Estrategias de umbral versus de memoria prescrita

Para concluir nuestro estudio numérico de los filtros de carbón activo, contrastamos las dos factorizaciones incompletas de Cholesky. De acuerdo a la figura 4.14, la estrategia de umbral es más eficiente que la de memoria prescrita, ya que requiere menos tiempo de CPU para una cantidad de memoria (número de no nulos) dada. Dicho resultado contrasta lo reportado por Lin y Moré (1999). Sin embargo, se debe enfatizar que los requerimientos de memoria de las estrategias de umbral no pueden predecirse. Así, si la memoria es una restricción crítica pueden ser preferibles las estrategias de memoria prescrita, más predecibles aunque menos eficientes.

También hemos realizado un estudio de convergencia de ambas FIC para diferentes parámetros y distintos pasos de tiempo como lo muestra las gráficas de la figura 4.15. Estos resultados están asociados al filtro B con la malla gruesa.

Al comparar en el número de iteraciones producidas por $p = 0$ y la FIC sin llenado (ver figura 4.15), vemos que no coinciden. Esto es debido a que la FIC de memoria prescrita respeta el número de elementos distintos de cero de la matriz original $\mathbf{A}$, pero no la posición de los mismos. En general, $p = 0$ es un mejor precondicionador que una FIC sin llenado, ver figura 4.14.

La figura 4.16 presenta las gráficas del número de iteraciones contra pasos de tiempo. Se emplea el filtro B para ambas mallas utilizando la FIC sin llenado. Observemos las gráficas 4.16.a y 4.16.b, el número máximo de iteraciones se alcanza en los pasos de tiempo $11\,985$ y $13\,315$, respectivamente. De la gráfica 4.16.c vemos que el flujo atraviesa las cámaras de aire del paso de tiempo $13\,050$ al $13\,725$. En consecuencia, el número de iteraciones es afectado por el material y la geometría de las cámaras. Por la gran variación de las iteraciones a lo largo del tiempo se define, como medida de análisis, el *promedio de iteraciones* igual al número acumulado de iteraciones entre el número de pasos de tiempo.

Un punto de vista más cuantitativo se ofrece en la tabla 4.6. Para el filtro A y una selección de FIC descritas en la figura 4.14, se muestra: (1) el número acumulado de iteraciones, (2) el número de pasos de tiempo, (3) el promedio de iteraciones, (4) el número de entradas no nulas en el factor $\mathbf{L}$; y los tiempos de CPU requeridos: (5) para calcular $\mathbf{L}$, (6) resolver los sistemas lineales en todos los pasos de tiempo, y (7) en total.

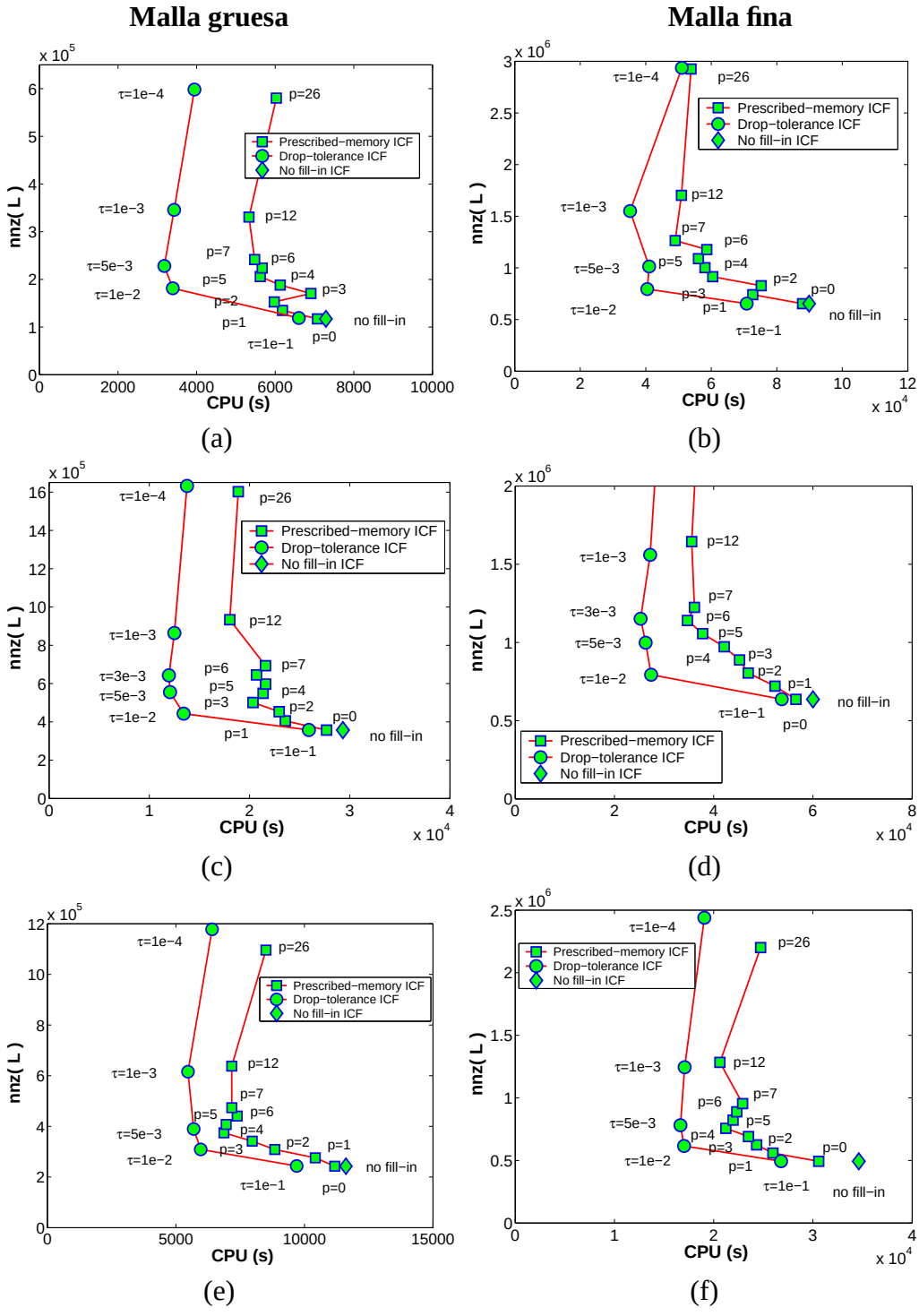


Figura 4.14: Coste computacional de las FIC de umbral y memoria prescrita para los tres filtros: (a) filtro A, malla gruesa, (b) filtro A, malla fina, (c) filtro B, malla gruesa, (d) filtro B, malla fina, (e) filtro C, malla gruesa, y (f) filtro C, malla fina.

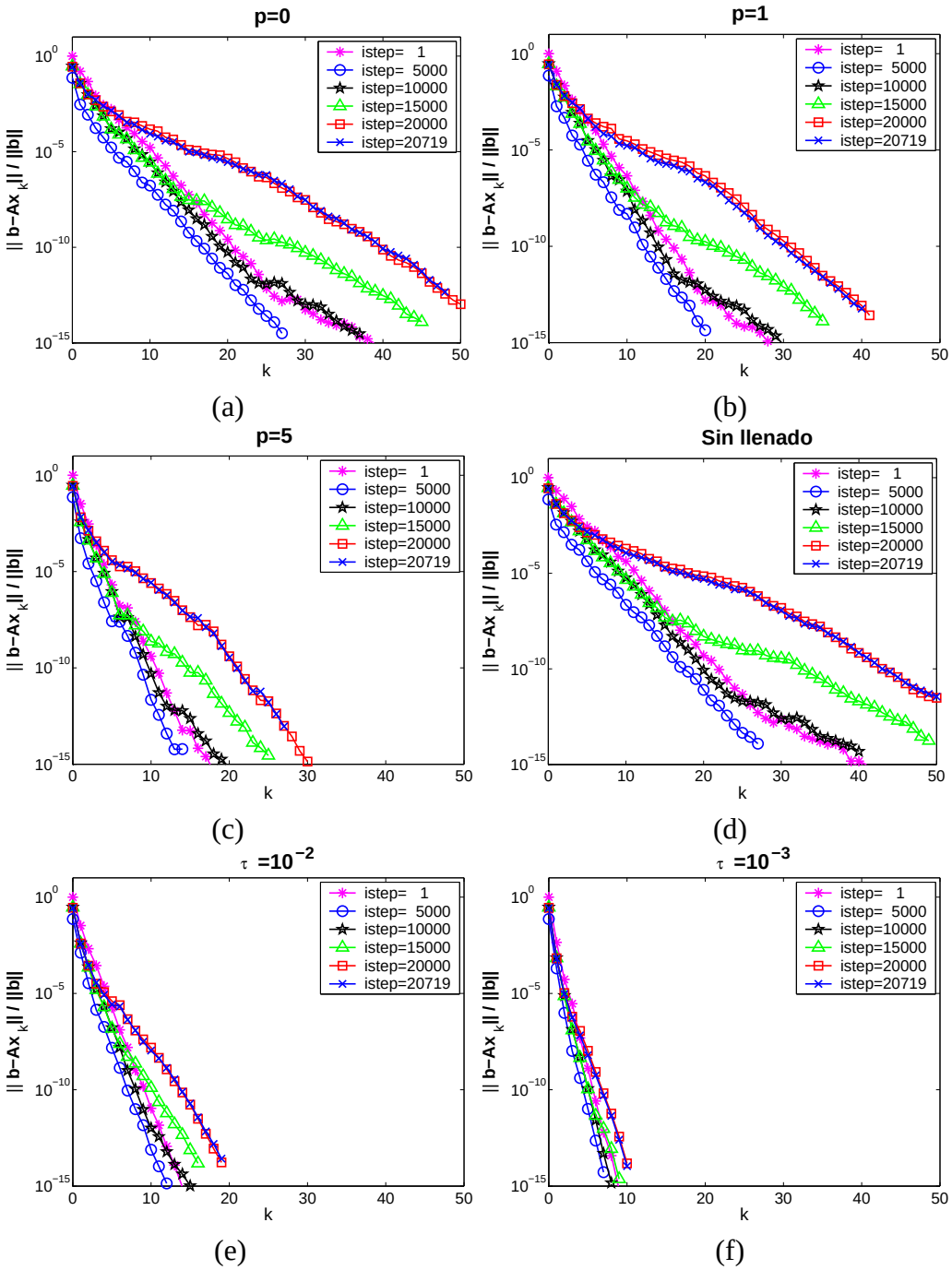
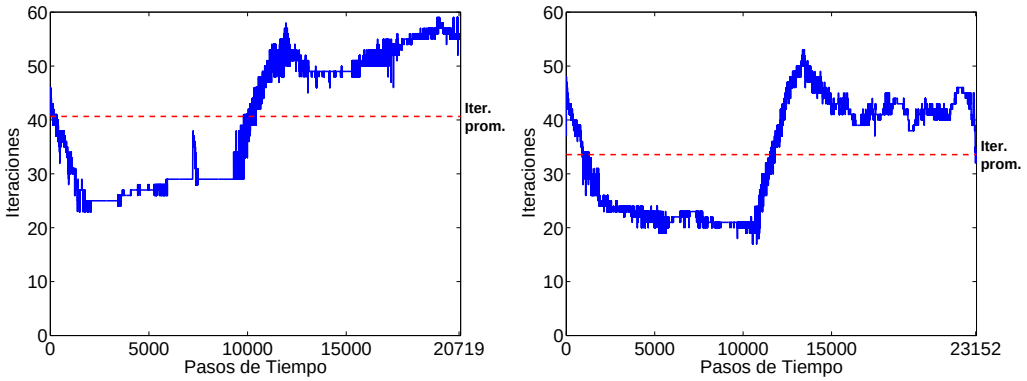


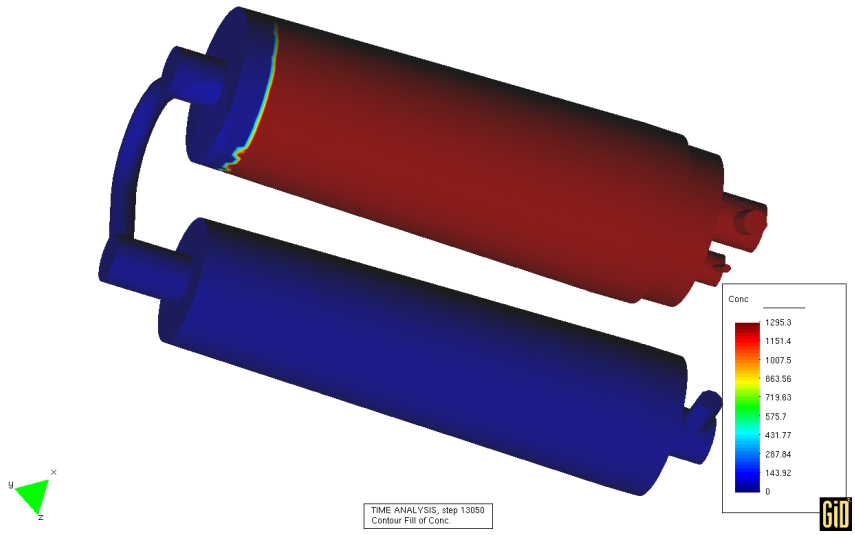
Figura 4.15: Gráficas de convergencia para el filtro B (malla gruesa). Se usan diferentes precondicionadores: (a) $p = 0$, (b) $p = 1$, (c) $p = 3$, (d) sin llenado, (e) $\tau = 10^{-2}$ y (f) $\tau = 10^{-3}$.

Los siguientes aspectos se pueden resaltar de la tabla 4.6:

- Para una estrategia de precondicionamiento dada, el número de iteraciones depende de la calidad del precondicionador, es decir, de su densidad: mayor número de entradas no nulas en $\mathbf{L}$ significan menor número de iteraciones.



(a) (b)



(c)

Figura 4.16: Gráficas iteraciones versus pasos de tiempo para la FIC sin llenado. Filtro B con las mallas: (a) gruesa y (b) fina. (c) Simulación en el paso de tiempo $13\,050$, malla fina.

- El tiempo de CPU, sin embargo, puede decrecer o incrementarse. Un precondicionador denso requiere menos iteraciones, pero cada iteración es más costosa (contrastar las dos FIC de umbral).

	FIC sin llenado		**FIC de umbral**			
			$\tau = 5 \cdot 10^{-3}$		$\tau = 10^{-4}$	
	Gruesa	Fina	Gruesa	Fina	Gruesa	Fina
# iteraciones	1 079 406	1 060 343	238 244	327 287	103 675	176 583
# pasos	17 961	32 867	17 961	32 867	17 961	32 867
iter/pasos	60.10	32.26	13.26	9.96	5.77	5.37
$\mathrm{nnz}(\mathbf{L})$	117 181	653 435	228 041	1 013 875	598 245	2 937 167
Precond. (s)	0.2	1.8	0.5	4.0	2.4	18.0
Solución (s)	6 918.7	86 523.9	2 871.1	37 679.6	3 639.0	47 558.9
TOTAL (s)	6 918.9	86 525.7	2 871.6	37 683.6	3 641.4	47 576.9

	FIC memoria prescrita					
	$p = 0$		$p = 3$		$p = 7$	
	Gruesa	Fina	Gruesa	Fina	Gruesa	Fina
# iteraciones	1 024 771	1 025 930	662 734	605 856	404 001	377 358
# pasos	17 961	32 867	17 961	32 867	17 961	32 867
iter/pasos	57.05	31.21	36.90	18.43	22.49	11.48
$\mathrm{nnz}(\mathbf{L})$	117 181	653 435	170 364	915 092	241 621	1 264 893
Precond. (s)	0.1	0.4	0.1	0.6	0.1	0.7
Solución (s)	6 779.3	84 561.5	6 587.6	57 167.5	5 173.8	45 746.4
TOTAL (s)	6 779.4	84 561.9	6 587.7	57 168.1	5 173.9	45 747.1

Tabla 4.6: Coste computacional de las FIC para el filtro A.

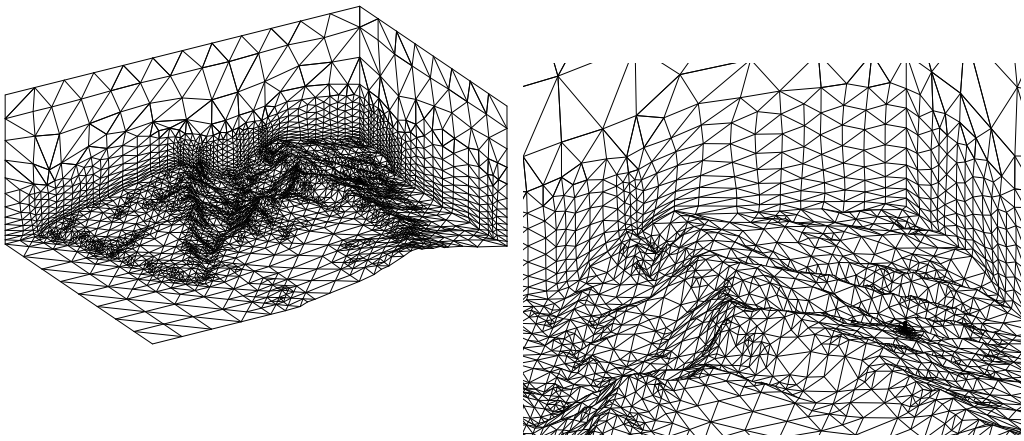
- Para los problemas de convección-difusión transitorios, el tiempo necesario para calcular el factor incompleto $\mathbf{L}$ es *despreciable* comparado con el tiempo requerido para resolver los sistemas lineales en todos los pasos de tiempo. Cuánto tiempo se invierte en calcular el precondicionador es un hecho irrelevante, debido a que se amortiza en muchos pasos de tiempo. Las dos cuestiones claves son: (1) el número de iteraciones y (2) el tiempo necesario para *aplicar* el precondicionador en cada iteración.

- En el caso del método directo de Cholesky ocurre lo mismo. Para el rango de problemas estudiados, es más importante el tiempo necesario para realizar las sustituciones en cada paso de tiempo que el requerido para efectuar la descomposición de la matriz $\mathbf{A}$.

4.8. Aplicación: dispersión de contaminantes

En todas las simulaciones de carbón activo discutidas anteriormente, la factorización completa de Cholesky es factible. Esto nos ha permitido realizar una comparación detallada, la cual muestra claramente que el método de CG precondicionado con una buena FIC gana al método directo en términos de tiempo de CPU y requerimientos de memoria. Sin embargo, se puede argumentar que esos ejemplos no ilustran la necesidad real de los métodos iterativos, puesto que en todo caso se logró calcular la descomposición completa, aunque sea costosa.

Considerando lo anterior, se presenta un ejemplo más grande relacionado con la modelización del transporte de contaminantes en la atmósfera descrito en el apartado 2.2.2.



(a) (b)

Figura 4.17: Malla de elementos finitos para el problema de dispersión de contaminantes: (a) malla de la superficie y (b) detalle mostrando la orografía compleja y la chimenea.

El dominio 3D de estudio está formado por una de región rectangular $15\,600$ m $\times 22\,803$ m en la zona sur de la Isla de La Palma (Islas Canarias), y un plano horizontal como frontera superior situado a $9\,000$ m de altura. Montenegro, Montero, Escobar y Rodríguez (2002) han desarrollado la discretización adaptativa del dominio, ver figura 4.17. Además, se han aplicado técnicas de suavizado y desen-

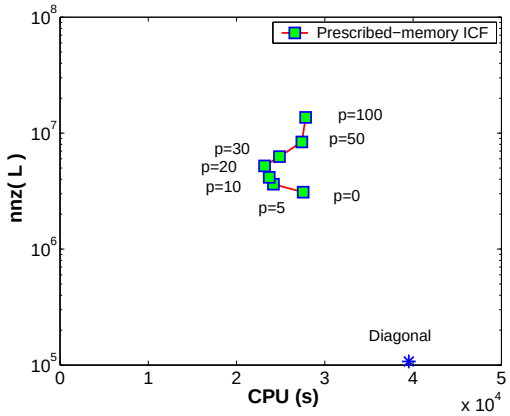


Figura 4.18: Coste computacional de las FIC de memoria prescrita para el problema de dispersión de contaminantes.

redo para construir la malla formada por $153\,085$ tetraedros y $28\,387$ nodos (ver Escobar, Rodríguez, Montenegro, Montero y González-Yuste 2003). Cabe recordar que el campo de viento ha sido simulado por Montero et al. (2005). Tanto la malla como el campo de viento han sido facilitados por sus autores para emplearlos en este estudio.

La simulación realiza $4\,000$ pasos de tiempo con $\Delta t = 0.13$ (seleccionado según el criterio de Courant). La figura 4.18 muestra la eficiencia de gradientes conjugados precondicionado. Claramente, las FIC de memoria prescrita superan al precondicionador diagonal en términos del tiempo de CPU mientras la memoria se mantiene controlada. Notemos que se precisan valores de p grandes para incrementar tanto los requerimientos de memoria como el tiempo de CPU.

El tamaño del problema es $N = 107\,548$ (cuatro variables por nodo no Dirichlet) y la matriz del sistema tiene $\mathrm{nnz}(\mathbf{A}) = 3\,094\,002$ entradas no nulas. Cabe señalar que se utiliza un Cuthill-McKee inverso y una ordenación por nodo de las especies con la finalidad de reducir el llenado del método directo. Aun así, este requiere 4.25 Gb de memoria, es decir, más de la mitad de la RAM total del servidor multi-usuarios. Esto provoca, en la práctica, que Cholesky no sea viable por lo que es necesario un método iterativo.

Como es imprescindible controlar la memoria, hemos elegido la FIC de memoria prescrita como precondicionador. La memoria necesaria para $p = 0$ es de 510 Mb. Además, por cada unidad de p se precisa reservar 1.64 Mb. Esto significa que

se puede alcanzar hasta un valor de $p = 100$ con 674 Mb de memoria total.

4.9. Conclusiones

En este capítulo se ha analizado la eficiencia numérica de dos familias de factorizaciones incompletas de Cholesky (FIC) para el método de gradientes conjugados precondicionado (PCG): de umbral y memoria prescrita. Con un valor apropiado de sus correspondientes parámetros numéricos (el umbral τ y la densidad de llenado p, respectivamente) las FIC son competitivas ante CG precondicionado diagonalmente y el método directo de Cholesky.

De acuerdo con nuestros experimentos numéricos sobre los filtros, el umbral τ se debe seleccionar en el rango $[0.005, 0.01]$ para los sistemas lineales de orden mayor que $30\,000$. Resultados similares están reportados en Dickinson y Forsyth (1994) para problemas de elasticidad. Por lo que concierne al parámetro de densidad p, nuestros ensayos numéricos sugieren tomar $p = 4$ o $p = 5$. El último valor también es recomendado en Lin y Moré (1999).

Nuestros resultados numéricos también muestran que, para problemas de convección-difusión con un campo de velocidad constante (la misma matriz en todas las etapas de tiempo), la eficiencia numérica de un precondicionador está completamente controlada por *su aplicación* en cada paso de tiempo. Puesto que el coste computacional por la construcción del precondicionador se puede amortizar en muchos pasos de tiempo (que en la aplicación de los filtros son de decenas de miles), cuánto cuesta *obtener* la factorización se vuelve un *factor irrelevante*.

Esta conclusión motiva a estudiar otros tipos de precondicionadores más eficientes, aunque sea más costoso obtenerlos, como se verá en el próximo capítulo.

Capítulo 5

Inversa aproximada simétrica

En este capítulo desarrollamos una inversa aproximada *sparse* simétrica (SSPAI) para resolver sistemas SDP, tridiagonales por bloques con múltiples lados derechos basada en la SPAI de Montero et al. (2002). Recordemos que de nuestro problema de interés (convección-difusión) se obtienen sistemas SDP debido a la estabilización con mínimos cuadrados, estructura tridiagonal causada por la reordenación bloque-interface y múltiples lados derechos debido al carácter transitorio del problema. El capítulo se divide esencialmente en tres partes, la primera introduce los conceptos necesarios, así como las relaciones fundamentales para construir la inversa aproximada (IA). En la segunda parte, apartado 5.3, se proponen diversas estrategias para generar una inversa aproximada simétrica hasta conseguir el algoritmo de la SSPAI. Finalmente, en la tercera parte se ilustra la eficiencia de estos precondicionadores mediante experimentos numéricos en 2D y 3D, incluyendo los filtros de carbón activo (ver apartado 5.4).

5.1. Introducción

En las últimas décadas, gracias el crecimiento de los ordenadores en paralelo, ha habido un gran interés en desarrollar algoritmos para construir inversas aproximadas tipo *sparse*. La idea común de esta clase de técnicas es calcular explícitamente una matriz *sparse* $\mathbf{M}$ que aproxime a $\mathbf{A}^{-1}$ y usarla como precondicionador en algún método iterativo de Krylov como se indica en el apartado 3.3. Existen varias ventajas en utilizar inversas aproximadas, la más importante es poder paralelizar

masivamente la operación de precondicionamiento en cada etapa del método iterativo. Recordemos que dicha operación consiste en realizar un producto de matriz por vector: $\mathbf{z} = \mathbf{Mr}$, ver figura 3.2. Otra virtud de las aproximadas principalmente para matrices fuertemente no simétricas y/o no definidas es su robustez ante las factorizaciones incompletas; se sabe que estas últimas fallan cuando los pivotes son cero o hay inestabilidades numéricas (ver apartado 4.4). En cambio la mayoría de las técnicas de inversas aproximadas no son vulnerables a estos problemas (Montero et al. 2002, Grote y Huckle 1997, Gould y Scott 1998, Huckle 1998, Cosgrove et al. 1992, Díaz y Macedo 1989, Benzi, Cullum y Tůma 2000, Kolotilina y Yeremin 1993). También se ha reportado que si se resuelven sistemas no simétricos con muchos lados derechos, las IA pueden competir con las factorizaciones incompletas aún en un ambiente secuencial, ver Benzi y Tůma (1999).

Las técnicas de inversas aproximadas pueden clasificarse en dos grandes grupos según si el precondicionador $\mathbf{M}$ se obtiene por medio de una sola matriz o como el producto de dos o más matrices. A su vez este último grupo se subdivide en las inversas aproximadas factorizadas y los métodos de precondicionamiento que consisten en realizar una factorización incompleta de $\mathbf{A}$ seguida de la aproximación de las inversas de sus factores (Benzi y Tůma 1998, Kolotilina y Yeremin 1993). Dentro del primer grupo se encuentran gran parte de los métodos de minimización en la norma de Frobenius (ver Montero et al. 2002, Grote y Huckle 1997, Gould y Scott 1998, Huckle 1998, Cosgrove et al. 1992, Díaz y Macedo 1989). Observemos que la FSAI de Kolotilina y Yeremin (1993) también se obtiene minimizando la norma de Frobenius.

Estamos interesados en construir un precondicionador $\mathbf{M}$ que sea *sparse* y simétrico tal que el coste de aplicarlo en una iteración de CG sea el mismo que el de realizar un producto de la matriz $\mathbf{A}$ por un vector. Es decir el número de entradas no nulas de $\mathbf{M}$ debe ser muy cercano al número de no nulos de la matriz $\mathbf{A}$.

Por lo regular la inversa de una matriz *sparse* es densa. Sin embargo, es frecuente que las entradas de dicha inversa sean pequeñas en valor absoluto. Por lo que es posible encontrar una matriz *sparse* $\mathbf{M}$ que sea una buena aproximación a $\mathbf{A}^{-1}$ si se eligen adecuadamente las "entradas grandes o relevantes". Se han desarrollado varias técnicas que están encaminadas a obtener en forma dinámica las posiciones

de las "entradas más relevantes" de $\mathbf{A}^{-1}$, ver por ejemplo Chow y Saad (1998), Benzi y Tůma (1999), Grote y Huckle (1997) y Montero et al. (2002). En particular, hemos considerado las propuestas de estas dos últimas referencias.

Especialmente para las matrices simétricas definidas positivas, no solamente se deben buscar las "entradas relevantes" de $\mathbf{A}^{-1}$, sino también aquellas que hagan que el precondicionador $\mathbf{M}$ sea SDP. Puesto que es frecuente encontrarse que al generar una inversa aproximada de una matriz simétrica definida positiva, ésta no sea simétrica y por tanto no se puede usar como precondicionador en gradientes conjugados. Se han propuesto diferentes precondicionadores para matrices SDP, por ejemplo: la inversa aproximada *sparse* factorizada (FSAI) desarrollada por Kolotilina y Yeremin (1993), la inversa aproximada *sparse* factorizada (AINV) de Benzi, Meyer y Tůma (1996) y su versión estabilizada (SAINV) de Benzi et al. (2000). Todas ellas pertenecen al grupo de IA obtenidas mediante dos o más matrices. Por su parte, para los precondiconadores explícitos construidos con una sola matriz, como las SPAIs de Grote y Huckle (1997) y de Montero et al. (2002) en donde el patrón de *sparsidad* es dinámico, no se ha desarrollado una versión para matrices SDP. Para el caso en donde el patrón de *sparsidad* está prescrito ver Carpentieri, Duff, Giraud y Magolu (2004).

Finalmente, se conoce que la fase de construcción de las SPAIs se caracteriza por ser relativamente costosa, pero se gana en robustez y sobre todo, el coste en la fase de aplicación puede ser menor que el de otros precondicionadores implícitos por la propiedad de paralelismo masivo. Esto resulta ser muy atractivo para los problemas transitorios como se verifica en nuestros experimentos numéricos.

En este capítulo se plantea un algoritmo para construir una inversa aproximada *sparse* simétrica (SSPAI) basada en la SPAI de Montero et al. (2002). Estamos interesados en resolver los sistemas (2.17) con múltiples vectores independientes, donde $\kappa(\mathbf{A})$ sea relativamente grande. En el apartado 5.3 proponemos diversas estrategias para generar un precondicionador simétrico hasta conseguir el algoritmo de la SSPAI. La efectividad o calidad de las inversa aproximadas la ilustramos mediante diferentes sistemas SDP asociados a problemas 2D y 3D, incluyendo la deformación de una laja elástica, un problema de calor libre en un dominio cuadrado, la simulación del campo de viento en la isla de Gran Canaria y los filtros de carbón

activo, ver apartado 5.4. Además, para complementar nuestro análisis comparamos los resultados numéricos (de los problemas de convección-difusión transitorios) con la familia de factorizaciones incompletas de Cholesky de umbral descritas en el capítulo anterior.

A continuación exponemos las ideas de los métodos de minimización en la norma de Frobenius y se obtienen las fórmulas necesarias para construir las IA propuestas.

5.2. Técnicas de minimización en la norma de Frobenius y expresión explícita para la SPAI simétrica

Existen varias técnicas para crear un precondicionador explícito que se basan en la minimización en la norma de Frobenius (Cosgrove et al. 1992, Grote y Huckle 1997, Gould y Scott 1998, Montero et al. 2002). El propósito de éstas es construir una matriz $\mathbf{M}$, en este caso por la derecha, tal que satisfaga el problema de minimización restringido al subespacio de matrices *sparse* $\mathscr{S}$ $(\subset \mathscr{M}_N(\mathbb{R}))$:

$$\min_{\mathbf{M}\in\mathscr{S}} \|\mathbf{AM}-\mathbf{I}\|_F, \tag{5.1}$$

donde $\mathbf{I}$ es la matriz identidad, $\mathbf{A}$ es una matriz no simétrica y $\|\cdot\|_F$ denota la norma de Frobenius definida por $\|\mathbf{A}\|_F^2 = \langle \mathbf{A}, \mathbf{A}\rangle_F$ con $\langle \mathbf{A}, \mathbf{B}\rangle_F = \mathrm{tr}(\mathbf{AB}^T)$. Esta norma tiene la virtud de satisfacer la propiedad

$$\|\mathbf{AM}-\mathbf{I}\|_F^2 = \sum_{k=1}^{N} \|\mathbf{Am}_k - \mathbf{e}_k\|_2^2, \tag{5.2}$$

con $\mathbf{m}_k$ y $\mathbf{e}_k$ las k-ésimas columnas de $\mathbf{M}$ y $\mathbf{I}$, respectivamente. Así, basta con resolver en forma independiente N problemas de mínimos cuadrados para encontrar la solución de (5.1). De esa manera, las columnas de la inversa aproximada *sparse* $\mathbf{M}$ pueden calcularse y aplicarse en paralelo.

Notemos que el subespacio de restricciones $\mathscr{S}$ determina la *sparsidad* de $\mathbf{M}$ filtrando las entradas de $\mathbf{A}^{-1}$ que contribuyen poco a la calidad del precondicionador. En general es difícil conocer las posiciones de la inversa donde las entradas tienen valores relevantes. Si por ejemplo, se utiliza el mismo patrón de *sparsidad* que el de la matriz $\mathbf{A}$ no siempre se produce un buen precondicionador (ver Benzi 2002).

Existen otros casos en donde una IA con un patrón de *sparsidad* estático funciona adecuadamente. Sin embargo la *sparsidad* de $\mathbf{M}$ depende las características del problema al cual esté asociado la matriz $\mathbf{A}$ (ver Carpentieri, Duff y Giraud 2000). Por ello, preferimos buscar el patrón de *sparsidad* adaptativamente (dinámicamente) como lo realizan los algoritmos de Montero et al. (2002) y Grote y Huckle (1997). El inconveniente de estas técnicas es no producir una inversa aproximada simétrica, aunque la matriz $\mathbf{A}$ lo sea. Esto obstaculiza el uso de los métodos iterativos diseñados para resolver sistemas simétricos, como PCG.

Nuestro objetivo es construir una inversa aproximada *sparse* simétrica basada en los métodos de minimización en la norma de Frobenius buscando un patrón de *sparsidad* dinámicamente. Para ello, nos apoyamos en la inversa aproximada *sparse* de Montero et al. (2002) y la SPAI de Grote y Huckle (1997) como se muestra en el apartado 5.3. Por la importancia de ambos algoritmos, los describimos brevemente.

5.2.1. Precondicionadores tipo SPAI

En general una inversa aproximada *sparse* (SPAI) se construyen columna a columna en forma iterativa. El proceso iterativo en cada columna k está formado por las siguientes etapas:

Primera etapa**.** Definir el conjunto de índices de entradas candidatas a entrar en el patrón de *sparsidad* de $\mathbf{M}$.

Segunda etapa**.** Elegir el o los índices óptimos.

Tercera etapa**.** Actualizar $\mathbf{m}_k$ y evaluar $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2^2$.

Estas etapas se repiten hasta conseguir la convergencia $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2^2 \leq \epsilon$ o alcanzar el número máximo de llenado n_k.

En la primera etapa de la SPAI, tanto de Montero et al. (2002) como de Grote y Huckle (1997), se define el conjunto de entradas candidatas $\mathcal{J}$ de la columna $\mathbf{m}_k$ mediante el siguiente criterio: sean $\mathbf{r} = \mathbf{A}\mathbf{m}_k - \mathbf{e}_k$ el residuo de la k-ésima columna e $\mathcal{I} = \{i \in \{1, 2, ..., N\} | r_{ik} \neq 0\}$ el conjunto de índices de las entradas no nulas en $\mathbf{r}$. Si $\mathcal{L} = \{l \in \{1, 2, ..., N\} | m_{lk} \neq 0\}$, entonces la nueva entrada candidata se busca en $\mathcal{J} = \{j \in \mathcal{L}^c | a_{ij} \neq 0, \forall i \in \mathcal{I}\}$, donde $\mathcal{L}^c$ es el complemento del conjunto $\mathcal{L}$.

Observemos que las únicas entradas consideradas en $\mathbf{m}_k$ son aquellas que afectan las entradas no nulas de $\mathbf{r}$.

En la segunda etapa del algoritmo de Montero et al. (2002) se utiliza una fórmula acumulativa para ver qué entrada candidata es la óptima. Esto es, para cada entrada candidata $j \in \mathcal{J}$ se asume que $j = i_p$ es la entrada no nula actual de $\mathbf{m}_k$. Entonces se admite que la *sparsidad* de $\mathbf{m}_k$ es $\mathcal{L} \cup \{j\} = \{i_1, i_2, ..., i_p\}$ y se evalúa la siguiente relación:

$$\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2^2 = 1 - \sum_{l=1}^{p} \frac{[\det(D_l)]^2}{\det(G_{l-1})\det(G_l)} \tag{5.3}$$

donde $\det(G_0) = 1$ y G_l es la matriz de Gram de las columnas $i_1, i_2, ..., i_l$ de la matriz $\mathbf{A}$ con respecto al producto interior euclidiano, D_l se calcula sustituyendo la última fila de la matriz G_l por $a_{ki_1}, a_{ki_2}, ..., a_{ki_l}$, con $1 \leq l \leq p$, ver teorema 3.1 de Montero et al. (2002).

Se inicia con el precondicionador diagonal óptimo
Para $k = 1 : N$
    Mientras $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2 > \epsilon$ y $p < n_k$ hacer
        Calcular $\mathbf{r} = \mathbf{A}\mathbf{m}_k - \mathbf{e}_k$
        Defi nir $\mathcal{I}, \mathcal{L}$ y $\mathcal{J}$
        $\forall j \in \mathcal{J}$ calcular $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2^2$ usando (5.3)
        Para j_k óptima actualizar $\mathbf{m}_k$ y $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2^2$ usando (5.4) y (5.3)
    Terminar
Terminar

Figura 5.1: Algoritmo de la SPAI de Montero, González, Flórez, García y Suárez

Posteriormente, se selecciona el índice óptimo j_k que minimiza $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2^2$. En la tercera etapa se actualiza $\mathbf{m}_k$ para $j_k = i_p$ con la expresión

$$\mathbf{m}_k = \sum_{l=1}^{p} \frac{[\det(D_l)]^2}{\det(G_{l-1})\det(G_l)} \widetilde{\mathbf{m}}_l \tag{5.4}$$

donde $\widetilde{\mathbf{m}}_l$ es un vector que se obtiene evaluando el determinante correspondiente que resulta de reemplazar la última fila de $\det(G_l)$ por $\mathbf{e}_h$, con $1 \leq h \leq p$, ver teorema 3.1 de Montero et al. (2002). Los aspectos teóricos y computacionales se analizan en Montero et al. (2002) y Vázquez (2003). Dicho algoritmo se presenta en la figura 5.1.

Por su parte, en la segunda etapa del algoritmo de Grote y Huckle (1997) se propone una forma barata de seleccionar la entrada que minimiza $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2$. Se resuelve para cada $j \in \mathcal{J}$ el problema de minimización unidimensional

$$\min_{\mu_j} \|\mathbf{r} + \mu_j \mathbf{A}\mathbf{e}_j\|_2,$$

cuya solución es $\mu_j = -\frac{\mathbf{r}^T \mathbf{A}\mathbf{e}_j}{\|\mathbf{A}\mathbf{e}_j\|_2^2}$. O sea, para cada j se calcula la norma de ρ_j del nuevo residuo $\mathbf{r} + \mu_j \mathbf{A}\mathbf{e}_j$:

$$\rho_j^2 = \|\mathbf{r}\|_2^2 - \frac{(\mathbf{r}^T \mathbf{A}\mathbf{e}_j)^2}{\|\mathbf{A}\mathbf{e}_j\|_2^2}. \tag{5.5}$$

Después, se seleccionan los s índices de las entradas con las ρ_j más pequeñas. Para los nuevos índices óptimos (tercera etapa) se establece un nuevo problema de mínimos cuadrados que se resuelve mediante una descomposición QR, ver detalles en Grote y Huckle (1997). Tal algoritmo se presenta en la figura 5.2.

Para $k = 1 : N$
  Elegir una *sparsidad* inicial $\mathcal{J}$ y defi nir $\mathcal{I}$ y $\mathcal{L}$
  Calcular la descomposición QR de $\widehat{\mathbf{A}} = \mathbf{A}(\mathcal{I}, \mathcal{J})$
  y $\mathbf{r} = \widehat{\mathbf{A}}\widehat{\mathbf{m}}_k - \widehat{\mathbf{e}_k}$, con $\widehat{\mathbf{m}}_k = \mathbf{R}^{-1}\mathbf{Q}^T\widehat{\mathbf{e}_k}$
  Mientras $\|\mathbf{r}\|_2 > \epsilon$ y $p < n_k$ hacer
    Defi nir $\mathcal{I}, \mathcal{L}$ y $\mathcal{J}$
    $\forall j \in \mathcal{J}$ calcular ρ_j con (5.5), y elegir s índices con las ρ_j más pequeñas
    Determinar los nuevos $\widetilde{\mathcal{I}}$ y $\widetilde{\mathcal{J}}$
    Actualizar la descomposición QR de $\mathbf{A}(\widetilde{\mathcal{I}} \cup \mathcal{I}, \widetilde{\mathcal{J}} \cup \mathcal{J})$ y
    calcular $\mathbf{m}_k$ y $\|\mathbf{r}\| = \|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2$ para $\mathcal{I} = \widetilde{\mathcal{I}} \cup \mathcal{I}, \mathcal{J} = \widetilde{\mathcal{J}} \cup \mathcal{J}$
  Terminar
Terminar

Figura 5.2: Algoritmo de la SPAI de Grote y Huckle

Cabe señalar que la estrategia iterativa en la SPAI puede iniciar con alguna estructura predeterminada del precondicionador o con el diagonal óptimo:

$$\operatorname{diag}\Big(\frac{a_{11}}{\|\mathbf{A}\mathbf{e}_1\|_2^2}, \frac{a_{22}}{\|\mathbf{A}\mathbf{e}_2\|_2^2}, ..., \frac{a_{NN}}{\|\mathbf{A}\mathbf{e}_N\|_2^2}\Big). \tag{5.6}$$

5.2.2. Expresión explícita para la inversa aproximada *sparse* simétrica

Estamos interesados en encontrar expresiones similares a las identidades (5.3) y (5.4) para construir una inversa aproximada simétrica. Esto es, relaciones para actualizar los valores de $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2^2$ y $\mathbf{m}_k$ si únicamente se eligen las entradas de la triangular inferior porque las entradas de la triangular superior se fijan por simetría. Con este fin, en este apartado y en el siguiente emplearemos las ideas y notación que a continuación se detallan.

De la propiedad (5.2) se verifica que la norma de Frobenius satisface la relación

$$\|\mathbf{AM} - \mathbf{I}\|_F^2 = \sum_{j=1}^{N} \|\mathbf{AM}_j - \mathbf{E}_{j,j}\|_F^2, \tag{5.7}$$

donde las $\mathbf{M}_j$ son matrices cuyos únicos no nulos están en la j-ésima columna y $\mathbf{E}_{i,j}$ es la matriz de dimensión $N \times N$ cuya única entrada no nula es $e_{ij} = 1$. Escribimos a $\mathbf{M}_j$ como $\mathbf{M}_j^L + \mathbf{M}_j^U$ donde los no nulos de las matrices $\mathbf{M}_j^L$ y $\mathbf{M}_j^U$ están en la triangular inferior (incluyendo la diagonal) y la triangular superior, respectivamente.

Definimos por $\mathscr{S}$ a un subespacio de $\mathscr{M}_N(\mathbb{R})$ formado por matrices simétricas y a $\mathscr{S}_j$ un subespacio de $\mathscr{S}$ constituído por matrices cuyas únicas entradas no nulas están en la j-ésima columna. A su vez $\mathscr{S}_j = \mathscr{S}_j^L \oplus \mathscr{S}_j^U$, donde los no nulos de las matrices en los subespacios $\mathscr{S}_j^L$ y $\mathscr{S}_j^U$ están en la triangular inferior (incluyendo la diagonal) y la triangular superior, respectivamente.

La idea es encontrar la matriz simétrica que sea solución del problema (5.1). Si utilizamos la propiedad (5.7) y la notación de los dos párrafos anteriores podemos reescribir el problema de minimización (5.1) como

$$\min_{\mathbf{M}\in\mathscr{S}} \|\mathbf{AM} - \mathbf{I}\|_F^2 = \min_{\forall\, j,\ \mathbf{M}_j^L \in \mathscr{S}_j^L} \sum_{j=1}^{N} \|\mathbf{AM}_j^L + \mathbf{AM}_j^U - \mathbf{E}_{j,j}\|_F^2, \tag{5.8}$$

donde las variables de diseño independientes son $\mathbf{M}_1^L, \mathbf{M}_2^L, \ldots, \mathbf{M}_N^L$ y las dependientes debido al acoplamiento por simetría son $\mathbf{M}_2^U, \mathbf{M}_3^U, \ldots, \mathbf{M}_N^U$. Notemos que $\mathbf{M}_j^U$ depende solamente de las matrices $\mathbf{M}_1^L, \mathbf{M}_2^L, \ldots, \mathbf{M}_{j-1}^L$.

El acoplamiento entre las variables $\mathbf{M}_j^L$ y $\mathbf{M}_j^U$ *rompe* la idea de separar el problema (5.8) en N problemas independientes. Debido a que el objetivo principal es encontrar una *aproximación* a la inversa de la matriz original $\mathbf{A}$ y no la inversa exacta $\mathbf{A}^{-1}$, proponemos construir una inversa aproximada simétrica $\mathbf{M}^0$ iterativamente columna a columna, aunque se pierda la independencia del cálculo entre columnas y no se encuentre el óptimo que satisface (5.8). Esto es, vamos a resolver los N problemas que están del lado derecho de la relación

$$\min_{\mathbf{M}\in\mathscr{S}} \|\mathbf{AM}-\mathbf{I}\|_F^2 \simeq \sum_{j=1}^{N} \min_{\mathbf{M}_j^L\in\mathscr{S}_j^L} \|\mathbf{AM}_j^L + \mathbf{AM}_j^{0,U} - \mathbf{E}_{j,j}\|_F^2, \tag{5.9}$$

donde cada $\mathbf{M}_j^{0,U}$ está predeterminada por simetría (depende de $\mathbf{M}_1^{0,L},\ldots,\mathbf{M}_{j-1}^{0,L}$). Nuestro objetivo es encontrar una fórmula explícita mediante la teoría de aproximación en espacios prehilbertianos para

$$\mathbf{M}_j^{0,L} = \arg\min_{\mathbf{M}_j^L\in\mathscr{S}_j^L} \|\mathbf{AM}_j^L + \mathbf{AM}_j^{0,U} - \mathbf{E}_{j,j}\|_F.$$

De esa manera la inversa aproximada $\mathbf{M}^0$ que construimos satisface que

$$\mathbf{M}^0 = \sum_{j=1}^{N} \mathbf{M}_j^0,$$

donde $\mathbf{M}_j^0 = \mathbf{M}_j^{0,L} + \mathbf{M}_j^{0,U}$.

Para fijar la dimensión de los subespacios de estudio, suponemos que la matriz $\mathbf{M}_j^0$ tiene p entradas no nulas, de las cuales q están en la triangular inferior. O sea, las dimensiones de los subespacios $\mathscr{S}_j$ y $\mathscr{S}_j^L$ son p y q, respectivamente. Así, el problema de minimización del lado derecho de la relación (5.9) se escribe como

$$\min_{\mathbf{M}_j^L\in\mathscr{S}_j^L} \|\mathbf{AM}_j^L - \mathbf{V}j,j\|_F^2 = \|\mathbf{AM}_j^{0,L} - \mathbf{V}j,j\|_F^2, \tag{5.10}$$

donde $\mathbf{V}j,j = \mathbf{E}j,j - \mathbf{AM}_j^{0,U}$.

Finalmente, las expresiones eficientes y acumulativas para actualizar $\mathbf{M}_j^{0,L}$ y evaluar $\|\mathbf{AM}_j^{0,L} - \mathbf{V}j,j\|_F^2$ se presentan a través de los siguientes dos teoremas. Se utiliza la misma teoría que en los resultados de Montero et al. (2002). El primero de ellos muestra una fórmula explícita para la solución del problema (5.10) expresada en una base ortogonal de $\mathbf{A}\mathscr{S}_j^L$. El segundo generaliza dicha expresión para una base arbitraria de $\mathscr{S}_j^L$.

Teorema 5.1. *Sean* $\mathbf{A}$ *una matriz SDP y* $\mathscr{S}_j$ *un subespacio de* $\mathscr{S}$ *de dimensión* p *formado por matrices cuyas únicas entradas no nulas están en la j-ésima columna. Se considera que* $\mathscr{S}_j = \mathscr{S}_j^L \oplus \mathscr{S}_j^U$*, donde los no nulos de las matrices en los subespacios* $\mathscr{S}_j^L$ *y* $\mathscr{S}_j^U$ *están en la triangular inferior (incluyendo la diagonal) y la triangular superior, respectivamente. Si* $\{\mathbf{S}_1, \ldots, \mathbf{S}_q\}$ *con* $q \leq p$ *es una base de* $\mathscr{S}_j^L$ *tal que* $\{\mathbf{AS}_1, \ldots, \mathbf{AS}_q\}$ *es una base ortogonal de* $\mathbf{A}\mathscr{S}_j^L$*. Entonces la solución al problema* (5.10) *es:*

$$\mathbf{M}_j^{0,L} = \sum_{l=1}^{q} \frac{\mathrm{tr}(\mathbf{AS}_l) - \langle \mathbf{AM}_j^{0,U}, \mathbf{AS}_l \rangle_F}{\|\mathbf{AS}_l\|_F^2} \mathbf{S}_l \tag{5.11}$$

$$\|\mathbf{AM}_j^{0,L} - \mathbf{V}_{j,j}\|_F^2 = \|\mathbf{V}_{j,j}\|_F^2 - \sum_{l=1}^{q} \frac{[\mathrm{tr}(\mathbf{AS}_l) - \langle \mathbf{AM}_j^{0,U}, \mathbf{AS}_l \rangle_F]^2}{\|\mathbf{AS}_l\|_F^2} \tag{5.12}$$

Demostración. La proyección ortogonal de $\mathbf{V}_{j,j}$ sobre $\mathbf{A}\mathscr{S}_j^L$ es $\mathbf{AM}_j^{0,L}$. Entonces la representación de $\mathbf{AM}_j^{0,L}$ por medio de la suma de Fourier de $\mathbf{V}_{j,j}$ relacionada a la base ortonormal $\{\mathbf{AS}_l / \|\mathbf{AS}_l\|_F\}_{l=1}^q$ es

$$\mathbf{AM}_j^{0,L} = \sum_{l=1}^{q} \frac{\langle \mathbf{V}_{j,j}, \mathbf{AS}_l \rangle_F}{\|\mathbf{AS}_l\|_F^2} \mathbf{AS}_l. \tag{5.13}$$

Tomando en cuenta que la única columna no nula de la matriz $\mathbf{AS}_l$ es la j-ésima se deduce que $\langle \mathbf{E}_{j,j}, \mathbf{AS}_l \rangle_F = \mathrm{tr}(\mathbf{AS}_l)$. De esa forma $\langle \mathbf{V}_{j,j}, \mathbf{AS}_l \rangle_F = \mathrm{tr}(\mathbf{AS}_l) - \langle \mathbf{AM}_j^{0,U}, \mathbf{AS}_l \rangle_F$. Al sustituir la última igualdad en la relación (5.13) se concluye la ecuación (5.11).

Por otro lado, dado que $\langle \mathbf{AM}_j^{0,L} - \mathbf{V}_{j,j}, \mathbf{AS}_l \rangle_F = 0$ para toda $\mathbf{S}_l \in \mathscr{S}_j^L$ y $\mathbf{V}j, j = \mathbf{E}j, j - \mathbf{AM}_j^{0,U}$ se obtiene

$$\|\mathbf{AM}_j^{0,L} - \mathbf{V}_{j,j}\|_F^2 = \langle \mathbf{V}_{j,j}, \mathbf{V}_{j,j} \rangle_F - \langle \mathbf{AM}_j^{0,L}, \mathbf{E}_{j,j} \rangle_F + \langle \mathbf{AM}_j^{0,L}, \mathbf{AM}_j^{0,U} \rangle_F.$$

Al sustituir la relación (5.11) en la igualdad anterior se deriva la ecuación (5.12). □

Este teorema, además de establecer una fórmula acumulativa para calcular $\|\mathbf{AM}_j^{0,L} - \mathbf{V}_{j,j}\|_F^2$, también propone una forma de actualizar la matriz $\mathbf{M}_j^{0,L}$. Por esta razón, resulta atractivo generalizar el teorema anterior a una base arbitraria del subespacio $\mathscr{S}_j^L$ como se muestra a continuación.

Teorema 5.2. *Sean* $\mathbf{A}$ *una matriz SDP y* $\mathscr{S}_j$ *un subespacio de* $\mathscr{S}$ *de dimensión* p *formado por matrices cuyas únicas entradas no nulas están en la j-ésima columna. Se considera que* $\mathscr{S}_j = \mathscr{S}_j^L \oplus \mathscr{S}_j^U$*, donde los no nulos de las matrices en los subespacios* $\mathscr{S}_j^L$ *y* $\mathscr{S}_j^U$ *están en la triangular inferior (incluyendo la diagonal) y la triangular superior, respectivamente. Si* $\{\mathbf{S}_1, \ldots, \mathbf{S}_q\}$ *con* $q \leq p$ *es una base de* $\mathscr{S}_j^L$*, entonces la solución al problema* (5.10) *es:*

$$\mathbf{M}_j^{0,L} = \sum_{l=1}^{q} \frac{\det(D_l) - \det(H_l)}{\det(G_{l-1})\det(G_l)} \widetilde{\mathbf{S}}_l \tag{5.14}$$

$$\|\mathbf{A}\mathbf{M}_j^{0,L} - \mathbf{V}_{j,j}\|_F^2 = \|\mathbf{V}_{j,j}\|_F^2 - \sum_{l=1}^{q} \frac{[\det(D_l) - \det(H_l)]^2}{\det(G_{l-1})\det(G_l)} \tag{5.15}$$

donde, $\det(G_0) = 1$ *y* G_l *es la matriz de Gram del sistema* $\{\mathbf{AS}_1, \ldots, \mathbf{AS}_l\}$ *con respecto al producto interior de Frobenius,* D_l *es la matriz que resulta de reemplazar la última fila de la matriz* G_l *por* $\mathrm{tr}(\mathbf{AS}_1), \mathrm{tr}(\mathbf{AS}_2), \ldots, \mathrm{tr}(\mathbf{AS}_l)$*,* H_l *es la matriz que resulta de reemplazar la última fila de la matriz* G_l *por* $\gamma_1, \gamma_2 \ldots, \gamma_l$*, donde* $\gamma_l = \langle \mathbf{A}\mathbf{M}_j^{0,U}, \mathbf{AS}_l \rangle_F$*; y* $\widetilde{\mathbf{S}}_l$ *es la matriz obtenida al evaluar el determinante simbólico que resulta de reemplazar la última fila de* $\det(G_l)$ *por* $\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_l$*, con* $1 \leq l \leq q$*.*

Demostración. Al aplicar el algoritmo de Gram-Schmidt a la base $\{\mathbf{AS}_l\}_{l=1}^q$ de $\mathbf{A}\mathscr{S}_j^L$ se obtiene la base ortogonal $\{\mathbf{A}\widetilde{\mathbf{S}}_l\}_{l=1}^q$. En efecto, si denotamos por

$$\widetilde{\mathbf{S}}_l = \begin{vmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1l} \\ \vdots & \vdots & & \vdots \\ \alpha_{(l-1)1} & \alpha_{(l-1)2} & \cdots & \alpha_{(l-1)l} \\ \mathbf{S}_1 & \mathbf{S}_2 & \cdots & \mathbf{S}_l \end{vmatrix}; \quad \alpha_{hk} = \langle \mathbf{AS}_h, \mathbf{AS}_k \rangle_F,$$

entonces

$$\forall h < k, \quad \langle \mathbf{AS}_h, \mathbf{A}\widetilde{\mathbf{S}}_k \rangle_F = \begin{vmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1k} \\ \vdots & \vdots & & \vdots \\ \alpha_{(k-1)1} & \alpha_{(k-1)2} & \cdots & \alpha_{(k-1)k} \\ \alpha_{h1} & \alpha_{h2} & \cdots & \alpha_{hk} \end{vmatrix} = 0$$

debido a que existen dos filas iguales. De esa forma, $\langle \mathbf{A}\widetilde{\mathbf{S}}_h, \mathbf{A}\widetilde{\mathbf{S}}_k \rangle_F = 0$ para toda $h < k$.

Para normalizar la base ortogonal de $\mathbf{A}\mathscr{S}_j^L$ tenemos que

$$\|\mathbf{A}\widetilde{\mathbf{S}}_l\|_F^2 = \langle \det(G_{l-1})\mathbf{AS}_l, \mathbf{A}\widetilde{\mathbf{S}}_l \rangle_F = \det(G_{l-1})\det(G_l), \quad \forall l = 1, 2, ..., q. \tag{5.16}$$

Por su parte,

$$\operatorname{tr}\left(\mathbf{A}\widetilde{\mathbf{S}}_l\right) = \begin{vmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1l} \\ \vdots & \vdots & & \vdots \\ \alpha_{(l-1)1} & \alpha_{(l-1)2} & \cdots & \alpha_{(l-1)l} \\ \operatorname{tr}(\mathbf{AS}_1) & \operatorname{tr}(\mathbf{AS}_2) & \cdots & \operatorname{tr}(\mathbf{AS}_l) \end{vmatrix} = \det(D_l), \quad \forall l = 1, ..., q. \tag{5.17}$$

También observemos que

$$\langle \mathbf{AM}_j^{0,U}, \mathbf{A}\widetilde{\mathbf{S}}_l \rangle_F = \begin{vmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1l} \\ \vdots & \vdots & & \vdots \\ \alpha_{(l-1)1} & \alpha_{(l-1)2} & \cdots & \alpha_{(l-1)l} \\ \gamma_1 & \gamma_2 & \cdots & \gamma_l \end{vmatrix} = \det(H_l), \quad \forall l = 1, ..., q. \tag{5.18}$$

donde $\gamma_l = \langle \mathbf{AM}_j^{0,U}, \mathbf{AS}_l \rangle_F$.

Si aplicamos el teorema 5.1 a la base ortogonal $\{\mathbf{A}\widetilde{\mathbf{S}}_l\}_{l=1}^q$ de $\mathbf{A}\mathscr{S}_j^L$ y consideramos las relaciones (5.16), (5.17) y (5.18) se deducen las ecuaciones (5.14) y (5.15). □

Notemos que las matrices G_l, D_l, H_l y $\widetilde{\mathbf{S}}_l$ dependen de la base del espacio $\mathscr{S}_j^L$. Si elegimos la base canónica se pueden simplificar los cálculos como se expone a continuación.

Denotemos por K_j, K_j^L y K_j^U a los patrones de *sparsidad* de $\mathbf{M}^0$, $\mathbf{M}_j^{0,L}$ y $\mathbf{M}_j^{0,U}$, respectivamente. Notemos que $K_j = K_j^L \cup K_j^U$. Supongamos que $K_j^L = \{(i_1, j), (i_2, j), \ldots, (i_q, j)\}$ y que el subespacio $\mathscr{S}_j^L$ está generado por las base canónica $\{\mathbf{E}_{i_1,j}, \ldots, \mathbf{E}_{i_q,j}\}$, con $j \leq i_1 < i_2 < \cdots < i_q \leq N$. En este caso, la j-ésima columna no nula de la matriz $\mathbf{AE}_{i,j}$ coincide con la i-ésima de $\mathbf{A}$, por lo que $\forall (i,j), (i',j) \in K_j^L : \langle \mathbf{AE}_{i,j}, \mathbf{AE}_{i',j} \rangle_F = \langle \mathbf{Ae}_i, \mathbf{Ae}_{i'} \rangle_2$. Además se satisface

que $\mathrm{tr}(\mathbf{A}\mathbf{E}_{l,j}) = a_{jl}$. Al considerar lo anterior en el teorema 5.2 se obtiene que las fórmulas (5.14) y (5.15) son válidas y se cumple para todo $l = 1, 2, \ldots, q$ que $\gamma_l = \langle \mathbf{A}\mathbf{M}_j^{0,U}, \mathbf{A}\mathbf{E}_{i_l,j} \rangle_F$, $\alpha_{hk} = \langle \mathbf{A}\mathbf{e}_{i_h}, \mathbf{A}\mathbf{e}_{i_k} \rangle_2$,

$$\det(G_l) = \begin{vmatrix} \alpha_{11} & \cdots & \alpha_{1l} \\ \vdots & & \vdots \\ \alpha_{l1} & \cdots & \alpha_{ll} \end{vmatrix}, \quad \det(D_l) = \begin{vmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1l} \\ \vdots & \vdots & & \vdots \\ \alpha_{(l-1)1} & \alpha_{(l-1)2} & \cdots & \alpha_{(l-1)l} \\ a_{ji_1} & a_{ji_2} & \cdots & a_{ji_l} \end{vmatrix},$$

$$\det(H_l) = \begin{vmatrix} \alpha_{11} & \cdots & \alpha_{1l} \\ \vdots & & \vdots \\ \alpha_{(l-1)1} & \cdots & \alpha_{(l-1)l} \\ \gamma_1 & \cdots & \gamma_l \end{vmatrix}, \quad \widetilde{\mathbf{S}}_l = \begin{vmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1l} \\ \vdots & \vdots & & \vdots \\ \alpha_{(l-1)1} & \alpha_{(l-1)2} & \cdots & \alpha_{(l-1)l} \\ \mathbf{E}_{i_1,j} & \mathbf{E}_{i_2,j} & \cdots & \mathbf{E}_{i_l,j} \end{vmatrix}. \tag{5.19}$$

Observemos que el cálculo del $\det(H_l)$ es costoso porque se necesita realizar una multiplicación matriz-vector para cada γ_l. Debido a esto, hemos optado por no incluir el $\det(H_l)$ en las relaciones (5.14) y (5.15). Por tanto las fórmulas que se utilizan en nuestros experimentos numéricos son

$$\mathbf{M}_j^{0,L} \simeq \sum_{l=1}^{q} \frac{\det(D_l)}{\det(G_{l-1})\det(G_l)} \widetilde{\mathbf{S}}_l, \tag{5.20}$$

$$\|\mathbf{A}\mathbf{M}_j^{0,L} - \mathbf{V}_{j,j}\|_F^2 \simeq \|\mathbf{V}_{j,j}\|_F^2 - \sum_{l=1}^{q} \frac{[\det(D_l)]^2}{\det(G_{l-1})\det(G_l)}. \tag{5.21}$$

De hecho, hemos verificado numéricamente que si la inversa aproximada simétrica $\mathbf{M}^0$ satisface columna a columna las relaciones (5.20) y (5.21) y cumple que

$$\|\mathbf{A}\mathbf{M}_j^{0,L} - \mathbf{V}j, j\|_F < \epsilon < \frac{\epsilon_0}{\sqrt{N}},$$

entonces para ϵ_0 suficientemente pequeñas, $\mathbf{M}^0$ tiende a $\mathbf{A}^{-1}$.

Cabe indicar que la inversa aproximada más simple que podemos obtener es el precondicionador diagonal óptimo (ecuación (5.6)). Estas inversas son definidas positivas para las matrices que estudiamos. Por otro lado, si $\epsilon_0 \to 0$ entonces $\mathbf{M}^0$ será definida positiva (DP). Sin embargo para cualquier valor de ϵ_0 dado, no podemos asegurar que la inversa aproximada obtenida sea DP. A pesar de ello, en

todos nuestros resultados numéricos no hemos tenido problema al aplicar la inversa aproximada $\mathbf{M}^0$.

5.3. Inversa aproximada *sparse* simétrica

En este apartado mostraremos cómo se construye la inversa aproximada sparse simétrica (SSPAI) para una matriz $\mathbf{A}$ SDP, tridiagonal por bloques. Debido a que esto no es una tarea fácil presentaremos diversas estrategias hasta obtener el algoritmo de la SSPAI. Además indicaremos las ventajas e inconvenientes de cada una de ellas. También, veremos que cada versión de la inversa aproximada es una modificación necesaria para aumentar la calidad del precondicionador.

Como ya ha sido indicado, hemos considerado que la generación de una SPAI está formada, para cada columna k, por tres etapas fundamentales. En la primera se definen el conjunto de índices de entradas candidatas $\mathcal{J}$, y los conjuntos de índices $\mathcal{I}$ y $\mathcal{L}$. La elección de la o las entradas candidatas óptima se realiza en la segunda etapa, ya sea mediante un problema de minimización unidimensional (5.5) o una fórmula acumulativa (5.3). Por último, se actualiza el precondicionador $\mathbf{m}_k$ y se calcula $\|\mathbf{A}\mathbf{m}_k - \mathbf{e}_k\|_2^2$ empleando las fórmulas (5.4) y (5.3) o mediante la descomposición QR.

Las inversas aproximadas *sparse* simétricas expuestas en este trabajo están basada en la SPAI de Montero et al. (2002). Sin embargo, la forma en que se realiza la búsqueda del candidato óptimo es costosa con respecto a cómo se selecciona en el algoritmo de Grote y Huckle (1997). Por ello, en todas las estrategias calculamos la relación (5.5) para cada entrada candidata $j \in \mathcal{J}$ y seleccionamos los s índices con las ρ_j más pequeñas. Además, al igual que las SPAI, nuestros algoritmos pueden iniciar con cualquier estructura del precondicionador, no obstante hemos preferido iniciar en todos ellos con el precondicionador diagonal óptimo definido en (5.6). Denotamos por $\mathbf{M}$ a la inversa aproximada de una matriz no simétrica construida con las características detalladas en este párrafo. A continuación describimos cada una de las estrategias.

5.3.1. Estrategia 1: precondicionador parte-simétrica

Una primera idea intuitiva para construir un precondicionador simétrico es utilizar la parte simétrica de la inversa aproximada $\mathbf{M}$,

$$\mathbf{M}_{sym-p} = \frac{\mathbf{M} + \mathbf{M}^T}{2}. \tag{5.22}$$

De esta forma conseguimos un precondicionador completamente paralelizable por columnas. Desafortunadamente, con la *estrategia 1* se pierde información de $\mathbf{M}$ y por tanto se reduce la calidad del precondicionador $\mathbf{M}_{sym-p}$. Esto es, se satisface $\|\mathbf{AM} - \mathbf{I}\|_F \leq \|\mathbf{AM}_{sym-p} - \mathbf{I}\|_F$ debido a que $\mathbf{M} = \arg\min_{\mathbf{M}\in\mathscr{S}} \|\mathbf{AM} - \mathbf{I}\|_F$ y cualquier otra matriz proporciona una norma mayor.

Notemos que el valor de $\|\mathbf{AM}_{sym-p} - \mathbf{I}\|_F$ no se puede calcular eficientemente.

5.3.2. Estrategia 2: precondicionador inferior

Buscamos entonces la construcción directa de la parte triangular inferior reformulando el algoritmo de la estrategia 1. Es decir, se buscan las posiciones candidatas óptimas por debajo de la diagonal en cada columna. La parte triangular superior se genera por simetría. Notemos que *es fundamental adecuar el cálculo de* $\mathbf{m}_k$ *y de la norma del residuo* (5.3) *de tal manera que únicamente se consideren las entradas de la triangular inferior. De ahí la importancia de obtener los resultados de los teoremas 5.1 y 5.2, así como las fórmulas* (5.20) *y* (5.21).

Supongamos que estamos calculando $\mathbf{m}_k^L$, sea $\mathbf{r} = \mathbf{Am}_k^L - \mathbf{v}_k$ el residuo en la k-ésima columna, donde $\mathbf{v}_k = \mathbf{e}_k - \mathbf{Am}_k^U$. Para la primera etapa de construcción de la inversa aproximada definimos los siguientes conjuntos de índices:

$$\begin{aligned} &\mathcal{I} = \{i \in \{k, k+1, ..., N\} | r_{ik} \neq 0\}, \quad \mathcal{L} = \{l \in \{k, k+1, ..., N\} | m_{lk} \neq 0\}, \\ &\mathcal{J} = \{j \in \mathcal{L}^c | a_{ij} \neq 0, \forall i \in \mathcal{I}\}, \end{aligned} \tag{5.23}$$

donde $\mathcal{L}^c = \{l \in \{k, k+1, ..., N\} | m_{lk} = 0\}$ es el complemento de $\mathcal{L}$. Observemos que el conjunto $\mathcal{J}$ de índices de entradas candidatas es un subconjunto de $\{k, k+1, ..., N\}$, es decir, son posiciones que están debajo de la entrada m_{kk}. En la segunda etapa, se calcula la relación (5.5) para cada entrada candidata $j \in \mathcal{J}$ y

se seleccionan los s índices con las ρ_j más pequeñas. Por último, se actualiza $\mathbf{m}_k^L$ y $\|\mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k\|_2$ para las s entradas candidatas óptimas: asumimos que el subconjunto $\mathcal{L} = \{i_1, i_2, ..., i_q\}$ es no vacío, con q el número actual de entradas no nulas de la parte triangular inferior de $\mathbf{m}_k$. Luego, de (5.20) y (5.21) se calcula para cada uno de los s índices óptimos

$$\mathbf{m}_k^L = \sum_{l=1}^{q} \frac{[\det(D_l)]^2}{\det(G_{l-1})\det(G_l)} \widetilde{\mathbf{m}}_l \tag{5.24}$$

$$\|\mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k\|_2^2 = \|\mathbf{v}_k\|_2^2 - \sum_{l=1}^{q} \frac{[\det(D_l)]^2}{\det(G_{l-1})\det(G_l)} \tag{5.25}$$

donde $\det(G_0) = 1$, $\det(G_l)$ y $\det(D_l)$ están definidos en (5.19). Además, cada vector $\widetilde{\mathbf{m}}_l$ se obtiene al evaluar el determinante simbólico que resulta de reemplazar la última fila de $\det(G_l)$ por $\mathbf{e}_{i_1}, \mathbf{e}_{i_2}, ..., \mathbf{e}_{i_l}$, $1 \leq l \leq q$. La figura 5.3 muestra el algoritmo para la IA de la *estrategia 2*, denotada por $\mathbf{M}_{low}$.

Se inicia con el precondicionador diagonal óptimo
Para $k = 1 : N$
    Mientras $\|\mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k\|_2 > \epsilon$ y $p < n_k$ hacer
        Calcular $\mathbf{r} = \mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k$
        Defi nir $\mathcal{I}, \mathcal{L}$ y $\mathcal{J}$ usando (5.23)
        $\forall j \in \mathcal{J}$ calcular ρ_j con (5.5), y elegir s índices con las ρ_j más pequeñas
        $\forall j$ óptima actualizar $\mathbf{m}_k^L$ y $\|\mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k\|_2^2$ usando (5.24) y (5.25)
    Terminar
    Calcular $[\mathbf{m}_k^L]^T$
Terminar

Figura 5.3: Algoritmo para construir $\mathbf{M}_{low}$, estrategia 2

Esta inversa aproximada simétrica gana ligeramente en calidad con respecto a la generada con la estrategia 1. Sin embargo, es más costosa la fase de creación de $\mathbf{M}_{low}$ y se pierde la propiedad de paralelismo masivo. Al comparar la entradas significativas de ambos precondicionadores se observa que $\mathbf{M}_{low}$ no incluye algunas entradas significativas que aparecen en la triangular superior de $\mathbf{M}_{sym-p}$. Esta pérdida de entradas significativas en la triangular superior de $\mathbf{M}_{low}$ provoca que la calidad de la inversa no sea tan buena. Una alternativa para mejorar esta deficiencia

es realizar un doble proceso: hacia adelante por la triangular inferior y hacia atrás por la triangular superior, lo que ocasiona doble coste computacional.

5.3.3. Estrategia 3: precondicionador inferior-superior

Nuestra propuesta para mejorar la calidad de $\mathbf{M}_{low}$, de tal forma que el coste computacional no se dispare, consiste en realizar un barrido inicial comenzando con el precondicionador diagonal óptimo y detectar las posiciones más significativas de cada columna situadas por encima de la diagonal. Esto es, para cada columna k:

(i) Se calcula $\mathbf{r} = \mathbf{A}\mathbf{m}_k - \mathbf{e}_k$.

(ii) Se definen los conjuntos $\mathcal{I} = \{i \in \{1, ..., N\} | r_{ik} \neq 0\}$, $\mathcal{L} = \{l \in \{1, \ldots, N\} | m_{lk} \neq 0\}$ y $\mathcal{J} = \{j \in \mathcal{L}^c | a_{ij} \neq 0, \forall i \in \mathcal{I}\}$.

(iii) Se seleccionan los t índices con las ρ_j más pequeñas, es decir $\{j_1, j_2, ..., j_t\} \subset \mathcal{J}$.

(iv) Del conjunto $\{j_1, j_2, ..., j_t\}$ de índices óptimos se eligen las posiciones que están por encima de la diagonal.

(v) Se fija (por simetría) cada una de las posiciones (k, j_l) donde $j_l < k$ y $l \in \{1, 2, ..., t\}$, o sea se actualiza el patrón de *sparsidad* de la j_l-ésima columna $K_{j_l} = K_{j_l} \cup \{(k, j_l)\}$.

Determinado el nuevo patrón de *sparsidad* K comenzamos a construir la inversa aproximada simétrica. El proceso del cálculo de valores de la IA se realiza por columnas, llenando previamente la diagonal y las posiciones que se han fijado. Luego se añaden las entradas que sean necesarias hasta satisfacer $\|\mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k\|_2 < \epsilon$ o el número máximo de llenado n_k. Por último se transpone la columna $\mathbf{m}_k^L$. El algoritmo de la *estrategia 3* se presenta en la figura 5.4 y la IA es denotada por $\mathbf{M}_{low-up}$.

La estrategia 3 mejora relativamente la calidad del precondicionador. Sin embargo, sigue siendo no paralelizable.

Se inicia con el precondicionador diagonal óptimo
Para $k = 1 : N$
  Calcular $\mathbf{r} = \mathbf{A}\mathbf{m}_k - \mathbf{e}_k$
  Defi nir $\mathcal{I}, \mathcal{L}$ y $\mathcal{J}$ usando (ii)
  $\forall j \in \mathcal{J}$ calcular ρ_j con (5.5), y elegir t índices con las ρ_j más pequeñas
  Fijar las posiciones $\{(k, j_l) \mid j_l < k, l \in \{1, 2, ..., t\}\}$
Terminar
Para $k = 1 : N$
  Para cada posición fi jada calcular $\mathbf{m}_k^L$ y $\|\mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k\|_2^2$ usando (5.24) y (5.25)
  Mientras $\|\mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k\|_2 > \epsilon$ y $p < n_k$ hacer
    Calcular $\mathbf{r} = \mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k$
    Defi nir $\mathcal{I}, \mathcal{L}$ y $\mathcal{J}$ usando (5.23)
    $\forall j \in \mathcal{J}$ calcular ρ_j con 5.5, y elegir s índices con las ρ_j más pequeñas
    $\forall j$ óptima actualizar $\mathbf{m}_k^L$ y $\|\mathbf{A}\mathbf{m}_k^L - \mathbf{v}_k\|_2^2$ usando (5.24) y (5.25)
  Terminar
  Calcular $[\mathbf{m}_k^L]^T$
Terminar

Figura 5.4: Algoritmo para construir $\mathbf{M}_{low-up}$, estrategia 3

5.3.4. Estrategia 4: precondicionador SSPAI

Para subsanar la no paralelización de $\mathbf{M}_{low-up}$, nos basamos en las propiedades de las matrices tridiagonales por bloques y el decaimiento de las entradas de su inversa a medida que nos alejamos del bloque diagonal (Axelsson 1996). La idea consiste en restringir el espacio de búsqueda de cada columna para cada bloque. Considerando lo anterior, se propone que el espacio de búsqueda para cada bloque sea la "caja" formada por un bloque anterior y un bloque posterior al bloque en estudio, ver figura 5.5. De esta forma se podrían paralelizar "cajas" cuyo solapes no afecten el cálculo de la IA. De hecho, esta técnica se puede generalizar para otro tipo de "cajas".

El algoritmo de la estrategia 4 comienza definiendo las "cajas" asociadas a cada bloque. Diremos que una "caja" w inicia en h_1^w y termina en h_f^w, con h_1^w la primera columna del bloque $\mathbf{A}_{w-1,w-1}$ y h_f^w la última columna del bloque $\mathbf{A}_{w+1,w+1}$, es decir, está formada por el subconjunto $\{h_1^w, h_2^w, ..., h_f^w\} \times \{h_1^w, h_2^w, ..., h_f^w\} \subset \{1, 2, ..., N\} \times \{1, 2, ..., N\}$, ver figura 5.5. Si n_b es el número de bloques, tendremos el mismo número de "cajas".

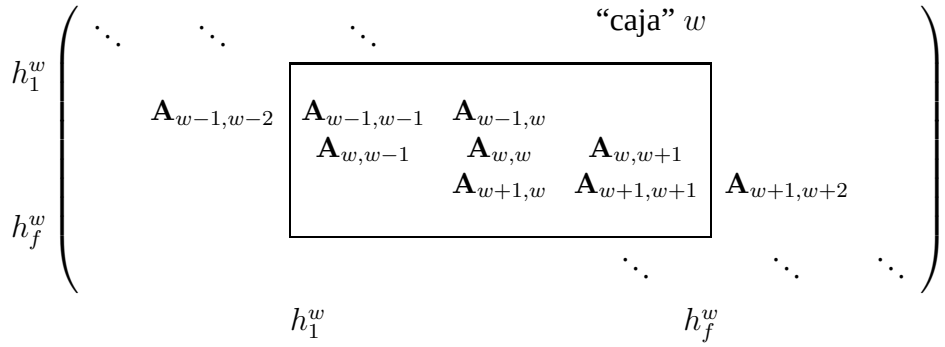


Figura 5.5: La "caja" w asociada al bloque $\mathbf{A}_{w,w}$

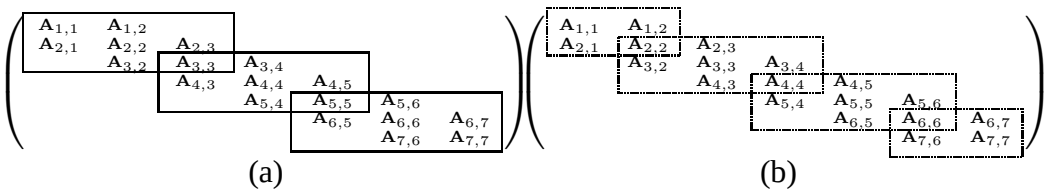


Figura 5.6: "Cajas" asociadas a bloques: (a) pares y (b) impares.

La figura 5.6 muestra que las "cajas" de los bloques pares se solapan en un bloque impar, al igual que las "cajas" impares en uno par. También es importante enfatizar que el precondicionador se puede construir en paralelo gracias a que los bloques de solape no intervienen en la generación de $\mathbf{M}_{low-up}$ por bloques. Por esta razón podemos construir un precondicionador en paralelo empleando $n_b/2$ procesadores, ver algoritmo en la figura 5.8. La inversa aproximada *sparse* simétrica generada con la estrategia 4 es denotada por SSPAI. Las figuras 5.7.a y 5.7.b esquematizan la construcción de la SSPAI para "cajas" impares y pares, respectivamente. La figura 5.7.c muestra que la SSPAI mantiene la misma estructura tridiagonal por bloques de $\mathbf{A}$.

Observemos que la parelización de la SSPAI no es masiva. Sin embargo las 4 estrategias conservan la propiedad de paralelismo masivo en la aplicación del precondicionador.

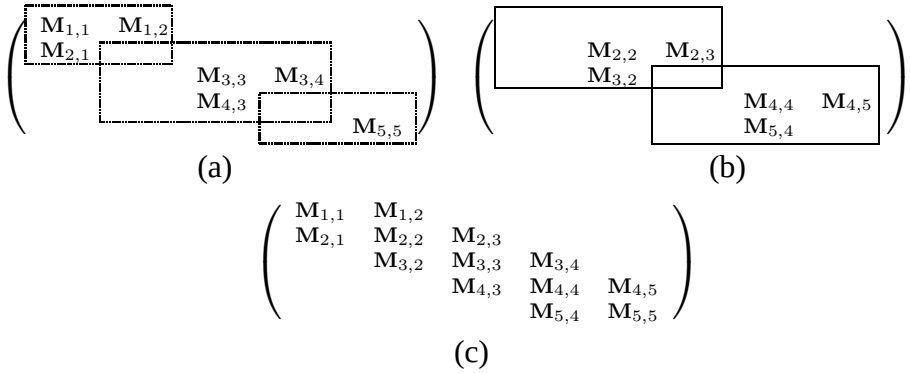


Figura 5.7: SSPAI asociada a “cajas”: (a) impares, (b) pares y (c) totales.

5.3.5. Aspectos computacionales y tipos de almacenamiento

Al igual que en la referencia (Montero et al. 2002), empleamos el método de Cholesky para obtener el $\det(G_l)$ y aprovechamos la descomposición previa de $\det(G_{l-1})$. Por lo que sólo realizamos la factorización de la última fila y columna. También, utilizamos el hecho de que la última incógnita del sistema $G_l d_l = (a_{ki_1}, a_{ki_2}, ..., a_{ki_l})^T$ sea igual al $\det(D_l)/\det(G_l)$, de esa manera efectuamos únicamente una sustitución hacia adelante. Finalmente, para determinar el valor de $\widetilde{\mathbf{m}}_l$ resolvemos el sistema $G_l v_l = e_l$, donde $\widetilde{m}_{i_h l} = v_{hl}$, $(1 \leq h \leq l)$.

Por otro lado, se ha empleado para las 4 propuestas una modificación del formato Ellpack-Itpack para almacenar la IA por columna (MCEI), ver Saad (2003). En dicho almacenamiento se emplea la matriz `Q_mat`$(n_k - 1, N)$ que guarda las entradas no nulas de la IA para las N columnas. La matriz `Qfil_mat`$(n_k - 1, N)$ contiene el número de fila de las entradas no nulas de la IA, `Qcol_vec`(N) almacena el número de entradas no nulas en cada columna sin los elementos diagonales y en `diag_Q`(N) están los elementos diagonales.

5.4. Experimentos numéricos

Nuestro interés en este apartado es mostrar la eficiencia de las cuatro inversas aproximadas *sparse* simétricas descritas en el apartado anterior. Para ello, ilustramos numéricamente las ventajas e inconvenientes de cada una de las estrategias. El

Defi nición de n_b "cajas": $\{h\ ^w_1, h^w_2, ..., h^w_f\} \times \{h^w_1, h^w_2, ..., h^w_f\}\ \forall w = 1, 2, ..., n_b$
Se inicia con el precondicionador diagonal óptimo
Para w "cajas" impares, después pares
  Para cada columna k de $\mathbf{A}_{w,w}$
    Calcular $\mathbf{r} = \mathbf{A}\mathbf{m}_k - \mathbf{e}_k$
    Defi nir $\mathcal{I} = \{i \in \{h^w_1, h^w_2, ..., h^w_f\} | r_{ik} \neq 0\}, \mathcal{L} = \{l \in \{h^w_1, h^w_2, ..., h^w_f\} |$
    $m_{lk} \neq 0\}$ y $\mathcal{J} = \{j \in \mathcal{L}^c / a_{ij} \neq 0, \forall i \in \mathcal{I}\}$
    $\forall j \in \mathcal{J}$ calcular ρ_j con (5.5), y elegir t índices con las ρ_j más pequeñas
    Fijar las posiciones $\{(k, j_l) | j_l < k, l \in \{1, 2, ..., t\}\}$
  Terminar
Termina
Para w "cajas" impares, después pares
  Para cada columna k de $\mathbf{A}_{w,w}$
    Para cada posición fi jada calcular $\mathbf{m}^L_k$ y $\|\mathbf{A}\mathbf{m}^L_k - \mathbf{v}_k\|^2_2$ usando (5.24) y (5.25)
    Mientras $\|\mathbf{A}\mathbf{m}^L_k - \mathbf{v}_k\|_2 > \epsilon$ y $p < n_k$ hacer
      Calcular $\mathbf{r} = \mathbf{A}\mathbf{m}^L_k - \mathbf{v}_k$
      Defi nir $\mathcal{I} = \{i \in \{k, k+1, ..., h^w_f\} | r_{ik} \neq 0\}$,
      $\mathcal{L} = \{l \in \{k, k+1, ..., h^w_f\} | m_{lk} \neq 0\}$ y $\mathcal{J} = \{j \in \mathcal{L}^c / a_{ij} \neq 0, \forall i \in \mathcal{I}\}$
      $\forall j \in \mathcal{J}$ calcular ρ_j con (5.5), y elegir s índices con las ρ_j más pequeñas
      $\forall j$ óptima actualizar $\mathbf{m}^L_k$ y $\|\mathbf{A}\mathbf{m}^L_k - \mathbf{v}_k\|^2_2$ usando (5.24) y (5.25)
    Terminar
    Calcular $[\mathbf{m}^L_k]^T$
  Terminar
Terminar

Figura 5.8: Algoritmo de la SSPAI, estrategia 4.

apartado se divide en dos partes, en la primera presentamos ejemplos 2D y 3D para verificar la efectividad esencialmente de las tres primeras estrategias. En la segunda parte se utilizan los filtros de carbón activo descritos en la figura 4.8 y tabla 4.1 con el fin de revisar nuevamente la calidad de los precondicionadores simétricos propuestos, pero ahora poniendo énfasis en la SSPAI, es decir la estrategia 4. Para colocarlos en contexto, los resultados se comparan con los precondicionadores: diagonal, diagonal óptimo y las FIC de umbral. Análogamente al capítulo anterior, en las confrontaciones del coste computacional se toman en cuenta los requerimientos de memoria y el tiempo de CPU.

Resumiendo, en el transcurso de este apartado analizamos los siguientes pre-

condicionadores y parámetros:

- M, inversa aproximada *sparse* para matriz no simétrica,
- $\mathbf{M}_{sym-p}$, inversa aproximada simétrica calculada con la estrategia 1,
- $\mathbf{M}_{low}$, inversa aproximada simétrica calculada con la estrategia 2,
- $\mathbf{M}_{low-up}$, inversa aproximada simétrica calculada con la estrategia 3,
- SSPAI, inversa aproximada simétrica calculada con la estrategia 4,
- FIC(τ), factorización incompleta de Cholesky de umbral descrita en el apartado 4.2,
- FIC(∞), factorización incompleta de Cholesky sin llenado, detallada en el apartado 4.3,
- ϵ, tolerancia para la columna k,
- n_k, número máximo de no nulos en la columna k,
- s, número de índices candidatos óptimos,
- t, máximo número de entradas significativas en la triangular superior,
- τ, umbral para las FIC.

Cabe señalar que todos los experimentos numéricos se han realizado sobre un procesador Itanium 2 (1.3GHz) de una HP rx5670. Los códigos están programados en Fortran 90 con aritmética en doble precisión y han sido compilados con la opción -tpp2.

5.4.1. Ejemplos generales de validación

Como ya ha sido mencionado, en este subapartado se comparan las estrategias 1, 2 y 3. Para estos experimentos numéricos el criterio de parada del método PCG es: $\left\|\mathbf{b}-\mathbf{A}\mathbf{x}^{(k+1)}\right\|_2 \leq tol_r \left\|\mathbf{b}\right\|_2$, con $tol_r = 10^{-8}$.

Ejemplo 1: deformación de una laja elástica. En este problema se estudia la deformación de una laja elástica delgada sometida a dos cargas, una constante y otra que varía linealmente, ver modelo completo en Suárez (1995). La discretización del problema conduce a un sistema simétrico de $8\,434$ ecuaciones con $\mathrm{nnz}(\mathbf{A}) = 33\,671$ entradas no nulas.

ϵ	n_k	s	t	$\frac{\mathrm{nnz}(\bullet)}{\mathrm{nnz}(\mathbf{A})}$	Its.	P-time	Its-time	T-time	$\|\mathbf{AM}-\mathbf{I}\|_F$	$\|\mathbf{A}\bullet-\mathbf{I}\|_F$
Estrategia 1: $\mathbf{M}_{sym-p}$										
0.5	5	5		0.39	468	23.209	0.371	23.580	44.5	44.2
0.4	8	4		0.68	316	47.352	0.292	47.644	33.7	34.3
0.3	15	15		1.78	242	100.572	0.320	100.892	25.5	29.4
0.1	50	25		7.50	153	992.880	0.523	993.403	16.5	17.7
Estrategia 2: $\mathbf{M}_{low}$										
0.5	5	5		0.41	438	28.451	0.350	28.801		42.7
0.4	8	4		0.71	323	50.307	0.304	50.611		34.1
0.3	15	15		1.88	274	99.528	0.374	99.902		31.8
0.1	50	25		6.38	269	542.042	0.809	542.851		24.3
Estrategia 3: $\mathbf{M}_{low-up}$										
0.5	5	5	3	0.73	321	43.173	0.303	43.476		42.5
0.4	8	4	4	1.07	293	70.436	0.315	70.751		41.9
0.3	15	15	6	1.99	251	122.945	0.354	123.299		42.8
0.1	50	25	20	6.38	165	531.866	0.494	532.360		64.9

Tabla 5.1: Resultados numéricos de las tres primeras estrategias para el problema de elasticidad.

El número de condición de la matriz es $\kappa(\mathbf{A}) \approx 9.03 \times 10^4$, por lo que es necesario acelerar la convergencia de CG. En la tabla 5.1 se presentan los resultados numéricos al utilizar las estrategias 1, 2 y 3. En ésta y en las tablas restantes de este capítulo se emplea Its. como abreviación del número de iteraciones; P-time y Its-time indican el tiempo de CPU (en segundos) al construir y aplicar el precondicionador, respectivamente. Además, T-time es la suma de los dos tiempos anteriores.

Notemos de la tabla 5.1 que efectivamente $\|\mathbf{AM}-\mathbf{I}\|_F \leq \|\mathbf{AM}_{sym-p}-\mathbf{I}\|_F$; por ejemplo para $\epsilon = 0.3$: $\|\mathbf{AM}-\mathbf{I}\|_F = 25.5$ y $\|\mathbf{AM}_{sym-p}-\mathbf{I}\|_F = 29.4$. También se observa que, conforme crece el llenado en $\mathbf{M}_{sym-p}$ el coste de su aplicación aumenta drásticamente con respecto a las inversas aproximadas de las estrategias 2 y 3.

Por su parte, la estrategia 2 construye directamente una IA simétrica y el cálculo de $\|\mathbf{A}\mathbf{M}_{low} - \mathbf{I}\|_F$ es barato debido al empleo de la relación (5.25). Sin embargo, al comparar los resultados para $\epsilon = 0.1$ de $\mathbf{M}_{sym-p}$ y $\mathbf{M}_{low}$, notamos que el precondicionador construido por la segunda estrategia perdió calidad, aumentó el número de iteraciones (aunque el tiempo de construcción, y por tanto, el tiempo total sean menores). Para subsanar esta deficiencia incorporamos la información de la triangular superior de $\mathbf{A}$ mediante el parámetro t. La tabla 5.1 muestra cómo la calidad del precondicionador aumenta cuando se utiliza el parámetro t. Por ejemplo para

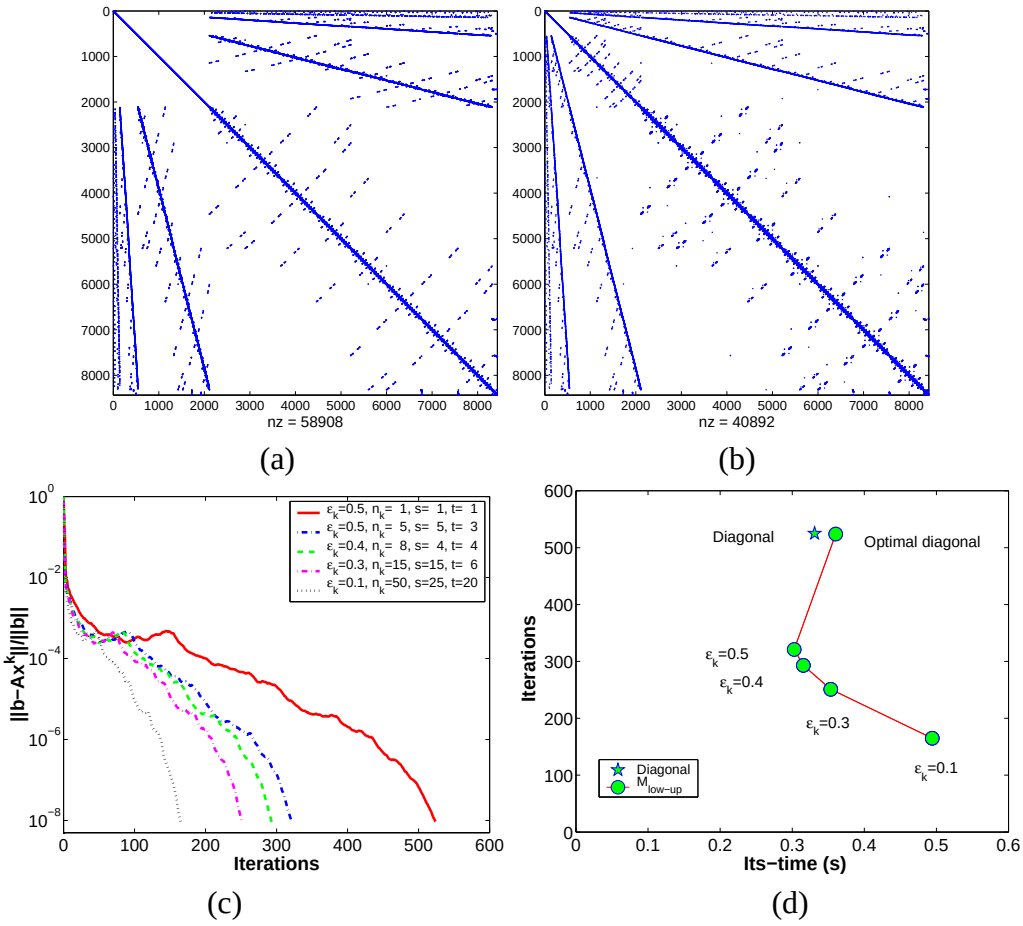


Figura 5.9: Problema de elasticidad. Patrones de *sparsidad* de las matrices: (a) $\mathbf{A}$ y (b) $\mathbf{M}_{low-up}$ para $\epsilon = 0.5$, $n_k = 5$, $s = 5$ y $t = 3$. Gráficas de: (a) convergencia y (b) iteraciones vs. its-time de $\mathbf{M}_{low-up}$

$\epsilon = 0.1$ y $t = 20$ se obtiene el mismo orden de iteraciones que $\mathbf{M}_{sym-p}$ requiriendo menor llenado y tiempo de CPU para aplicarlo. Sin embargo $\|\mathbf{A}\mathbf{M}_{low-up} - \mathbf{I}\|_F$ es

mucho mayor que $\|\mathbf{AM}_{sym-p} - \mathbf{I}\|_F$, este problema se explica con detalle en el ejemplo 2.

La figura 5.9 presenta los patrones de *sparsidad*, a la izquierda de la matriz $\mathbf{A}$, y a la derecha de la inversa aproximada utilizando la estrategia 3. Observemos que las posiciones de las entradas no nulas de $\mathbf{M}_{low-up}$ no coinciden con las de $\mathbf{A}$, aunque el primero tiene un menor número de entradas no nulas $(73\,\%)$.

La figura 5.9.c muestra la suavidad de las gráficas de convergencia de $\mathbf{M}_{low-up}$ para diferentes valores de ϵ. Con los parámetros $\epsilon = 0.5$, $n_k = 1$, $s = 1$ y $t = 1$ se obtiene el precondicionador diagonal óptimo definido en (5.6). Es claro de la figura 5.9.d que $\mathbf{M}_{low-up}$ con $\epsilon = 0.5$ o $\epsilon = 0.4$ supera al precondicionador diagonal. Esto significa que si este problema (para $\epsilon = 0.5$) se resolviera más de 314 veces en un procesamiento secuencial sería más eficiente que los otros precondicionadores, puesto que $n(0.303) + 43.173 \leq n(0.350) + 28.451$.

Ejemplo 2: calor libre en un dominio cuadrado. Este es un problema de trasmisión de calor sobre un dominio cuadrado 1, definido por la ecuación $-\lambda(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) = f$, con condiciones de contorno Dirichlet: $u = u_1$ en $y = 0$ y $x = 0$, $u = u_2$ en $x = 1$; y condición mixta en $y = 1$: $-\lambda\frac{\partial u}{\partial x} = H(u - u_\infty)$, donde λ es la conductividad térmica dada en $[\frac{Kcal}{hm^oC}]$, f representa las fuentes volumétricas externas en $[\frac{Kcal}{hm^3}]$, y H es el coeficiente de convección en $[\frac{Kcal}{hm^{2o}C}]$.

En nuestro experimento numérico se han considerado los siguientes valores: $\lambda = 10^{-5}$, $f = 1$, $H = 20$, y como condiciones de temperatura $u_1 = 0.1$, $u_2 = 100$ y $u_\infty = 1000$. En particular, se trabaja con un sistema simétrico de $16\,412$ ecuaciones con $\mathrm{nnz}(\mathbf{A}) = 48\,783$ entradas no nulas.

La tabla 5.2 muestra los resultados numéricos utilizando las primeras tres estrategias. Notemos que los valores de las normas $\|\mathbf{AM} - \mathbf{I}\|_F$ y $\|\mathbf{AM}_{sym-p} - \mathbf{I}\|_F$ son prácticamente iguales. De hecho, hemos analizado el precondicionador $\mathbf{M}$ y hemos observado que es casi simétrico, esto por supuesto, rara vez ocurre. Por otro lado, se verifica nuevamente que el parámetro t aumenta la calidad de $\mathbf{M}_{low-up}$ con respecto a $\mathbf{M}_{low}$. Sin embargo, un inconveniente de dicho parámetro es que puede fijar entradas en la triangular superior que al principio del proceso eran significativas pero después, conforme se va construyendo la inversa, dejan de serlo. Eso provoca

Estrategia 1: $\mathbf{M}_{sym-p}$										
ϵ	n_k	s	t	$\frac{\text{nnz}(\bullet)}{\text{nnz}(\mathbf{A})}$	Its.	P-time	Its-time	T-time	$\|\mathbf{AM}-\mathbf{I}\|_F$	$\|\mathbf{A}\bullet-\mathbf{I}\|_F$
0.5	1	1		0.33	240	9.488	0.271	9.759	56.9	56.9
0.4	6	3		0.83	182	78.350	0.277	78.627	49.8	47.8
0.3	10	5		1.00	150	111.911	0.250	112.161	36.5	36.5
0.2	20	5		3.55	97	737.799	0.307	738.106	24.6	24.7
0.1	40	15		7.30	69	2058.165	0.365	2058.530	17.2	18.1
Estrategia 2: $\mathbf{M}_{low}$										
0.5	1	1		0.33	240	39.723	0.272	39.995		56.9
0.4	6	3		0.75	184	91.152	0.277	91.429		50.0
0.3	10	5		1.02	158	113.945	0.268	114.213		48.4
0.2	20	5		3.31	100	428.360	0.304	428.664		26.5
0.1	40	15		6.74	89	1114.592	0.441	1115.033		26.5
Estrategia 3: $\mathbf{M}_{low-up}$										
0.5	1	1	1	0.33	240	34.446	0.273	34.719		56.9
0.4	6	3	4	1.09	137	117.753	0.237	117.990		49.7
0.3	10	5	6	1.83	111	188.572	0.239	188.811		54.8
0.2	20	5	10	3.53	97	407.968	0.305	408.273		68.1
0.1	40	15	13	6.85	84	1091.307	0.419	1091.726		85.2

Tabla 5.2: Resultados numéricos de las tres primeras estrategias para el problema de calor libre.

que $\|\mathbf{AM}_{low-up} - \mathbf{I}\|_F$ crezca al disminuir ϵ, y por tanto cause inestabilidad en el precondicionador como se aprecia en las tablas 5.1 y 5.2. Observemos que conforme ϵ disminuye, los valores de las normas $\|\mathbf{AM}_{sym-p} - \mathbf{I}\|_F$ y $\|\mathbf{AM}_{low} - \mathbf{I}\|_F$ también decrecen.

Los patrones de *sparsidad* de las matrices $\mathbf{A}$ y $\mathbf{M}_{low-up}$ se presentan en las figuras 5.10.a y 5.10.b, respectivamente. En este caso el número de condición $\kappa(\mathbf{A})$ es 1.89×10^7, lo que provoca que el error residual disminuya lentamente, ver figura 5.10.c. Por último, se necesitaría resolver $3\,177$ veces el problema de calor libre para que $\mathbf{M}_{low-up}$ con $\epsilon = 0.4$ fuera más eficiente que los otros precondicionadores usando un solo procesador, ver figura 5.10.d. Con este sistema pequeño se percibe la necesidad de paralelizar el precondicionador.

Ejemplo 3: simulación del campo de viento en la isla de Gran Canaria. En este problema se realiza la simulación numérica del campo de viento en la isla de Gran Canaria (Islas Canarias), desarrollado por el grupo formado por las divisiones

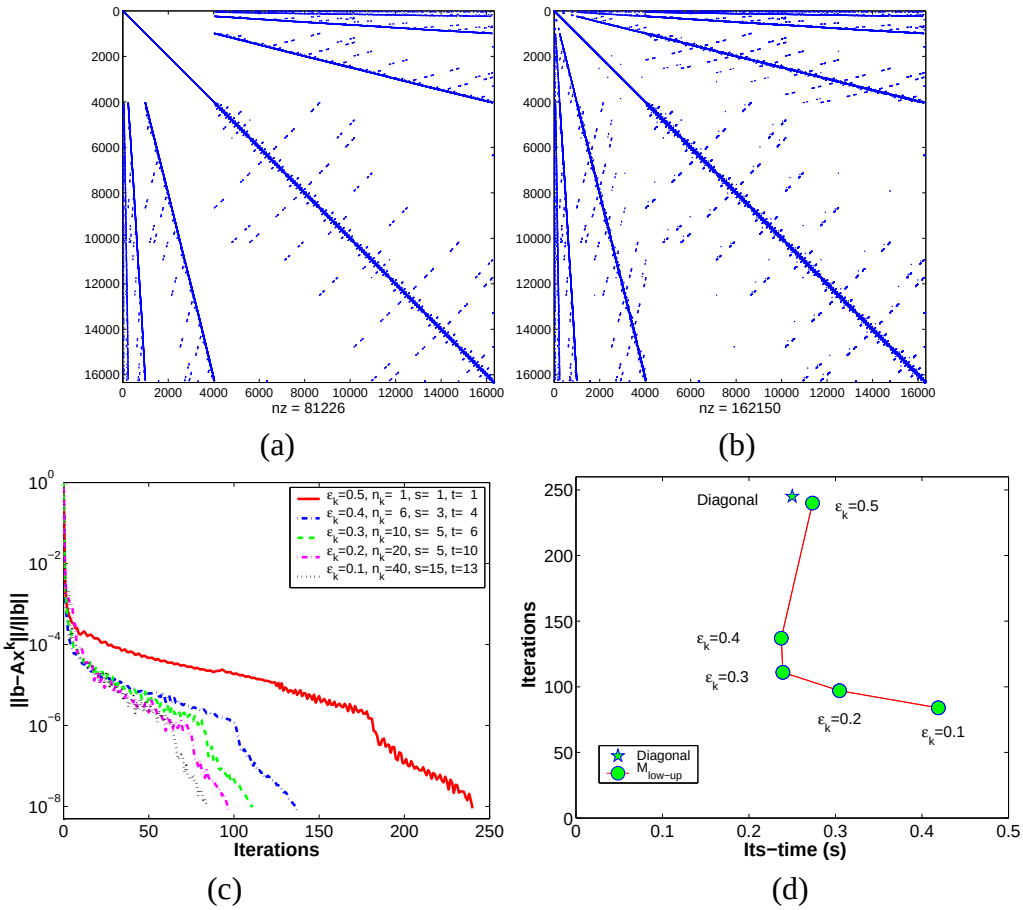


Figura 5.10: Problema de calor libre. Patrones de *sparsidad* de las matrices: (a) $\mathbf{A}$ y (b) $\mathbf{M}_{low-up}$ para $\epsilon = 0.3$, $n_k = 10$, $s = 5$ y $t = 6$. Gráficas de: (c) convergencia y (d) iteraciones vs. its-time de $\mathbf{M}_{low-up}$.

GANA y DDA del Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería de la Universidad de Las Palmas de Gran Canaria.

El problema se plantea sobre un dominio de 16.5 km $\times 9.5$ km $\times 7$ km situado en la zona noroeste de la isla de Gran Canaria. Se dispone de una digitalización del terreno con una resolución de 25 m $\times 25$ m y un error máximo de 5 m en la cota. El fenómeno está determinado por las leyes físicas definidas para un fluido incompresible ($\boldsymbol{\nabla} \cdot \mathbf{u} = 0$). Se impone la condición de no permeabilidad ($\mathbf{n} \cdot \mathbf{u} = 0$) sobre el terreno y la frontera superior del dominio; además se asume que la densidad del aire es constante en todo el dominio. De esta forma, se emplea un modelo de masa consistente para diagnosticar los campos de velocidades de viento

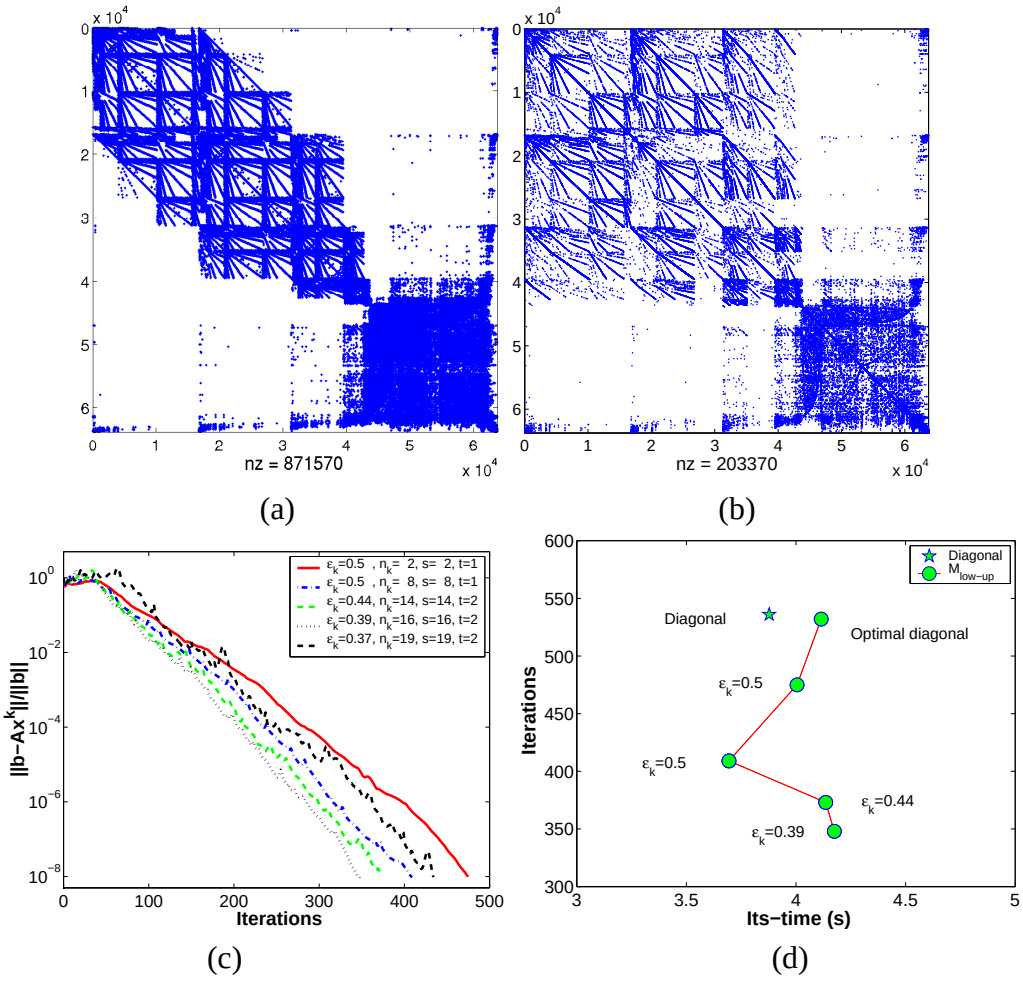


Figura 5.11: Problema de viento. Patrones de *sparsidad* de las matrices: (a) $\mathbf{A}$ y (b) $\mathbf{M}_{low-up}$ para $\epsilon = 0.5$, $n_k = 8$, $s = 8$ y $t = 1$. Gráficas de: (a) convergencia y (b) iteraciones vs. its-time de $\mathbf{M}_{low-up}$

a partir de un número determinado de datos experimentales obtenidos de los mapas de previsión que el Instituto Nacional de Meteorología proporciona (ver Rodríguez 2004, Rodríguez, Montero, Montenegro, Escobar y González-Yuste 2002, Montero et al. 2005).

Después de realizar varios suavizados de una malla 3D (ver Escobar et al. 2003) se obtiene un sistema simétrico de $63\,746$ ecuaciones con $\mathrm{nnz}(\mathbf{A}) = 467\,658$ entradas no nulas. Los valores propios mínimo y máximo de $\mathbf{A}$ son $\lambda_{min} = 2.04 \times 10^{-3}$ y $\lambda_{max} = 3.07 \times 10^{2}$, respectivamente. Así, el número de condición $\kappa(\mathbf{A})$ es 1.51×10^{5}. La figura 5.11 muestra los patrones de *sparsidad* de las matrices $\mathbf{A}$

y $\mathbf{M}_{low-up}$. La gráfica 5.11.b muestra las posiciones de las entradas no nulas que dan el mejor tiempo de CPU, ver figura 5.11.b.

De los problemas que hasta el momento se han presentado, el ejemplo del campo de viento es el más difícil de estudiar. Esto se debe a la sensibilidad de los parámetros ϵ, n_k, s y t. Basta con analizar la tabla 5.3 que resume los resultados de las tres primeras estrategias y la gráfica de convergencia de $\mathbf{M}_{low-up}$, figura 5.11.c. El menor número de iteraciones que se ha conseguido utilizando la estrategia 3 es de 348; si aumentamos el llenado se incrementa considerablemente el número de iteraciones. Por ello se requiere estabilizar el precondicionador inferior-superior. Para

Estrategia 1: $\mathbf{M}_{sym-p}$										
ϵ	n_k	s	t	$\frac{\text{nnz}(\bullet)}{\text{nnz}(\mathbf{A})}$	Its.	P-time	Its-time	T-time	$\|\mathbf{AM}-\mathbf{I}\|_F$	$\|\mathbf{A}\bullet-\mathbf{I}\|_F$
0.5	2	2		0.20	476	3306.897	4.043	3310.940	118	127
0.5	8	8		0.23	448	3184.443	3.874	3188.317	113	126
0.44	14	14		0.33	411	6753.555	3.856	6757.411	100	112
0.39	16	16		0.43	380	8072.822	3.805	8076.627	93.5	105
Estrategia 2: $\mathbf{M}_{low}$										
0.5	2	2		0.18	485	2195.795	4.015	2199.810		120
0.5	8	8		0.23	426	3303.595	3.682	3307.277		111
0.44	14	14		0.35	463	5517.662	4.368	5522.030		100
0.39	16	16		0.46	592	7152.059	5.997	7158.056		94
Estrategia 3: $\mathbf{M}_{low-up}$										
0.5	2	2	1	0.20	475	6894.014	4.006	6894.020		124
0.5	8	8	1	0.29	409	9927.658	3.694	9931.352		111
0.44	14	14	2	0.65	373	15115.63	4.136	15119.77		116
0.39	16	16	2	0.82	348	16762.97	4.175	16767.15		114

Tabla 5.3: Problema de viento. Resultados de las tres primeras estrategias.

este ejemplo funciona mejor la estrategia 1, debido a que es más barato construir y aplicar $\mathbf{M}_{sym-p}$.

5.4.2. Ejemplos de convección-difusión transitorios

En este subapartado se evalúa la eficiencia numérica de las 4 estrategias propuestas, poniendo énfasis en la SSPAI, a través de los problemas de convección-difusión transitorios.