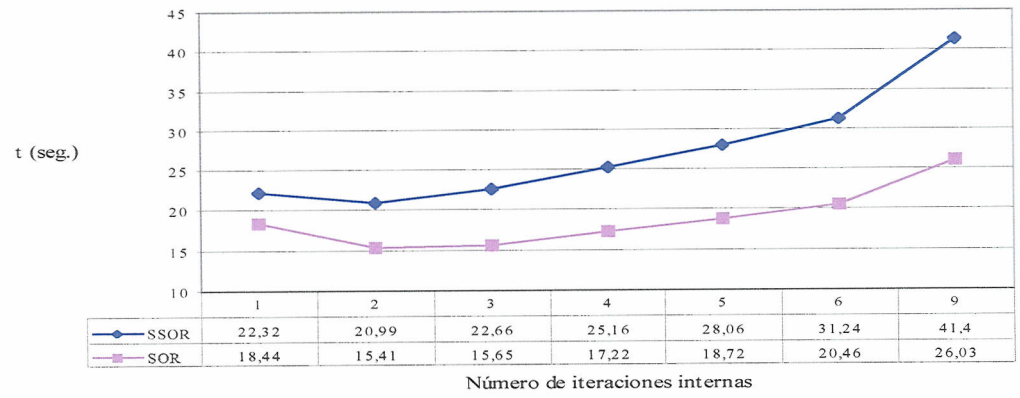


(b) $\omega = 1,4$

Figura 18: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR y SSOR. Matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

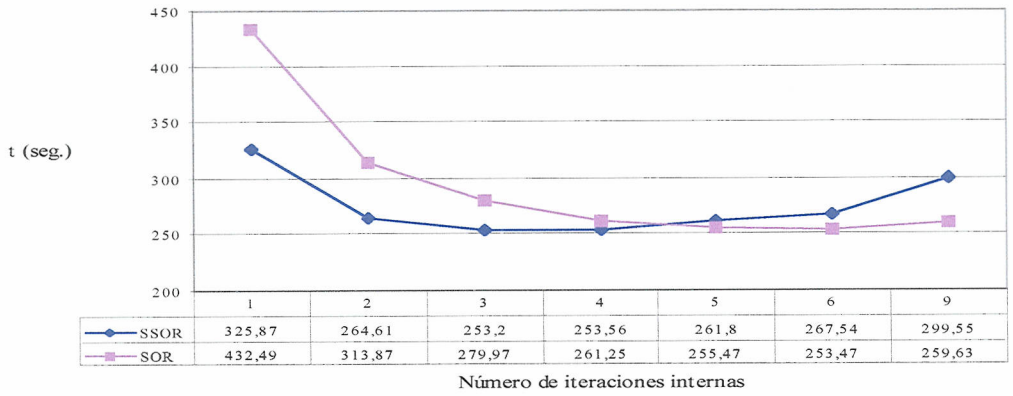


(a) $\omega = 1,6$

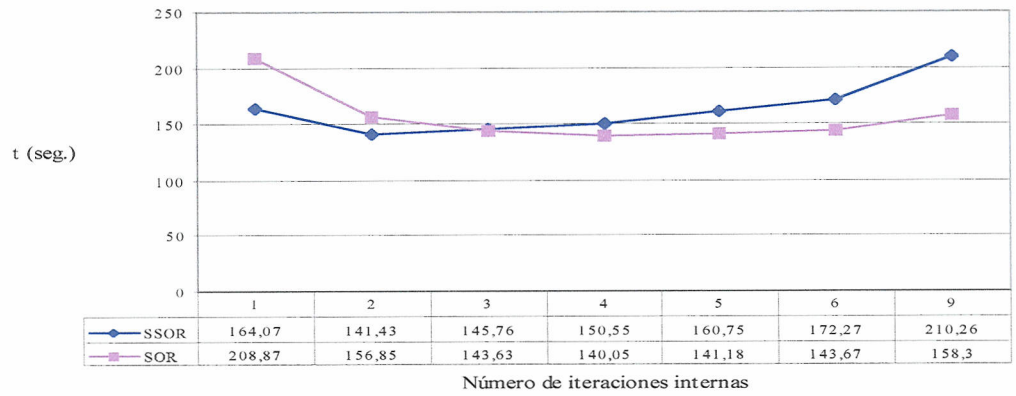


(b) $\omega = 1,8$

Figura 19: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR y SSOR. Matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

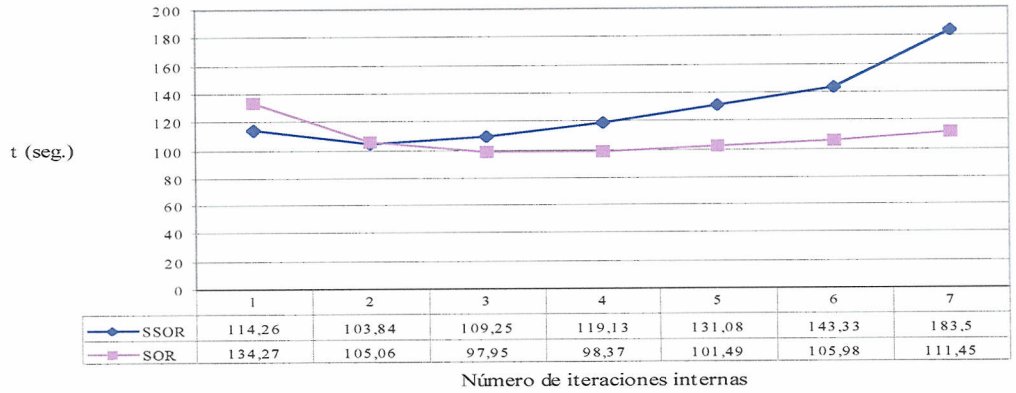


(a) $\omega = 1$

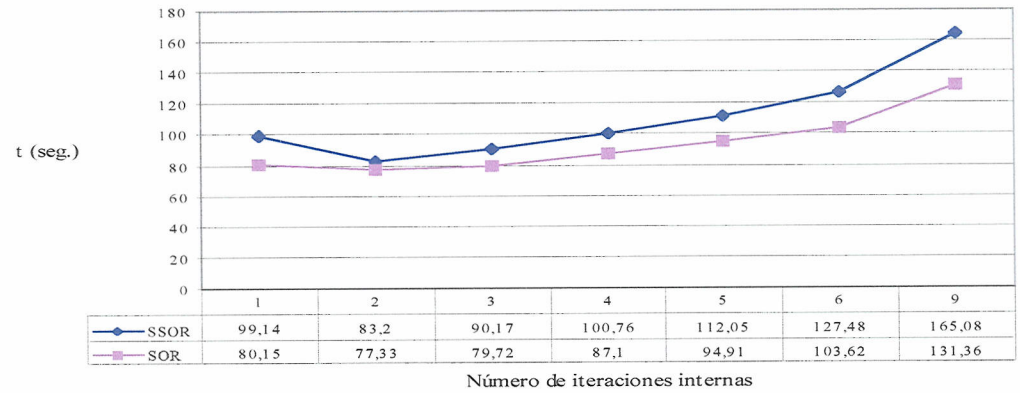


(b) $\omega = 1,4$

Figura 20: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR y SSOR. Matriz de Laplace 40000, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

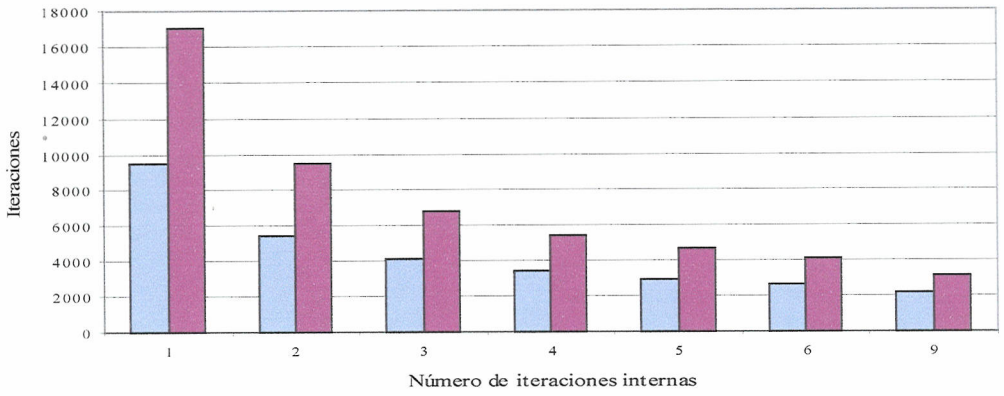


(a) $\omega = 1,6$

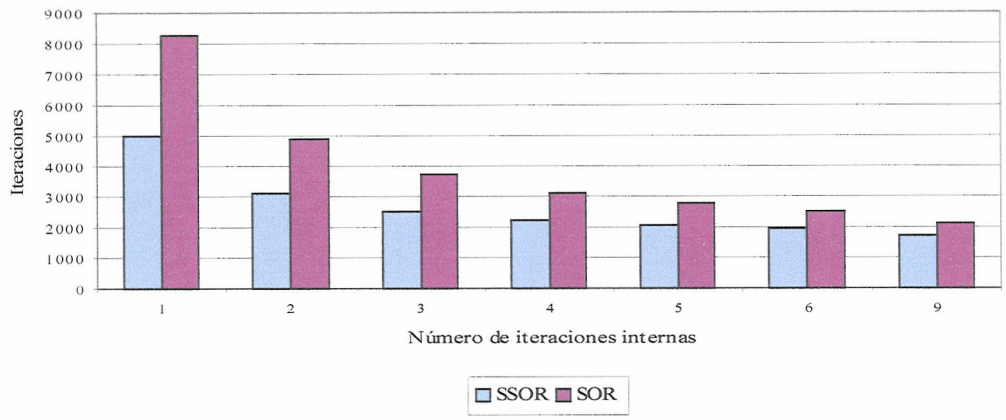


(b) $\omega = 1,8$

Figura 21: *Tiempos experimentales del método por bloques en dos etapas con iteraciones internas SOR y SSOR. Matriz de Laplace 40000, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

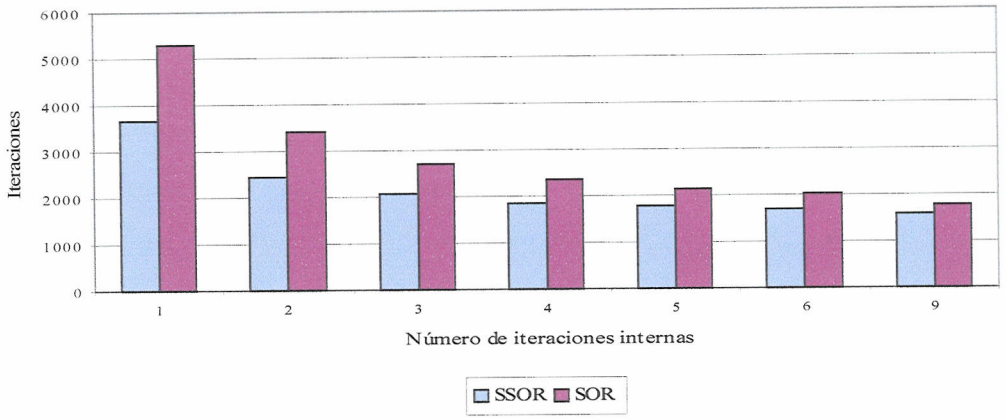


(a) $\omega = 1$

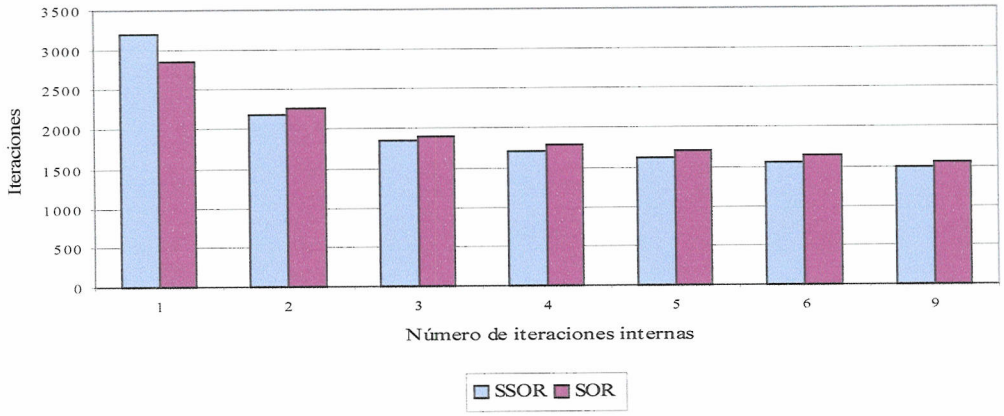


(b) $\omega = 1,4$

Figura 22: *Iteraciones del método por bloques en dos etapas con iteraciones internas SOR y SSOR. Matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.



(a) $\omega = 1,6$



(b) $\omega = 1,8$

Figura 23: *Iteraciones del método por bloques en dos etapas con iteraciones internas SOR y SSOR. Matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

7.4.8 Variando las iteraciones internas

Uno de los problemas que se plantea, cuando se estudia un método en dos etapas, es la elección del número de iteraciones internas. Hasta ahora en la elección hecha del número de procesadores y del número de filas con las que trabaja cada procesador, hay un equilibrio que hace que todos los procesadores tengan aproximadamente la misma carga, es decir, todos aproximan sistemas de ecuaciones con un número de elementos similar. Es por esto, por lo que en los experimentos mostrados hasta el momento, se ha mantenido fijo el número de iteraciones internas para cada bloque.

Nos interesa por tanto estudiar qué ocurre, cuando se produce un desequilibrio en la carga asignada a cada uno de los procesadores. La Tabla 12 ilustra esta situación comparando distintos métodos por bloques en dos etapas atendiendo a la elección del número de iteraciones internas Gauss-Seidel realizadas por cada procesador. En ellas se ha considerado la matriz de Laplace de tamaño 10000 y se han utilizado 2 procesadores. Se han distinguido dos casos. En el primer caso se ha asignado a cada uno de los procesadores 5000 filas y en el segundo, un procesador trabaja con 2500 filas y el otro con 7500. Dicha tabla refleja que si la carga asignada a cada procesador es parecida, lo más lógico parece elegir los factores no estacionarios algo mayores que 1, e iguales para todos los bloques. Por el contrario, si hay un desequilibrio en la carga entre los procesadores, se puede elegir el número de iteraciones internas realizadas, para cada bloque, diferente con el fin de conseguir un equilibrio de carga a priori, resolviendo el problema en menor tiempo.

Siguiendo con el estudio de la elección del número de iteraciones internas,

	Filas/Bl: 5000^2		Filas/Bl: 2500,7500	
q	Iter.	T.Par.	Iter.	T.Par.
1	10309	39,96	10184	55,99
5	2680	29,03	2531	39,4
6	2338	29,39	2185	39,5
4,2	4587	39,66	5303	42,68
8,4	2622	40,29	2917	38,09
12,4	2469	54,52	2469	54,52

Tabla 12: *Comparación de tiempos para el método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 10000, usando 2 procesadores con carga de trabajo compensada y descompensada entre ellos. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

nos planteamos variar éstas cada cierto número de iteraciones externas. Los dos criterios elegidos para el estudio del comportamiento del método han sido:

$$\left.\begin{array}{l} q(niter) = \max(1, q - [niter/k]) \\ q(niter) = q + [niter/k] \end{array}\right\} \; k \in \{250, 500\}.$$

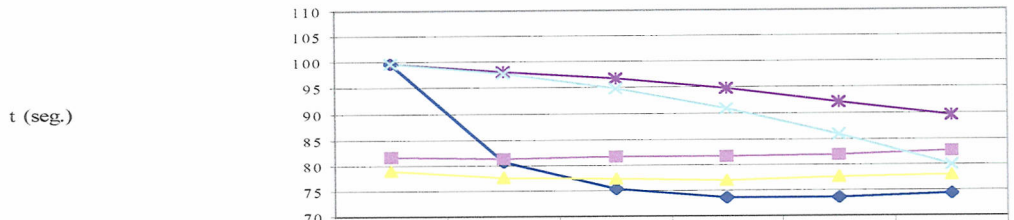
En estas expresiones, *niter* indica el valor actual de la iteración externa y $[\cdot]$ denota la parte entera de un número real. Notemos que la elección de $q(niter) = \max(1, q - [niter/k])$ indica que el número de iteraciones internas se inicia con el valor de q y se reduce en una unidad cada k iteraciones externas hasta llegar a un mínimo de 1 , en el que permanece constante; es obvio que si $q = 1$, entonces $q(niter) = \max(1, 1 - [niter/k]) = 1$. Por otro lado, la elección de $q(niter) = q + [niter/k]$, indica que el número de iteraciones internas va aumentando en una unidad cada k iteraciones externas a partir de un valor inicial q. Los resultados obtenidos con estas elecciones para el número de iteraciones internas, los comparamos con el criterio usual que hemos considerado hasta el momento, es decir, que cada procesador realice un número fijo q de iteraciones

internas en su ejecución.

Para comparar estos criterios, presentamos en las Figuras 24 y 25 resultados del método por bloques en dos etapas con iteraciones internas SOR para la matriz de Laplace de tamaño 16384 usando 2 procesadores. Para $\omega = 1$, el mejor resultado lo obtenemos para el criterio usual de mantener el número de iteraciones internas fijo, siendo el criterio $q(niter) = q+[niter/500]$, el que ofrece resultados más próximos. Por el contrario, conforme aumenta el parámetro de relajación, la situación se invierte hasta llegar a obtener mejores resultados con el criterio $q(niter) = \max(1, q - [niter/250])$.

En la Tabla 13 comparamos estos dos criterios en la elección de las iteraciones internas con la elección de factores no estacionarios fijos, cuando resolvemos un problema mayor con más procesadores. La matriz utilizada, en este caso, ha sido la matriz de Laplace de tamaño 40000 usando 4 procesadores. Observamos una sitación similar a la comentada anteriormente.

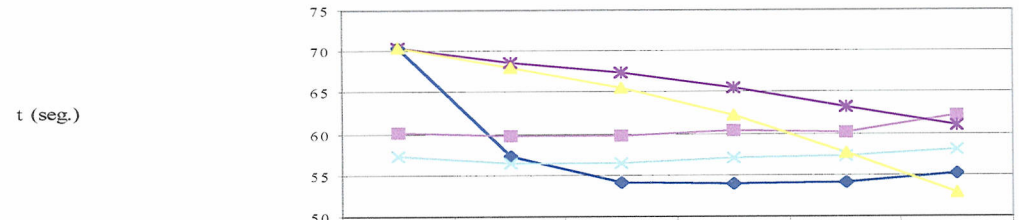
De todo lo expuesto en relación a los criterios de elección de la iteraciones internas, pensamos que es muy difícil el poder asegurar a priori que la elección de uno de estos criterios redundará en un beneficio frente a la elección de un número fijo de iteraciones internas. Además, el beneficio que se obtendría podría ser, en ocasiones, casi inapreciable.



	1	2	3	4	5	6
q (fijo)	99,88	80,56	75,33	73,8	73,67	74,17
q = q + [niter/250]	81,62	81,22	81,43	81,5	82,01	82,67
q = q + [niter/500]	78,84	77,74	77,15	76,91	77,62	77,88
q = max(1,q - [niter/250])	99,88	98,26	96,79	94,75	92,31	89,41
q = max(1,q - [niter/500])	99,88	97,64	94,82	90,98	85,96	80,04

Número de iteraciones internas

(a) $\omega = 1$

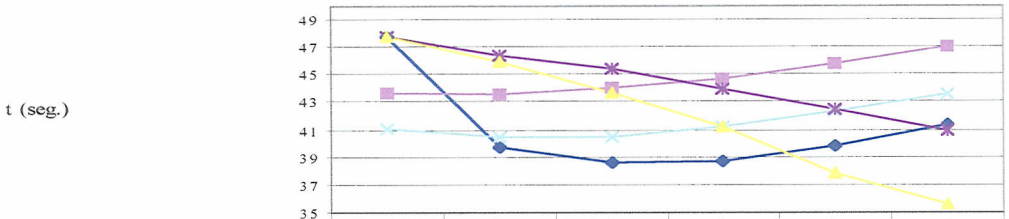


	1	2	3	4	5	6
q (fijo)	70,43	57,24	54,09	53,94	54,1	55,22
q = q + [niter/250]	60,13	59,75	59,71	60,37	60,05	62,18
q = q + [niter/500]	57,17	56,32	56,37	56,93	57,32	58,04
q = max(1,q - [niter/250])	70,43	68,59	67,28	65,5	63,32	60,95
q = max(1,q - [niter/500])	70,43	67,97	65,45	62,28	57,68	52,95

Número de iteraciones internas

(b) $\omega = 1,2$

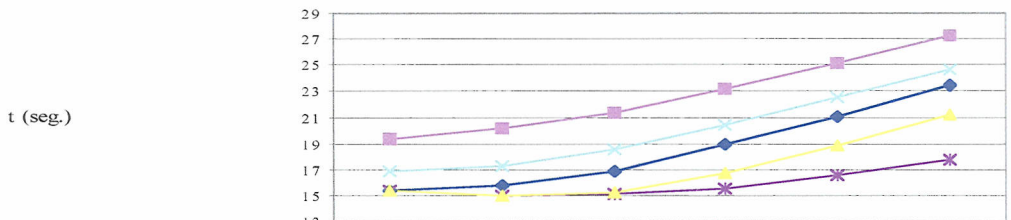
Figura 24: *Comparación de tiempos entre los diferentes criterios de elección del número de iteraciones internas. Método en dos etapas con iteraciones internas SOR. Matriz de Laplace 16384, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^l\| < 10^{-2}$.



	1	2	3	4	5	6
q (fijo)	47,83	39,75	38,58	38,72	39,82	41,28
q = q + [niter/250]	43,71	43,55	44,05	44,67	45,83	47,02
q = q + [niter/500]	41,13	40,51	40,48	41,21	42,31	43,5
q = max(1,q - [niter/250])	47,83	46,42	45,47	43,92	42,41	40,9
q = max(1,q - [niter/500]	47,83	45,89	43,73	41,18	37,86	35,61

Número de iteraciones internas

(a) $\omega = 1,4$



	1	2	3	4	5	6
q (fijo)	15,35	15,83	16,79	18,96	21,07	23,4
q = q + [niter/250]	19,29	20,16	21,27	23,18	25,18	27,34
q = q + [niter/500]	16,82	17,25	18,6	20,46	22,53	24,66
q = max(1,q - [niter/250])	15,35	15,01	15,09	15,56	16,55	17,78
q = max(1,q - [niter/500])	15,35	15,02	15,24	16,72	18,83	21,17

Número de iteraciones internas

(b) $\omega = 1,8$

Figura 25: *Comparación de tiempos entre los diferentes criterios de elección del número de iteraciones internas. Método en dos etapas con iteraciones internas SOR. Matriz de Laplace 16384, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^l\| < 10^{-2}$.

		Iter. internas $\max(1, q - [niter/250])$		Iter. internas $q + [niter/250]$		Iter. internas q fijo	
w	q	Iter.	T.Par.	Iter.	T.Par.	Iter.	T.Par.
1	1	40769	432,43	6144	310,7	40769	432,43
	2	40558	432,28	5912	309,53	22391	313,87
	6	37512	408,98	5156	313,96	9203	253,47
1,4	1	19487	208,87	4564	182,15	19487	208,87
	2	19271	204,34	4348	181,94	11131	156,85
	6	16790	190,24	3734	191,47	5204	143,67
1,8	1	6387	80,15	3156	95,85	6387	80,15
	2	6262	78,18	2995	97,78	4573	77,33
	6	5159	78,94	2677	118,17	2957	103,62

Tabla 13: *Comparación de tiempos entre los diferentes criterios de elección del número de iteraciones internas. Método en dos etapas con iteraciones internas SOR. Matriz de Laplace 40000, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^l\| < 10^{-2}$.

7.4.9 Comportamiento del método atendiendo al multiprocesador usado

Las Tablas 14 y 15 ilustran cómo se comportan estos métodos atendiendo al multiprocesador usado. Los resultados de la Tabla 14 corresponden al método en dos etapas por bloques con iteraciones internas Gauss-Seidel y los de la Tabla 15 con iteraciones internas SSOR. En ambas tablas se ha usado la matriz de Laplace 4096-N. Como podemos observar, aunque las conclusiones que se obtienen en lo que se refiere al comportamiento de dichos métodos son similares en el IBM RS/6000 SP y en el Cluster de Pentiums, los tiempos aumentan considerablemente de utilizar una u otra plataforma. Obviamente, la velocidad de comunicación y CPU del Cluster es mucho menor que la del IBM RS/6000 SP. No obstante, cabe mencionar que mientras en el IBM RS/6000 SP no conseguimos, en ocasiones, disminuir el tiempo al aumentar el número de iteraciones internas, ocurre justamente lo contrario en el Cluster (ver por ejemplo, los tiempos para $q = 1$, 2 y $\omega = 1,4$ en la Tabla 15). La explicación la encontramos en que en el IBM RS/6000 SP la reducción en el número de iteraciones globales puede no compensar el hecho de realizar más iteraciones internas, debido a que las comunicaciones son muy rápidas; en el Cluster, al ser las comunicaciones más lentas, la misma reducción en el número de iteraciones globales sí puede afectar, positivamente, en el tiempo total.

Así, para ilustrar la diferencia de tiempos en el Cluster, en las Figuras 26, 27, 28 y 29 se compara el tiempo de CPU y el tiempo REAL de convergencia obtenidos en el Cluster de Pentiums para estos métodos por bloques en dos etapas al incrementar el número de iteraciones internas Gauss-Seidel. En las

Figuras 26 y 27, se presentan los resultados para la matriz 4096-N usando 2 y 4 procesadores, respectivamente. Las Figuras 28 y 29 ilustran el mismo tipo de resultados para la matriz de Laplace de tamaño 50000, que recordemos tiene 100 bloques B de tamaño 500. Observamos que la gran diferencia existente entre ambos tiempos se reduce considerablemente conforme aumentamos el número de iteraciones internas. Esto es lógico ya que se reduce el número de comunicaciones entre procesadores.

En estas figuras también se presenta el porcentaje que representa el tiempo que no es de CPU (diferencia entre el REAL y el de CPU) frente al tiempo REAL de ejecución. Obviamente este porcentaje crece para una matriz dada, al aumentar el número de procesadores ya que cuantos más procesadores intervengan en la resolución del problema se disminuye el tamaño de los bloques asignados a cada procesador y por tanto, se reduce el tiempo de CPU, pero aumenta el número de comunicaciones. Por otra parte, si fijamos ahora el número de procesadores (comparar Figura 26 con Figura 28, y las Figuras 27 y 29) observamos que conforme aumenta el tamaño de la matriz este porcentaje disminuye, es decir, el tiempo de cálculo aumenta más que el de comunicación.

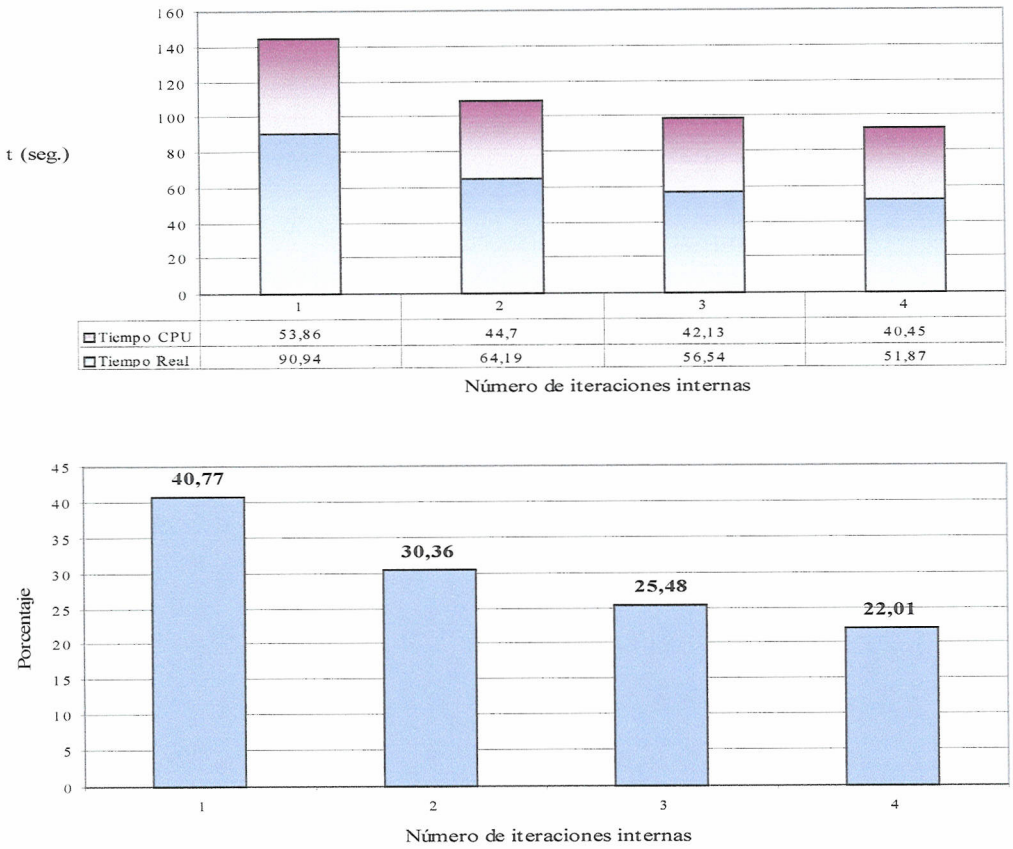


Figura 26: *Comparación de los tiempos REALES y de CPU para el método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 4096-N usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-7}$.

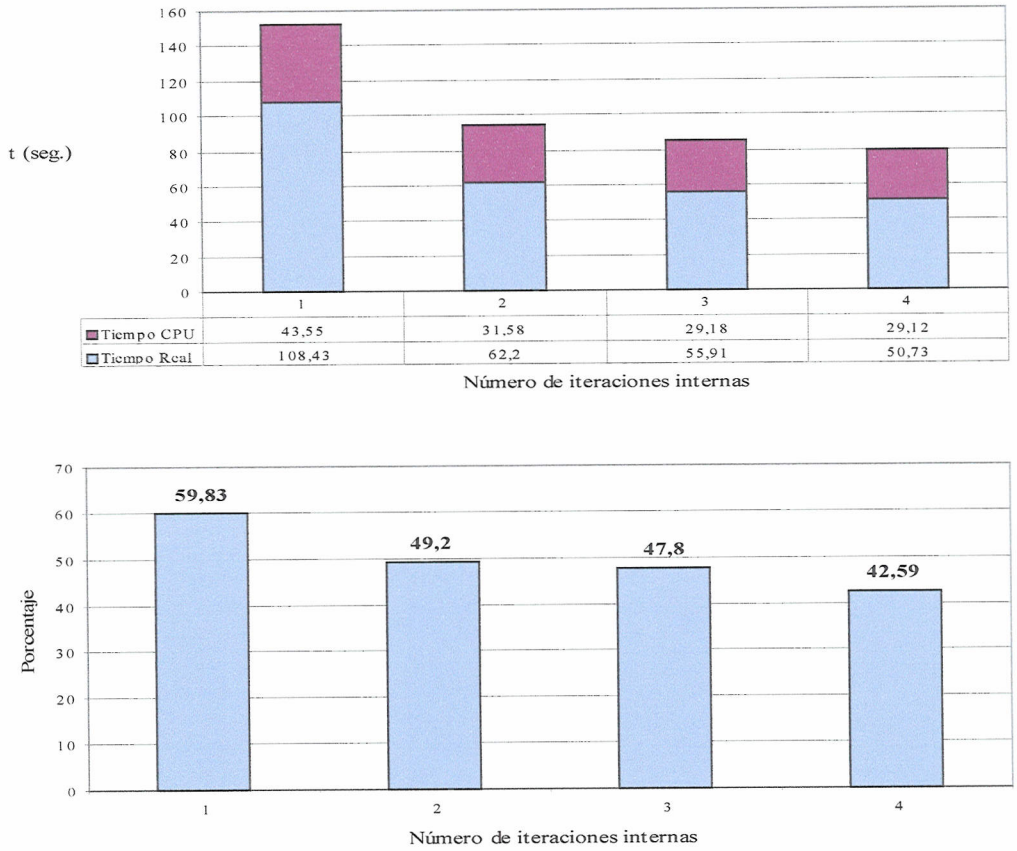


Figura 27: *Comparación de los tiempos REALES y de CPU para el método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 4096-N usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-7}$.

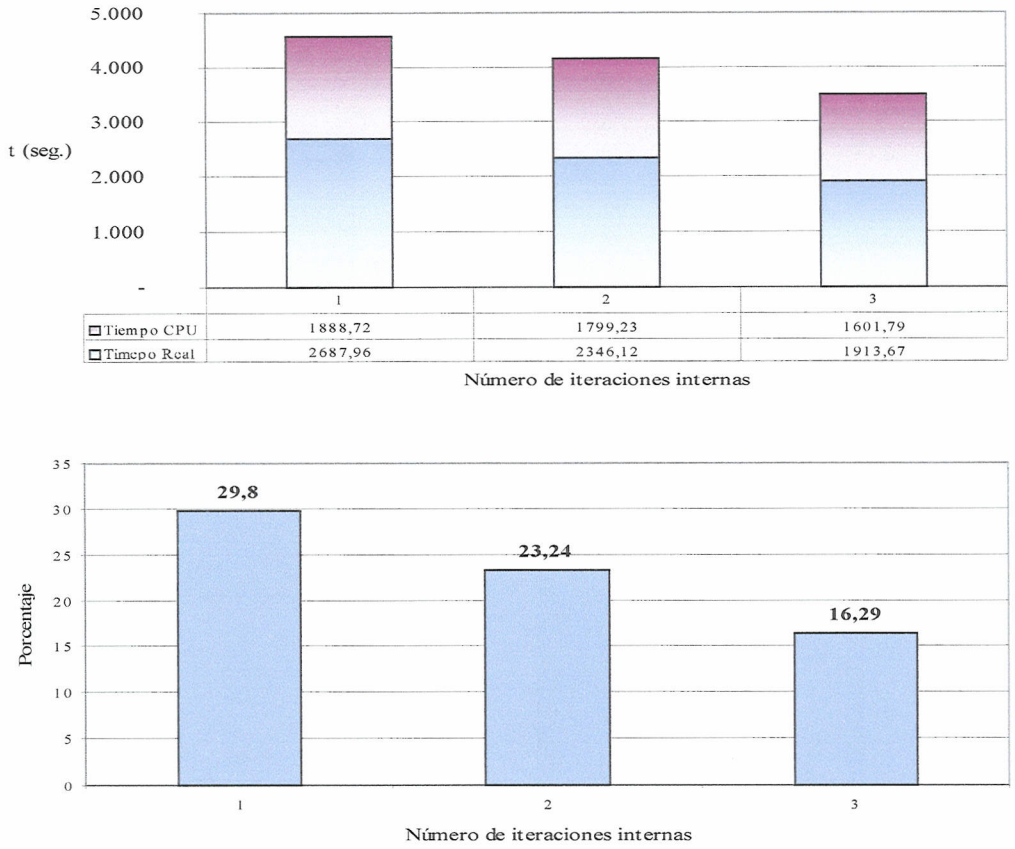


Figura 28: *Comparación de los tiempos REALES y de CPU para el método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 50000 usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-4}$.

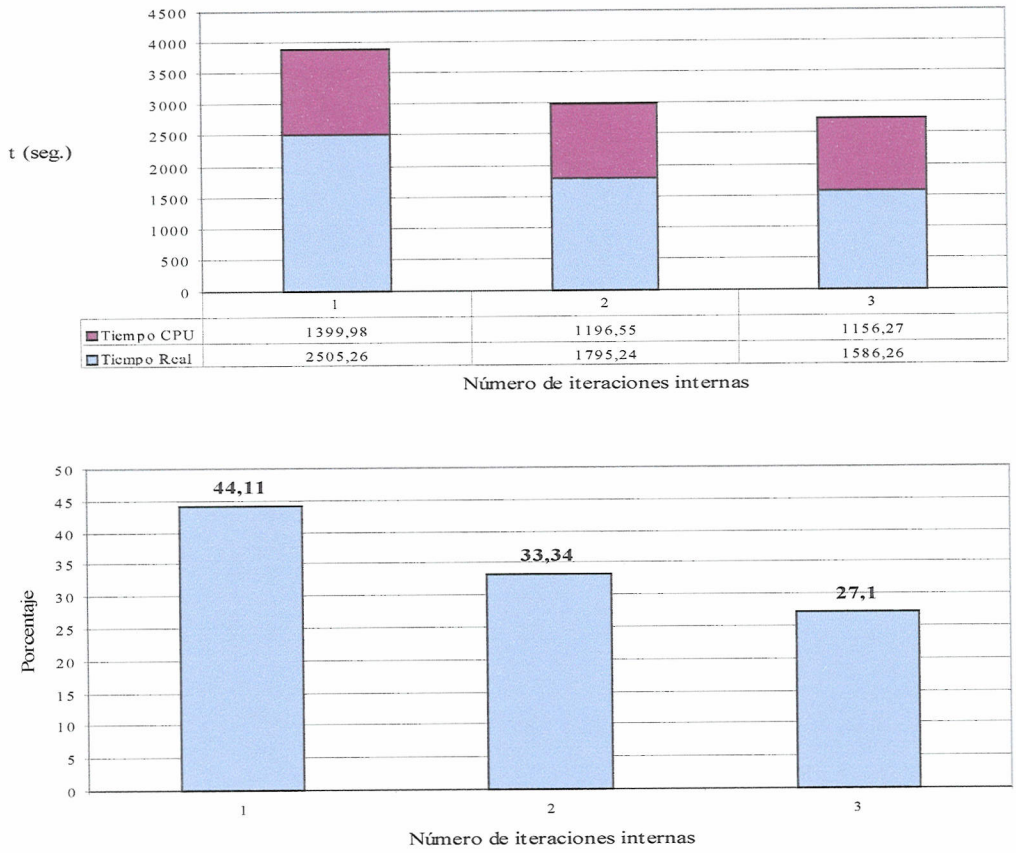


Figura 29: *Comparación de los tiempos REALES y de CPU para el método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 50000 usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-4}$.

Número Proc.	q	Iter.	IBM RS/6000 SP	CLUSTER
$r = 2$	1	4337	8,52	41,92
	2	2412	6,39	30,49
	3	1741	5,81	26,95
	5	1194	5,62	26,06
$r = 3$	1	4428	9,13	44,19
	2	2506	6,31	29,52
	3	1804	5,43	25,25
	5	1300	5,01	23,20

Tabla 14: *Método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Matriz de Laplace 4096-N, usando 2 y 3 procesadores. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

q	w	Iter.	IBM RS/6000 SP	CLUSTER
1	1	2512	5,5	30,02
2		1505	5,43	24,1
3		1165	5,34	23,29
5		897	5,89	25,16
1	1,4	1393	3,55	16,75
2		935	3,6	14,75
3		789	3,71	15,64
5		679	4,39	18,98
1	1,6	1104	2,9	13,15
2		786	2,98	12,41
3		688	3,25	13,64
5		619	4,01	17,03
1	1,8	1113	2,91	13,27
2		784	2,96	12,39
3		678	3,21	13,41
5		602	3,87	16,83
1	1,9	1603	3,79	19,13
2		1033	4,59	16,33
3		837	3,86	16,03
5		686	4,39	19,36

Tabla 15: *Método por bloques en dos etapas con iteracions internas SSOR. Matriz de Laplace 4096-N, usando 3 procesadores. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

7.4.10 Speed-up y eficiencia

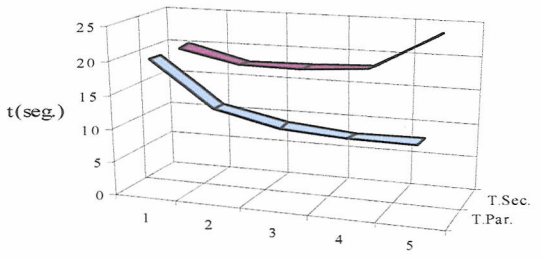
A continuación estudiamos el rendimiento de estos métodos en relación con su algoritmo secuencial. La Figura 30 ilustra el comportamiento del algoritmo paralelo, usando iteraciones internas Gauss-Seidel, frente a su correspondiente secuencial, para la matriz 4096-N usando 3 procesadores. Se observa que para $q > 1$, se obtienen mejores tiempos en la resolución del algoritmo en paralelo que en secuencial, siendo las diferencias más significativas conforme aumentamos el número de iteraciones internas ya que en este caso, el número de comunicaciones entre procesadores decrece consiguiendo speed-up's en torno a 2,25, que se corresponden con una eficiencia del 75%. Sin embargo, para $q = 1$, el tiempo paralelo y secuencial son prácticamente iguales.

Conforme aumenta el tamaño de la matriz el tiempo paralelo siempre es menor que el secuencial para cualquier q, como puede apreciarse en las Figuras 31 y 32, en las que se comparan los tiempos paralelos y secuenciales para la matriz de tamaño 16384, usando 2 y 4 procesadores, respectivamente. Notemos, en dichas figuras, que la eficiencia decrece notablemente cuando el número de procesadores aumenta, haciendose más patente este decrecimiento cuanto menor es el orden de la matriz. Esto se debe a que el coste de las operaciones ejecutadas en paralelo puede ser menor que el coste de las comunicaciones. Estas conclusiones también pueden ser observadas prestando atención a la Tabla 16 que presenta resultados para matrices de Laplace de tamaños 4096, 10000 y 16384, usando 2, 3 y 4 procesadores.

Las Figuras 33 y 34, ilustran el comportamiento del método, utilizando iteraciones internas SOR con $\omega = 1,9$ para la matriz de Laplace de tamaño 16384.

Hacemos notar que se han elegido como factores de relajación los óptimos en cada caso. Como se puede observar las conclusiones son análogas a las obtenidas para iteraciones internas Gauss-Seidel. Estas conclusiones son también extensibles al caso en que no utilicemos un factor de relajación óptimo, tal y como se muestra en las Tablas 17 y 18 para distintas matrices de Laplace, usando 3 y 4 procesadores, respectivamente.

q	Speep-up
1	1
2	1,34
3	1,59
4	1,75
5	2,23



	1	2	3	4	5
T.Par.	20,13	13,43	11,35	10,56	10,64
T.Sec.	20,16	17,99	18,02	18,49	23,76

Figura 30: *Método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Relación entre los tiempos paralelos y secuenciales de una matriz de Laplace 4096-N, usando 3 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-7}$.

q	Eficiencia
1	0,73
2	0,81
3	0,85
4	0,87
5	0,89
6	0,90
9	0,93

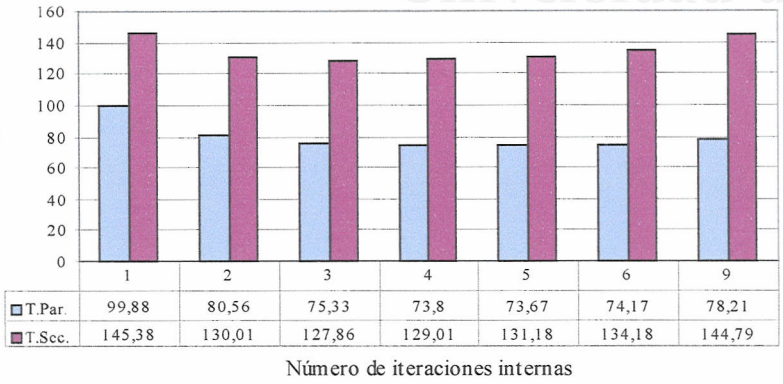


Figura 31: *Método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Relación entre los tiempos paralelos y secuenciales de una matriz de Laplace 16384, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

q	Eficiencia
1	0,37
2	0,51
3	0,54
4	0,64
5	0,68
6	0,70
9	0,70

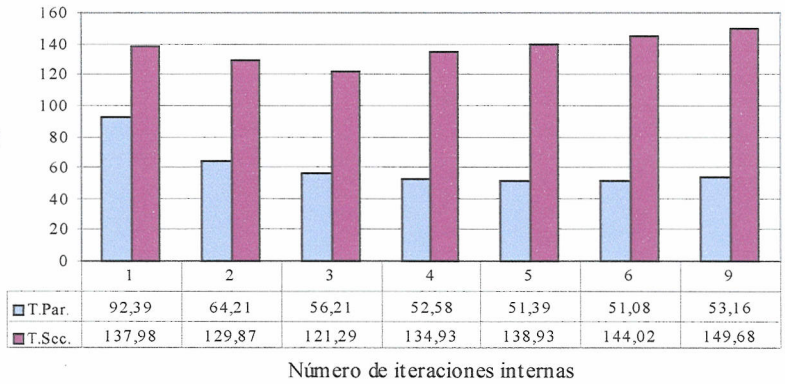


Figura 32: *Método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Relación entre los tiempos paralelos y secuenciales de una matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

		Matriz de Laplace 4096-N				Matriz de Laplace 10000				Matriz de Laplace 16384			
r	q	Iter.	T.Par.	T.Sec.	Efic.	Iter.	T.Par.	T.Sec.	Efic.	Iter.	T.Par.	T.Sec.	Efic.
2	1	4337	8,52	8,8	0,52	10309	39,96	54,3	0,68	16714	99,88	145,38	0,73
	2	2412	6,39	8,16	0,64	5658	31,78	49,09	0,77	9126	80,56	130,01	0,81
	3	1741	5,81	8,25	0,71	4027	29,65	48,62	0,82	6457	75,33	127,86	0,85
	4	1400	5,62	8,54	0,76	3190	29,05	49,37	0,85	5085	73,8	129,01	0,87
	5	1194	5,63	8,91	0,79	2680	29,03	50,57	0,87	4246	73,67	131,18	0,89
	6	1056	5,72	9,31	0,81	2338	29,39	50,09	0,85	3681	74,17	134,18	0,90
	9	828	6,16	10,7	0,87	1762	31,31	57,26	0,91	2727	78,21	144,79	0,93
3	1	4428	9,13	9,03	0,33	10449	38,26	54,92	0,48	16894	89,88	146,78	0,54
	2	2506	6,31	8,4	0,44	5803	27,84	50,37	0,60	9312	67,11	132,47	0,66
	3	1840	5,43	8,57	0,53	4179	24,86	50,43	0,68	6654	60,9	131,53	0,72
	4	1503	5,13	8,97	0,58	3350	23,83	51,84	0,73	5291	58,1	133,87	0,77
	5	1300	5,01	9,47	0,63	2846	23,55	53,77	0,76	4461	57,52	137,7	0,80
	6	1164	5,01	9,97	0,66	2508	23,63	55,89	0,79	3902	57,59	141,99	0,82
	9	941	5,27	11,77	0,74	1941	25,03	63,13	0,84	2960	60,54	157,02	0,86
4	1	4513	10,75	9,23	0,21	10581	40,82	55,74	0,34	17062	92,39	137,98	0,37
	2	2593	7,01	8,76	0,31	5939	28,02	51,44	0,46	9485	64,21	129,87	0,51
	3	1932	5,85	8,98	0,38	4324	24,11	53,23	0,55	6837	56,21	121,29	0,54
	4	1599	5,41	9,12	0,42	3501	22,53	55,12	0,61	5484	52,58	134,93	0,64
	5	1399	5,14	9,57	0,47	3003	21,87	56,58	0,65	4661	51,39	138,93	0,68
	6	1266	5,07	10,02	0,49	2670	21,72	61,01	0,70	4108	51,08	144,02	0,70
	9	1048	5,25	13,22	0,63	2113	22,83	68,48	0,75	3180	53,16	149,68	0,70

Tabla 16: *Método por bloques en dos etapas con iteraciones internas Gauss-Seidel. Resultados para distintas matrices de Laplace usando 2, 3 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

q	Eficiencia
1	0,48
2	0,59
3	0,66
4	0,7
5	0,75
6	0,71
9	0,68

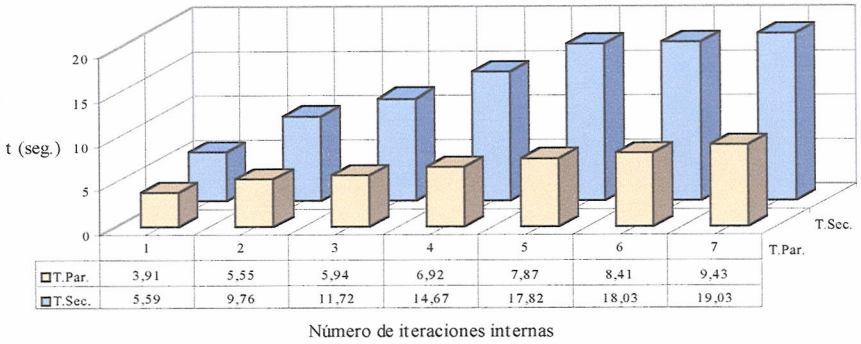


Figura 33: *Método por bloques en dos etapas con iteraciones internas SOR. Relación entre los tiempos paralelos y secuenciales para una matriz de Laplace 10000, usando 3 procesadores y* $\omega = 1,9$. *IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

q	Eficiencia
1	0,38
2	0,49
3	0,6
4	0,66
5	0,72
6	0,73
9	0,72

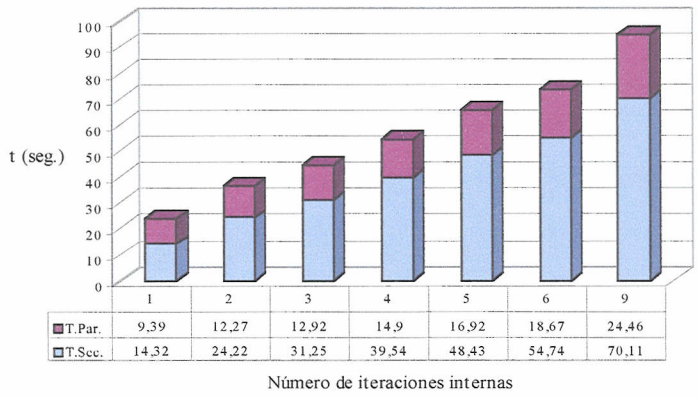


Figura 34: *Método por bloques en dos etapas con iteraciones internas SOR. Relación entre los tiempos paralelos y secuenciales para una matriz de Laplace 16384, usando 4 procesadores y* $\omega = 1,9$. *IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

		Matriz de Laplace 4096-N				Matriz de Laplace 10000				Matriz de Laplace 16384			
q	ω	Iter.	T.Sec.	T.Par.	Efic.	Iter.	T.Sec.	T.Par.	Efic.	Iter.	T.Sec.	T.Par.	Efic.
1	1	4428	9,03	9,13	0,33	10449	54,92	38,26	0,48	16894	146,78	89,88	0,54
2		2506	8,4	6,31	0,44	5803	50,37	27,84	0,60	9312	132,47	67,11	0,66
5		1300	8,57	5,01	0,57	4179	50,43	23,55	0,71	6654	131,53	57,52	0,76
9		941	11,78	5,27	0,74	3350	51,84	25,03	0,69	5291	133,87	58,10	0,74
1	1,2	3142	6,41	6,64	0,32	7360	38,67	26,96	0,48	11865	101,11	63,12	0,53
2		1835	6,15	4,62	0,44	4172	36,14	20,09	0,60	6645	94,62	47,79	0,66
5		1029	7,48	3,98	0,63	2167	40,9	17,96	0,76	3337	102,95	43,05	0,80
9		795	9,94	4,51	0,73	1835	42,81	19,32	0,74	2329	123,31	49,92	0,82
1	1,4	2182	4,45	4,57	0,32	5053	26,56	18,51	0,48	8108	70,45	43,21	0,54
2		1351	4,53	3,41	0,44	2981	25,81	14,35	0,60	4688	66,67	33,78	0,66
5		839	6,11	3,23	0,63	1681	31,72	13,93	0,76	2526	77,91	32,53	0,80
9		695	8,71	3,92	0,74	1143	33,11	15,21	0,73	1881	99,69	38,44	0,86
1	1,6	1427	2,91	2,98	0,33	3237	17,01	11,84	0,48	5150	44,71	27,41	0,54
2		994	3,34	2,52	0,44	2081	18,02	10,02	0,60	3195	45,42	22,9	0,66
5		702	5,12	2,71	0,63	1321	24,93	10,95	0,76	1918	59,13	24,79	0,80
9		624	7,81	3,48	0,75	980	26,11	11,26	0,77	1552	82,23	31,68	0,87
1	1,8	806	1,63	1,71	0,32	1740	9,12	6,54	0,46	2709	23,49	14,41	0,54
2		736	2,45	1,85	0,44	1395	12,06	6,73	0,60	2037	28,94	14,64	0,66
5		603	4,39	2,31	0,63	1051	19,82	8,79	0,75	1457	44,91	18,78	0,80
9		528	7,19	1,11	2,16	829	21,01	8,99	0,78	1310	69,44	26,71	0,87
1	1,9	528	1,06	1,11	0,32	1069	5,59	3,91	0,48	1613	13,95	8,55	0,54
2		652	2,12	1,65	0,43	1130	9,76	5,55	0,59	1567	22,24	11,28	0,66
5		567	4,13	2,13	0,65	945	17,82	7,87	0,75	1272	39,21	16,34	0,80
9		432	7,01	3,22	0,73	773	19,03	9,43	0,69	1216	64,41	24,86	0,86

Tabla 17: *Comparando el método por bloques en dos etapas con iteraciones internas SOR, para distintas matrices de Laplace, usando 3 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

		Matriz de Laplace 4096-N				Matriz de Laplace 10000				Matriz de Laplace 16384			
q	ω	Iter.	T.Sec.	T.Par.	Efic.	Iter.	T.Sec.	T.Par.	Efic.	Iter.	T.Sec.	T.Par.	Efic.
1	1	4513	9,28	10,75	0,21	10581	55,74	40,82	0,34	17062	99,89	92,39	0,27
2		2593	8,66	7,01	0,30	5939	51,46	28,02	0,46	9485	87,43	64,21	0,34
5		1399	10,10	5,14	0,49	3003	56,58	21,87	0,65	4661	89,34	51,39	0,44
9		1048	12,95	5,25	0,62	2113	68,48	22,83	0,75	3180	102,21	53,16	0,48
1	1,2	3220	6,62	7,65	0,22	7483	39,47	28,89	0,34	12021	74,54	64,38	0,29
2		1925	6,43	5,19	0,31	4314	27,37	20,33	0,34	6825	52,32	46,28	0,28
5		1134	8,17	4,18	0,49	2334	43,95	16,99	0,65	3551	48,23	39,04	0,31
9		905	12,95	4,53	0,71	2113	68,5	22,67	0,76	3180	55,65	52,92	0,26
1	1,4	2256	4,63	5,34	0,22	5170	27,2	19,94	0,34	8256	53,43	44,46	0,30
2		1447	4,85	3,89	0,31	3133	27,13	14,79	0,46	4881	66,32	33,07	0,50
5		948	6,83	3,49	0,49	1857	34,97	13,51	0,65	2751	79,99	30,36	0,65
9		808	9,98	4,03	0,62	1484	48,08	15,94	0,75	2119	99,12	35,40	0,70
1	1,6	1500	3,09	3,58	0,22	3351	17,63	12,91	0,34	5294	34,45	28,38	0,30
2		1099	3,67	2,97	0,31	2248	19,46	10,58	0,46	3408	22,21	23,11	0,24
5		815	5,86	2,99	0,49	1504	28,31	10,97	0,65	2154	39,11	23,76	0,41
9		740	9,14	3,69	0,62	1296	42,01	13,86	0,76	1796	48,21	29,98	0,40
1	1,8	878	1,80	2,08	0,22	1854	9,73	7,19	0,34	2854	24,56	18,44	0,33
2		849	2,83	2,27	0,31	1584	13,68	7,47	0,46	2277	29,21	15,41	0,47
5		721	5,19	2,66	0,49	1239	23,32	9,02	0,65	1701	37,21	18,72	0,50
9		554	8,56	3,21	0,67	1160	37,58	12,41	0,76	1558	59,01	26,03	0,57
1	1,9	601	1,22	1,41	0,22	1186	6,21	4,56	0,34	1761	14,32	9,39	0,38
2		751	2,52	2,03	0,31	1351	11,68	6,38	0,46	1819	24,22	12,27	0,47
5		689	4,95	2,54	0,49	1123	21,15	8,15	0,65	1522	48,43	16,92	0,72
9		445	8,32	2,98	0,70	1108	35,92	11,91	0,75	1468	70,11	24,46	0,72

Tabla 18: *Comparando el método por bloques en dos etapas con iteraciones internas SOR, para distintas matrices de Laplace, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

7.4.11 Sobre la matriz del operador biarmónico

En la Sección 7.3 hemos planteado dos problemas modelo, correspondientes a la matriz de Laplace y a la matriz que hemos denominado Biarmónica. Todos los resultados obtenidos hasta el momento se han dedicado al primer problema. A continuación, en la Tabla 19 ilustramos el comportamiento de estos métodos para matrices Biarmónicas de tamaño 4096 y 10000 sobre el IBM RS/6000 SP. Como puede apreciarse, para conseguir convergencia mediante los métodos por bloques en dos etapas para estas matrices es necesario realizar demasiadas iteraciones, lo que produce tiempos de ejecución excesivamente altos. Es por esto que no nos vamos a extender más en la exposición de resultados para estas matrices en esta sección. Volveremos a ellas en la Sección 7.5, donde cabe esperar que, al utilizar precondicionadores paralelos, los resultados sean satisfactorios.

		$\omega = 1,2$		$\omega = 1,4$		$\omega = 1,6$	
	q	Iter.	T.Par.	Iter.	T.Par.	Iter.	T.Par.
4096	2	451488	2222,18	368562	1812,82	313507	1543,73
	3	373215	2497,43	318090	2125,96	279363	1869,03
	4	334473	2829,03	293481	2482,21	263936	2231,12
	6	296216	3556,91	269384	3234,95	249737	2997,69
10000	2	3916601	26942,73	1878772	21525,19	1499680	17189,03
	3	2417216	26792,23	1550807	24834,74	1288571	20630,13

Tabla 19: *Método por bloques en dos etapas con iteraciones internas SSOR. Resultados para distintas matrices Biarmónicas, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\|x^{(l+1)} - x^{(l)}\|_1 < 10^{-2}$.

7.5 Implementación de los precondicionadores paralelos

7.5.1 Introducción

En esta sección vamos a exponer los resultados obtenidos para el método del gradiente conjugado precondicionado, calculando el precondicionador de las dos formas vistas en el Capítulo 6.

Para construir dichos precondicionadores, consideramos que la matriz A está dividida en $r \times r$ bloques con los bloques diagonales de tamaño n_j, $\sum_{j=1}^{r} n_j = n$, de tal forma que el sistema $Ax = b$ puede ser escrito de la forma

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1r} \\ A_{21} & A_{22} & \cdots & A_{2r} \\ \vdots & \vdots & & \vdots \\ A_{r1} & A_{r2} & \cdots & A_{rr} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_r \end{bmatrix}, \tag{7.12}$$

donde x y b son vectores que están divididos de acuerdo al tamaño de los bloques de A.

Precondicionador basado en la técnica en dos etapas

Los resultados de convergencia para este precondicionador, se basan en particiones P-regulares de una matriz simétrica. En los Teoremas 6.1 y 6.2 se supone que en la partición $A = M - N$, la matriz N es semidefinida positiva. Para asegurar esta condición en la práctica, podemos elegir la siguiente partición de A:

- Sea $A = \tilde{M} - \tilde{N}$ la partición de Jacobi por bloques de la matriz A, es decir, $\tilde{M}$ es de la forma

$$\tilde{M} = \text{diag}(A_{11}, \ldots, A_{rr}). \tag{7.13}$$

- Consideramos una matriz cuadrada diagonal no negativa D, de tamaño n, definida de la forma

$$D = \text{diag}\left(\sum_{j=1, j=1}^{n} |\tilde{n}_{1j}|, \ldots, \sum_{j=1, j=n}^{n} |\tilde{n}_{nj}| \right).$$

- Construimos la partición $A = M - N$, donde

$$M = \text{diag}(A_{11}, \ldots, A_{rr}) + D = \text{diag}(M_1, \ldots, M_r), \tag{7.14}$$

$$N = \tilde{N} + D. \tag{7.15}$$

La partición así construida satisface las hipótesis de los Teorema 6.1 y 6.2 (Sección 6.2), pues la matriz M es hermítica y la matriz N es semidefinida positiva.

Para la elección de la partición que utilizaremos para aproximar los sistemas internos tenemos que hacer una serie de consideraciones previas según resolvamos el problema de Laplace o el de la ecuación Biarmónica. Sabemos que la partición que induce el método SSOR es P-regular siempre que se obtenga sobre una matriz simétrica y definida positiva. Por otro lado, la partición de Jacobi, en general, no tiene por que ser P-regular, aunque se obtenga de una matriz simétrica y definida positiva. Como las matrices de Laplace son además irreduciblemente diagonal dominantes, por los Teoremas 1.13 y 1.14, la partición de Jacobi sí es en este caso P-regular. Sin embargo, para la matriz Biarmónica la

partición de Jacobi resultante no es P-regular, hecho que hemos contrastado, experimentalmente, con diversos ejemplos. Por tanto, las particiones de Jacobi y SSOR verificarán los Teoremas 6.1 y 6.2 para la matriz de Laplace, mientras que para la matriz Biarmónica, sólo los verificará la partición obtenida con el método SSOR. Así, en los resultados que presentaremos, para la aproximación del sistema interno, utilizaremos el método de Jacobi y el método SSOR cuando utilicemos las matrices de Laplace y el método SSOR para las matrices Biarmónicas.

Precondicionador basado en la técnica de dos etapas con factorización incompleta de Choleski como partición interna.

Para el cálculo de este precondicionador se utiliza la técnica en dos etapas tomando, en primer lugar, como partición externa la partición de Jacobi, es decir,

$$A = M - N, \text{ donde } M = \text{diag}(A_{11}, \ldots, A_{rr}). \tag{7.16}$$

Después se eligen subconjuntos no ceros G de índices definidos en el Teorema 6.3 (Sección 6.2). Estos subconjuntos consistirán en variar el número de diagonales no cero de las matrices L_j de la descomposición incompleta de Choleski. En el algoritmo que daremos a continuación, estos subconjuntos vienen referenciados por los parámetros $N1$ y $N2$, donde $N1$ será el número de diagonales que se factorizarán a partir de la diagonal principal y $N2$ será el número de subdiagonales que se factorizarán desde la última subdiagonal distinta de cero hasta la diagonal principal. Para cada bloque A_{jj} se considera una partición basada en la factorización incompleta de Choleski de una matriz simétrica y definida positiva, teniendo en cuenta la elección del conjunto G de la factorización, es

decir, las particiones internas se definen ahora como

$$A_{jj} = B_j - C_j, \quad \text{con} \quad B_j = L_j L_j^T \quad \text{y} \quad C_j = O, \tag{7.17}$$

donde L_j es una matriz triangular inferior. La partición así elegida satisface las hipótesis de los Teoremas 6.4 y 6.5.

Puesto que L_j es no singular, la resolución del sistema $\mathcal{M}s = \tau$, se llevará a cabo resolviendo estos dos sistemas triangulares

$$L_j x_j^{(l)} = s_j^{(l)}, \quad L_j^T s_j^{(l)} = \tau_j^{(l)}, \quad 1 \leq j \leq r.$$

La factorización incompleta puede calcularse independientemente por cada procesador, es decir, cada procesador calcula una matriz triangular inferior L_j tal que $L_j L_j^T = B_j$. Como los elementos de B_j cuyos índices están en el conjunto no cero G coinciden con los correspondientes elementos de A_{jj} y $A_{jj} = B_j - C_j$, calcularemos sólo aquellos elementos de la matriz C_j cuyos índices no pertenezcan al conjunto G.

Los resultados que presentaremos para contrastar el uso de este precondionador hacen referencia al problema de la ecuación de Laplace. Hacemos notar que la matriz de la ecuación Biarmónica no es una matriz de $Z^{n\times n}$ y por tanto no tiene porqué cumplir los Teoremas 6.4 y 6.5. De hecho, se ha comprobado con diversos ejemplos que la partición obtenida con la descomposición incompleta de Choleski no es regular (ver Teorema 6.3) y que el precondicionador no es válido.

A continuación definimos el algoritmo para la ejecución en paralelo del método del gradiente conjugado precondicionado, donde tendremos que tener

en cuenta el método utilizado para el cálculo de la matriz precondicionadora $\mathcal{M}$, a la hora de resolver el sistema $\mathcal{M}s = \tau$.

Algoritmo 7.2. Algoritmo paralelo para el método del gradiente conjugado precondicionado

Etapa 0 $\rightsquigarrow$ Declaración de datos:

Matriz de coeficientes: A.

Término independiente: b.

Número de subdiagonales: $N1$.

Número de subdiagonales: $N2$.

Factor de relajación: ω.

Etapa 1 $\rightsquigarrow$ Elegir los vectores iniciales: $x^{(0)}$, $s^{(0)}$ y $l = 0$.

Asignar n_j, $1 \leq j \leq r$ filas de la matriz A a cada procesador de manera que cada uno dispone de las filas representadas por $A_j \equiv [A_{j1}, A_{j2}, \ldots, A_{jr}]$.

Cada procesador j, identifica su bloque diagonal A_{jj} para considerar una partición de la forma dada en (7.14) y (7.15), o en (7.16).

En paralelo calcular:

- el primer residuo $\tau_j^0 = b_j - A_j x_j^{(0)}$.
- resolver el sistema $\mathcal{M}s^{(0)} = \tau^{(0)}$, según método (ver Nota-1).
- la primera dirección conjugada $p^{(0)} = s^{(0)}$.
- el producto $\vartheta_j^{(l)} = \langle s_j^{(l)}, \tau_j^{(l)} \rangle$ y comunicar.

Etapa 2 $\rightsquigarrow$ En paralelo calcular: $v_j^{(l)} = \langle p_j^{(l)}, A_j p^{(l)} \rangle$ y comunicar.

Calcular $\alpha^{(l)} = -\dfrac{\sum_{j=1}^{r} \vartheta_j^{(l)}}{\sum_{j=1}^{r} v_j^{(l)}}$ y comunicar.

Calcular nueva solución $x^{(l+1)} = x^{(l)} - \alpha^{(l)} x^{(l)}$.

En paralelo calcular:

- nuevo residuo $\tau_j^{(l+1)} = \tau_j^{(l)} - \alpha^{(l)} A_j p^{(l)}$.
- resolver el sistema $\mathcal{M} s^{(l+1)} = \tau^{(l+1)}$ según método (ver Nota-2).
- $\vartheta_j^{(l)} = \langle s_j^{(l)}, \tau_j^{(l)} \rangle$
- Comunicar $\vartheta_j^{(l+1)}$ y $\tau_j^{(l+1)}$.

Etapa 3 $\rightsquigarrow$ Realizar el test de convergencia:

Si $\sum_{j=1}^{r} \vartheta_j^{(l+1)} < \epsilon$ entonces

Calcular $\xi^{(l+1)} = \langle \tau^{(l+1)}, \tau^{(l+1)} \rangle$.

Si $\xi^{(l+1)} < \epsilon$ entonces FIN.

Calcular $\beta = -\dfrac{\sum_{j=1}^{r} \vartheta_j^{(l+1)}}{\sum_{j=1}^{r} \vartheta_j^{(l)}}$.

Calcular nueva dirección conjugada $p^{(l+1)} = s^{(l+1)} - \beta p^{(l)}$ y comunicar.

$l \longleftarrow l + 1$.

Ir a la Etapa 2.

Nota-2: La resolución del sistema $\mathcal{M}s = \tau$, del Algoritmo 7.2, se llevará a cabo según la forma en que elijamos el cálculo del precondicionador.

Prácticamente todos los resultados presentados se han obtenido en el IBM RS/6000 SP (ver descripción en Sección 7.2) para diferentes tamaños de matrices de Laplace y matrices Biarmónicas.

Enumeramos a continuación, las matrices con las que trabajaremos en el resto de esta sección, el vector inicial $x^{(0)}$, el término independiente considerado y el criterio de parada usado:

- Matriz de coeficientes A:

  Laplace: tamaños 1024, 4096, 10000, 16384, 40000 y 262144 (ver Tabla 1).

  Biarmónica: tamaños 1024, 10000, 16384 y 40000 (ver Tabla 2).

- Vector inicial:

  Laplace: $x^{(0)} = (0, \ldots, 0)^T$.

  Biarmónica: $x^{(0)} = (0, \ldots, 0)^T$.

- Término independiente:

  Laplace: $b = (b_1^T, b_2^T, \ldots, b_J^T)^T,\ b_i \in \mathbb{R}^K,\ b_i = (0, 0, \ldots, 100)^T$.

  Biarmónica: $b = (1, \ldots, 1)^T$.

- Criterio de parada: $\langle \tau, \tau \rangle < \epsilon,\ \epsilon = 10^{-5}, 10^{-7}$, donde τ es el residuo que se comete en cada iteración ($\tau = Ax - b$).

A continuación pasamos a presentar los resultados numéricos más representativos, del método del gradiente conjugado precondicionado, utilizando las dos técnicas descritas anteriormente.

7.5.2 Resultados del método GCP basado en la técnica en dos etapas

En esta sección presentamos los resultados numéricos del método del gradiente conjugado precondicionado (GCP), cuyo precondicionador se calcula aplicando el método en dos etapas utilizando como particiones internas las obtenidas a partir de los métodos de Jacobi y SSOR. Denotaremos a estos métodos GCP/2E-Jacobi y GCP/2E-SSOR, respectivamente.

7.5.2.1 Sobre el número de condición

Uno de los criterios generales para la elección de la matriz precondicionadora $\mathcal{M}$, en el método del gradiente conjugado precondicionado, es elegirla de forma que la matriz $\hat{A} = SAS^T$ tenga número de condición mucho menor que A. Se observa que la matriz $\hat{A}$ satisface $S^T\hat{A}S^{-T} = S^TSA = \mathcal{M}^{-1}A$, y por tanto, las matrices $\hat{A}$ y $\mathcal{M}^{-1}A$ son semejantes. De aquí que $\mathrm{cond}(\hat{A}) = \frac{\lambda_{max}(\mathcal{M}^{-1}A)}{\lambda_{min}(\mathcal{M}^{-1}A)}$. Puesto que el objetivo del precondicionador es conseguir que la matriz $\hat{A}$ tenga menor número de condición que A, se debe elegir $\mathcal{M}$ de forma que sea simétrica

y definida positiva y tal que la razón entre el mayor y el menor valor propio de $\mathcal{M}^{-1}A$ sea lo más próximo a 1.

En la Sección 6.2 hemos demostrado que el precondicionador que nos ocupa es simétrico y definido positivo. Nos interesa, por tanto, ver cuál es su comportamiento respecto al número de condición. Para ello, damos a continuación una serie de ejemplos ilustrativos.

En las Tablas 20 y 21 se han considerado dichos precondicionadores, utilizando iteraciones internas Jacobi y Gauss-Seidel simétrico, respectivamente, y siendo A la matriz de Laplace de tamaño 1024. Dichas tablas presentan el número de condición de la matriz del sistema precondicionado $\hat{A}$. El número de condición se ha calculado atendiendo al número de iteraciones internas q realizadas y al número de pasos del precondicionador, para $r = 2$ y 4 procesadores. En ambos casos se ha supuesto que el número de filas asignadas a cada procesador es el mismo. De forma análoga, en la Tabla 22 se ilustra el comportamiento de estos precondicionadores para la matriz Biarmónica de tamaño 1024, utilizando iteraciones internas Gauss-Seidel simétrico.

Estos resultados se han calculado utilizando el programa de cálculo matricial MATLAB. El número de condición obtenido para la matriz de Laplace ha sido 440,68 y para la matriz Biarmónica, 65549,09.

Como puede apreciarse en la Tabla 20, cuando se usa el precondicionador con iteraciones internas Jacobi, el número de condición de $\hat{A}$ es menor que el número de condición de A salvo cuando se realiza un único paso y una única iteración interna. Por otro lado, se puede observar, en esta misma tabla, que para un número de pasos par fijo, el número de condición decrece conforme aumentamos el número de iteraciones internas, mientras que para un número

2 PROCESADORES			4 PROCESADORES		
Número de pasos (m)	q	Número de Condición	Número de pasos (m)	q	Número de Condición
1	1	452,64	1	1	460,55
	2	118,57		2	126,52
	3	162,49		3	172,78
	4	64,90		4	73,87
	5	105,85		5	63,69
2	1	114,05	2	1	117,54
	2	59,53		2	63,50
	3	41,56		3	45,84
	4	32,70		4	37,18
	5	27,46		5	32,09
3	1	149,85	3	1	148,71
	2	39,85		2	42,51
	3	53,05		3	52,74
	4	21,97		4	24,96
	5	34,06			21,56
4	1	57,27	4	1	59,02
	2	30,02		2	32,01
	3	21,03		3	23,17
	4	16,60		4	18,84
	5	13,98		5	16,30
5	1	89,30	5	1	86,56
	2	24,11		2	25,71
	3	31,20		3	29,33
	4	13,38		4	15,18
	5	19,74		5	13,14
6	1	38,35	6	1	39,51
	2	20,18		2	21,51
	3	14,19		3	15,62
	4	11,24		4	12,73
	5	9,49		5	11,04

Tabla 20: *Número de condición de la matriz $\hat{A}$, usando iteraciones internas Jacobi. Matriz de Laplace 1024, cond(A)=440,68.*

2 PROCESADORES			4 PROCESADORES		
Número de pasos (m)	q	Número de Condición	Número de pasos (m)	q	Número de Condición
1	1	66,67	1	1	76,89
	2	39,84		2	50,03
	3	31,53		3	41,80
	4	27,66		4	37,96
	5	25,47		5	35,79
2	1	35,59	2	1	38,69
	2	20,17		2	25,26
	3	16,02		3	21,15
	4	14,08		4	19,23
	5	12,99		5	18,14
3	1	22,56	3	1	25,96
	2	13,62		2	17,01
	3	10,85		3	14,27
	4	9,56		4	12,99
	5	8,83		5	12,26
4	1	17,04	4	1	19,60
	2	10,34		2	12,88
	3	8,26		3	10,83
	4	7,30		4	9,87
	5	6,75		5	9,33
5	1	13,74	5	1	15,78
	2	8,37		2	10,41
	3	6,72		3	8,77
	4	5,94		4	8
	5	5,51		5	7,56
6	1	11,53	6	1	13,23
	2	7,07		2	8,76
	3	5,68		3	7,39
	4	5,04		4	6,75
	5	4,68		5	6,39

Tabla 21: *Número de condición de la matriz $\hat{A}$, usando iteraciones internas Gauss-Seidel simétrico. Matriz de Laplace 1024, cond(A)=440,68.*

2 PROCESADORES			4 PROCESADORES		
Número de pasos (m)	q	Número de Condición	Número de pasos (m)	q	Número de Condición
1	1	7227,35	1	1	8669,72
	2	4686,32		2	6116,39
	3	3904,07		3	5359,72
	4	3532,66		4	5008,51
	5	3317,80		5	4808,51
2	1	3613,92	2	1	4335,11
	2	2343,41		2	3058,44
	3	1952,28		3	2680,11
	4	1766,58		4	2504,50
	5	1659,15		5	2404,50
3	1	2409,45	3	1	2890,24
	2	1562,44		2	2039,13
	3	1301,69		3	1786,90
	4	1177,88		4	1669,83
	5	1106,26		5	1603,17
4	1	1807,21	4	1	2167,80
	2	1171,95		2	1592,47
	3	976,39		3	1340,30
	4	883,54		4	1252,50
	5	829,82		5	1202,50
5	1	1445,87	5	1	1734,34
	2	937,66		2	1223,67
	3	781,21		3	1072,34
	4	706,93		4	1002,10
	5	663,96		5	962,10
6	1	1204,97	6	1	1445,37
	2	781,47		2	1019,81
	3	651,09		3	893,70
	4	589,19		4	835,16
	5	553,38		5	801,83

Tabla 22: *Número de condición de la matriz $\hat{A}$, usando iteraciones internas Gauss-Seidel simétrico. Matriz Biarmónica 1024, cond(A)=65549,09.*

de pasos impar el número de condición va disminuyendo o aumentando según sea el número de iteraciones internas par o impar, respectivamente. Además, el número de condición de $\hat{A}$, utilizando un precondicionador con un número impar de pasos, cuando realizamos también un número impar de iteraciones internas, es en todos los casos mayor que el número de condición de $\hat{A}$ para el precondicionador previo con un número par de pasos y el mismo número de iteraciones internas.

Por otro lado, como se observa en las Tablas 21 y 22, cuando se usa el precondicionador con iteraciones internas Gauss-Seidel simétrico, el número de condición de $\hat{A}$ es siempre menor que el número de condición de A. En este caso, el número de condición decrece conforme aumentamos el número de iteraciones internas independientemente de que el número de pasos sea par o impar. Esto es lógico que ocurra a la vista de lo que hemos observado para el caso de iteraciones internas Jacobi, ya que efectuar iteraciones Gauss-Seidel simétrico supone realizar dos iteraciones (un número par) del tipo Gauss-Seidel. Además, el número de condición también decrece al aumentar el número de pasos.

Cabe destacar, además, que al aumentar el número de procesadores (es decir, el número de bloques diagonales en la partición tipo Jacobi), también aumenta el número de condición del precondicionador correspondiente.

7.5.2.2 Comportamiento del método GCP con iteraciones internas Jacobi

En esta sección estudiamos el comportamiento del método del gradiente conjugado precondicionado (GCP), cuyo precondicionador se calcula aplicando el método por bloques en dos etapas, usando para las etapas internas el método iterativo de Jacobi (GCP/2E-Jacobi). Como matrices de prueba utilizaremos las matrices de Laplace ya que, tal y como hemos visto en la Sección 7.5.1, este precondicionador no es válido para las matrices Biarmónicas. En la Tabla 23 se presentan los resultados obtenidos para la matriz de Laplace 4096-N. La cota utilizada en este caso ha sido $\langle \tau, \tau \rangle < 10^{-5}$. Dichos resultados se han obtenido en el Cluster de Pentiums y en el IBM RS/6000 SP. Como se puede apreciar, el comportamiento en ambas plataformas es análogo, es decir, los tiempos disminuyen a medida que aumentamos el número de iteraciones internas hasta llegar a un cierto valor óptimo. Obviamente, la velocidad de comunicación y de CPU del Cluster es mucho menor que la del IBM RS/6000 SP, como hemos puesto de manifiesto en la Sección 7.4.9 y por tanto los tiempos aumentan considerablemente de utilizar una u otra plataforma. Estos resultados se han comparado en la misma tabla, con el método resultante de aplicar como precondicionador un Jacobi por bloques (GCP/Bl-Jacobi); en este caso, los subsistemas diagonales se resuelven utilizando la factorización completa de Choleski. Como se puede apreciar, con matrices pequeñas, el método GCP/Bl-Jacobi ya obtiene peores resultados. Tengamos en cuenta que al utilizar dicho método se va a consumir gran parte del tiempo de CPU en realizar la factorización completa en cada bloque diagonal. No obstante, en las Tablas 24 y 25, donde se muestran resultados

r	m	GCP/2E-Jacobi				GCP/Bl-Jacobi		
		q	Iter.	Tiempo Cluster	Tiempo SP2	Iter.	Tiempo Cluster	Tiempo SP2
3	1	1	102	5,93	0,56			
	1	4	47	3,18	0,31	19	16,47	9,01
	2	1	52	4,12	0,36			
	2	4^3	32	3,02	0,32	11	16,40	8,06
4	1	1	101	7,21	0,63			
	1	4	38	3,04	0,27	21	9,58	4,01
	2	1	51	4,95	0,41			
	2	4	27	3,12	0,27	14	9,72	4,02
	3	4	22	3,36	0,29	13	10,28	4,08

Tabla 23: *Tiempos paralelos del método GCP en dos etapas con iteraciones internas mediante el método de Jacobi. Matriz de Laplace 4096-N. Cota de parada* $\langle \tau, \tau \rangle < 10^{-5}$.

análogos a los anteriores para la matriz de Laplace de tamaño 10000 usando 2 y 4 procesadores, se puede apreciar que incluso descontando el tiempo que utiliza un procesador en realizar dicha factorización en su bloque, el precondicionador que nos ocupa consigue obtener mejores resultados.

Atendiendo al comportamiento del método, hemos observado que los mejores tiempos se obtienen para $m = 1$ o $m = 2$ pasos del precondicionador, y siempre con un número par de iteraciones internas. Por otro lado, conforme aumentan los procesadores, observamos que como el método GCP/2E-Jacobi consume poco tiempo de CPU, predomina el tiempo de comunicación sobre el de cálculo y por tanto, por regla general, como un aumento de procesadores conlleva más tiempo de comunicación, esto nos podría hacer pensar que no interesará utilizar muchos procesadores. No obstante, como al aumentar el número de iteraciones internas se reduce el tiempo de comunicación, se observa que conforme aumenta

el tamaño de la matriz, (ver Tablas 26 y 27) el mejor tiempo en paralelo con 4 procesadores es menor que el mejor con dos procesadores, obteniendo una reducción en el tiempo de ejecución de aproximadamente un 20%.

Además, queremos hacer notar, que dado un número de iteraciones internas, se observa en determinados casos, que cuando el número de pasos es impar, el tiempo de convergencia es mayor respecto al paso previo par, tal y como muestran las Tablas 26 y 27, para la matriz Laplace de tamaño 16384. Este aumento de tiempo es debido a que el número de iteraciones con dicho número de pasos impar es también mayor respecto al número de iteraciones en el paso previo par. Esto sucede porque como el número de condición de la matriz $\hat{A} = SAS^T$ es similar al de la matriz $\mathcal{M}_m^{-1}A$. Entonces, $\text{cond}(\hat{A}) = \frac{1-\lambda_{min}(T^m)}{1-\lambda_{max}(T^m)}$, donde $\lambda_{min}(T^m)$ y $\lambda_{max}(T^m)$ son respectivamente el mínimo y el máximo valor propio de T^m. Entonces, si T tiene los valores propios negativos y m es impar, el numerador de $\text{cond}(\hat{A})$ es mayor que 1. Sin embargo, si m es par, el numerador es siempre menor que 1. Por lo tanto, debemos esperar más decrecimiento en el número de condición de A, para valores pares de m (ver la Tabla 20 de la Subsección 7.5.2.1, donde se ilustra el comportamiento del número de condición para este tipo de matrices).

			GCP/2E-Jacobi		GCP/Bl-Jacobi		
r	m	q	Iter.	Tiempo	Iter.	Tiempo	Tiempo de FCC
2	1	1	242	3,55			108,1
		2	122	1,92			
		3	142	2,45	28	217,51	
		4	87	1,62			
		5	111	2,22			
		6	73	1,58			
2	2	1	121	2,45			
		2	86	2,11			
		3	70	2	19	110,01	
		4	61	2,02			
		5	55	2,04			
		6	52	2,15			
2	3	1	140	3,69			
		2	70	2,34			
		3	81	3,25	16	110,67	
		4	50	2,37			
		5	63	3,42			
		6	42	2,58			
2	4	1	85	2,74			
		2	61	2,6			
		3	50	2,6	14	111,22	
		4	44	2,73			
		5	39	2,79			
		6	37	3,07			

Tabla 24: *Tiempos paralelos del método GCP/2E-Jacobi. Matriz de Laplace 10000, usando 2 procesadores. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

			GCP/2E-Jacobi		GCP/Bl-Jacobi		
r	m	q	Iter.	Tiempo	Iter.	Tiempo	Tiempo de FCC
4	1	1	242	4,65			26,26
		2	123	2,46			
		3	144	2,95	39	53,84	
		4	90	1,92			
		5	115	2,51			
		6	76	1,72			
4	2	1	120	3,14			
		2	86	2,44			
		3	72	2,17	22	55,95	
		4	63	2,02			
		5	57	1,97			
		6	53	2,07			
4	3	1	139	4,61			
		2	70	2,59			
		3	83	3,29	23	57,22	
		4	51	2,22			
		5	66	3,08			
		6	44	2,22			
4	4	1	85	3,39			
		2	61	2,72			
		3	51	2,54	16	28,36	
		4	45	2,44			
		5	41	2,42			
		6	38	3,65			

Tabla 25: *Tiempos paralelos del método GCP/2E-Jacobi. Matriz de Laplace 10000, usando 4 procesadores. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

GCP/2E-Jacobi				
r	m	q	Iter.	Tiempo
2	1	1	307	6,84
		2	154	3,79
		3	180	4,86
		4	110	3,23
		5	141	4,45
		6	90	3,05
2	2	1	153	4,82
		2	109	4,2
		3	90	4,11
		4	78	4,11
		5	70	4,18
		6	60	4,08
2	3	1	178	7,3
		2	89	4,69
		3	104	6,67
		4	64	4,93
		5	80	7,07
		6	52	5,2
2	4	1	108	5,44
		2	77	5,15
		3	63	5,25
		4	55	5,48
		5	46	5,82
		6	42	6,15
2	5	1	137	8,2
		2	69	5,58
		3	80	8,17
		4	50	6,17
		5	62	8,95
		6	41	6,83

Tabla 26: *Tiempos paralelos del método GCP/2E-Jacobi. Matriz de Laplace 16384, usando 2 procesadores. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

GCP/2E-Jacobi				
r	m	q	Iter.	Tiempo
4	1	1	307	8,97
		2	155	4,67
		3	182	5,71
		4	113	3,67
		5	144	4,85
		6	94	3,26
4	2	1	153	6,08
		2	109	4,7
		3	90	4,19
		4	79	3,99
		5	72	3,9
		6	67	2,54
4	3	1	178	8,98
		2	89	5,51
		3	105	6,51
		4	65	4,44
		5	83	6,11
		6	54	4,31
4	4	1	108	6,62
		2	77	5,36
		3	64	4,99
		4	56	4,82
		5	51	4,81
		6	47	4,83
4	5	1	137	9,84
		2	69	5,73
		3	80	7,44
		4	51	5,29
		5	63	7,21
		6	42	5,27

Tabla 27: *Tiempos paralelos del método GCP/2E-Jacobi. Matriz de Laplace 16384, usando 4 procesadores. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

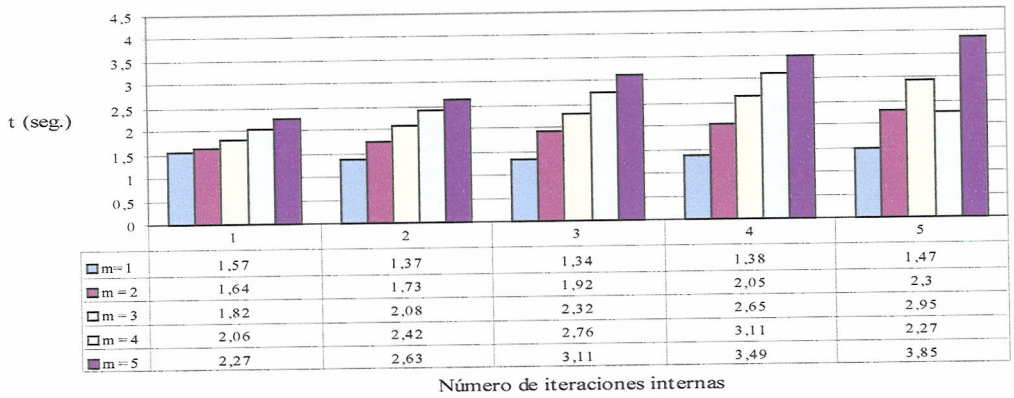
7.5.2.3 Comportamiento del método GCP con iteraciones Gauss-Seidel simétrico

En esta sección estudiamos el comportamiento del método del gradiente conjugado precondicionado (GCP), cuyo precondicionador se calcula aplicando el método por bloques en dos etapas, utilizando para las etapas internas el método iterativo Gauss-Seidel simétrico (GCP/2E-GSS). Como matrices de prueba usaremos las matrices de Laplace y las matrices Biarmónicas.

Las Figuras 35 y 36 muestran los resultados obtenidos para la matriz de Laplace de tamaño 10000 usando, respectivamente, 2 y 4 procesadores. En la Figura 35 se presentan los tiempos experimentales obtenidos, atendiendo al número de pasos del precondicionador y al número de iteraciones internas, y en la Figura 36 se indica el número de iteraciones globales para cada uno de los métodos de la Figura 35. Los tiempos obtenidos para la matriz de Laplace de tamaño 40000, usando 4 procesadores, se pueden ver en la Figura 37. La Figura 38 muestra los tiempos obtenidos para las matriz Biarmónica de tamaño 16384 usando 2 y 4 procesadores.

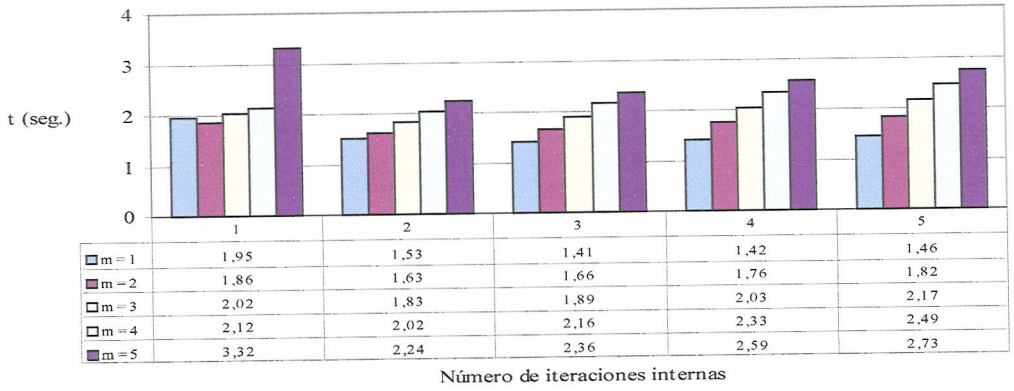
De estas figuras se deduce, que independientemente del número de iteraciones internas realizadas, el número de pasos óptimos parece ser 1, tanto para las matrices de Laplace como para las matrices Biarmónicas. Esto ha resultado ser independiente del número de procesadores utilizado, y del tamaño de los bloques asignados a cada procesador.

Sin embargo, la elección del número de iteraciones internas q depende del tamaño de la matriz o, más bien, del número de elementos no nulos que posea. Así, para un número de procesadores fijo, el factor no estacionario óptimo



	1	2	3	4	5
m=1	1,57	1,37	1,34	1,38	1,47
m=2	1,64	1,73	1,92	2,05	2,3
m=3	1,82	2,08	2,32	2,65	2,95
m=4	2,06	2,42	2,76	3,11	2,27
m=5	2,27	2,63	3,11	3,49	3,85

(a) 2 procesadores.



	1	2	3	4	5
m=1	1,95	1,53	1,41	1,42	1,46
m=2	1,86	1,63	1,66	1,76	1,82
m=3	2,02	1,83	1,89	2,03	2,17
m=4	2,12	2,02	2,16	2,33	2,49
m=5	3,32	2,24	2,36	2,59	2,73

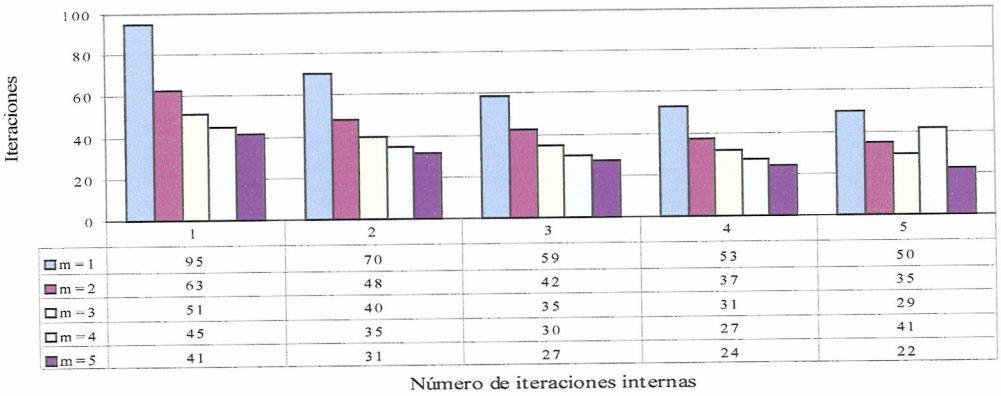
(b) 4 procesadores.

Figura 35: *Tiempos experimentales del método GCP/2E-GSS. Matriz de Laplace 10000, usando 2 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

aumenta ligeramente conforme aumenta el tamaño de la matriz. Por ejemplo, para $m = 1$ y 4 procesadores, la Figura 37 muestra que el mejor tiempo para la matriz de tamaño 40000 se obtiene para $q = 5$, mientras que para la matriz de tamaño 10000 el número de iteraciones internas óptimo es 3, tal y como se puede ver en la Figura 35(b).

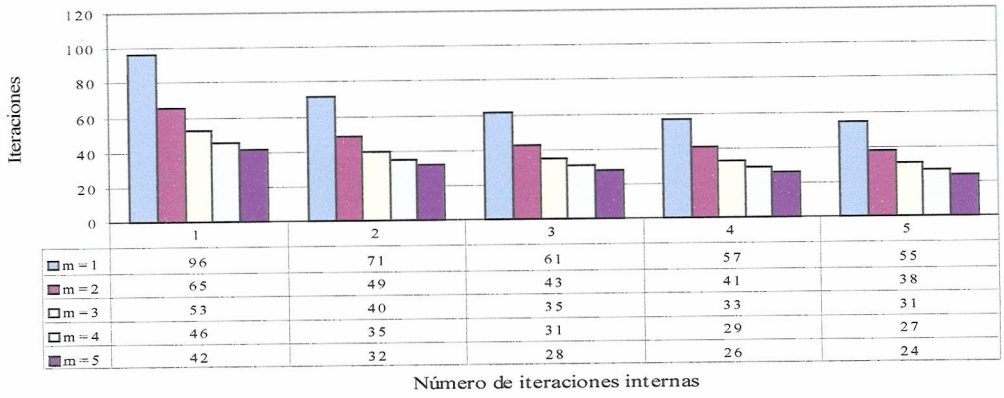
La Figura 39 muestra los resultados para la matriz de Laplace de tamaño 262144 usando 2 procesadores. Vemos en dicha figura, que cuando se realiza una única iteración interna los tiempos realizando 1 ó 2 pasos del precondicionador son similares, y conforme aumentamos el número de iteraciones internas, la diferencia de tiempos entre ellos también se incrementa. Como puede apreciarse en la Figura 40, el comportamiento, en este sentido, de las matrices Biarmónicas también es similar, aunque las diferencias de tiempos para 1 y 2 pasos ya son apreciables de forma clara para $q = 1$.

En las Figuras 39 y 40 aparece también el número de iteraciones globales necesarias en cada caso para obtener convergencia. Claramente, para un número de pasos fijo del precondicionador, el número de iteraciones globales disminuye conforme aumentamos el número de iteraciones internas. Por tanto, al igual que ocurría en los métodos por bloques en dos etapas, si esa reducción compensa la realización de más iteraciones internas, entonces se observa un menor tiempo de ejecución. Esto se ha podido constatar, a la vista de los resultados, ya que para una misma matriz al aumentar el número de procesadores, y por tanto al aumentar las comunicaciones, el q óptimo aumenta ligeramente.



	1	2	3	4	5
m = 1	95	70	59	53	50
m = 2	63	48	42	37	35
m = 3	51	40	35	31	29
m = 4	45	35	30	27	41
m = 5	41	31	27	24	22

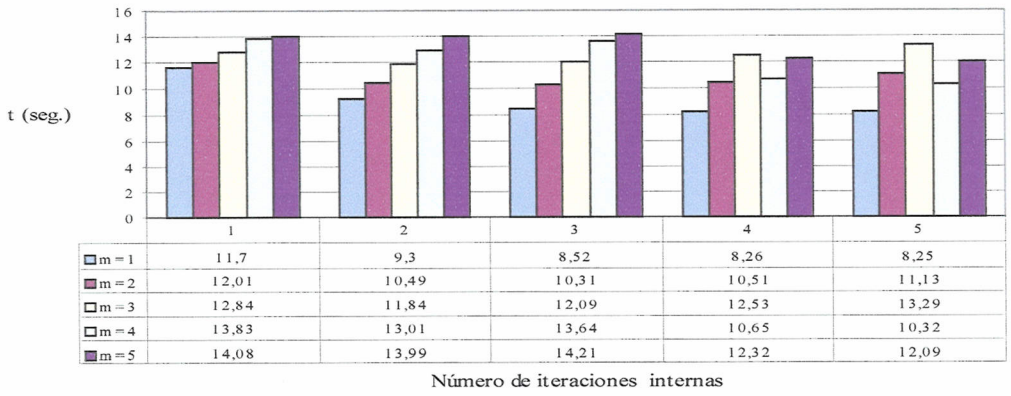
(a) 2 procesadores.



	1	2	3	4	5
m = 1	96	71	61	57	55
m = 2	65	49	43	41	38
m = 3	53	40	35	33	31
m = 4	46	35	31	29	27
m = 5	42	32	28	26	24

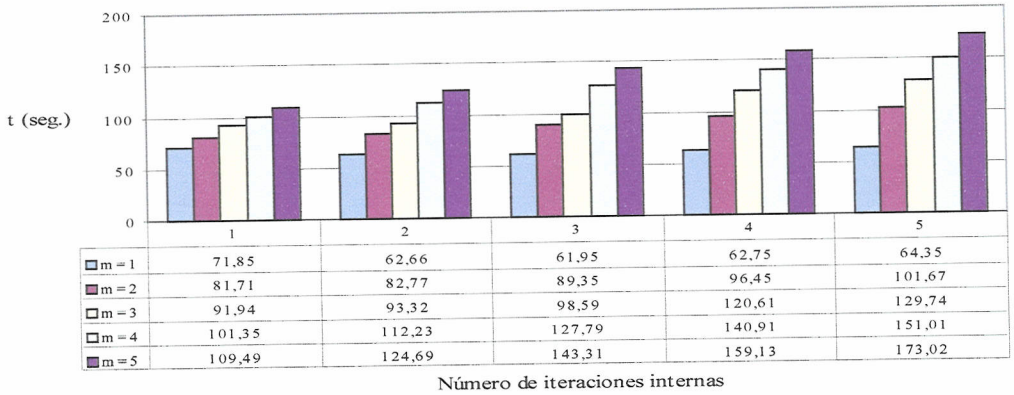
(b) 4 procesadores.

Figura 36: *Iteraciones para el método GCP/2E-GSS. Matriz de Laplace 10000, usando 2 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



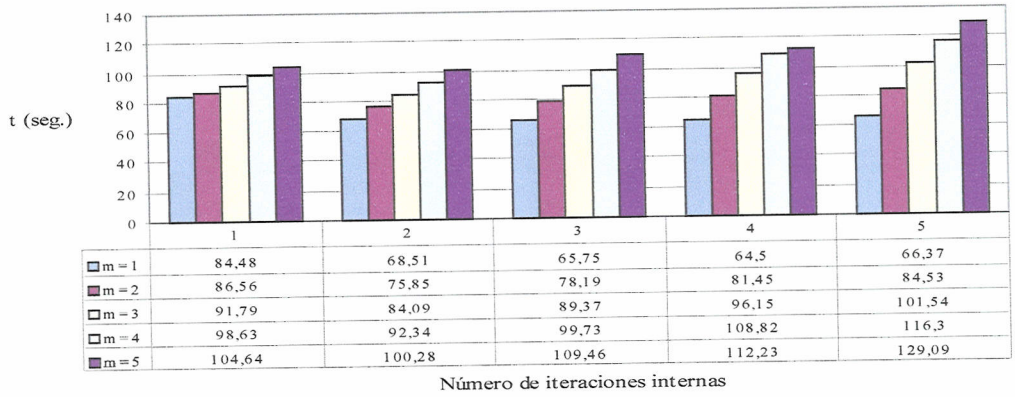
	1	2	3	4	5
m=1	11,7	9,3	8,52	8,26	8,25
m=2	12,01	10,49	10,31	10,51	11,13
m=3	12,84	11,84	12,09	12,53	13,29
m=4	13,83	13,01	13,64	10,65	10,32
m=5	14,08	13,99	14,21	12,32	12,09

Figura 37: *Tiempos experimentales del método GCP/2E-GSS. Matriz de Laplace 40000, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



	1	2	3	4	5
m = 1	71,85	62,66	61,95	62,75	64,35
m = 2	81,71	82,77	89,35	96,45	101,67
m = 3	91,94	93,32	98,59	120,61	129,74
m = 4	101,35	112,23	127,79	140,91	151,01
m = 5	109,49	124,69	143,31	159,13	173,02

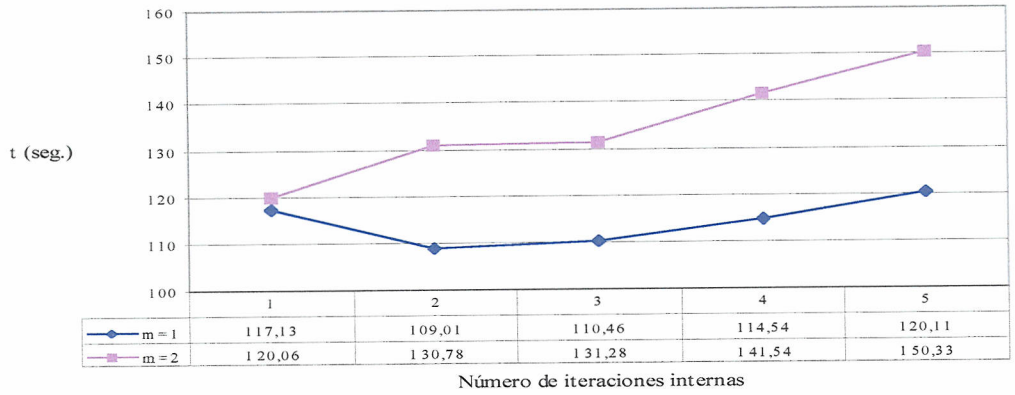
(a) 2 procesadores.



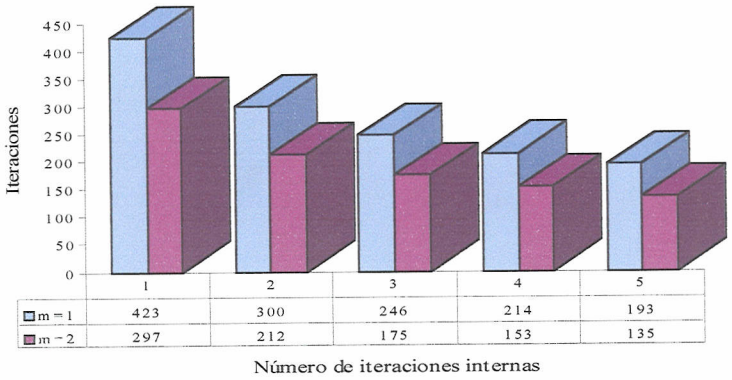
	1	2	3	4	5
m = 1	84,48	68,51	65,75	64,5	66,37
m = 2	86,56	75,85	78,19	81,45	84,53
m = 3	91,79	84,09	89,37	96,15	101,54
m = 4	98,63	92,34	99,73	108,82	116,3
m = 5	104,64	100,28	109,46	112,23	129,09

(b) 4 procesadores.

Figura 38: *Resultados del método GCP/2E-GSS. Matriz Biarmónica 16384, usando 2 y 4 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

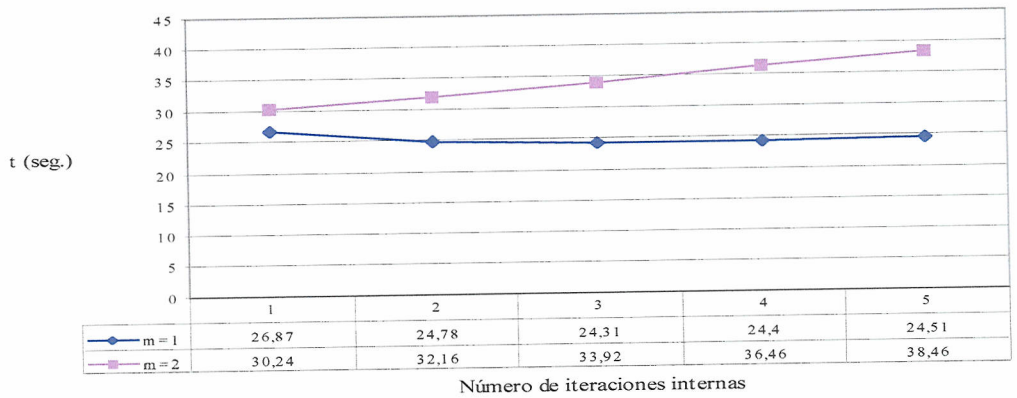


(a)

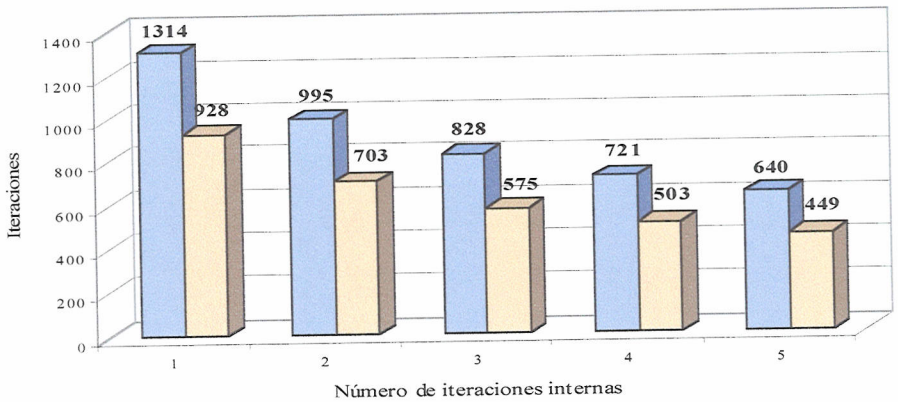


(b)

Figura 39: *Resultados del método del GCP/2E-GSS. Matriz de Laplace 262144, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



(a)



(b)

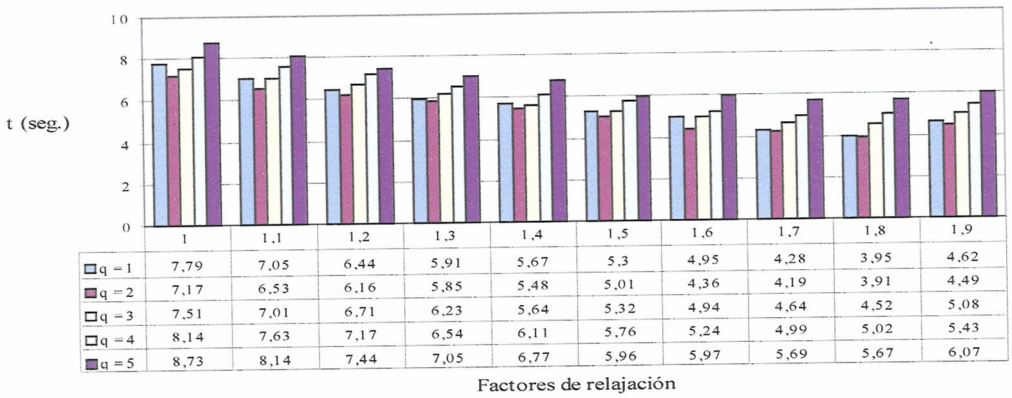
Figura 40: *Resultados del método GCP/2E-GSS. Matriz Biarmónica 10000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

7.5.2.4 Influencia del parámetro de relajación

Las Figuras 41 y 42 ilustran la influencia del parámetro de relajación ω en el tiempo de ejecución del Algoritmo 7.2 calculando el precondicionador mediante el método en dos etapas, con iteraciones internas SSOR (GCP/2E-SSOR), usando la matriz de Laplace de tamaño 40000 y distinto número de pasos del precondicionador. Para la obtención de los resultados de estas figuras se han utilizado 2 procesadores. Por otro lado, en las Figuras 43 y 44 se presentan resultados numéricos, análogos a los anteriores, para la matriz Biarmónica de tamaño 40000.

Como se puede observar, muchas de las conclusiones extraidas para $\omega = 1$, son generalizables al caso en que $\omega > 1$, aunque obviamente se consiguen mejores resultados con estos últimos. Así, como se refleja en estas figuras el número de pasos con el que se obtiene los mejores tiempos es con $m = 1$, tanto para las matrices de Laplace como para las matrices Biarmónicas. En este caso los tiempos bajan al aumentar el número de iteraciones internas hasta cierto valor óptimo a partir del cual empiezan a subir. Por otro lado, cuando $m > 1$, generalmente el mejor método se obtiene con $q = 1$. Tambien se observa cuando $m > 1$, que para las matrices de Laplace los tiempos suben generalmente conforme se aumenta el número de iteraciones internas. Sin embargo, en las matrices Biarmónicas esto empieza a ocurrir para valores más altos del factor de relajación (ver, por ejemplo, la Figura 43).

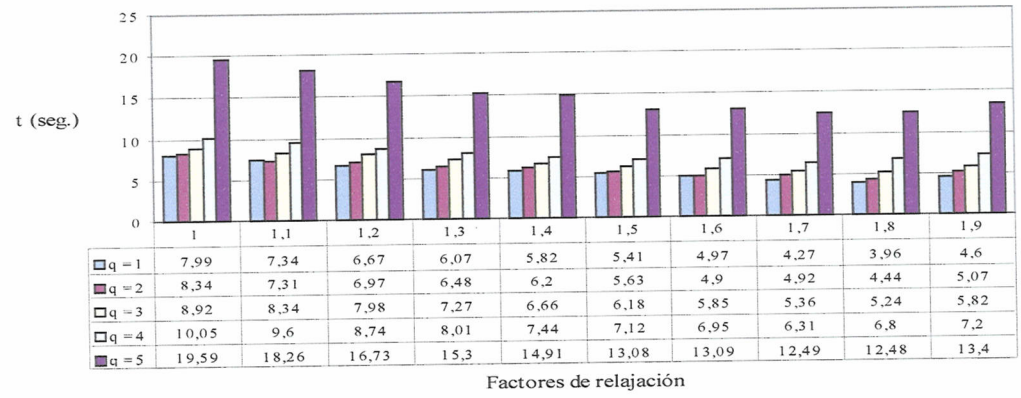
Cabe destacar también que, independientemente del factor de relajación, el número de iteraciones globales se reduce, al aumentar el número de pasos del precondicionador, como ocurría cuando se usaban iteraciones internas Gauss-Seidel.



	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	7,79	7,05	6,44	5,91	5,67	5,3	4,95	4,28	3,95	4,62
q = 2	7,17	6,53	6,16	5,85	5,48	5,01	4,36	4,19	3,91	4,49
q = 3	7,51	7,01	6,71	6,23	5,64	5,32	4,94	4,64	4,52	5,08
q = 4	8,14	7,63	7,17	6,54	6,11	5,76	5,24	4,99	5,02	5,43
q = 5	8,73	8,14	7,44	7,05	6,77	5,96	5,97	5,69	5,67	6,07

Factores de relajación

(a) Número de pasos del precondicionador: 1.

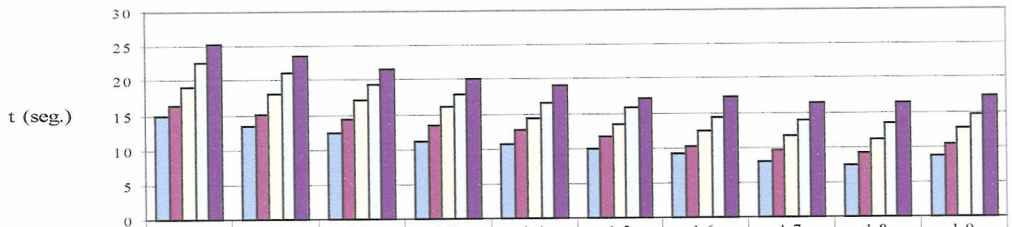


	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	7,99	7,34	6,67	6,07	5,82	5,41	4,97	4,27	3,96	4,6
q = 2	8,34	7,31	6,97	6,48	6,2	5,63	4,9	4,92	4,44	5,07
q = 3	8,92	8,34	7,98	7,27	6,66	6,18	5,85	5,36	5,24	5,82
q = 4	10,05	9,6	8,74	8,01	7,44	7,12	6,95	6,31	6,8	7,2
q = 5	19,59	18,26	16,73	15,3	14,91	13,08	13,09	12,49	12,48	13,4

Factores de relajación

(b) Número de pasos del precondicionador: 2.

Figura 41: *Resultados del método GCP/2E-SSOR. Matriz de Laplace 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

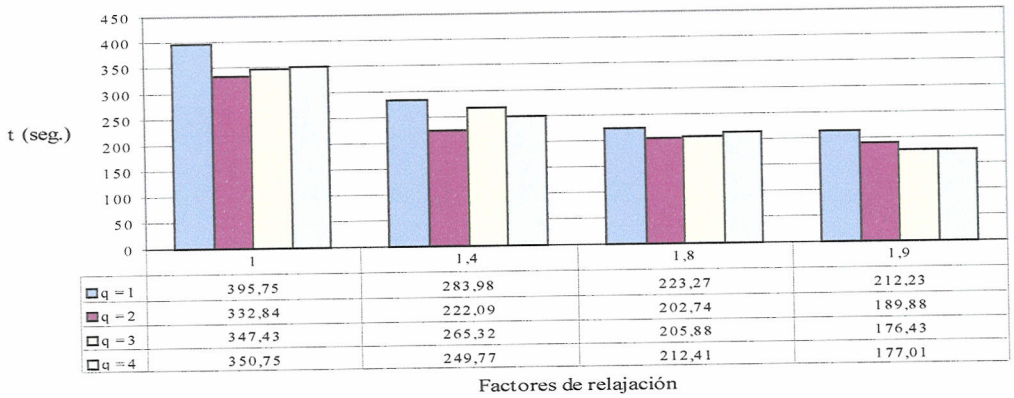


	1	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
q = 1	14,82	13,38	12,33	11,11	10,69	9,95	9,22	8,01	7,54	8,61
q = 2	16,41	15,01	14,3	13,38	12,7	11,54	10,14	9,68	9,24	10,38
q = 3	19,11	18,13	17,23	16,02	14,42	13,46	12,51	11,6	11,25	12,53
q = 4	22,55	20,96	19,41	17,81	16,6	15,82	14,27	13,87	13,45	14,67
q = 5	25,23	23,43	21,5	20,09	19,13	17,23	17,24	16,29	16,29	17,24

Factores de relajación

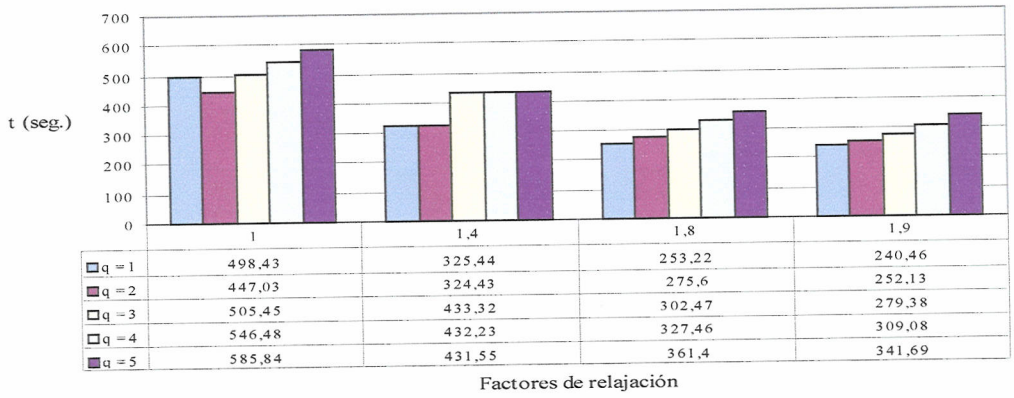
(a) Número de pasos del precondicionador: 3.

Figura 42: *Resultados del método GCP/2E-SSOR. Matriz de Laplace 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\tau, \tau\rangle < 10^{-7}$.



	1	1,4	1,8	1,9
q = 1	395,75	283,98	223,27	212,23
q = 2	332,84	222,09	202,74	189,88
q = 3	347,43	265,32	205,88	176,43
q = 4	350,75	249,77	212,41	177,01

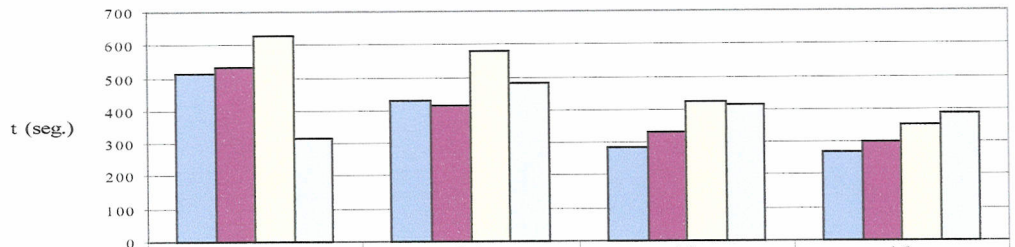
(a) Número de pasos del precondicionador: 1.



	1	1,4	1,8	1,9
q = 1	498,43	325,44	253,22	240,46
q = 2	447,03	324,43	275,6	252,13
q = 3	505,45	433,32	302,47	279,38
q = 4	546,48	432,23	327,46	309,08
q = 5	585,84	431,55	361,4	341,69

(b) Número de pasos del precondicionador: 2.

Figura 43: *Resultados del método GCP/2E-SSOR. Matriz Biarmónica 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



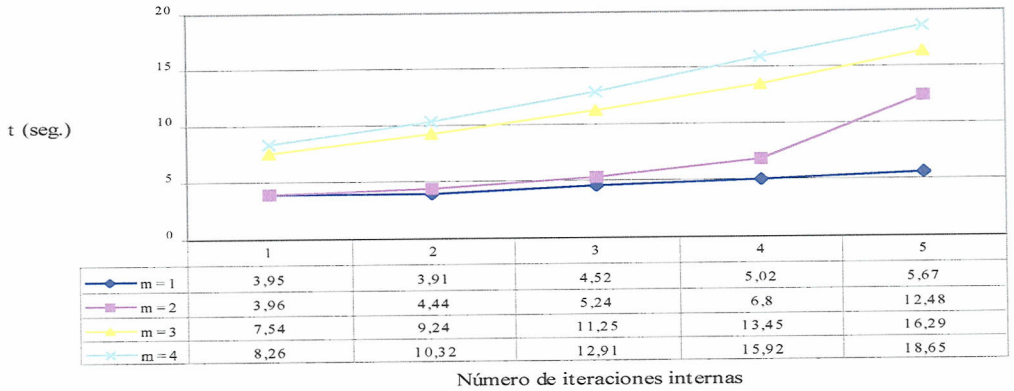
	1	1,4	1,8	1,9
q = 1	515,2	431,31	286,57	271,87
q = 2	535,93	416,01	332,01	303,21
q = 3	625,81	581,21	423,03	350,42
q = 4	317,77	481,05	414,05	391,03

Factores de relajación

(a) Número de pasos del precondicionador: 3.

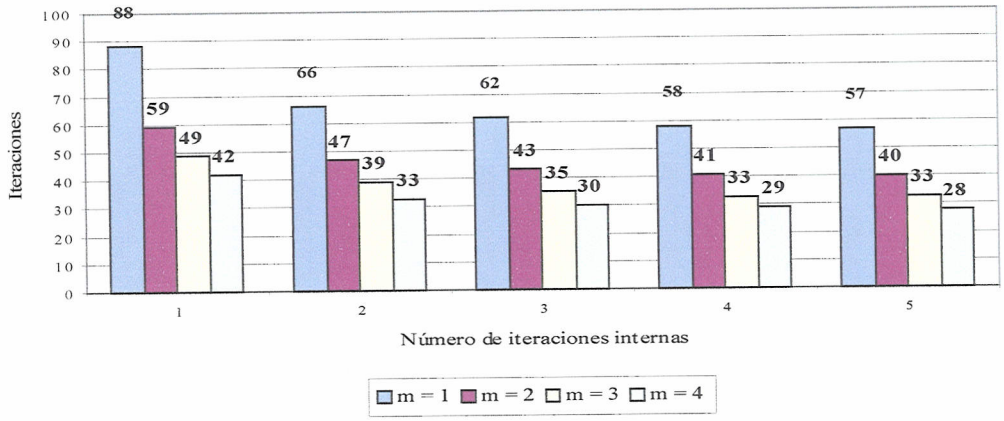
Figura 44: *Resultados del método GCP/2E-SSOR. Matriz Biarmónica 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\tau, \tau\rangle < 10^{-7}$.

Esto queda reflejado en las Figuras 45 y 46, donde también se puede apreciar que $m = 1$ es el número de pasos óptimo independientemente del número de procesadores utilizados.



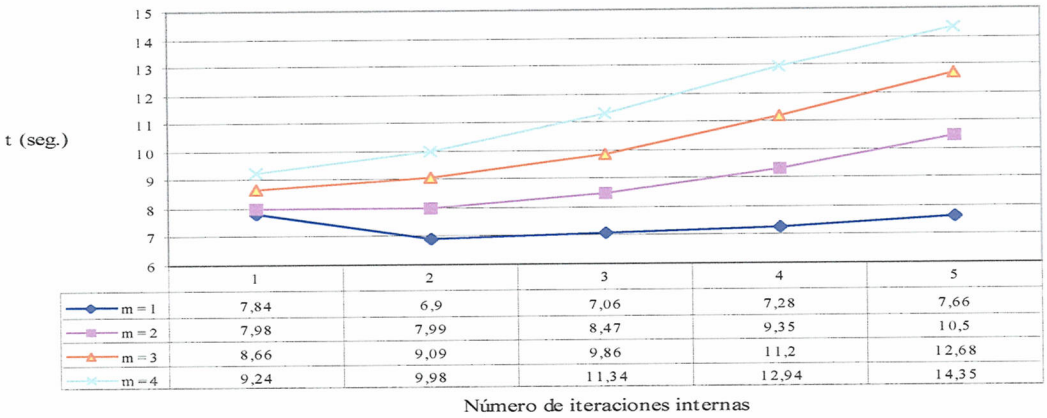
	1	2	3	4	5
m = 1	3,95	3,91	4,52	5,02	5,67
m = 2	3,96	4,44	5,24	6,8	12,48
m = 3	7,54	9,24	11,25	13,45	16,29
m = 4	8,26	10,32	12,91	15,92	18,65

(a)



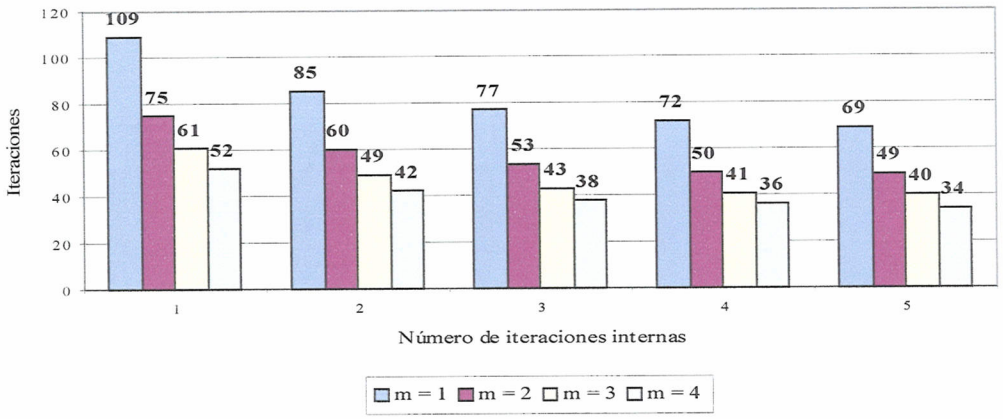
(b)

Figura 45: *Resultados del método GCP/2E- SSOR. Matriz de Laplace 40000, usando 2 procesadores y* $\omega = 1,8$. *IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



	1	2	3	4	5
m = 1	7,84	6,9	7,06	7,28	7,66
m = 2	7,98	7,99	8,47	9,35	10,5
m = 3	8,66	9,09	9,86	11,2	12,68
m = 4	9,24	9,98	11,34	12,94	14,35

(a)



(b)

Figura 46: *Resultados del método GCP/2E-SSOR. Matriz de Laplace 40000, usando 4 procesadores y* $\omega = 1,7$. *IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

7.5.2.5 Speed-up y eficiencia

A continuación vamos a ilustrar el comportamiento de estos métodos frente a su correspondiente secuencial. Como puede apreciarse en las Figuras 47 y 48, independientemente del valor del factor de relajación, el algoritmo paralelo siempre acelera al correspondiente secuencial. Las eficiencias obtenidas son también similares y están en torno al $75-80\%$. Sin embargo, como se explicó en la Sección 2.5, puesto que estos algoritmos paralelos no tienen porqué ser más eficientes cuando se ejecutan en secuencial, parece lógico evaluar dichos algoritmos tomando como referencia un "buen" algoritmo secuencial. Teniendo en cuenta las excelentes propiedades que tiene el método del gradiente conjugado respecto a su velocidad de convergencia desde el punto de vista secuencial, parece lógico comparar nuestros mejores métodos con dicho algoritmo secuencial.

La Tabla 28 muestra el comportamiento del método del gradiente conjugado precondicionado, con iteraciones internas SSOR, para la matriz de Laplace 4096-N, usando 2 procesadores, frente al método secuencial del gradiente conjugado precondicionado, usando como precondicionador el método SSOR en m-pasos. Como se puede apreciar, el método paralelo es más lento que el citado secuencial. Sin embargo, conforme aumenta el tamaño de la matriz esto cambia. Así por ejemplo, la Tabla 29 muestra los resultados para el método del gradiente conjugado precondicionado con iteraciones internas Gauss-Seidel simétrico (SSOR, con $\omega = 1$) y para el método del gradiente conjugado precondicionado con precondicionador Gauss-Seidel simétrico en m-pasos, para la matriz Laplace de tamaño 40000 y 2 procesadores. Se observa que el método del gradiente conjugado precondicionado paralelo acelera a este algoritmo secuencial clásico.

		GCP/2E-SSOR			GCP SSOR secuencial	
m	ω	q	Iter.	tiempo	Iter.	tiempo
1	1	1	65	0,52	62	0,30
1	1,7	1	42	0,32	33	0,16
1	1,9	1	59	0,48	27	0,13
1	1	2	48	0,44		
1	1,7	2	34	0,32		
1	1,9	2	44	0,40		
1	1	3	39	0,40		
1	1,7	3	33	0,35		
1	1,9	3	40	0,42		
2	1	1	46	0,56	43	0,33
2	1,7	1	29	0,36	22	0,17
2	1,9	1	41	0,50	18	0,14
2	1	2	48	0,55		
2	1,7	2	34	0,39		
2	1,9	2	44	0,53		
2	1	3	39	0,56		
2	1,7	3	33	0,47		
2	1,9	3	40	0,56		

Tabla 28: *Comparando el método GCP/2E-SSOR con el método secuencial GCP SSOR. Matriz de Laplace 4096-N, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

Las Figuras 49 y 50 comparan el comportamiento de ambos métodos para matrices de tamaño 40000 y 262144, respectivamente, en relación al parámetro de relajación usado en el método SSOR.

En general, las conclusiones observadas son parecidas a las obtenidas cuando se utiliza en la etapa interna Gauss-Seidel simétrico. Aunque cabe decir que, en determinados casos para factores de relajación altos, el comportamiento del mejor algoritmo secuencial obtenido de forma experimental ($m = 1$) ha sido

		GCP/2E-GSS				GCP GSS secuencial	
m	q	Iter.	T. Par.	T.Sec.	Speed-up	Iter.	Tiempo
1	1	171	7,79	8,22	1,05	167	8,12
1	2	122	7,17	9,05	1,26		
1	3	104	7,51	10,38	1,38		
2	1	120	7,99	9,28	1,66	117	9,17
2	2	86	8,06	11,11	1,37		
2	3	74	8,92	13,42	1,50		

Tabla 29: *Comparando el método GCP/2E-GSS con el método secuencial GCP GSS. Matriz de Laplace 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

similar al del algoritmo paralelo óptimo. Teniendo en cuenta que a priori no se conoce el factor de relajación óptimo, podemos pensar que en la práctica resolveríamos nuestro problema concreto con un valor de ω ni demasiado próximo a 1, por ser más lento en estos casos, ni demasiado próximo a 2, para evitar obtener pobre convergencia. Así, por ejemplo, en la Figura 51, se compara el comportamiento de los precondicionadores en dos etapas, usando SSOR en las iteraciones internas, en relación al número de iteraciones internas realizadas para la matriz de 262144, para $\omega = 1,5$, observando en dicho caso, una eficiencia del mejor algoritmo paralelo frente al mejor secuencial de en torno al 45%.

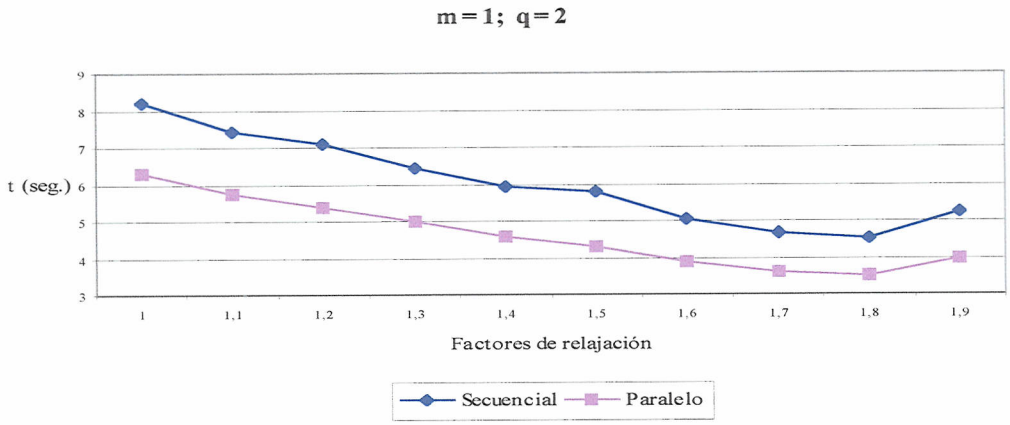


Figura 47: *Comparando del método GCP/2E-SSOR con su correspondiente secuencial. Matriz de Laplace 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

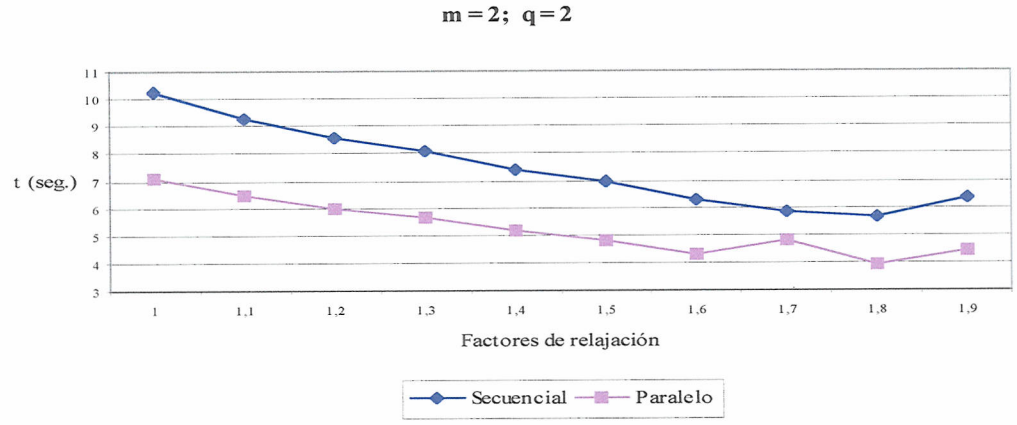


Figura 48: *Comparando del método GCP/2E-SSOR con su correspondiente secuencial. Matriz de Laplace 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

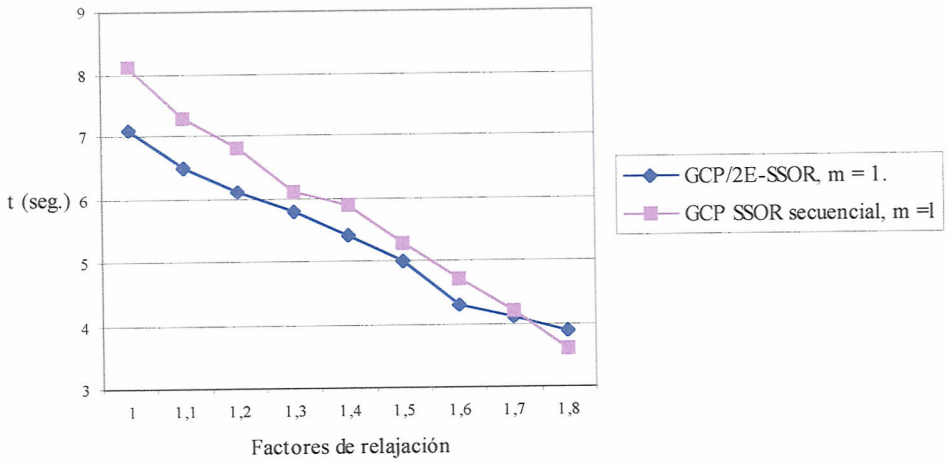


Figura 49: *Comparando el método GCP/2E-SSOR con el método secuencial GCP SSOR ($q = 2$). Matriz de Laplace 40000, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

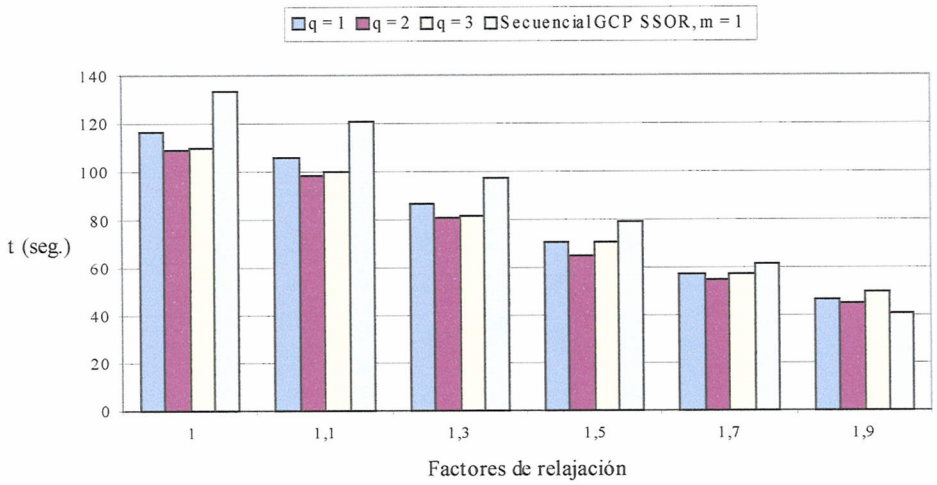


Figura 50: *Comparando el método GCP/2E-SSOR con el método secuencial GCP SSOR. Matriz de Laplace 262144, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

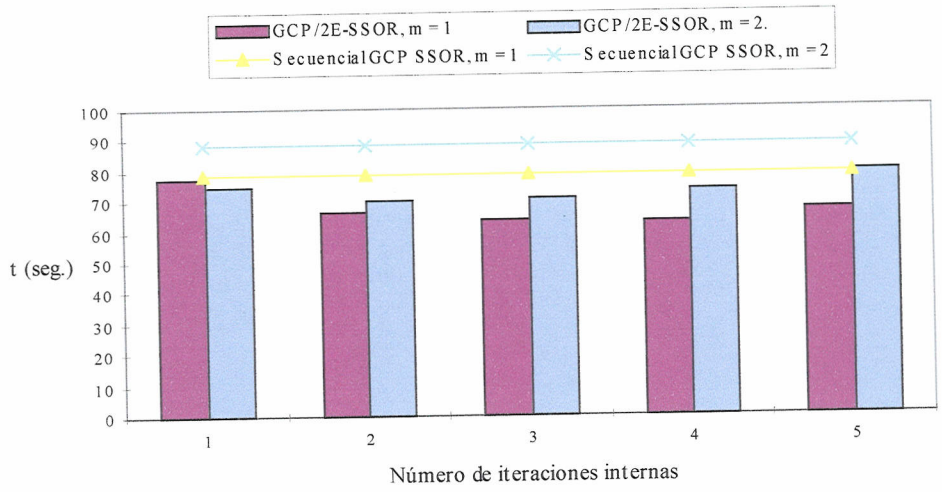


Figura 51: *Comparando el método GCP/2E-SSOR ($\omega = 1,5$) con el método secuencial GCP SSOR ($\omega = 1,5$). Matriz de Laplace 262144, usando 3 procesadores. IBM RS/6000 SP. Cota de parada $\langle \tau, \tau \rangle < 10^{-7}$.*

7.5.3 Resultados para el método GCP basado en la técnica en dos etapas con FIC como partición interna

A lo largo de esta sección presentamos los resultados numéricos del método del gradiente conjugado precondicionado (GCP), cuyo precondicionador se calcula aplicando el método en dos etapas utilizando para obtener las particiones internas, la factorización incompleta de Choleski (FIC). Denotamos a este método GCP/2E-FIC.

7.5.3.1 Sobre el número de condición

Al igual que se hizo en la Sección 7.5.2.1 vamos a dar, a continuación, una serie de ejemplos ilustrativos de cuál es el comportamiento del precondicionador en dos etapas basado en la factorización incompleta de Choleski. Tenemos que recordar, para esto, que los parámetros $N1$ y $N2$ que aparecen en todos los resultados de esta sección nos indican el número de subdiagonales que se factorizan. Así, $N1$ indica el número de subdiagonales que se factorizan a partir de la diagonal principal y $N2$ indica el número de subdiagonales que se factorizan desde la última subdiagonal distinta de cero hacia la diagonal principal. Estos parámetros definen el conjunto no cero G introducido en el Capítulo 6 (Sección 6.3), como conjunto no cero de la factorización.

En las Tablas 30 y 31 se han considerado dichos precondicionadores, utilizando los niveles de llenado $N1 = 2$, $N2 = 0$, y $N1 = N2 = 2$, para una matriz de Laplace de tamaño 1024. Dichas tablas presentan el número de condición

de la matriz del sistema precondicionado $\hat{A}$. El número de condición se ha calculado atendiendo al número de iteraciones internas q realizadas y al número de pasos del precondicionador, para 2 y 4 procesadores. En ambos casos, se ha supuesto que el número de filas asignadas a cada procesador es el mismo. Como en la Sección 7.5.2.1, estos resultados se han calculado utilizando el programa de cálculo matricial MATLAB. Recordemos que el número de condición obtenido para la matriz de Laplace ha sido 440,68.

De estas tablas se deduce que el número de condición de $\hat{A}$ es siempre menor que el número de condición de A. Además, el número de condición de $\hat{A}$ disminuye, tanto si aumentamos el número de pasos del precondicionador, como si aumentamos el número de iteraciones internas. Por otro lado, observamos que el conjunto no cero de la factorización definido por $N1 = N2 = 2$ da lugar a un número de condición menor que en el caso $N1 = 1, N2 = 0$, ya que en este último caso, se factorizan menos elementos que en el primero y por lo tanto, la aproximación que se obtiene de A es peor.

Cabe destacar, además, que al aumentar el número de procesadores, también aumenta el número de condición del precondicionador correspondiente. Esto se debe al hecho de que cuando aumentamos el número de bloques diagonales, los conjuntos no cero de la factorización incompleta de Choleski son más pequeños y, por tanto, las factorizaciones resultan "más incompletas", por lo que es de esperar un mayor número de condición.

2 PROCESADORES			4 PROCESADORES		
Número de pasos (m)	q	Número de Condición	Número de pasos (m)	q	Número de Condición
1	1	37,84	1	1	42,49
	2	22,20		2	27,06
	3	17,23		3	22,16
	4	14,91		4	19,87
	5	13,61		5	18,57
2	1	19,17	2	1	21,50
	2	11,35		2	13,78
	3	8,87		3	11,33
	4	7,71		4	10,19
	5	7,06		5	9,54
3	1	12,95	3	1	14,50
	2	7,74		2	9,36
	3	6,09		3	7,73
	4	5,32		4	6,96
	5	4,88		5	6,53
4	1	9,84	4	1	11,00
	2	5,93		2	7,15
	3	4,70		3	5,92
	4	4,12		4	5,35
	5	3,80		5	5,03
5	1	7,97	5	1	8,90
	2	4,85		2	5,82
	3	3,87		3	4,85
	4	3,41		4	4,39
	5	3,15		5	4,13
6	1	6,73	6	1	7,51
	2	4,13		2	4,94
	3	3,31		3	4,13
	4	2,93		4	3,75
	5	2,72		5	3,53

Tabla 30: *Número de condición de la matriz* $\hat{A}$, $N1 = 2$, $N2 = 0$. *Matriz de Laplace 1024, cond(A)=440,68.*

2 PROCESADORES			4 PROCESADORES		
Número de pasos (m)	q	Número de Condición	Número de pasos (m)	q	Número de Condición
1	1	13,96	1	1	19,43
	2	11,40		2	16,60
	3	10,74		3	15,93
	4	10,48		4	15,71
	5	10,35		5	15,63
2	1	7,24	2	1	9,97
	2	5,96		2	8,56
	3	5,63		3	8,22
	4	5,50		4	8,11
	5	5,43		5	8,07
3	1	5,00	3	1	6,82
	2	4,15		2	5,88
	3	3,93		3	5,66
	4	3,85		4	5,58
	5	3,80		5	5,55
4	1	3,88	4	1	5,25
	2	3,25		2	4,54
	3	3,09		3	4,38
	4	3,02		4	4,32
	5	2,99		5	4,30
5	1	3,22	5	1	4,30
	2	2,71		2	3,74
	3	2,58		3	3,61
	4	2,53		4	3,56
	5	2,51		5	3,55
6	1	2,78	6	1	3,68
	2	2,36		2	3,21
	3	2,25		3	3,10
	4	2,21		4	3,06
	5	2,19		5	3,05

Tabla 31: *Número de condición de la matriz $\hat{A}$, $N1 = 2$, $N2 = 2$. Matriz de Laplace 1024, cond(A)=440,68.*

7.5.3.2 Comportamiento del método

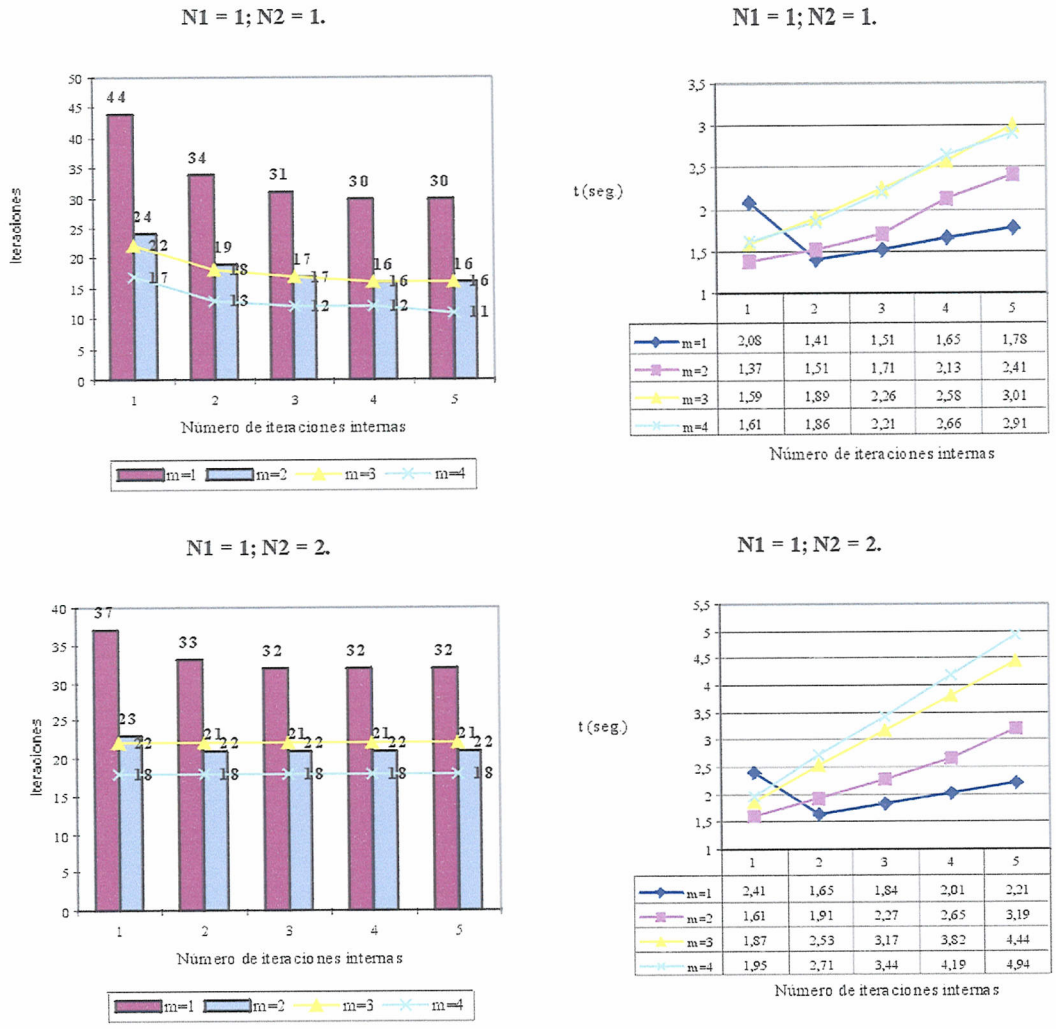
En esta sección estudiamos el comportamiento del método del gradiente conjugado precondicionado, cuyo precondicionador se calcula aplicando el método en dos etapas utilizando para obtener las particiones internas, la factorización incompleta de Choleski (GCP/2E-FIC). El estudio se realiza para distintos tamaños de matrices de Laplace.

Las Figuras 52 y 53 reflejan los resultados del método GCP/2E-FIC, para la matriz de Laplace 4096-N, y para distintos niveles de llenado, usando 2 procesadores. Análogamente, las Figuras 54, 55 y 56 reflejan los resultados obtenidos cuando se utilizan 3 procesadores.

En estas figuras se puede apreciar que para matrices pequeñas (o con pocos elementos no nulos), para $m = 1$, independientemente del nivel de llenado utilizado, generalmente, el número de iteraciones globales decrece conforme se aumenta el número de iteraciones internas. Por tanto, cuando dicha disminución en el número de iteraciones globales compense la realización de más iteraciones internas, se obtendrá un buen tiempo. Observamos en dichas figuras, por ejemplo, que para $m = 1$, los mejores tiempos, para esta matriz, se obtienen con $q = 2$. Sin embargo, cuando $m > 1$, el método no se comporta igual, ya que conforme se aumenta el número de iteraciones internas también aumenta el tiempo de ejecución. Teniendo en cuenta que realizando 2 pasos del precondicionador con una única iteración interna, se obtienen, para esta matriz, mejores resultados que para el precondicionador en 1 paso con q óptimo, concluimos que para esta matriz los mejores tiempos se obtienen con $m = 2$ y $q = 1$, independientemente del nivel de llenado. Sin embargo, esto no es generalizable a matrices de

mayor tamaño. Conforme aumenta el tamaño de la matriz, encontramos siempre un número de paso mayor que 1, en el que el método se comporta, como en el caso de la matriz de Laplace 4096-N, para $m = 1$, es decir los tiempos bajan al aumentar el número de iteraciones internas hasta cierto valor óptimo, a partir del cual comienzan a subir. Esto ocurre en dicho paso y todos los anteriores. Esto queda reflejado, en las Figuras 57 y 58, para la matriz Laplace de tamaño 40000.

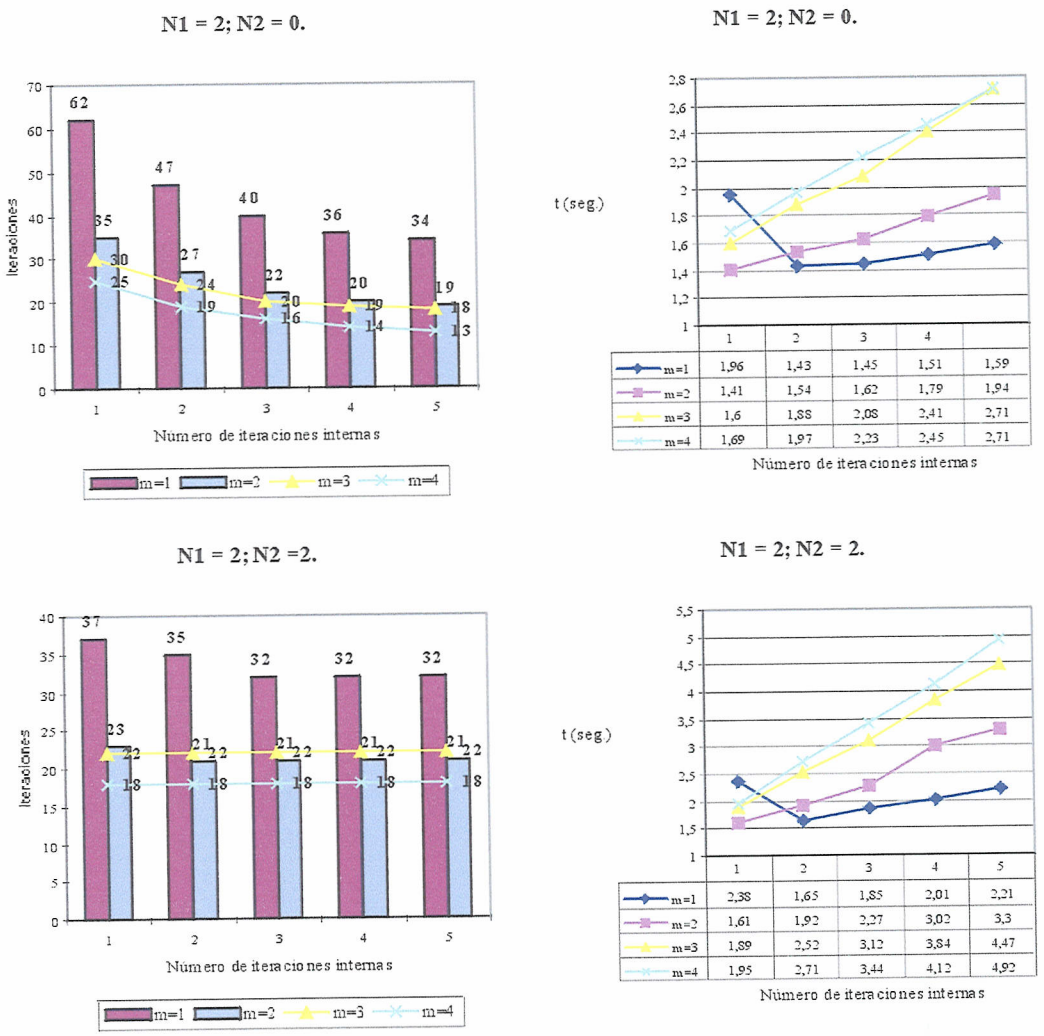
Por otra parte, si volvemos a las Figuras 54, 55 y 56 y las comparamos con las Figuras 52 y 53 se observa claramente, que para matrices pequeñas (por ejemplo la matriz de Laplace 4096-N), los tiempos disminuyen al aumentar el número de procesadores de 2 a 3. Este hecho es independiente del nivel de llenado utilizado. Además, al contrario de lo que ocurría en el método por bloques en dos etapas (ver Sección 7.4.4), el tiempo sigue disminuyendo al usar 4 procesadores. La Figura 59(a) ilustra este comportamiento. También podemos observar en la Figura 59(b) que al aumentar el número de procesadores (y por tanto disminuir el tamaño de los bloques), se obtiene mayor número de iteraciones globales, las cuales van bajando a medida que aumentamos el número de iteraciones internas q, lo cual parece lógico si nos atenemos a lo dicho sobre el comportamiento del número de condición para estos precondicionadores, en la Sección 7.5.3.1. Sin embargo, como dicho aumento es prácticamente insignificante, no afecta al comportamiento descrito del método, es decir, los tiempos disminuyen al aumentar el número de procesadores . Este hecho es independiente del tamaño de la matriz, como puede observarse, si comparamos la Figura 59, para la matriz de Laplace 4096-N, con la Figura 60, para la matriz Laplace de tamaño 16384, y la Figura 61, para la matriz de Laplace de tamaño 40000.



	1	2	3	4	5
m=1	2,08	1,41	1,51	1,65	1,78
m=2	1,37	1,51	1,71	2,13	2,41
m=3	1,59	1,89	2,26	2,58	3,01
m=4	1,61	1,86	2,21	2,66	2,91

	1	2	3	4	5
m=1	2,41	1,65	1,84	2,01	2,21
m=2	1,61	1,91	2,27	2,65	3,19
m=3	1,87	2,53	3,17	3,82	4,44
m=4	1,95	2,71	3,44	4,19	4,94

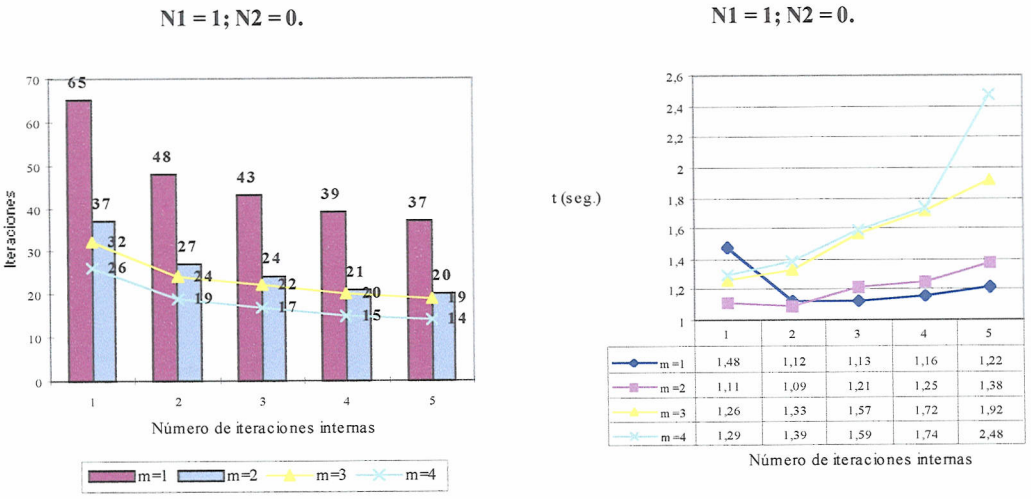
Figura 52: *Resultados del método del GCP/2E-FIC. Matriz de Laplace 4096-N, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



	1	2	3	4	
m=1	1,96	1,43	1,45	1,51	1,59
m=2	1,41	1,54	1,62	1,79	1,94
m=3	1,6	1,88	2,08	2,41	2,71
m=4	1,69	1,97	2,23	2,45	2,71

	1	2	3	4	5
m=1	2,38	1,65	1,85	2,01	2,21
m=2	1,61	1,92	2,27	3,02	3,3
m=3	1,89	2,52	3,12	3,84	4,47
m=4	1,95	2,71	3,44	4,12	4,92

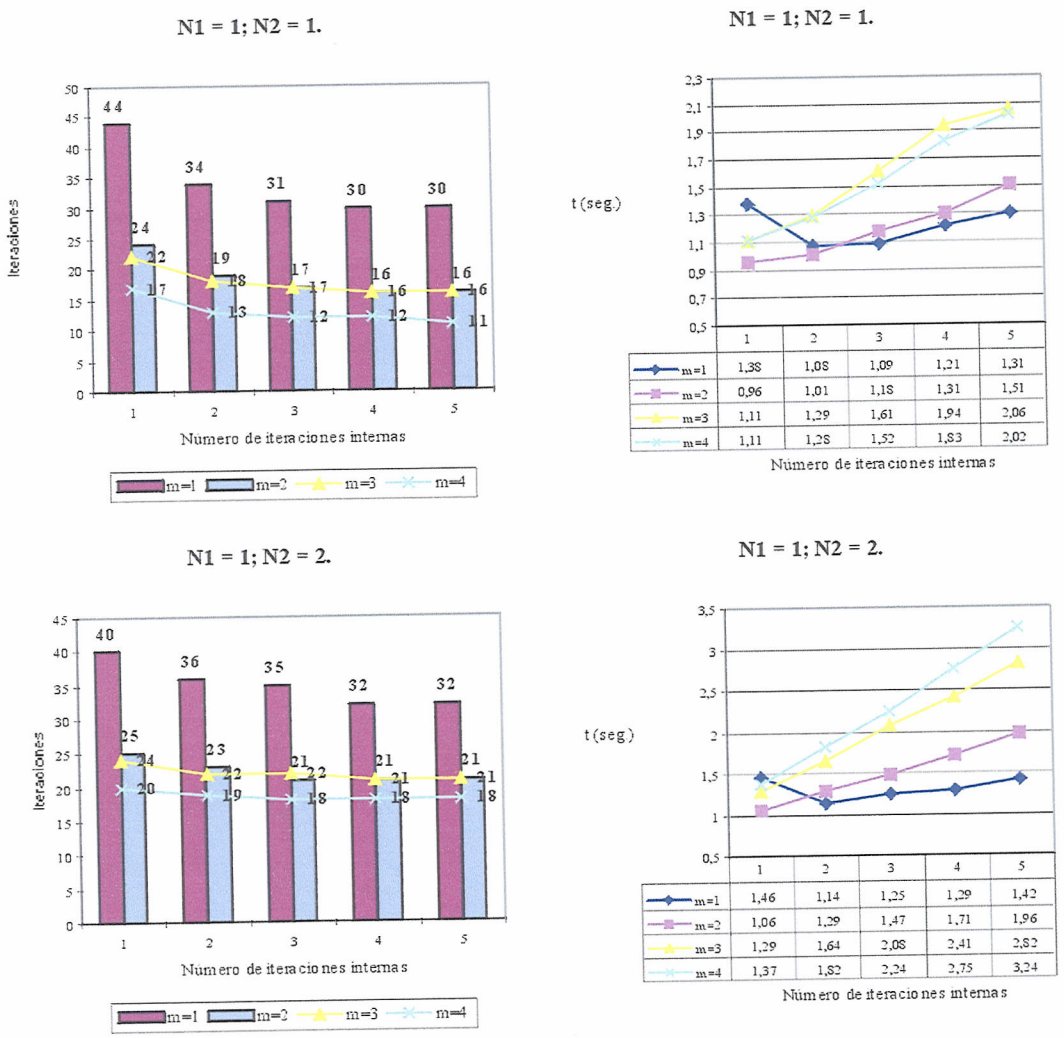
Figura 53: *Resultados del método del GCP/2E-FIC. Matriz de Laplace 4096-N, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



	1	2	3	4	5
m=1	1,48	1,12	1,13	1,16	1,22
m=2	1,11	1,09	1,21	1,25	1,38
m=3	1,26	1,33	1,57	1,72	1,92
m=4	1,29	1,39	1,59	1,74	2,48

Figura 54: *Resultados del método del GCP/2E-FIC. Matriz de Laplace 4096-N, usando 3 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

Queremos hacer notar que se hicieron pruebas con otros niveles de llenado, pero los que aparecen aquí, son con los que obtuvimos mejores resultados. Destacamos, que no se han encontrado demasiadas diferencias entre la elección del nivel de llenado $N1 = 1$ y $N2 = 0$ y la elección $N1 = 2$ y $N2 = 0$, (ver por ejemplo las Figuras 62, 63), siendo estos niveles de llenado, los que en general, obtienen mejores resultados. Por otra parte, se ha observado, como muestra la Figura 64, que de entre todos los niveles de llenado probados, el tiempo de ejecución es mayor para la elección del nivel de llenado $N1 = 2$ y $N2 = 2$.



	1	2	3	4	5
m=1	1,38	1,08	1,09	1,21	1,31
m=2	0,96	1,01	1,18	1,31	1,51
m=3	1,11	1,29	1,61	1,94	2,06
m=4	1,11	1,28	1,52	1,83	2,02

	1	2	3	4	5
m=1	1,46	1,14	1,25	1,29	1,42
m=2	1,06	1,29	1,47	1,71	1,96
m=3	1,29	1,64	2,08	2,41	2,82
m=4	1,37	1,82	2,24	2,75	3,24

Figura 55: *Resultados del método del GCP/2E-FIC. Matriz de Laplace 4096-N, usando 3 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

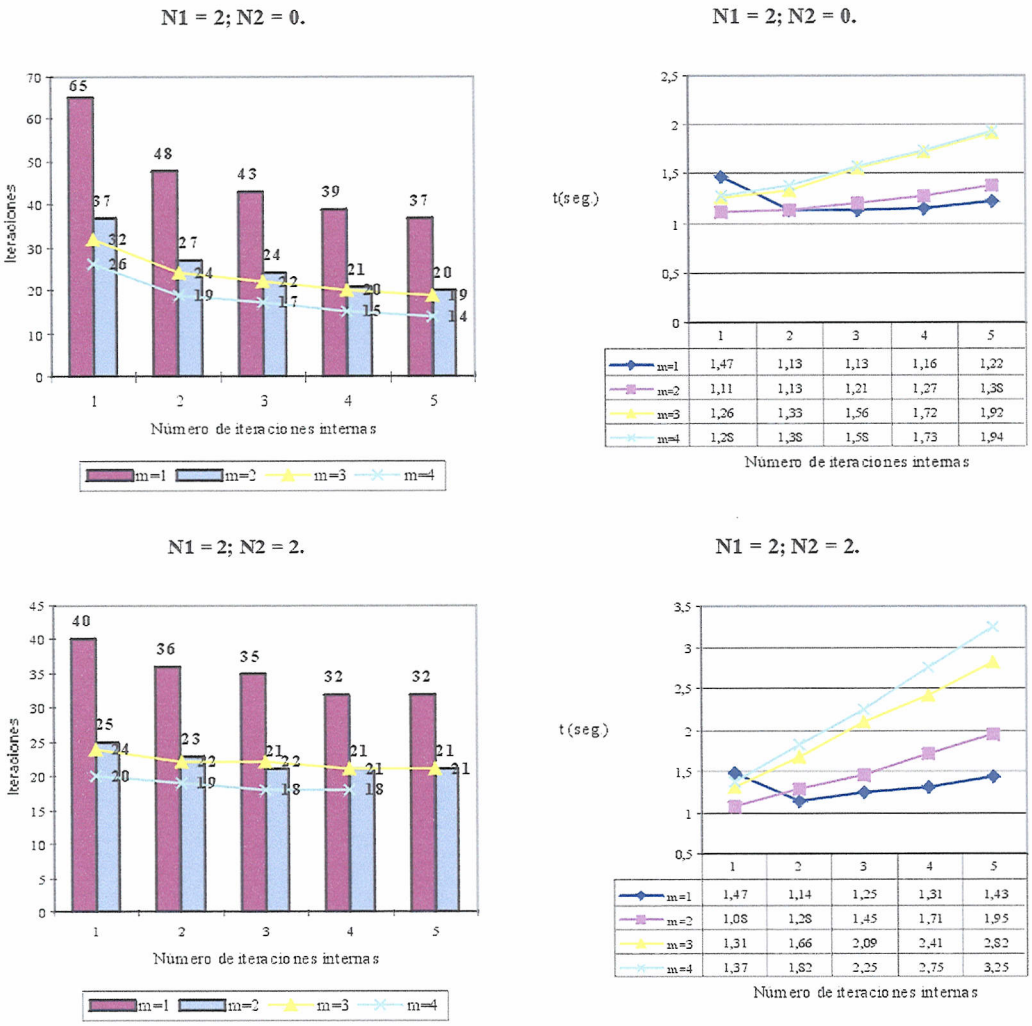
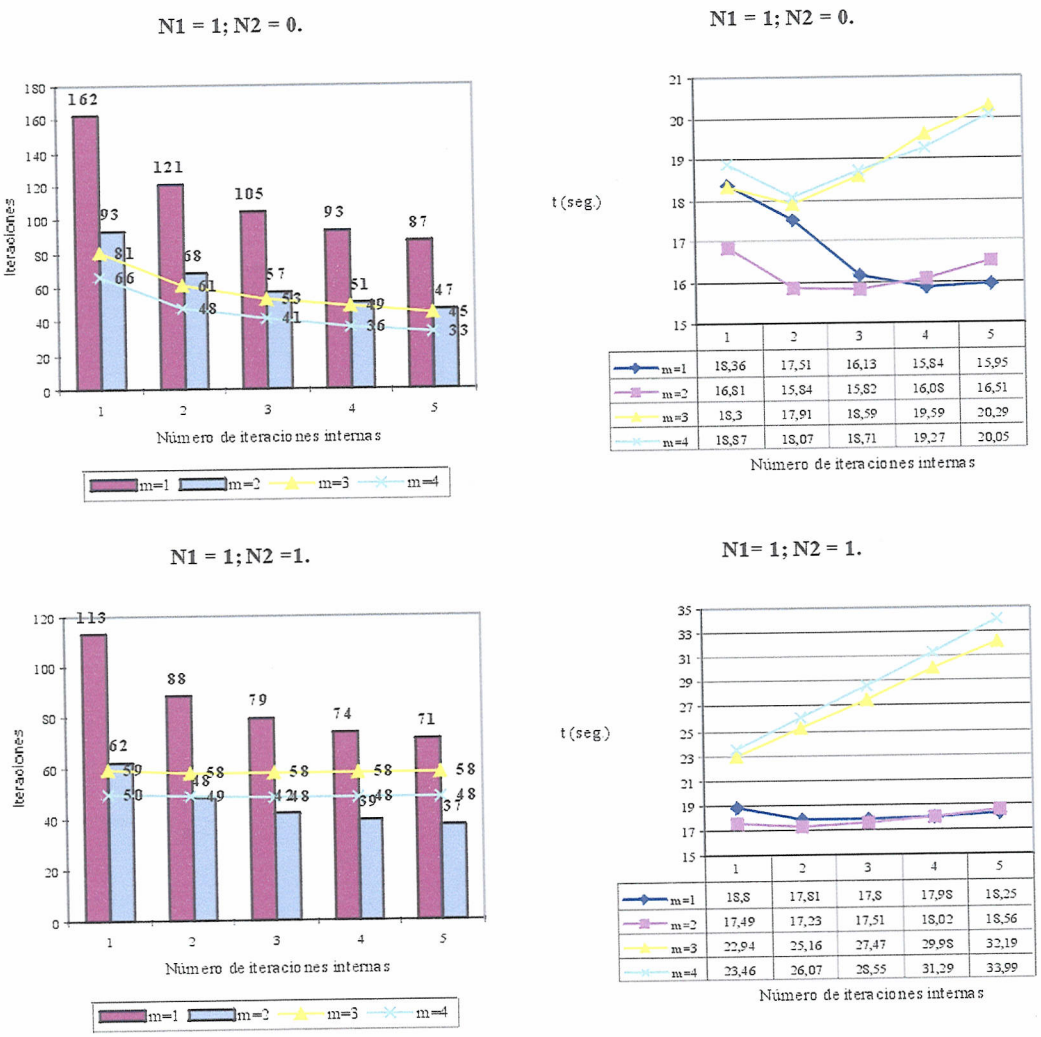


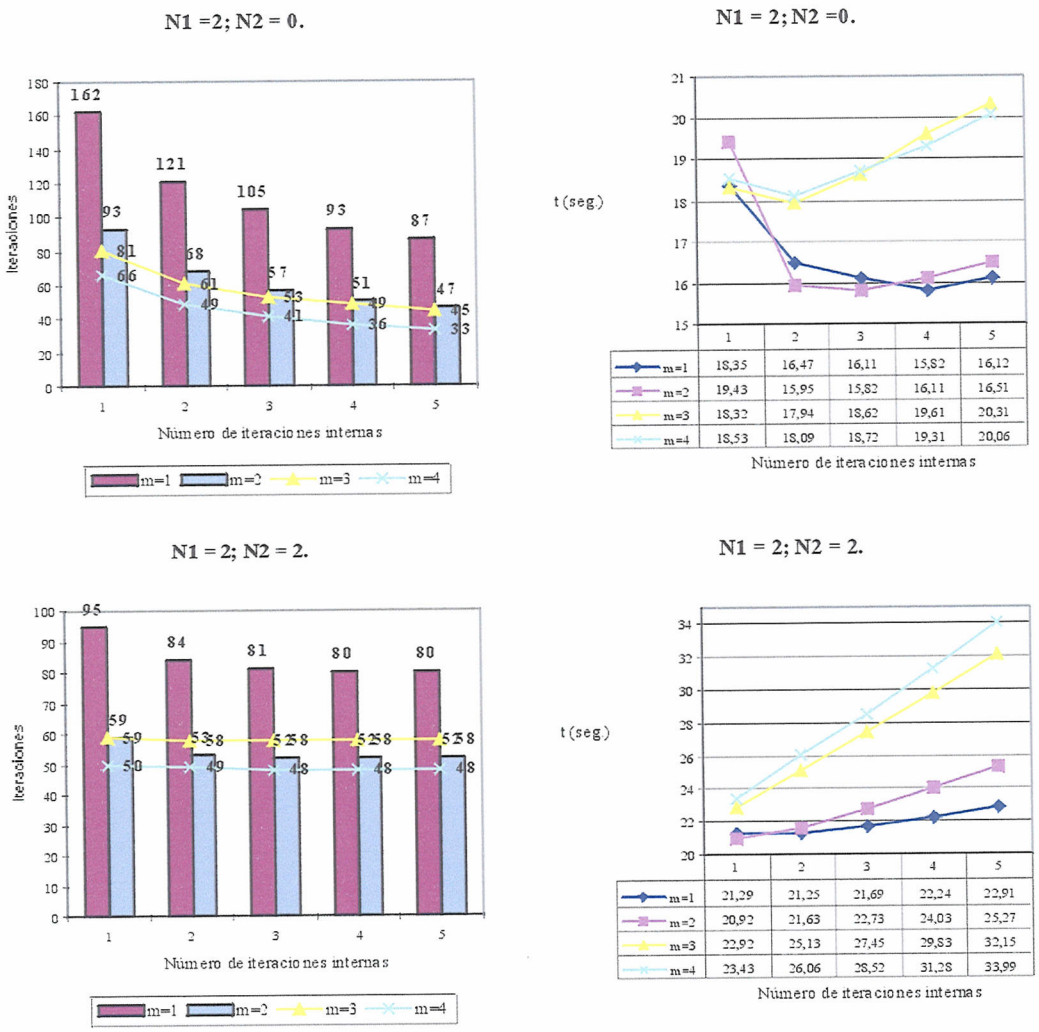
Figura 56: *Resultados del método del GCP/2E-FIC. Matriz de Laplace 4096-N, usando 3 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



	1	2	3	4	5
m=1	18,36	17,51	16,13	15,84	15,95
m=2	16,81	15,84	15,82	16,08	16,51
m=3	18,3	17,91	18,59	19,59	20,29
m=4	18,87	18,07	18,71	19,27	20,05

	1	2	3	4	5
m=1	18,8	17,81	17,8	17,98	18,25
m=2	17,49	17,23	17,51	18,02	18,56
m=3	22,94	25,16	27,47	29,98	32,19
m=4	23,46	26,07	28,55	31,29	33,99

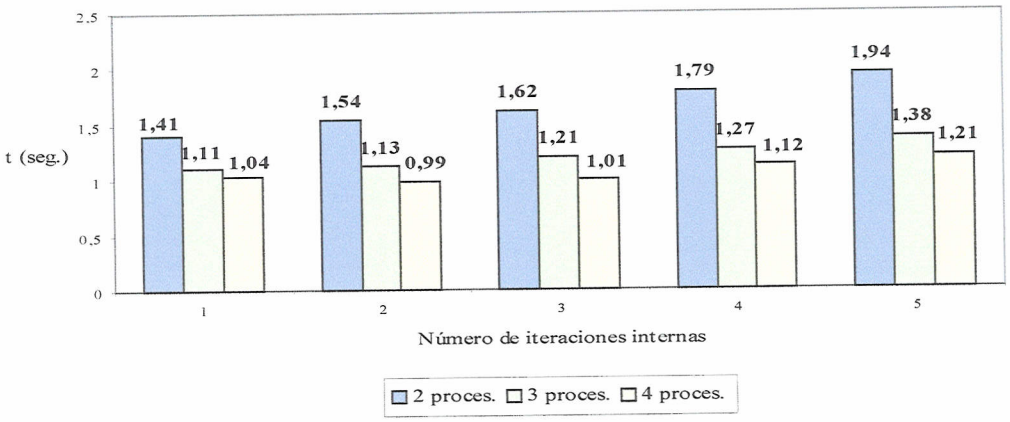
Figura 57: *Resultados del método del GCP/2E-FIC. Matriz de Laplace 40000 con 200 bloques de tamaño 200, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



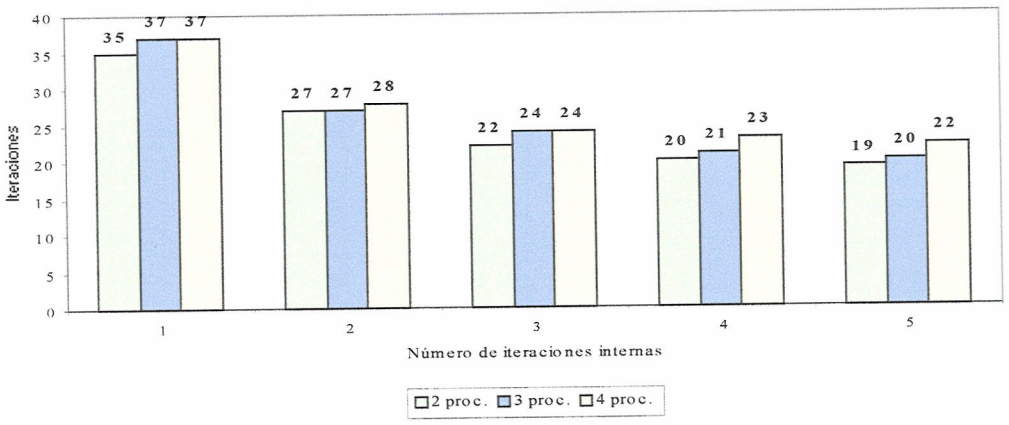
	1	2	3	4	5
m=1	18,35	16,47	16,11	15,82	16,12
m=2	19,43	15,95	15,82	16,11	16,51
m=3	18,32	17,94	18,62	19,61	20,31
m=4	18,53	18,09	18,72	19,31	20,06

	1	2	3	4	5
m=1	21,29	21,25	21,69	22,24	22,91
m=2	20,92	21,63	22,73	24,03	25,27
m=3	22,92	25,13	27,45	29,83	32,15
m=4	23,43	26,06	28,52	31,28	33,99

Figura 58: *Resultados del método del GCP/2E-FIC. Matriz de Laplace 40000 con 200 bloques de tamaño 200, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

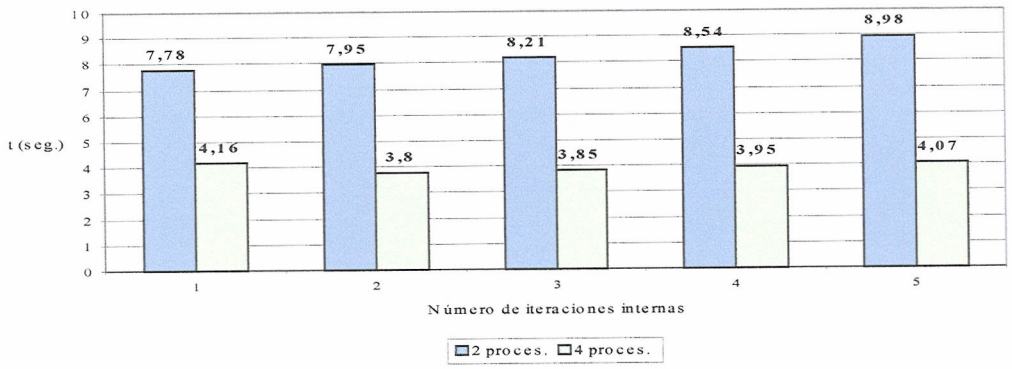


(a) Tiempos paralelos, para N1=2, N2=0 y m=2.

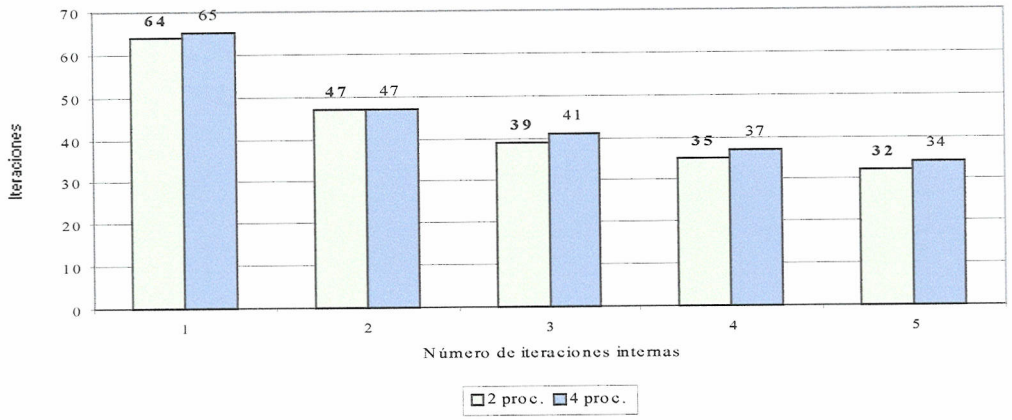


(b) Iteraciones para N1=2, N2=0 y m=2.

Figura 59: *Resultados del método del GCP/2E-FIC para 2, 3 y 4 procesadores. Matriz de Laplace 4096-N. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



(a) Tiempos paralelos, para N1=2, N2=0 y m=2.



(b) Iteraciones para N1=2, N2=0 y m=2.

Figura 60: *Resultados del método GCP/2E-FIC para 2 y 4 procesadores. Matriz de Laplace 16384 con 128 bloques de tamaño 128. IBM RS/6000 SP. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

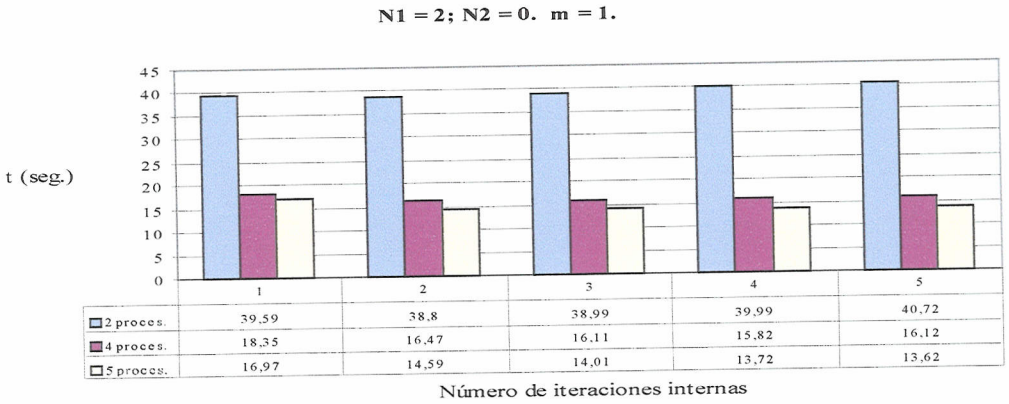


Figura 61: *Tiempos experimentales del método GCP/2E-FIC. Matriz de Laplace 40000, usando 2, 4 y 5 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

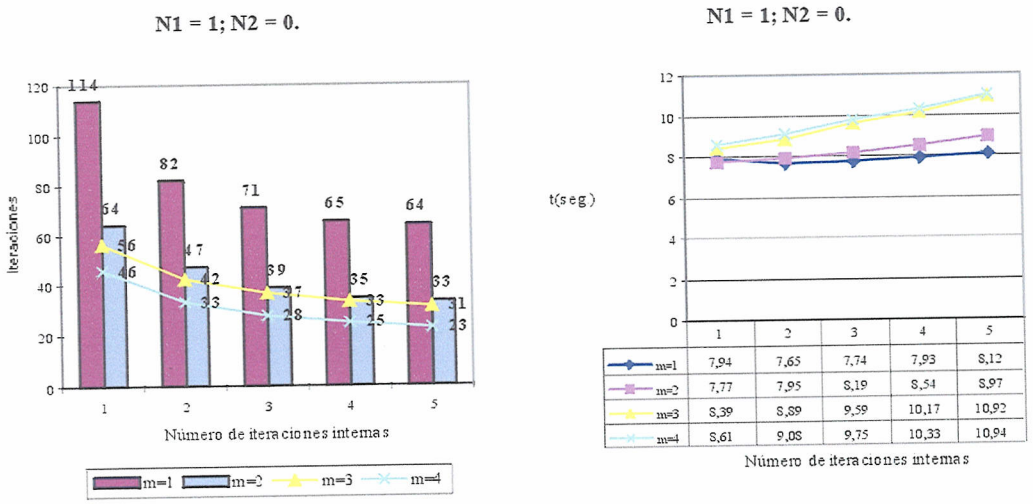
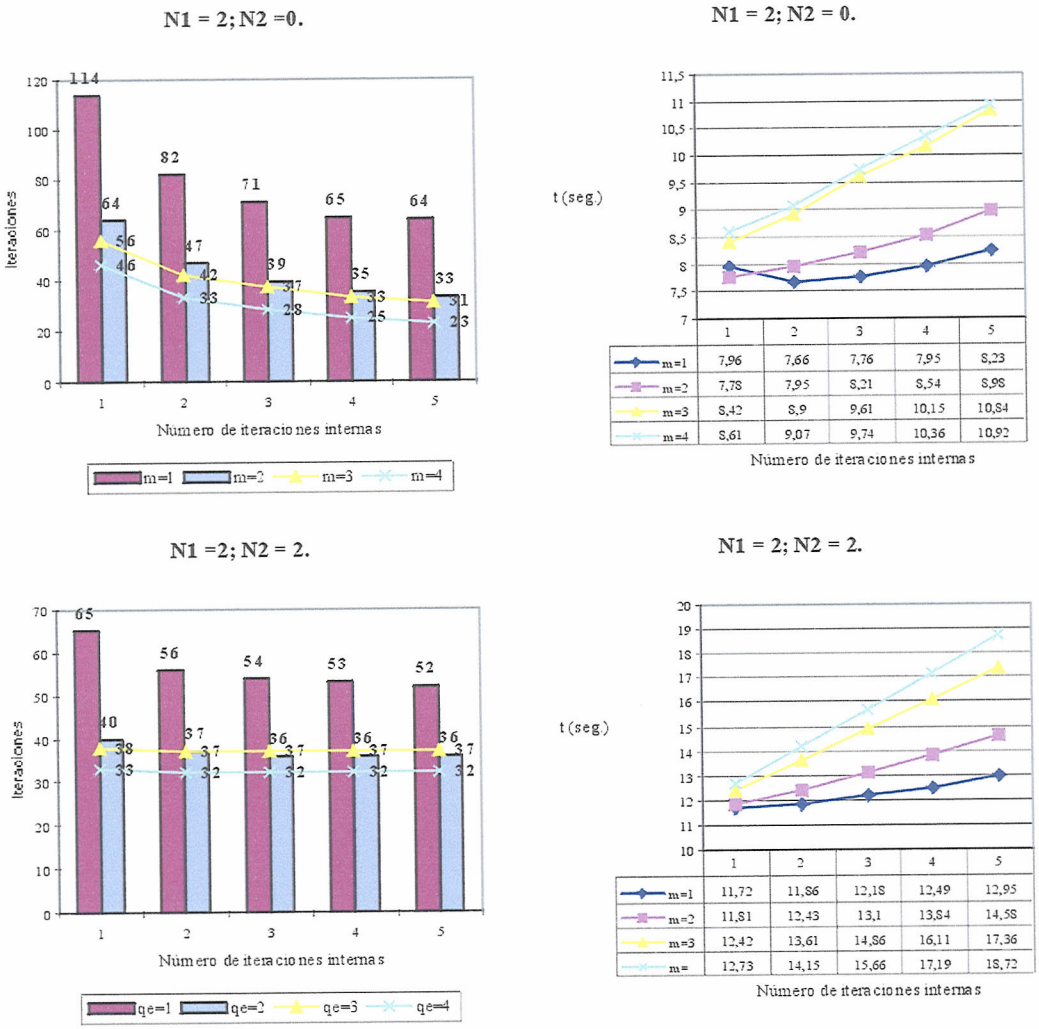


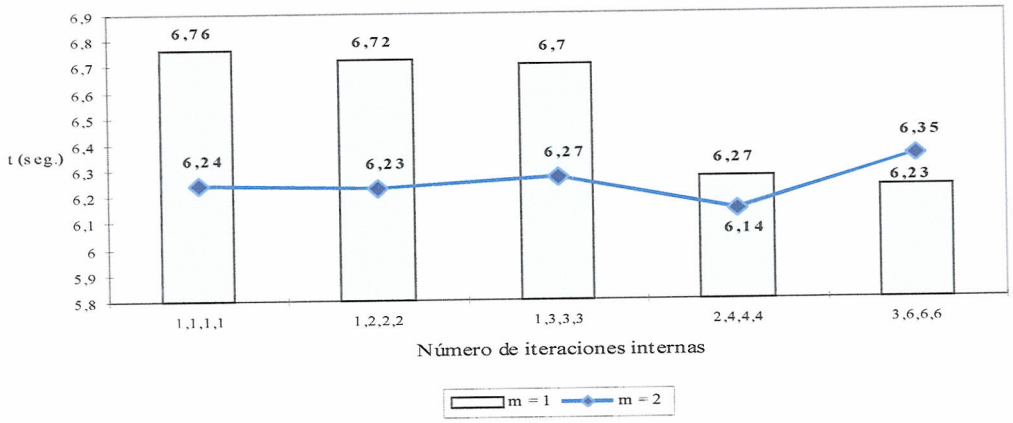
Figura 62: *Resultados del método GCP/2E-FIC. Matriz de Laplace 16384 con 128 bloques de tamaño 128, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



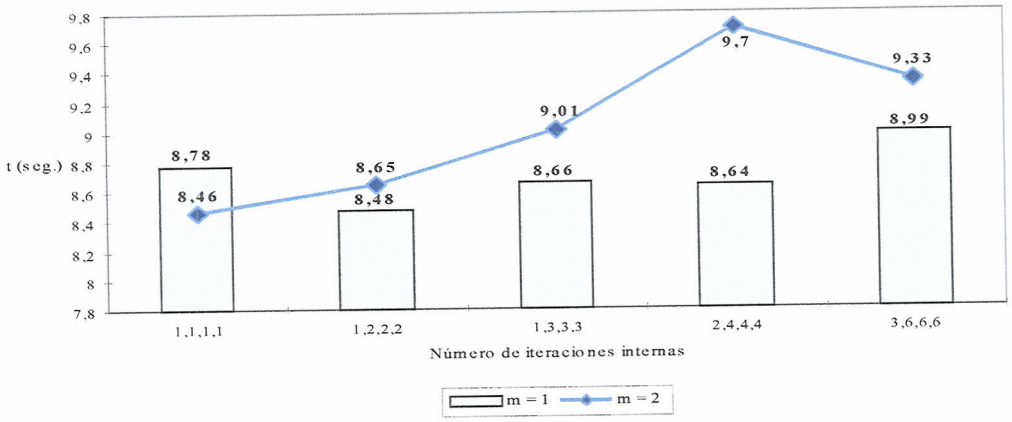
	1	2	3	4	5
m=1	7,96	7,66	7,76	7,95	8,23
m=2	7,78	7,95	8,21	8,54	8,98
m=3	8,42	8,9	9,61	10,15	10,84
m=4	8,61	9,07	9,74	10,36	10,92

	1	2	3	4	5
m=1	11,72	11,86	12,18	12,49	12,95
m=2	11,81	12,43	13,1	13,84	14,58
m=3	12,42	13,61	14,86	16,11	17,36
m=	12,73	14,15	15,66	17,19	18,72

Figura 63: *Resultados del método GCP/2E- FIC. Matriz de Laplace 16384 con 128 bloques de tamaño 128, usando 2 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.



(a) Tiempos paralelos, para N1=1, N2=0.



(b) Tiempos paralelos, para N1=2, N2=2.

Figura 64: *Resultados del método del GCP/2E-FIC con carga descompensada de 6400, 3328^3 filas/proces. Matriz de Laplace 16384 con 128 bloques de tamaño 128, usando 4 procesadores. IBM RS/6000 SP. Cota de parada* $\langle \tau, \tau \rangle < 10^{-7}$.

7.6 Comparación de los métodos. Conclusiones

En el presente capítulo hemos estudiado, desde el punto de vista experimental, el compotamiento de diversos métodos paralelos introducidos en esta memoria. Concretamente, en la Sección 7.4, se ha analizado el comportamiento de los métodos por bloques en dos etapas, introducidos en el Capítulo 5. Como matrices de prueba se han elegido, básicamente las matrices de Laplace, aunque también se da una reseña sobre el comportamiento de dichos métodos para las matrices que provienen de la ecuación Biarmónica, mostrando que para estas matrices dichos métodos obtienen la convergencia de forma más lenta que para las matrices de Laplace. En dichos experimentos hemos utilizado para la aproximación de los sistemas internos, principalmente, la partición del SOR y la partición del SSOR. En líneas generales, se muestra que el número de iteraciones globales de los algoritmos decrece al aumentar el número de iteraciones internas, reduciéndose además la diferencia entre el tiempo REAL y de CPU. Cuando dicha disminución en las iteraciones globales compensa el aumento de iteraciones internas es cuando se obtiene el mejor resultado del correspondiente algoritmo. Sin embargo, si comparamos los mejores resultados obtenidos para cada caso, observamos que aunque para $\omega = 1$, es más eficiente el método utilizando iteraciones internas SSOR, o lo que es lo mismo es más eficiente usar iteraciones internas Gauss-Seidel simétrico, que Gauss-Seidel, sin embargo, conforme aumentamos el factor de relajación, llegando a los factores de relajación óptimos esto se invierte, consiguiendo por tanto, el mejor resultado para el método SOR. Esto lo ilustramos en la Figura 65, donde aparecen los tiempos de ejecución de

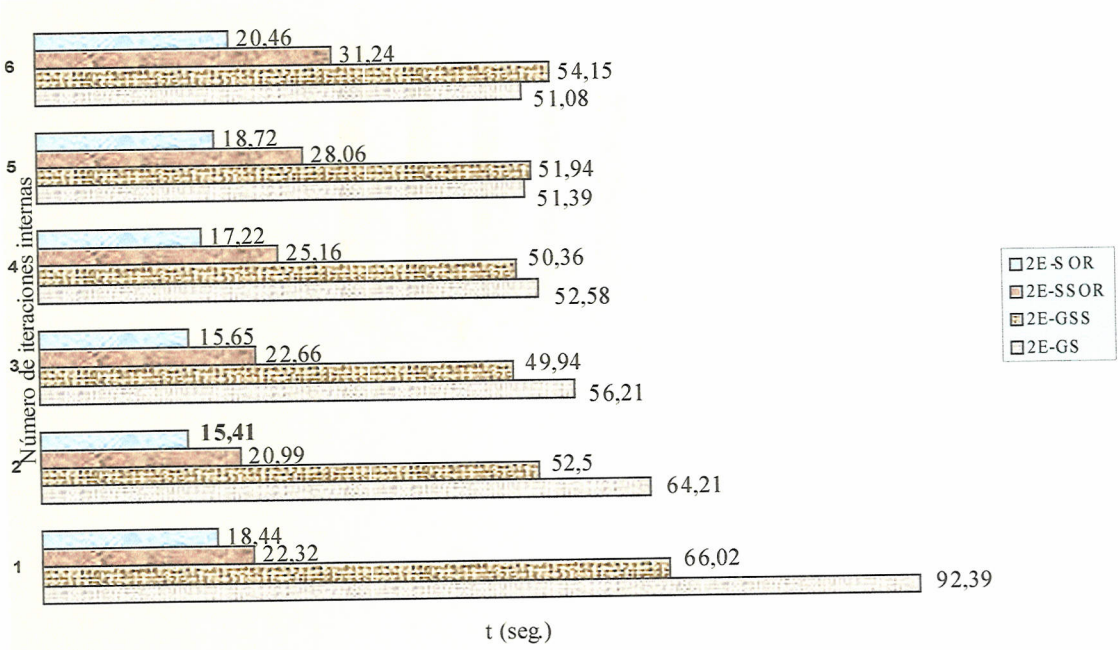
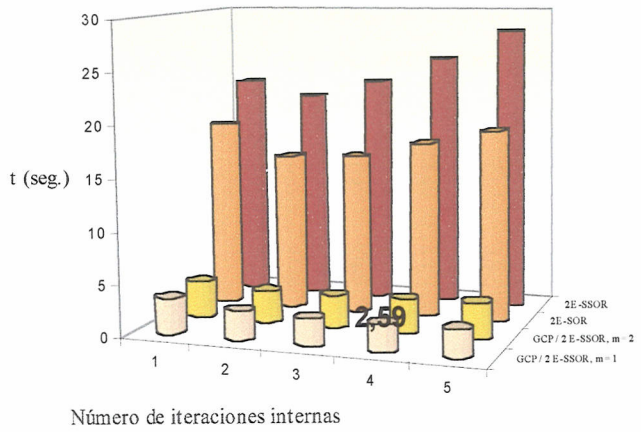


Figura 65: *Comparación de los métodos en dos etapas con iteraciones internas SOR y SSOR ($\omega = 1$, $\omega = 1,8$). Matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP.*

ambos métodos, tomando como factores de relajación $\omega = 1$ y $\omega = 1,8$, para la matriz de tamaño 16384, usando 4 procesadores. Como se puede apreciar, el mejor tiempo se obtiene con $\omega = 1,8$, realizando dos iteraciones internas.

En la Sección 7.5 se analiza, experimentalmente el comportamiento del método del gradiente conjugado precondicionado utilizando las dos técnicas de precondicionamiento estudiadas en el Capítulo 6.

El primer precondicionador está basado en el método por bloques en dos etapas analizado en la Sección 7.4. Utilizamos iteraciones internas Jacobi y SSOR y como matrices de prueba las matrices de Laplace y las matrices Biarmónicas.



	1	2	3	4	5
GCP/2E-SSOR, m = 1	3,51	2,82	2,61	2,59	2,67
GCP/2E-SSOR, m = 2	3,59	3,12	3,13	3,27	3,42
2E-SOR	18,44	15,41	15,65	17,22	18,72
2E-SSOR	22,32	20,99	22,66	25,16	28,06

Figura 66: *Comparación de los métodos en dos etapas con iteraciones internas SOR y SSOR con los métodos GCP/2E-SSOR ($\omega = 1{,}8$). Matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP.*

No obstante queremos hacer notar, que para estas últimas sólo se han utilizado iteraciones internas SSOR, ya que el precondicionador usando iteraciones internas Jacobi no es válido porque la partición interna de Jacobi no es convergente para este tipo de matrices. Para estos precondicionadores se ha analizado, experimentalmente, el número de condición, observando, cuando se usan interaciones internas tipo Jacobi, que el número de condición de la matriz del sistema precondicionado es menor que el número de condición de la matriz inicial, excepto cuando se realiza un único paso y una única iteración interna. Además, para un

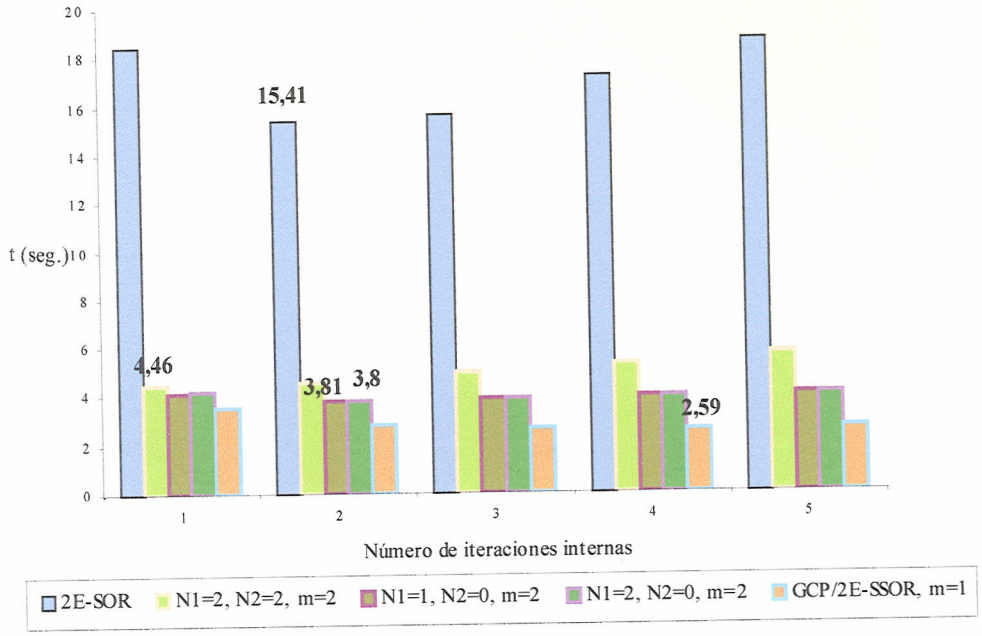


Figura 67: *Comparación de los métodos GCP/2E-SSOR, GCP/2E-FIC y los métodos en dos etapas con iteraciones internas SOR. Matriz de Laplace 16384, usando 4 procesadores. IBM RS/6000 SP.*

número de pasos par fijo, el número de condición decrece conforme aumentamos el número de iteraciones internas, mientras que para un número de pasos impar el número de condición va disminuyendo o aumentando según sea el número de iteraciones internas par o impar, respectivamente. Sin embargo, cuando se usa el precondicionador con iteraciones internas tipo SSOR, el número de condición de la matriz del sistema precondicionado es siempre menor que el número de condición de A. Además, el número de condición decrece conforme aumentamos el número de iteraciones internas, independientemente de que el número de

pasos sea par o impar. Respecto a los tiempos de ejecución diremos, como se puede ver en la Figura 66, que estos métodos aceleran de forma importante la convergencia frente al uso de los métodos por bloques en dos etapas analizados en la Sección 7.4.

Se ha finalizado este capítulo analizando el precondicionador construido en la Sección 6.3. Dicho precondicionador aúna la técnica de dos etapas y el uso de la factorización incompleta de Choleski. Al igual que se hizo con el anterior precondicionador se ha analizado, experimentalmente, el número de condición concluyendo que el número de condición de la matriz del sistema precondicionado es menor que el de la matriz del sistema original y disminuye, tanto si aumentamos el número de pasos del precondicionador, como si aumentamos el número de iteraciones internas. Por otro lado, desde el punto de vista de tiempos de ejecución, se ha observado que los mejores tiempos se consiguen cuando realizamos 1 ó 2 pasos del precondicionador, para cualquier tamaño de los bloques diagonales y para todos los niveles de llenado. Respecto a estos últimos, de todos los llenado analizados, los mejores tiempos se han conseguido con los niveles de llenado $N1 = 1,\ N2 = 0$ y $N1 = 2,\ N2 = 0$, que han resultado ser similares, mientras que el llenado $N1 = 2,\ N2 = 2$, es el que peor resultados ha obtenido.

Por otra parte, aunque con este precondicionador, se obtienen mejores resultados que con los métodos por bloques en dos etapas, analizados en la Sección 7.4, tenemos que decir que da peores resultados que el precondicionador anterior. Esto queda reflejado en la Figura 67, en la que se comparan los dos precondicionadores y el mejor método por bloques en dos etapas obtenido experimentalmente, para la matriz de Laplace de tamaño 16384.

Volver al índice/Tornar a l'índex

Concluimos por tanto que de todos los métodos estudiados en esta memoria, el método que mejor se comporta experimentalmente es el método del gradiente conjugado precondicionado cuyo precondicionador se calcula aplicando el método en dos etapas, y utilizando como particiones internas las obtenidas mediante el método SSOR.

Líneas Futuras

En esta memoria se han estudiado distintos métodos iterativos para la resolución de sistemas de ecuaciones lineales en el contexto de las matrices hermíticas y definidas positivas.

Los métodos han sido implementados en un IBM RS 6000/SP y un cluster de Pentiums, usando como soporte software PVM (parallel virtual machine). PVM es un paquete de software que permite que una red de computadores heterogéneos puedan actuar como un solo recurso computacional concurrente.

Actualmente no existe un modelo único que fundamente el desarrollo de la computación paralela al igual que el modelo Von Neuman lo ha hecho con la computación secuencial, sin embargo el modelo BSP (Bulk Synchronous Parallel) es uno de los que más seriamente se está considerando. Fue propuesto por Valiant [96] en 1990 y una de sus características es que incluye un modelo de costes para evaluar los algoritmos paralelos. En esta línea parece interesante comparar los resultados obtenidos experimentalmente usando PVM con los que se obtendrían usando BSP. Por otro lado, con la ayuda del modelo de costes, parece factible la obtención de heurísticos eficientes que no exijan la elección de

los factores no estacionarios a priori.

La consecuencia de estos objetivos nos permitirá crear unas librerías portables y eficientes que incluyan, entre otras, los algoritmos paralelos diseñados en esta memoria de manera que puedan ser utilizados por cualquier usuario.

Por otro lado, el diseño de algoritmos paralelos para sistemas no lineales está poco estudiado desde el punto de vista paralelo. Parece lógico entonces, plantear algoritmos paralelos no lineales basados en las técnicas estudiadas en esta memoria, expecialmente las técnicas de precondicionamiento. Por otra parte, el tema de asincronismo merece especial interés tanto desde el punto de vista lineal como no lineal. Éstas son básicamente las líneas en las que se enmarcarán nuestros próximos trabajos de investigación.

Bibliografía

[1] Loyce M. Adams. M-step preconditioned conjugate gradient method. *SIAM Journal on Scientifc and Statistical Computing*, 1:452–462, 1985.

[2] Loyce M. Adams. An m step preconditioned conjugate gradient method for parallel computation. *En Proceedings of the International Conference on Parallel Processing Laplacian*, 36–43, 1985.

[3] Loyce M. Adams y Elizabeth G. Ong. Additive polynomial preconditioners for parallel computers. *Parallel Computing*, 9:333–345, 1988.

[4] P. Albrecht y M. P. Klein. Extrapolated iterative methods for linear systems. *SIAM Journal on Numerical Analysis*, 21:192–201, 1984.

[5] E. O. Amedjoe. Splitting and multisplitting of matrices. *Annales Universitatis Scientiarum Budapestensis*, 13:195–224, 1992.

[6] O. Axelsson, S. Brinkkemper y V. P. Ill'n. On some versions of incomplete block-matrix factorization iterative methods. *Linear Algebra and its Applications*, 58:3–15, 1984.

[7] R. E. Bank y Donald J. Rose. Global approximate Newton methods. *Numerische Mathematik*, 37:279–295, 1981.

[8] R. Beauwens. Factorization iterative methods, M-operators and H-operators. *Numerische Mathematik*, 31:335–357, 1979.

[9] Michele Benzi y Daniel B. Szyld. Existence and uniqueness of splittings for stationary iterative methods with applications to alternating methods. *Numerische Mathematik*, 76:309–322, 1997.

[10] Abraham Berman y Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Academic Press, New York, tercera edición, 1979. Reimpreso por SIAM, Philadelphia, 1994.

[11] D. P. Bertsekas y J. N. Tsitsiklis. *Parallel and Distributed Computation.* Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[12] BLAS. Basic Linear Algebra Subroutine. Disponible en http://www.netlib.org/index.html.

[13] Rafael Bru, Cristina Corral, Ángeles Martinez y José Mas. Overlapping multisplitting preconditioners for the conjugate gradient methods. En R. F. Sincovec et al., editor, *Proceedings of the Sixth SIAM Conference on Parallel Processing for Cientific Computing*, 660–663. SIAM, Philadelphia, 1993.

[14] Rafael Bru, Cristina Corral, Ángeles Martinez y José Mas. Multisplitting preconditioners based on incomplete Choleski factorizations. *SIAM Journal on Matrix Analysis*, 16(4):1210–1222, 1995.

[15] Rafael Bru, Ludwig Elsner y Michael Neumann. Models of parallel chaotic iteration methods. *Linear Algebra and its Applications*, 103:175–192, 1988.

[16] Rafael Bru y Robert Fuster. Parallel chaotic extrapolated Jacobi method. *Applied Mathematics Letters*, 3(4):65–69, 1990.

[17] Rafael Bru y Charles R. Johnson. The spectral radius of a product of nonnegative matrices. *Linear Algebra and its Applications*, 141:227–240, 1990.

[18] Rafael Bru, Violeta Migallón y José Penadés. Chaotic inner-outer iterative schemes. En J. G. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, 434–438. SIAM Press, Philadelphia, 1994.

[19] Rafael Bru, Violeta Migallón y José Penadés. Chaotic methods for the parallel solution of linear systems. *Computing Systems in Engineering*, 6(4,5):385–390, 1995.

[20] Rafael Bru, Violeta Migallón, José Penadés y Daniel B. Szyld. Parallel, synchronous and asynchronous two–stage multisplitting methods. *Electronic Transactions on Numerical Analysis*, 3:24–38, 1995.

[21] Zhi-Hao Cao. Nonstationary two-stage multisplitting methods with overlapping blocks. *Linear Algebra and its Applications*, 285:153–163, 1998.

[22] Zhi-Hao Cao y Zhong-Yun Liu. Symmetric multisplitting of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 285:309–319, 1998.

[23] M. Jesús Castel, Violeta Migallón y José Penadés. Block two-stage preconditioners. Aceptado para su publicación en *Applied Mathematics Letters.*

[24] M. Jesús Castel, Violeta Migallón y José Penadés. Convergence of non-stationary parallel multisplitting methods for Hermitian positive definite matrices. *Mathematics of Computation*, 67(221):209–220, 1998.

[25] M. Jesús Castel, Violeta Migallón y José Penadés. Block two-stage precondictioners based on incomplete Choleski factorizations. *International Journal of Applied Science and Computations*, 6(2):96–101, 1999.

[26] D. Chazan y W. Miranker. Chaotic relaxation. *Linear Algebra and its Applications*, 2:199–222, 1969.

[27] K. L. Clark y Steven Gregory. PARLOG: parallel programming in logic. *ACM Transactions on Programming Languages and Systems*, 8(1):1–49, 1986.

[28] Joan Josep Climent y Carmen Perea. Comparison theorems for weak nonnegative splittings of K-monotone matrices. *Eletronic Journal of Linear Algebra*, 5:24–38, 1999.

[29] Joan Josep Climent y Carmen Perea. Convergence and comparison theorems for multisplittings. *Numerical Linear Algebra with applications*, 6:93–107, 1999.

[30] P. Concus, Gene H. Golub y Gérard Meurant. Block preconditioning for the conjugate gradient method. *SIAM Journal on Scientific and Statistical Computing*, 6:309–332, 1985.

[31] C. Conde y G. Winter. *Métodos y Algoritmos básicos del Álgebra Lineal.* Reveté, S.A., Barcelona, España, 1990.

[32] Cristina Corral. *Precondicionadores Paralelos para el Método del Gradiente Conjugado.* Tesis Doctoral, Departamento de Matemática Aplicada, Universidad Politécnica de Valencia, Mayo 1995.

[33] J. Dancis. The optimal ω is not best for the SOR iteration method. *Linear Algebra and its Applications*, 154:819–845, 1991.

[34] R. S. Dembo, S. C. Eisenstat y T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 18:400–408, 1982.

[35] J. Douglas. Alternating direction iteration for midly non linear difference equations. *Numerische Mathematik*, 3:92–98, 1961.

[36] P. F. Dubois, A. Greenbaum y G. H. Rodrigue. Approximating the inverse of a matrix for use on iterative algorithms on vector processors. *Computing*, 22:257–268, 1979.

[37] Iain Duff y Gérard A. Meurant. The effect of ordering on preconditioned conjugated gradients. *BIT*, 29:635–657, 1989.

[38] T. Dupont. Factorization procedure for the solution of elliptic difference equations. *SIAM Journal on Numerical Analysis*, 5:753–782, 1968.

[39] E. G. D'Yakonov. On the solution of some elliptic difference equations. *Numerische Mathematik*, 7:1–20, 1971.

[40] Ludwig Elsner. Comparisons of weak regular splittings and multisplittings methods. *Numerische Mathematik*, 56:283–290, 1989.

[41] Ludwig Elsner, Israel Koltracht y Michael Neumann. On the convergence of asynchronous paracontractions with application to tomographic reconstruction from incomplete data. *Linear Algebra and its Applications*, 130:65–82, 1990.

[42] Michael J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9):948–960, 1972.

[43] Andreas Frommer y Günter Mayer. Convergence of relaxed parallel multisplitting methods. *Linear Algebra and its Applications*, 119:141–152, 1989.

[44] Andreas Frommer y Günter Mayer. On the theory and practice of multisplitting methods in parallel computation. *Computing*, 49:63–74, 1992.

[45] Andreas Frommer y Bert Pohl. Comparison results for splittings based on overlapping blocks. En J.G. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, 29–33. SIAM Press, Philadelphia, 1994.

[46] Andreas Frommer y Bert Pohl. A comparison result for multisplittings and waveform relaxation methods. *Numerical Linear Algebra with Applications*, 2:335–346, 1995.

[47] Andreas Frommer y Daniel B. Szyld. H-splittings and two-stage iterative methods. *Numerische Mathematik*, 63:345–356, 1992.

[48] Andreas Frommer y Daniel B. Szyld. Asynchronous two-stage iterative methods. *Numerische Mathematik*, 69:141–153, 1994.

[49] Robert Fuster, Violeta Migallón y José Penadés. Non-stationary parallel multisplitting AOR methods. *Electronic Transactions on Numerical Analysis*, 4:1–13, 1996.

[50] Robert Fuster, Violeta Migallón y José Penadés. Parallel chaotic extrapolated Jacobi-like methods. *Linear Algebra and its Applications*, 247:237–250, 1996.

[51] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek y Vaidy Sunderam. PVM 3 User's Guide and Reference Manual. Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, Tennessee, Septiembre 1994.

[52] Gene H. Golub y James M. Ortega. *An Introduction with Parallel Computing.* Academic Press, San Diego, 1993.

[53] Gene H. Golub y M. L. Overton. Convergence of a two-stage Richardson iterative procedure for solving systems of linear equations. *Lecture Notes in Mathematics*, 912:128–139, 1982.

[54] Gene H. Golub y M. L. Overton. The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems. *Numerische Mathematik*, 53:571–593, 1988.

[55] A. Hadjidimos. The optimal solution to the problem of complex extrapo-

lation of a first-order scheme. *Linear Algebra and its Applications*, 62:241–261, 1984.

[56] M. Hestenes y E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, 49:409–436, 1952.

[57] R. A. Horn y C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[58] IBM Corporation. IBM PVMe for AIX User's Guide and Subroutine Reference. Technical Report GC23-3884-00, IBM Corp., Poughkeepsie, NY, Octubre 1995.

[59] O. Johnson, C. Micchelli y G. Paul. Polynomial precondictioners for conjugate gradient calculations. *SIAM Journal Numerical Analysis*, 20:362–376, 1983.

[60] Mark T. Jones y Daniel B. Szyld. Two-stage multisplitting methods with overlapping blocks. *Numerical Linear Algebra with Applications*, 3(2):113–124, 1996.

[61] Herbert B. Keller. On the solution of singular and semidefinite linear systems by iteration. *SIAM Journal on Numerical Analysis*, 2(2):281–290, 1965.

[62] R. Kettler. *Linear multigrid methods in numerical reservoir simulations.* Tesis Doctoral, Delft University of Technology, Diciembre 1987.

[63] Paul J. Lanzkron, Donald J. Rose y Daniel B. Szyld. Convergence of nested

iterative methods for linear systems. *Numerische Mathematik*, 58:685–702, 1991.

[64] K. Löwner. Über monotone matrixfunktionen. *Math. Z.*, 38:177–216, 1934.

[65] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computation*, 34:473–497, 1980.

[66] Ivo Marek y Daniel B. Szyld. Comparison theorems for weak splittings of bounded operators. *Numerische Mathematik*, 58:389–397, 1990.

[67] José Mas, Violeta Migallón, José Penadés y Daniel B. Szyld. Non-stationary parallel relaxed multisplitting methods. *Linear Algebra and its Applications*, 241/243:733–748, 1996.

[68] J. A. Meijerink y H. A. Van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matriz. *Mathematics of Computation*, 31(137):148–162, 1977.

[69] Gérard Meurant. The block preconditioned conjugate gradient method on vector computers. *BIT*, 24:623–633, 1984.

[70] Gérard Meurant. Multitasking the conjugate gradient method on the CRAY X-MP/48. *Parallel Computing*, 5:267–280, 1987.

[71] Violeta Migallón y José Penadés. Convergence of two-stage iterative methods for Hermitian positive definite matrices. *Applied Mathematics Letters*, 10(3):79–83, 1997.

[72] Violeta Migallón y José Penadés. The monotonicity of two-stage iterative methods. *Applied Mathematics Letters*, 12(8):73–76, 1999.

[73] Violeta Migallón, José Penadés y Daniel B. Szyld. Nonstationary multisplittings with general weighting matrices. Technical Report 00-1-16, Department of Mathematics, Temple University, Enero 2000.

[74] Reinhard Nabben. A note on comparison theorems for splittings and multisplittings of Hermitian positive definite matrices. *Linear Algebra and its Applications*, 233:67–80, 1995.

[75] Michael Neumann y Robert J. Plemmons. Convergence of parallel multisplitting iterative methods for M-matrices. *Linear Algebra and its Applications*, 88–89:559–573, 1987.

[76] Nancy K. Nichols. On the convergence of two-stage iterative processes for solving linear equations. *SIAM Journal on Numerical Analysis*, 10:460–469, 1973.

[77] Dianne P. O'Leary y Robert E. White. Multi-splittings of matrices and parallel solution of linear systems. *SIAM Journal on Algebraic Discrete Methods*, 6:630–640, 1985.

[78] James O'Neil y Daniel B. Szyld. A block ordering method for sparse matrices. *SIAM Journal on Scientific and Statistical Computing*, 11:811–823, 1990.

[79] James M. Ortega. *Numerical Analysis, A second Course.* Academic Press, New York, 1972. Reimpreso por SIAM, Philadelphia, 1990.

[80] James M. Ortega. *Matrix Theory.* Plenum Press, New York, 1987.

[81] James M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems.* Plenum Press, New York, 1988.

[82] James M. Ortega y Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, San Diego, 1970. Reimpreso por SIAM, Philadelphia, 2000.

[83] Alexander M. Ostrowski. Iterative solution of linear systems of functional equations. *Journal of Mathematical Analysis and Applications*, 2:351–369, 1961.

[84] José Penadés. *Métodos Iterativos Paralelos para la Resolución de Sistemas Lineales basados en Multiparticiones.* Tesis Doctoral, Departamento de Tecnología Informática y Computación, Universidad de Alicante, Diciembre 1993.

[85] G. Radicati y Y. Robert. Vector and parallel CG-like algorithms for sparse non-symmetric systems. Technical Report 681-M, Grenoble, France, 1987.

[86] F. Robert, M. Charnay y F. Musy. Itérations chaotiques série-parallèle pour des équations non-linéaires de point fixe. *Aplikace Matematiky*, 20:1–38, 1975.

[87] Heinz Rutishauser. Theory of gradient methods. En *Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, Basel- Stuttgart.* Institute of Applied Mathematics, Zurich, BirkHauser Verlag, 1959.

[88] Heinz Rutishauser. *Lectures on Numerical Mathematics.* Birkhäuser Boston, Boston, Massachusetts, 1990.

[89] Yousef Saad. SPARSKIT: A basic tool kit for sparse matrix computations. Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990. Segunda versión de SPARSKIT disponible en http://www.cs.emn.edu/Research/arpa/SPARSKIT.

[90] Yousef Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, 1996. Reimpreso por Y. Saad, segunda edición, enero 2000, disponible en http://www-users.cs.umn.edu/~saad/books.html.

[91] M. K. Seager. Parallelizing conjugate gradient for the CRAY X-MP. *Parallel Computing*, 3:35–47, 1986.

[92] E. Shapiro. Concurrent prolog. *IEEE Computer*, 19(1):44–58, 1986.

[93] J. Smith. The coupled equation approach to the numerical solution of the biharmonic equation by finite differences. *SIAM Journal on Numerical analysis*, 5:323–339, 1969.

[94] E. L. Stiefel. *Kernel polynomials in linear algebra and their applications.* CBMS Regional Conference Series in Applied Mathematics. Bur. Standars Applied Mathematics Series, U.S., 1958.

[95] Daniel B. Szyld y Mark T. Jones. Two–stage and multisplitting methods for the parallel solution of linear systems. *SIAM Journal on Matrix Analysis and Applications*, 13:671–679, 1992.

[96] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33:103–111, 1990.

[97] Richard S. Varga. Factorization and normalized iterative methods. En Rudolph E. Langer, editor, *Boundary Problems in Differential Equations*, 121–142. The University of Wiscocsin Press, Madison, 1960.

[98] Richard S. Varga. *Matrix Iterative Analysis.* Prentice Hall, 1962. Reimpreso en versión aumentada por Springer, Berlín, 1999.

[99] Deren R. Wang. On the convergence of the parallel multisplitting AOR algorithm. *Linear Algebra and its Applications*, 154–156:473–486, 1991.

[100] Robert E. White. Multisplittings and parallel iterative methods. *Computer Methods in Applied Mechanics and Engineering*, 64:567–577, 1987.

[101] Robert E. White. Multisplitting with different weighting schemes. *SIAM Journal on Matrix Analysis and Applications*, 10:481–493, 1989.

[102] Robert E. White. Multisplitting of a symmetric positive definite matrix. *SIAM Journal on Matrix Analysis and Applications*, 11:69–82, 1990.

[103] Z. I. Woźnicki. The AGA two-sweep methods for the solutions of linear equation systems. *Linear Algebra and its Applications*, 121:702–710, 1989.

[104] Z. I. Woźnicki. Nonnegative splitting theory. *Japan Journal on Industrial and Applied Mathematics*, 11:289–342, 1994.

[105] David M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1972.