Universidad de Las Palmas de Gran Canaria
Departamento de Matemáticas

Tesis Doctoral

Estrategias para la resolución de grandes sistemas de ecuaciones lineales. Métodos de Cuasi-Mínimo Residuo Modificados.

M. Dolores García León

Las Palmas de Gran Canaria, Marzo de 2003

Universidad de Las Palmas de Gran Canaria
Departamento de Matemáticas

Tesis Doctoral

Estrategias para la resolución de grandes sistemas de ecuaciones lineales. Métodos de Cuasi-Mínimo Residuo Modificados.

Autora: M. Dolores García León
Directores: Gustavo Montero García
Antonio Suárez Sarmiento

La Doctorando | El Director | El Director

Fdo.: M. Dolores García León | Fdo.: Gustavo Montero García | Fdo.: Antonio Suárez Sarmiento

Las Palmas de Gran Canaria, Marzo de 2003

A mi
familia

Agradecimientos

A Gustavo Montero García y Antonio Suárez Sarmiento, directores de la tesis, sin cuyo esfuerzo, asesoramiento y tutela no hubiera sido posible la realización de este trabajo.

A Eduardo Rodríguez Barrera por su ayuda para resolver los problemas informáticos que surgieron durante el desarrollo de esta tesis.

A todos los compañeros del Departamento de Matemáticas que de alguna forma han prestado su colaboración.

Al Colegio Oficial de Ingenieros Industriales de Canarias por la ayuda económica prestada para la realización de esta tesis.

Esta tesis ha sido desarrollada en el marco del proyecto subvencionado por el Ministerio de Ciencia y Tecnología, REN2001-0925-C03-02/CLI, titulado *Modelización numérica de transporte de contaminantes en la atmósfera*.

Resumen

Las aplicaciones de métodos como diferencias finitas, elementos finitos, elementos de contorno, volúmenes finitos, etc., para la obtención de soluciones aproximadas de problemas de contorno en derivadas parciales, desembocan en la resolución de grandes sistemas de ecuaciones lineales de matriz tipo *sparse*.

Para resolver estos sistemas, además de los métodos directos basados generalmente en la factorización de la matriz del sistema utilizando la eliminación gaussiana y de los métodos iterativos clásicos (Jacobi, Gauss-Seidel, Relajación,...), se han desarrollado en los últimos años otros métodos, basados en los subespacios de Krylov, que presentan algunas ventajas respecto a los anteriores.

El objeto de esta tesis es el estudio de estos métodos de Krylov, principalmente de su aplicación a la resolución de sistemas no simétricos, así como el de algunas técnicas de precondicionmiento, almacenamiento y reordenación de los sistemas que los hacen más efectivos.

El presente trabajo se estructura en dos partes. En una primera parte se presenta un estado del arte de los métodos basados en los subespacios de Krylov y de las técnicas anteriormente mencionadas. La segunda parte, pretende hacer una nueva aportación a algunos de estos métodos, concretamente a los métodos de cuasi-mínimo residuo, introduciendo una variante en su desarrollo que consiste en resolver el problema de mínimos cuadrados, correspondiente a la cuasi-minimización, utilizando un método directo. Por último, se presentan una serie de experimentos numéricos para contrastar la eficacia de los distintos algoritmos estudiados, utilizando diferentes formas de precondicionamiento y reordenación en cada caso y exponiendo las conclusiones extraídas de estos y las posibles líneas de trabajo futuras.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Antecedentes

El carácter numérico de las matemáticas data de la antiguedad en la que esta ciencia debía hacer frente a problemas prácticos con la finalidad de obtener una solución en forma de números. La necesidad de nuevos planteamientos para métodos númericos ha supuesto que los modelos matemáticos para representar los fenómenos físicos se hayan ido perfeccionando de tal forma, que resulta casi siempre más apropiado buscar una solución aproximada de un modelo matemático complicado que una solución más exacta del problema simplificado.

La solución numérica de un problema con formulación en derivadas parciales pasa por un proceso de discretización (diferencias finitas, elementos finitos, etc.), que permita valorar la solución en un número finito de puntos del dominio. Esta discretización, conduce a la resolución de un sistema lineal de ecuaciones cuyas incognitas son precisamente estos valores numéricos puntuales de la solución aproximada al problema físico que ha generado dicho sistema.

La resolución de estos sistemas de ecuaciones, fundamental en el proceso de encontrar solución a un problema, con las técnicas y métodos apropiados, es el principal objeto de esta tesis.

La mayoría de las veces, el proceso de discretización da lugar a grandes sistemas de ecuaciones lineales de matriz tipo *sparse*. Los errores de redondeo y el efecto de relleno que se produce en la aplicación de los métodos directos de factorización de la matriz del sistema, hacen más adecuados los métodos iterativos [4], teniendo especial relevancia entre estos últimos los métodos basados en los subespacios de Krylov, de más reciente desarrollo.

Los métodos de Krylov [17, 35, 38, 84], utilizados para la resolución de grandes sistemas lineales se obtienen para adaptarse, en principio, a dos requerimientos básicos, esto es, minimizar una cierta norma del vector residuo sobre un subespacio de Krylov generado por la matriz del sistema y que se traduce en una convergencia suave sin grandes fluctuaciones y ofrecer un bajo coste

computacional por iteración y no exigir alta disponibilidad de almacenaje.

Para los sistemas de ecuaciones lineales cuya matriz de coeficientes es simétrica y definida positiva, el algoritmo del Gradiente Conjugado propuesto por Hestenes y Stiefel en 1952 [39], y desarrollado en la práctica a partir de 1970, cumple los requisitos anteriores de minimalidad y optimalidad. Sin embargo, para sistema no simétricos, no existen métodos que cumplan estos dos requisitos simultáneamente sin añadir inconvenientes y/o desventajas, por lo que los métodos desarrollados (ver por ejemplo [86]) para estos casos, se obtienen para adaptarse a uno de ellos, bien a la minimización, o bien métodos que ofrezcan un bajo coste computacional y de almacenamiento.

Por otro lado, la convergencia de los métodos basados en los subespacios de Krylov mejora con el uso de las técnicas de precondicionamiento y reordenación de los sistemas. Además generalmente las matrices de los sistemas lineales que se resuelven son *sparse*, por lo que es fundamental elegir un buen esquema de almacenamiento que debe básicamente reducir el coste computacional y el requerimiento de memoria en el ordenador [27].

1.2. Objetivos

Con los antecedentes anteriormente expuestos, los objetivos específicos de esta tesis, con el fin de proporcionar herramientas adecuadas para obtener soluciones aproximadas de sistemas de ecuaciones lineales, son:

- Dar una visión general de los distintos métodos basados en los subespacios de Krylov.

- Comprobar el efecto del precondicionamiento de los sistemas en la convergencia de los algoritmos.

- Estudiar el efecto que producen algunas técnicas de renumeración en la utilización de algunos precondicionadores.

- Implementar una variante en los métodos de cuasi-mínimo residuo y estudiar su comportamiento con respecto a la implementación clásica de éstos.

- Realizar un estudio comparativo de estos métodos entre sí, en algunos casos prácticos, en general derivados de problemas de convección-difusión.

1.3. Metodología

El presente trabajo se estructura en dos grandes bloques. Un primer bloque que abarca desde el capítulo 1 hasta el capítulo 7, en el que se presenta un estado del arte de los métodos y de las técnicas anteriormente mencionadas. El segundo bloque que consta de los capítulos 8 y 9, pretende hacer una nueva aportación a algunos de estos métodos, concretamente a los métodos de cuasi-mínimo residuo, introduciendo una variante en su desarrollo. Por último

se presentan algunos experimentos numéricos de aplicación de todos los métodos analizados. Las conclusiones obtenidas y las futuras líneas de investigación concluyen este trabajo.

En el capítulo 1 y a modo de introducción se presentan los antecedentes y los objetivos propuestos en el desarrollo de esta tesis.

En el capítulo 2 se definen los subespacios de Krylov y se hace un breve resumen de los métodos de proyección, dado que los métodos de Krylov están basados en un proceso de proyección sobre un subespacio de Krylov. Asímismo se describen algunos algoritmos de utilidad para el siguiente capítulo.

El tercer capítulo está dedicado al estudio de las tres grandes familias de métodos de Krylov (ver por ejemplo [52]): métodos de ortogonalización, métodos de biortogonalización, y por último, los métodos basados en la ecuación normal.

Dentro de los métodos de ortogonalización, que utilizan para su implementación el algoritmo de ortogonalización de Arnoldi [1], se estudian los métodos FOM (*Full Orthogonalization Method*) y GMRES (*Generalized Minimum Residual Method*) [70]. (Distintas versiones de este algoritmo pueden verse en [33, 42, 43, 62, 69, 87, 89]).

Para el caso particular de sistemas simétricos se estudia el método del Gradiente Conjugado, basado en el método de ortogonalización de Lanczos [44], que es una simplificación del algoritmo de Arnoldi para sistemas simétricos.

De entre los métodos de biortogonalización, basados en el algoritmo de biortogonalización de Lanczos se estudian los métodos Bi-CG (*Biconjugate Gradient*) [17] y sus variante, CGS (*Conjugate Grandient Squared*) [76] y Bi-CGSTAB (*Biconjugate Gradient Stabilized*) [85], y los métodos de cuasi-minimización, QMR (*Quasi-Minimal Residual*) [21] y sus variantes, TFQMR (*Transpose-Free QMR*) [20] y QMRCGSTAB [6]. Por último, dentro de los basados en la Ecuación Normal, los métodos CGN (*Gradiente Conjugado para la Ecuación Normal*) [39] y LSQR (*Least-square QR*) [63].

El capítulo 4 esta dedicado a las técnicas de precondicionamiento, que mejoran sensiblemente la convergencia de los métodos anteriormente mencionados. Además de presentar las distintas formas de precondicionamiento, por la izquierda, por la derecha y por ambos lados, se estudian algunos tipos de precondicionadores, el de Jacobi, el SSOR, el ILUT, y como caso particular de éste el ILU(0), y, por último, el de la Aproximada Inversa de estructura Diagonal, denominado en este trabajo como precondicionador Diagonal Óptimo. Otros tipos de precondicionadores han sido propuestos en los últimos años (ver por ejemplo [7, 53, 77, 78, 83, 95]).

El capítulo 5, está dedicado a los esquemas de almacenamiento [67, 68, 72], que están orientados a reducir el coste computacional y el requerimiento de memoria en el ordenador, ya que las matrices de los sistemas que se pretenden resolver son de tipo *sparse*.

El estudio de distintas técnicas de reordenación son presentadas en el capítulo 6. Estas técnicas, que se aplicaban fundamentalmente en la resolución

de sistemas por métodos directos y que están basadas en la teoría de grafos, proporcionan matrices con ancho de banda o perfil menor, lo que reduce, en ciertos casos, el coste de almacenamiento y son utilizadas en los algoritmos precondicionados con ILU o SSOR (distintas formas de reordenación para diversos algoritmos precondicionados pueden verse en [13, 18, 45, 46, 49, 57])

En el capítulo 7 se presentan las versiones precondicionadas de los algoritmos estudiados en el capítulo 3. En el caso particular del algoritmo GMRES, el precondicionamiento da lugar a nuevas versiones de este, como es el caso del FGMRES (Flexible GMRES) [71] y una variante de éste, el VGMRES [24, 52].

En el segundo bloque, el capitulo 8 introduce una variante en los métodos de cuasi-minimo residuo, QMR, TFQMR y QMRCGSTAB, proponiendo, de forma análoga a como se hace en la versión del GMRES propuesta por Galan [25, 26], la resolución directa del problema de mínimos cuadrados que plantean estos métodos, esto es, haciendo una factorización LU en lugar de la factorización QR tradicional.

En el capítulo 9 se estudian las diferentes formas de precondicionamiento (ver por ejemplo [34, 51, 79, 82]) para los métodos de cuasi-mínimo residuo modificados.

Por último, en el capítulo 10 se realizan diversos experimentos numéricos en los que se aplican a distintos problemas algunos de los algoritmos anteriores y con los que se pretende comparar el comportamiento de unos respecto de otros y estudiar la eficacia de los distintos precondicionadores y de las técnicas de reordenación. Existen muchos trabajos publicados que presentan estudios comparativos de métodos iterativos para la resolución de grandes sistemas de ecuaciones lineales aplicados a la resolución de distintos problemas de ingeniería, ver por ejemplo en Elasticidad [10, 88], Electromagnetismo [14], Análisis estructural [9, 73], Mecánica de fluidos [11], Campos de viento [55, 91, 92], y problemas de Convección-Difusión [50, 66, 80, 94], entre otros.

En esta tesis se han realizado distintos experimentos tanto para sistemas simétricos como no simétricos y para distintos casos prácticos, que se han extraído entre otros de problemas resueltos con *Código NEPTUNO*, desarrollado por Ferragut y otros [16], de la *Harwell-Boeing Sparse Matrix Collection* [12] y de la simulación de un filtro de carbón activo desarrollado por el grupo LaCan de la Universitat Politécnica de Catalunya [15]. Se ha prestado especial interés a los problemas de Convección-Difusión por ser una de las principales líneas de trabajo que abarca el proyecto de investigación del que forma parte esta tesis.

Asimismo, en este capítulo se compara el comportamiento de la variante desarrollada de los métodos de cuasimínimo residuo con el de los métodos implementados tradicionalmente, para algunos casos prácticos.

Por último en el capítulo 11 se presentan las conclusiones y futuras líneas de investigación.

Capítulo 2

Preliminares

Los métodos basados en los subespacios de Krylov, se han desarrollado para la resolución de grandes sistemas de ecuaciones lineales

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.1}$$

donde la matriz $\mathbf{A}$ es no singular y de tipo *sparse*. Están basados en un proceso de proyección sobre un subespacio de Krylov que es generado por vectores de la forma $p(\mathbf{A})\mathbf{v}$. Estas técnicas aproximan $\mathbf{A}^{-1}\mathbf{b}$ por $p(\mathbf{A})\mathbf{b}$ donde $p(\mathbf{A})$ es un polinomio matricial elegido adecuadamente.

2.1. Subespacios de Krylov

En los métodos iterativos, las sucesivas aproximaciones del sistema (2.1), vienen dadas por una relación de recurrencia de la forma,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{B}^{-1}\left(\mathbf{b} - \mathbf{A}\mathbf{x}_k\right) \tag{2.2}$$

o bien,

$$\mathbf{B}\mathbf{x}_{k+1} = \mathbf{C}\mathbf{x}_k + \mathbf{b}, \text{ siendo } \mathbf{C} = \mathbf{B} - \mathbf{A} \tag{2.3}$$

La elección de las matrices $\mathbf{B}$ y $\mathbf{C}$ en función de la matriz $\mathbf{A}$ permiten obtener los métodos clásicos de Jacobi, Gauss-Seidel o de Relajación.

Estas expresiones, (2.2) y (2.3), también se pueden escribir en función del vector residuo $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ como,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{B}^{-1}\mathbf{r}_k \tag{2.4}$$

De esta forma, dada una aproximación inicial $\mathbf{x}_0$, las sucesivas iteraciones se podrían expresar,

$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{B}^{-1}\mathbf{r}_0$

$$\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{B}^{-1}\mathbf{r}_1 = \mathbf{x}_0 + \mathbf{B}^{-1}\mathbf{r}_0 + \mathbf{B}^{-1}\left(\mathbf{b} - \mathbf{A}\mathbf{x}_1\right) =$$
$$= \mathbf{x}_0 + \mathbf{B}^{-1}\mathbf{r}_0 + \mathbf{B}^{-1}\left(\mathbf{b} - \mathbf{A}\mathbf{x}_0 - \mathbf{A}\mathbf{B}^{-1}\mathbf{r}_0\right) =$$
$$= \mathbf{x}_0 + 2\mathbf{B}^{-1}\mathbf{r}_0 - \mathbf{B}^{-1}\mathbf{A}\left(\mathbf{B}^{-1}\mathbf{r}_0\right)$$

$$\mathbf{x}_3 = \mathbf{x}_2 + \mathbf{B}^{-1}\mathbf{r}_2$$

$$\mathbf{x}_4 = \mathbf{x}_3 + \mathbf{B}^{-1}\mathbf{r}_3$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{B}^{-1}\mathbf{r}_{k-1}$$

En el caso en que $\mathbf{B} = \mathbf{I}$ quedaría,

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{r}_0$$

$$\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{r}_1 = \mathbf{x}_0 + \mathbf{r}_0 + (\mathbf{b} - \mathbf{A}\mathbf{x}_1) = \mathbf{x}_0 + \mathbf{r}_0 + (\mathbf{b} - \mathbf{A}\mathbf{x}_0 - \mathbf{A}\mathbf{r}_0) =$$
$$= \mathbf{x}_0 + 2\mathbf{r}_0 - \mathbf{A}\mathbf{r}_0$$

$$\mathbf{x}_3 = \mathbf{x}_2 + \mathbf{r}_2 = \mathbf{x}_0 + 2\mathbf{r}_0 - \mathbf{A}\mathbf{r}_0 + (\mathbf{b} - \mathbf{A}\mathbf{x}_2) =$$
$$= \mathbf{x}_0 + 2\mathbf{r}_0 - \mathbf{A}\mathbf{r}_0 + (\mathbf{b} - \mathbf{A}\mathbf{x}_0 + 2\mathbf{A}\mathbf{r}_0 - \mathbf{A}^2\mathbf{r}_0) = \mathbf{x}_0 + 2\mathbf{r}_0 + \mathbf{A}\mathbf{r}_0 - \mathbf{A}^2\mathbf{r}_0$$

$$\mathbf{x}_4 = \mathbf{x}_3 + \mathbf{r}_3$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{r}_{k-1}$$

Es decir que la k-ésima iteración de la solución aproximada se puede expresar como suma de la aproximación inicial y una combinación lineal de k vectores,

$$\mathbf{x}_k = \mathbf{x}_0 + C.L.\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, ..., \mathbf{A}^{k-1}\mathbf{r}_0\right\}$$

Por tanto,

$$\mathbf{x}_k = \mathbf{x}_0 + \left[\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, ..., \mathbf{A}^{k-1}\mathbf{r}_0\right] \tag{2.5}$$

El subespacio $\mathcal{K}_k\left(\mathbf{A}; \mathbf{r}_0\right)$ de base $\left[\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, ..., \mathbf{A}^{k-1}\mathbf{r}_0\right]$, es llamado subespacio de Krylov de dimensión k correspondiente a la matriz $\mathbf{A}$ y al residuo inicial $\mathbf{r}_0$.

2.2. Métodos de proyección

Dado el sistema (2.1), tal que la matriz $\mathbf{A}$ es real y no singular, sean $\mathcal{L}$ y $\mathcal{K}$ dos subespacios de $\Re^n$ de dimensión k. Una técnica de proyección sobre el subespacio $\mathcal{K}$ y ortogonal a $\mathcal{L}$ es un proceso para encontrar una solución $\widetilde{\mathbf{x}}$

aproximada del sistema anterior, imponiendo la condición de que $\widetilde{\mathbf{x}}$ pertenezca a $\mathcal{K}$ y que el nuevo vector residuo sea ortogonal a $\mathcal{L}$. Esto es,

$$\text{Encontrar } \widetilde{\mathbf{x}} \in \mathcal{K}, \text{ tal que } \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}} \perp \mathcal{L}$$

Redefinimos pues, nuestro problema. Sea $\mathbf{x}_0$ una aproximación inicial, entonces $\widetilde{\mathbf{x}}$ debe estar en el espacio afín $\mathbf{x}_0 + \mathcal{K}$,

$$\text{Encontrar } \widetilde{\mathbf{x}} \in \mathbf{x}_0 + \mathcal{K}, \text{ tal que } \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}} \perp \mathcal{L}$$

Si expresamos la solución aproximada por $\widetilde{\mathbf{x}} = \mathbf{x}_0 + \delta$ y el vector residuo incial es $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ entonces tendrá que cumplirse,

$$\mathbf{b} - \mathbf{A}\left(\mathbf{x}_0 + \delta\right) \perp \mathcal{L} \Rightarrow \mathbf{r}_0 - \mathbf{A}\delta \perp \mathcal{L}$$

Es decir que la solución aproximada $\widetilde{\mathbf{x}}$ del sistema $\mathbf{A}\mathbf{x} = \mathbf{b}$, queda definida como,

$$\widetilde{\mathbf{x}} = \mathbf{x}_0 + \delta, \delta \in \mathcal{K} / \ \langle \mathbf{r}_0 - \mathbf{A}\delta, \mathbf{w} \rangle = 0, \forall \mathbf{w} \in \mathcal{L} \tag{2.6}$$

La representación matricial de la condición de ortogonalidad del residuo, expresada en la ecuación (2.6) la haremos de la siguiente forma:

Sea $\mathbf{V} = [\mathbf{v}_{1,}\mathbf{v}_2, ..., \mathbf{v}_k]$ la matriz $n \times k$ cuyos vectores columnas forman una base del subespacio $\mathcal{K}$, y sea $\mathbf{W} = [\mathbf{w}_{1,}\mathbf{w}_2, ..., \mathbf{w}_k]$ la matriz $n \times k$ cuyos vectores columnas forman una base del subespacio $\mathcal{L}$. Si la solución aproximada del sistema $\mathbf{A}\mathbf{x} = \mathbf{b}$ la escribimos como,

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{V}\mathbf{y}$$

por (2.6),

$$\langle \mathbf{r}_0 - \mathbf{A}\mathbf{V}\mathbf{y}, \mathbf{w} \rangle = 0, \forall \mathbf{w} \in \mathcal{L}$$

o lo que es lo mismo,

$$\mathbf{W}^T\mathbf{A}\mathbf{V}\mathbf{y} = \mathbf{W}^T\mathbf{r}_0$$

entonces, si la matriz $n \times n$ $\mathbf{W}^T\mathbf{A}\mathbf{V}$ es no singular, la solución aproximada vendrá dada por,

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{V}\left(\mathbf{W}^T\mathbf{A}\mathbf{V}\right)^{-1}\mathbf{W}^T\mathbf{r}_0 \tag{2.7}$$

donde, si $\mathbf{A}$ es simétrica definida positiva y $\mathcal{L} = \mathcal{K}$, o bien si $\mathbf{A}$ es no singular y $\mathcal{L} = \mathbf{A}\mathcal{K}$, entonces la matriz $\mathbf{W}^T\mathbf{A}\mathbf{V}$ es no singular para las bases $\mathbf{V}$ y $\mathbf{W}$ de $\mathcal{K}$ y $\mathcal{L}$ respectivamente (ver por ejemplo [72]).

En el primer caso, $\mathbf{A}$ simétrica definida positiva y $\mathcal{L} = \mathcal{K}$, la solución aproximada $\widetilde{\mathbf{x}}$ es el resultado de un método de proyección ortogonal sobre $\mathcal{K}$, si y sólo si minimiza la norma euclídea del error sobre $\mathbf{x}_0 + \mathcal{K}$,

$$E\left(\widetilde{\mathbf{x}}\right) = \min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}} E\left(\mathbf{x}\right)$$

siendo,

$$E(\mathbf{x}) \equiv [\mathbf{A}(\mathbf{x}_* - \mathbf{x}), \mathbf{x}_* - \mathbf{x}]^{\frac{1}{2}}$$

donde $\mathbf{x}_*$ representa la solución exacta del sistema (2.1).

La condición necesaria y suficiente para que el resultado de minimizar $E(\mathbf{x})$ sea $\widetilde{\mathbf{x}}$ es que $\mathbf{x}_* - \widetilde{\mathbf{x}}$ sea $\mathbf{A}$-ortogonal a todo el subespacio $\mathcal{K}$, es decir que,

$$[\mathbf{A}(\mathbf{x}_* - \widetilde{\mathbf{x}}), \mathbf{v}] = 0, \quad \forall \mathbf{v} \in \mathcal{K}$$

que equivale a,

$$(\mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}, \mathbf{v}) = 0, \quad \forall \mathbf{v} \in \mathcal{K}$$

que es la condición de Galerkin definiendo un proceso de proyección ortogonal para la aproximación $\widetilde{\mathbf{x}}$.

En el seguno caso, si $\mathbf{A}$ es no singular y $\mathcal{L} = \mathbf{A}\mathcal{K}$, la solución aproximada $\widetilde{\mathbf{x}}$ es el resultado de un método de proyección oblicua sobre $\mathcal{K}$, ortogonalmente a $\mathcal{L}$ si y sólo si minimiza la norma euclídea del vector residuo $\mathbf{b} - \mathbf{A}\mathbf{x}$ sobre $\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}$,

$$R(\widetilde{\mathbf{x}}) = \min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}} R(\mathbf{x})$$

siendo,

$$R(\mathbf{x}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|$$

La condición necesaria y suficiente para que el resultado de minimizar $R(\mathbf{x})$ sea $\widetilde{\mathbf{x}}$ es que $\mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}$ sea ortogonal a todos los vectores de la forma $\mathbf{v} = \mathbf{A}\mathbf{y}$, donde $\mathbf{y} \in \mathcal{K}$, es decir que,

$$(\mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}, \mathbf{v}) = 0, \quad \forall \mathbf{v} \in \mathbf{A}\mathcal{K}$$

que es la condición de Petrov-Galerkin que define la solución aproximada $\widetilde{\mathbf{x}}$.

2.3. Método de Arnoldi

Introducido en 1951 [1] se basa en reducir una matriz densa a una matriz de Hessemberg cuyos autovalores, obtenidos en un número de pasos menor que n, proporcionan una aproximación exacta de algunos autovalores de la matriz original, a la vez que genera un sistema de vectores ortonormados.

ALGORITMO DE ARNOLDI
Elegir un vector $\mathbf{v}_1 / \parallel \mathbf{v}_1 \parallel = 1$
Desde $j = 1, ..., k$ hacer
    Desde $i = 1, ..., j$ hacer
        $\{\mathbf{H}\}_{ij} = \langle \mathbf{A}\mathbf{v}_j, \mathbf{v}_i \rangle$;

$\mathbf{w}_j = \mathbf{A}\mathbf{v}_j - \{\mathbf{H}\}_{ij}\,\mathbf{v}_i$;
Fin
$\{\mathbf{H}\}_{j+1,j} = \| \mathbf{w}_j \|$. Si $\{\mathbf{H}\}_{j+1,j} = 0$ parar
$\mathbf{v}_{j+1} = \dfrac{1}{\{\mathbf{H}\}_{j+1,j}}\,\mathbf{w}_j$;
Fin

Los vectores $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k$ obtenidos en el algoritmo de Arnoldi forman una base ortogonal del subespacio de Krylov,

$$\mathcal{K}_k\left(\mathbf{A}; \mathbf{v}_1\right) = C.L.\left\{\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \mathbf{A}^2\mathbf{v}_1, ..., \mathbf{A}^{k-1}\mathbf{v}_1\right\}$$

Estos vectores $\mathbf{v}_j (j = 1, ..., k)$ son ortonormados por construcción. Cada vector del subespacio $\mathcal{K}_k\left(\mathbf{A}; \mathbf{v}_1\right)$ es de la forma $q_{j-1}(\mathbf{A})\mathbf{v}_1$, donde q_{j-1} es un polinomio de grado $(j-1)$. En efecto, se puede demostrar por inducción,

Para $j = 1, \qquad \mathbf{v}_1 = q_0(\mathbf{A})\mathbf{v}_1 / \qquad q_0(\mathbf{A}) \equiv \mathbf{I}$.

Asumiendo que el resultado es válido para todo entero $i \leq j$ y considerando $\mathbf{v}_{j+1}$, entonces,

$$h_{j+1,j}\mathbf{v}_{j+1} = \mathbf{A}\mathbf{v}_j - \sum_{i=1}^{j} h_{ij}\mathbf{v}_i = \mathbf{A}q_{j-1}(\mathbf{A})\mathbf{v}_1 - \sum_{i=1}^{j} h_{ij}q_{i-1}(\mathbf{A})\mathbf{v}_1 \tag{2.8}$$

Por tanto $\mathbf{v}_{j+1}$ puede ser expresado como $q_j(\mathbf{A})\mathbf{v}_1$, donde q_j es de grado j.

Llamando $\mathbf{V}_k$ a la matriz $n \times k$ cuyas columnas son los vectores $\mathbf{v}_j$ con $j = 1, ..., k$ y $\overline{\mathbf{H}}_k$ a la matriz $(k+1) \times k$ de Hessemberg cuyas entradas no nulas h_{ij} son obtenidas en el algoritmo de Arnoldi, y por $\mathbf{H}_k$ la matriz obtenida eliminando la última fila de $\overline{\mathbf{H}}_k$, entonces es inmediato comprobar que:

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{H}_k + \mathbf{w}_k\mathbf{e}_k^T = \mathbf{V}_{k+1}\overline{\mathbf{H}}_k \tag{2.9}$$

$$\mathbf{V}_k^T\mathbf{A}\mathbf{V}_k = \mathbf{H}_k \tag{2.10}$$

En la práctica, el algoritmo básico de Arnoldi se sustituye por el Modificado Gram-Schmidt, que es en aritmética exacta matemáticamente equivalente pero más eficiente. Este algoritmo se describe a continuación:

ALGORITMO DE ARNOLDI MODIFICADO GRAM-SCHMIDT
Elegir un vector $\mathbf{v}_1 / \| \mathbf{v}_1 \| = 1$
Desde $j = 1, ..., k$ hacer
$\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$;
Desde $i = 1, ..., j$ hacer
$\{\mathbf{H}\}_{ij} = \langle \mathbf{w}_j, \mathbf{v}_i \rangle$;
$\mathbf{w}_j = \mathbf{w}_j - \{\mathbf{H}\}_{ij}\,\mathbf{v}_i$;
Fin

$\{\mathbf{H}\}_{j+1,j} = \| \mathbf{w}_j \|$; Si $\{\mathbf{H}\}_{j+1,j} = 0$ parar

$\mathbf{v}_{j+1} = \dfrac{1}{\{\mathbf{H}\}_{j+1,j}} \mathbf{w}_j$;

Fin

2.4. Método de Lanczos

Es una simplificación del algoritmo de Arnoldi [1] para el caso de matrices simétricas ya que los coeficientes h_{ij} generados por el algoritmo son tales que,

$$\begin{cases} h_{ij} = 0 \text{ para } 1 \leq i \leq j-1 \\ h_{j,j+1} = h_{j+1,j}, \, j = 1, 2, ..., k \end{cases}$$

con lo que la matriz $\mathbf{H}_k$ obtenida del proceso de Arnoldi es tridiagonal y simétrica.

La solución estándar del método es,

$$\begin{cases} h_{ij} \equiv \alpha_j \\ h_{j-1,j} \equiv \beta_j \end{cases}$$

y si denotamos por $\mathbf{T}_k$ la matriz resultante del algoritmo de Lanczos,

$$\mathbf{T}_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \cdot & \cdot & \cdot & & \\ & & & \beta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & & \beta_k & \alpha_k \end{pmatrix} \tag{2.11}$$

ALGORITMO DE LANCZOS

Elegir un vector $\mathbf{v}_1 / \| \mathbf{v}_1 \| = 1$, $\beta_1 \equiv 0$, $\mathbf{v}_0 \equiv 0$

Desde $j = 1, ..., k$ hacer

$\mathbf{w}_j = \mathbf{A}\mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$;

$\alpha_j = \langle \mathbf{w}_j, \mathbf{v}_j \rangle$;

$\mathbf{w}_j = \mathbf{w}_j - \alpha_j \mathbf{v}_j$;

$\beta_{j+1} = \| \mathbf{w}_j \|$. Si $\beta_{j+1} = 0$ parar;

$\mathbf{v}_{j+1} = \dfrac{1}{\beta_{j+1}} \mathbf{w}_j$

Fin

Donde los vectores $\mathbf{v}_i$ $(i = 1, 2, ...)$ son ortonormados.

2.5. Método de biortogonalización de Lanczos

Es una extensión del método de Lanczos [44], [47], para el caso de matrices no simétricas. Genera dos sistemas de vectores $\mathbf{v}_1, \ldots, \mathbf{v}_k$ y $\mathbf{w}_1, \ldots, \mathbf{w}_k$, $k = 1, 2, \ldots$, tal que,

$$\mathcal{K}_k\left(\mathbf{A}; \mathbf{v}_1\right) = C.L.\left\{\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \mathbf{A}^2\mathbf{v}_1, ..., \mathbf{A}^{k-1}\mathbf{v}_1\right\}$$

y,

$$\mathcal{K}_k\left(\mathbf{A}^T; \mathbf{w}_1\right) = C.L.\left\{\mathbf{w}_1, \mathbf{A}^T\mathbf{w}_1, \left(\mathbf{A}^T\right)^2\mathbf{w}_1, ..., \left(\mathbf{A}^T\right)^{k-1}\mathbf{w}_1\right\}$$

y que satisfacen la condición de biortogonalidad,

$$\langle \mathbf{v}_i, \mathbf{w}_j \rangle = \begin{cases} 0 \text{ si } j \neq i \\ d_j \neq 0 \text{ si } j = i \end{cases} \tag{2.12}$$

ALGORITMO DE BIORTOGONALIZACIÓN DE LANCZOS
Elegir dos vectores $\mathbf{v}_1, \mathbf{w}_1 / \langle \mathbf{v}_1, \mathbf{w}_1 \rangle = 1$
$\beta_1 \equiv \delta_1 \equiv 0$, $\mathbf{w}_0 \equiv \mathbf{v}_0 \equiv 0$
Desde $j = 1, ..., k$ hacer
$\alpha_j = \langle \mathbf{A}\mathbf{v}_j, \mathbf{w}_j \rangle$;
$\widehat{\mathbf{v}}_{j+1} = \mathbf{A}\mathbf{v}_j - \alpha_j\mathbf{v}_j - \beta_j\mathbf{v}_{j-1}$;
$\widehat{\mathbf{w}}_{j+1} = \mathbf{A}^T\mathbf{w}_j - \alpha_j\mathbf{w}_j - \delta_j\mathbf{w}_{j-1}$;
$\delta_{j+1} = |\langle \widehat{\mathbf{v}}_{j+1}, \widehat{\mathbf{w}}_{j+1} \rangle|^{1/2}$. Si $\delta_{j+1} = 0$ parar;
$\beta_{j+1} = \dfrac{\langle \widehat{\mathbf{v}}_{j+1}, \widehat{\mathbf{w}}_{j+1} \rangle}{\delta_{j+1}}$;
$\mathbf{w}_{j+1} = \dfrac{1}{\beta_{j+1}}\widehat{\mathbf{w}}_{j+1}$
$\mathbf{v}_{j+1} = \dfrac{1}{\delta_{j+1}}\widehat{\mathbf{v}}_{j+1}$
Fin

Los parámetros δ_{j+1} y β_{j+1} son factores de escala para los vectores $\mathbf{v}_{j+1}$ y $\mathbf{w}_{j+1}$, respectivamente, y deben ser elegidos de forma que se asegure que $\langle \mathbf{v}_{j+1}, \mathbf{w}_{j+1} \rangle = 1$, por tanto,

$$\delta_{j+1}\beta_{j+1} = \langle \widehat{\mathbf{v}}_{j+1}, \widehat{\mathbf{w}}_{j+1} \rangle \tag{2.13}$$

También se podría escalar los vectores $\mathbf{v}_{j+1}$ y $\mathbf{w}_{j+1}$ por su norma-2, pero en ese caso el producto $\langle \mathbf{v}_{j+1}, \mathbf{w}_{j+1} \rangle$ sería distinto de 1 y habría que modificar el algoritmo.

Consideramos el caso en que δ_{j+1}, β_{j+1} satisfacen la relación (2.13). Denotando por $\mathbf{T}_k$ la matriz tridiagonal,

$$\mathbf{T}_k = \begin{pmatrix} \alpha_1 & \beta_2 & & . & & & \\ \delta_2 & \alpha_2 & \beta_3 & . & & & \\ & \delta_3 & \alpha_3 & . & & & \\ . & . & . & . & . & . & . \\ & & & . & \alpha_{k-2} & \beta_{k-1} & \\ & & & . & \delta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & . & & \delta_k & \alpha_k \end{pmatrix} \tag{2.14}$$

si se determinan δ_{j+1} y β_{j+1} en el algoritmo, entonces los δ_{j+1} son positivos y $\beta_{j+1} = \pm\delta_{j+1}$.

Como los vectores $\mathbf{v}_i$, $i = 1, ..., k$ pertenecen al subespacio $\mathcal{K}_k(\mathbf{A}; \mathbf{v}_1)$ y los $\mathbf{w}_j, j = 1, ..., k$ al subespacio $\mathcal{K}_k\left(\mathbf{A}^T; \mathbf{w}_1\right)$, podemos demostrar que forman un sistema biortogonal tal que,

$$\langle \mathbf{v}_i, \mathbf{w}_j \rangle = \delta_{ij} \quad 1 \leq i,\ j \leq k$$

Además, $\{\mathbf{v}_i\}_{i=1,...,k}$ es una base de $\mathcal{K}_k(\mathbf{A}; \mathbf{v}_1)$ y $\{\mathbf{w}_j\}_{j=1,...,k}$ es una base de $\mathcal{K}_k\left(\mathbf{A}^T; \mathbf{w}_1\right)$ y se cumplen las siguientes relaciones,

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{T}_k + \delta_{k+1}\mathbf{v}_{k+1}\mathbf{e}_k^T \tag{2.15}$$

$$\mathbf{A}^T\mathbf{W}_k = \mathbf{W}_k\mathbf{T}_k^T + \beta_{k+1}\mathbf{w}_{k+1}\mathbf{e}_k^T \tag{2.16}$$

$$\mathbf{W}_k^T\mathbf{A}\mathbf{V}_k = \mathbf{T}_k \tag{2.17}$$

La demostración de la biortogonalidad de los vectores $\mathbf{v}_i, \mathbf{w}_i$ la podemos hacer por inducción. Si $\langle \mathbf{v}_1, \mathbf{w}_1 \rangle = 1$ y asumiendo que los vectores $\mathbf{v}_1, ..., \mathbf{v}_j$ y $\mathbf{w}_1, ..., \mathbf{w}_j$ son biortogonales, vamos a demostrar que los vectores $\mathbf{v}_1, ..., \mathbf{v}_{j+1}$y $\mathbf{w}_1, ..., \mathbf{w}_{j+1}$ también son biortogonales.

Primero veamos que $\langle \mathbf{v}_{j+1}, \mathbf{w}_i \rangle = 0$ para $i \leq j$. Cuando $i = j$, entonces,

$$\langle \mathbf{v}_{j+1}, \mathbf{w}_j \rangle = \delta_{j+1}^{-1} \left[\langle \mathbf{A}\mathbf{v}_j, \mathbf{w}_j \rangle - \alpha_j \langle \mathbf{v}_j, \mathbf{w}_j \rangle - \beta_j \langle \mathbf{v}_{j-1}, \mathbf{w}_j \rangle \right]$$

Pero por hipótesis de inducción $\langle \mathbf{v}_{j-1}, \mathbf{w}_j \rangle = 0$. Los otros dos términos se cancelan uno con otro por la definición de α_j y porque $\langle \mathbf{v}_j, \mathbf{w}_j \rangle = 1$.

Consideremos ahora el producto $\langle \mathbf{v}_{j+1}, \mathbf{w}_i \rangle$ cono $i < j$,

$$\begin{aligned} \langle \mathbf{v}_{j+1}, \mathbf{w}_i \rangle &= \delta_{j+1}^{-1} \left[\langle \mathbf{A}\mathbf{v}_j, \mathbf{w}_i \rangle - \alpha_j \langle \mathbf{v}_j, \mathbf{w}_i \rangle - \beta_j \langle \mathbf{v}_{j-1}, \mathbf{w}_i \rangle \right] = \\ &= \delta_{j+1}^{-1} \left[\left\langle \mathbf{v}_j, \mathbf{A}^T\mathbf{w}_i \right\rangle - \beta_j \langle \mathbf{v}_{j-1}, \mathbf{w}_i \rangle \right] = \\ &= \delta_{j+1}^{-1} \left[\langle \mathbf{v}_j, \beta_{i+1}\mathbf{w}_{i+1} + \alpha_i\mathbf{w}_i + \delta_i\mathbf{w}_{i-1} \rangle - \beta_j \langle \mathbf{v}_{j-1}, \mathbf{w}_i \rangle \right] \end{aligned}$$

Para $i < j-1$, todos los productos en la anterior expresión se anulan por la hipótesis de inducción. Para $i = j-1$, el producto es,

$$\begin{aligned}\langle \mathbf{v}_{j+1}, \mathbf{w}_{j-1}\rangle &= \delta_{j+1}^{-1}\left[\langle \mathbf{v}_j, \beta_j \mathbf{w}_j + \alpha_{j-1}\mathbf{w}_{j-1} + \delta_{j-1}\mathbf{w}_{j-2}\rangle - \beta_j \langle \mathbf{v}_{j-1}, \mathbf{w}_{j-1}\rangle\right] \\ &= \delta_{j+1}^{-1}\left[\beta_j \langle \mathbf{v}_j, \mathbf{w}_j\rangle - \beta_j \langle \mathbf{v}_{j-1}, \mathbf{w}_{j-1}\rangle\right] = 0\end{aligned}$$

Se puede probar de la misma forma que $\langle \mathbf{v}_i, \mathbf{w}_{j+1}\rangle = 0$ para $i \leq j$. Finalmente, por construcción $\langle \mathbf{v}_{j+1}, \mathbf{w}_{j+1}\rangle = 1$, con lo que se completa la demostración.

Las relaciones anteriores (2.15-2.17) nos permiten interpretar el algoritmo. La matriz $\mathbf{T}_k$ es la proyección de $\mathbf{A}$ obtenida por un proceso de proyección oblícua sobre $\mathcal{K}_k(\mathbf{A}; \mathbf{v}_1)$ y ortogonalmente a $\mathcal{K}_k\left(\mathbf{A}^T; \mathbf{w}_1\right)$. De forma similar $\mathbf{T}_k^T$, representa la proyección de $\mathbf{A}^T$ sobre $\mathcal{K}_k\left(\mathbf{A}^T; \mathbf{w}_1\right)$ y ortogonalmente a $\mathcal{K}_k(\mathbf{A}; \mathbf{v}_1)$.

Observamos que con este método debemos resolver dos sistemas lineales, uno con la matriz $\mathbf{A}$ y otro con la matriz $\mathbf{A}^T$, aunque desde un punto de vista práctico, este algoritmo presenta algunas ventajas sobre el de Arnoldi, porque requiere sólo unos pocos vectores de almacenamiento. Además, se han desarrollado algunas técnicas alternativas para no tener que resolver el sistema con $\mathbf{A}^T$.

Sin embargo, el método presenta en ocasiones bastantes inconvenientes ya que los factores de escala para los vectores de Lanczos hacen que puedan acumularse excesivos errores de redondeo, e incluso llegar a anularse produciendo el *"rompimiento"* del algoritmo. Una modificación de éste, *The Look-ahead Lanczos* [64], propuesto por Parlett-Taylor-Liu, trata de subsanar en lo posible estos problemas.

2.5.1. Variante *Look-Ahead* del método de Lanczos (LAL)

La variante *"look-ahead"* del proceso de biortogonalización de Lanczos, desarrollada por Parlett, Taylor y Liu [64], para generar una base en el subespacio de Krylov inducido por la matriz del sistema, se plantea para evitar el posible "*rompimiento*" que podría suceder en el proceso de Lanczos.

Como en el proceso clásico de biortogonalización de Lanczos, el LAL genera dos sistemas de vectores $\mathbf{v}_1, \ldots, \mathbf{v}_k$ y $\mathbf{w}_1, \ldots, \mathbf{w}_k$, $k = 1, 2, \ldots$, tal que,

$$\mathcal{K}_k(\mathbf{A}; \mathbf{v}_1) = C.L.\left\{\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \mathbf{A}^2\mathbf{v}_1, ..., \mathbf{A}^{k-1}\mathbf{v}_1\right\}$$

y,

$$\mathcal{K}_k\left(\mathbf{A}^T; \mathbf{w}_1\right) = C.L.\left\{\mathbf{w}_1, \mathbf{A}^T\mathbf{w}_1, \left(\mathbf{A}^T\right)^2\mathbf{w}_1, ..., \left(\mathbf{A}^T\right)^{k-1}\mathbf{w}_1\right\}$$

y que satisfacen la condición de biortogonalidad,

$$\langle \mathbf{v}_l, \mathbf{w}_j\rangle = \begin{cases} 0 \text{ si } j \neq l \\ d_j \neq 0 \text{ si } j = l \end{cases} \tag{2.18}$$

Debido a esta condición podría ocurrir que $\mathbf{w}_{k+1}^T \mathbf{v}_{k+1} \approx 0$ con $\mathbf{w}_{k+1} \neq 0$ y $\mathbf{v}_{k+1} \neq 0$ pero muy próximos a cero y ocasionar el "*rompimiento*" del proceso de Lanczos.

Los vectores $\mathbf{v}_1, \ldots, \mathbf{v}_k$ y $\mathbf{w}_1, \ldots, \mathbf{w}_k$ son agrupados en $N = N(k)$ bloques,

$$\left\{ \begin{array}{c} \mathbf{V}_l = \left[\mathbf{v}_{k_l}, \mathbf{v}_{k_l+1}, \ldots \mathbf{v}_{k_{l+1}-1}\right] \\ \mathbf{W}_l = \left[\mathbf{w}_{k_l}, \mathbf{w}_{k_l+1}, \ldots \mathbf{w}_{k_{l+1}-1}\right] \end{array} \right. , \quad l = 1, 2, \ldots N-1 \tag{2.19}$$

$$\left\{ \begin{array}{c} \mathbf{V}_N = \left[\mathbf{v}_{k_N}, \mathbf{v}_{k_N+1}, \ldots \mathbf{v}_k\right] \\ \mathbf{W}_N = \left[\mathbf{w}_{k_N}, \mathbf{w}_{k_N+1}, \ldots \mathbf{w}_k\right] \end{array} \right. \tag{2.20}$$

con,

$$1 = k_1 < k_2 < \ldots < k_l < \ldots < k_N < k_{N+1}$$

Los bloques son construidos tal que, en lugar de (2.18), tenemos,

$$\mathbf{W}_j^T \mathbf{V}_l = \left\{ \begin{array}{c} 0 \text{ si } j \neq l \\ \mathbf{D}_l \text{ si } j = l \end{array} \right. \qquad j, l = 1, 2, \ldots N \tag{2.21}$$

donde,

$$\mathbf{D}_l \text{ es no singular, } l = 1, 2, \ldots N-1 \tag{2.22}$$

y,

$$\mathbf{D}_l \text{ es no singular si } k = k_{N+1} - 1 \tag{2.23}$$

En consecuencia, si denotamos el número de vectores en cada bloque por,

$$h_l = k_{l+1} - k_{l,}\, l = 1, 2, \ldots N-1, \widetilde{h}_N = k - k_N$$

Los primeros vectores $\mathbf{v}_{k_l}$ y $\mathbf{w}_{k_l}$ en cada bloque son llamados "*regulares*" y los restantes vectores son llamados "*internos*". El $N-$ésimo bloque es llamado "*completo*" si $k = k_{N+1} - 1$; en este caso, en el siguiente paso $k+1$, es colocado un nuevo bloque con vectores regulares $\mathbf{v}_{k_{N+1}}$ y $\mathbf{w}_{k_{N+1}}$. Pero si por el contrario $k < k_{N+1} - 1$, el $N-$ésimo bloque es "*incompleto*" y en el siguiente paso, los vectores de Lanczos $\mathbf{v}_{k+1}$ y $\mathbf{w}_{k+1}$ son añadidos al $N-$ésimo bloque como vectores "*internos*".

ALGORITMO LAL
Elegir dos vectores $\mathbf{v}_1, \mathbf{w}_1 / \|\mathbf{v}_1\| = \|\mathbf{w}_1\| = 1$;
$\mathbf{V}_1 = \mathbf{v}_1$, $\mathbf{W}_1 = \mathbf{w}_1$, $\mathbf{D}_1 = \mathbf{W}_1^T \mathbf{V}_1$;
$k_1 = 1$, $N = 1$, $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$, $\mathbf{V}_0 = \mathbf{W}_0 = (0)$, $\rho_1 = \zeta_1 = 1$;
Para $k = 1, 2, \ldots$:
1) Decidir para hallar $\mathbf{v}_{k+1}$ y $\mathbf{w}_{k+1}$como vectores "*regulares*" o "*internos*"
e ir a 2) ó 3), respectivamente;
2) (Paso *regular*) Hacer

$$\widetilde{\mathbf{v}}_{k+1} = \mathbf{A}\mathbf{v}_k - \mathbf{V}_N \mathbf{D}_N^{-1} \mathbf{W}_N^T \mathbf{A}\mathbf{v}_k - \mathbf{V}_{N-1} \mathbf{D}_{N-1}^{-1} \mathbf{W}_{N-1}^T \mathbf{A}\mathbf{v}_k$$
$$\widetilde{\mathbf{w}}_{k+1} = \mathbf{A}^T \mathbf{w}_k - \mathbf{W}_N \mathbf{D}_N^{-T} \mathbf{V}_N^T \mathbf{A}^T \mathbf{w}_k - \mathbf{W}_{N-1} \mathbf{D}_{N-1}^{-T} \mathbf{V}_{N-1}^T \mathbf{A}^T \mathbf{w}_k$$

$k_{N+1} = k+1$, $N = N+1$, $\mathbf{V}_N = \mathbf{W}_N = (0)$, e ir a 4)
3) (Paso *interno*) Hacer
$\widetilde{\mathbf{v}}_{k+1} = \mathbf{A}\mathbf{v}_k - \zeta_{k-k_N}\mathbf{v}_k - (\eta_{k-k_N}/\rho_k)\,\mathbf{v}_{k-1} - \mathbf{V}_{N-1}\mathbf{D}_{N-1}^{-1}\mathbf{W}_{N-1}^T\mathbf{A}\mathbf{v}_k$
$\widetilde{\mathbf{w}}_{k+1} = \mathbf{A}^T\mathbf{w}_k - \zeta_{k-k_N}\mathbf{w}_k - (\eta_{k-k_N}/\zeta_k)\,\mathbf{w}_{k-1} -$
$-\mathbf{W}_{N-1}\mathbf{D}_{N-1}^{-T}\mathbf{V}_{N-1}^T\mathbf{A}^T\mathbf{w}_k$
4) Hacer $\rho_{k+1} = \|\widetilde{\mathbf{v}}_{k+1}\|$ y $\zeta_{k+1} = \|\widetilde{\mathbf{w}}_{k+1}\|$;
Si $\rho_{k+1} = 0$ o $\zeta_{k+1} = 0$, parar;
En caso contrario,
$\mathbf{v}_{k+1} = \dfrac{\widetilde{\mathbf{v}}_{k+1}}{\rho_{k+1}}, \quad \mathbf{w}_{k+1} = \dfrac{\widetilde{\mathbf{w}}_{k+1}}{\zeta_{k+1}}$
$\mathbf{V}_N = [\mathbf{V}_N, \mathbf{v}_{k+1}], \quad \mathbf{W}_N = \left[\mathbf{W}_N, \mathbf{w}_{k+1}\right], \quad \mathbf{D}_N = \mathbf{W}_N^T\mathbf{V}_N$
Fin

Nótese que si sólo se ejecutan pasos del tipo 2) (*regulares*), entonces todos los bloques tienen dimensión $h_l = 1$ y el algoritmo se reduce al clásico proceso de biortogonalización de Lanczos.

Es conveniente introducir una nueva notación de la forma,

$$\begin{aligned} \mathbf{V}^{(k)} &= [\mathbf{v}_1, \mathbf{v}_2, \dots \mathbf{v}_k] \qquad \left(= \left[\begin{array}{cccc} \mathbf{V}_1 & \mathbf{V}_2 & \dots & \mathbf{V}_N \end{array}\right]\right), \\ \mathbf{W}^{(k)} &= [\mathbf{w}_1, \mathbf{w}_2, \dots \mathbf{w}_k] \qquad \left(= \left[\begin{array}{cccc} \mathbf{W}_1 & \mathbf{W}_2 & \dots & \mathbf{W}_N \end{array}\right]\right) \end{aligned}$$

Entonces,

$$\begin{aligned} \mathcal{K}_k\left(\mathbf{A}; \mathbf{v}_1\right) &= \left\{\mathbf{V}^{(k)}\mathbf{z}\right\}, \\ \mathcal{K}_k\left(\mathbf{A}^T; \mathbf{w}_1\right) &= \left\{\mathbf{W}^{(k)}\mathbf{z}\right\} \end{aligned}$$

El resultado del algoritmo *look-ahead* Lanczos es una relación de la forma,

$$\mathbf{A}\mathbf{V}^{(k)} = \mathbf{V}^{(k+1)}\overline{\mathbf{T}}^{(k)} \tag{2.24}$$

donde, $\overline{\mathbf{T}}^{(k)}$ es una matriz $(k+1) \times k$,

$$\overline{\mathbf{T}}^{(k)} = \begin{pmatrix} \mathbf{T}^{(k)} \\ \rho_{k+1}\left(\mathbf{e}_k^{(k)}\right)^T \end{pmatrix}, \qquad \left(\mathbf{e}_k^{(k)}\right) = \left[\begin{array}{cccc} 0 & \dots & 0 & 1 \end{array}\right]^T$$

siendo $\mathbf{T}^{(k)}$ una matriz tridiagonal por bloques,

$$\mathbf{T}^{(k)} = \begin{pmatrix} \alpha_1 & \beta_2 & & . & & & \\ \delta_2 & \alpha_2 & \beta_3 & . & & & \\ & \delta_3 & \alpha_3 & . & & & \\ . & . & . & . & . & . & . \\ & & & . & \alpha_{N-2} & \beta_{N-1} & \\ & & & . & \delta_{N-1} & \alpha_{N-1} & \beta_N \\ & & & . & & \delta_N & \alpha_N \end{pmatrix}$$

con,

$$\alpha_l = \begin{pmatrix} * & * & 0 & \cdots & 0 & * \\ \rho_{k_l+1} & * & \ddots & \ddots & \vdots & \vdots \\ 0 & \rho_{k_l+2} & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & * & * \\ \vdots & & \ddots & \ddots & * & * \\ 0 & \cdots & \cdots & 0 & \rho_{k_{l+1}-1} & * \end{pmatrix}, \quad \delta_l = \begin{pmatrix} 0 & \cdots & \cdots & 0 & \rho_{k_l} \\ \vdots & \ddots & & & 0 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

y los bloque β_l son generalmente matrices llenas. Además, las matrices α_l, β_l y δ_l para $l = 1, 2, \ldots N-1$ tienen dimensiones $h_l \times h_l$, $h_{l-1} \times h_l$ y $h_l \times h_{l-1}$, respectivamente. Las matrices α_N, β_N y δ_N correspondientes a los bloques N tienen dimensiones $\widetilde{h}_N \times \widetilde{h}_N$, $h_{N-1} \times \widetilde{h}_N$ y $\widetilde{h}_N \times h_{N-1}$, respectivamente y $\widetilde{h}_N = h_N$ si el bloque es "*completo*".

Capítulo 3

Métodos basados en los subespacios de Krylov

Estos métodos utilizados para la resolución de grandes sistemas lineales, se obtienen para adaptarse, en principio, a dos requerimientos básicos que son:

a) Minimizar una cierta norma del vector residuo sobre un subespacio de Krylov generado por la matriz del sistema y que se traduce en una convergencia suave sin grandes fluctuaciones.

b) Ofrecer un bajo coste computacional por iteración y no exigir alta disponibilidad de almacenaje.

Sin embargo, esto no es siempre posible. Sólo en sistemas de ecuaciones lineales cuya matriz de coeficientes es simétrica y definida positiva, el algoritmo del Gradiente Conjugado propuesto por Hestenes y Stiefel en 1952 [39] y desarrollado en la práctica a partir de 1970, alcanza, salvo errores de redondeo, teóricamente la solución exacta en un número de iteraciones igual a la dimensión del sistema y cumple los requisitos esenciales anteriores de minimalidad y optimalidad.

Cuando la matriz del sistema no cumple las condiciones de simetría y positividad, el método del Gradiente Conjugado no es aplicable y en general, no existen métodos que cumplan los dos requisitos anteriores simultáneamente sin añadir inconvenientes y/o desventajas. Por todo ello, los métodos utilizados para estos sistemas, se obtienen para adaptarse a uno de ellos, bien a la minimización, o bien métodos que ofrezcan un bajo coste computacional y de almacenamiento.

Los podemos agrupar en tres grandes familias (ver [61]): métodos de ortogonalización, métodos de biortogonalización y métodos basados en la Ecuación Normal.

3.1. Métodos de ortogonalización

Están basados en un proceso de proyección ortogonal sobre un subespacio de Krylov de dimensión menor que la del sistema. Las largas recurrencias de estos métodos hacen que el coste computacional y la memoria ocupada aumenten considerablemente con el número de iteraciones. Se fundamentan en el algoritmo de ortogonalización de Arnoldi.

3.1.1. Método de Arnoldi para sistemas lineales (Método FOM)

La aplicación del método de Arnoldi a los sistemas lineales de ecuaciones se realiza de la siguiente forma: dado el sistema $\mathbf{Ax} = \mathbf{b}$, sea $\mathbf{x}_0$ una aproximación inicial a partir de la cual se construye una secuencia de vectores $\mathbf{x}_k \in \mathbf{x}_0 + \mathcal{K}_k$, siendo $\mathcal{K}_k \equiv C.L.\left\{\mathbf{r}_0, \mathbf{Ar}_0, \mathbf{A}^2\mathbf{r}_0, ..., \mathbf{A}^{k-1}\mathbf{r}_0\right\}$ con $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, imponiendo la condición de ortogonalidad,

$$\mathbf{b} - \mathbf{Ax}_k \perp \mathcal{K}_k$$

como la solución aproximada del sistema viene dada por,

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\mathbf{y}_k \tag{3.1}$$

entonces,

$$\mathbf{b} - \mathbf{A}\left(\mathbf{x}_0 + \mathbf{V}_k\mathbf{y}_k\right) \perp \mathcal{K}_k \Rightarrow \mathbf{r}_0 - \mathbf{AV}_k\mathbf{y}_k \perp \mathcal{K}_k$$

es decir que,

$$\mathbf{V}_k^T\mathbf{AV}_k\mathbf{y}_k = \mathbf{V}_k^T\mathbf{r}_0$$

y teniendo en cuenta la ecuación (2.10),

$$\mathbf{H}_k\mathbf{y}_k = \mathbf{V}_k^T\mathbf{r}_0$$

si $\mathbf{v}_1 = \mathbf{r}_0/ \parallel \mathbf{r}_0 \parallel$ entonces,

$$\mathbf{V}_k^T\mathbf{r}_0 = \mathbf{V}_k^T\left(\beta\mathbf{v}_1\right) = \beta\mathbf{e}_1$$

siendo $\beta = \parallel \mathbf{r}_0 \parallel$, por tanto,

$$\mathbf{y}_k = \mathbf{H}_k^{-1}\left(\beta\mathbf{e}_1\right) \tag{3.2}$$

Basado en esta aproximación y llamado Método FOM (*Full Orthogonalization Method*), este método aplica en cada paso el algoritmo de Arnoldi modificado Gram-Schmidt para resolver el sistema de ecuaciones (2.1).

ALGORITMO FOM
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, $\beta = \parallel \mathbf{r}_0 \parallel$, y $\mathbf{v}_1 = \mathbf{r}_0/\beta$
Definir la $k \times k$ matriz $\mathbf{H}_k = \{\mathbf{H}\}_{i,j=1,...,k}$; poner $\mathbf{H}_k = 0$

Desde $j = 1, ..., k$ hacer
  $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$;
  Desde $i = 1, ..., j$ hacer
    $\{\mathbf{H}\}_{ij} = \langle \mathbf{w}_j, \mathbf{v}_i \rangle$;
    $\mathbf{w}_j = \mathbf{w}_j - \{\mathbf{H}\}_{ij} \mathbf{v}_i$;
  Fin
  $\{\mathbf{H}\}_{j+1,j} = \| \mathbf{w}_j \|$; Si $\{\mathbf{H}\}_{j+1,j} = 0$ poner $k = j$ y parar
  $\mathbf{v}_{j+1} = \frac{1}{\{\mathbf{H}\}_{j+1,j}} \mathbf{w}_j$;
Fin
Resolver $\mathbf{H}_k \mathbf{y}_k = \beta \mathbf{e}_1$
Calcular $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k$

El vector residuo de la solución aproximada $\mathbf{x}_k$ obtenido con el algoritmo FOM será,

$$\mathbf{b} - \mathbf{A}\mathbf{x}_k = -h_{k+1,k} \mathbf{e}_k^T \mathbf{y}_k \mathbf{v}_{k+1}$$

y entonces,

$$\| \mathbf{b} - \mathbf{A}\mathbf{x}_k \| = \mathbf{h}_{k+1,k} \left| \mathbf{e}_k^T \mathbf{y}_k \right| \tag{3.3}$$

El coste computacional de este método es de orden $(k^2 n)$ debido al proceso de ortogonalización de Gram-Schmidt y el coste de almacenamiento es de (kn), lo que supone que para órdenes grandes de n, la dimensión del subespacio $\mathcal{K}_k$ también es grande. Existen alternativas para paliar este problema como son el algoritmo *Restarted* FOM que periodicamente hace un *restart*, esto es, se ejecutan un número prefijado de iteraciones para obtener una solución aproximada y se usa ésta como inicialización en una siguiente aplicación del algoritmo (ver por ejemplo [2]), y los algoritmos IOM y DIOM que se basan en el truncamiento de la ortogonalización de Arnoldi [72].

3.1.2. Método de Mínimo Residuo Generalizado (GMRES)

El algoritmo GMRES [70, 72] aproxima la solución de forma similar al algoritmo FOM visto anteriormente.

Es un método de proyección basado en que $\mathcal{K} = \mathcal{K}_k$ y $\mathcal{L} = \mathbf{A}\mathcal{K}_k$ donde $\mathcal{K}_k$ es un subespacio de Krylov de dimensión k, lo que equivale a minimizar la norma del residuo.

Así, el desarrollo del algoritmo consiste en encontrar un vector $\mathbf{x}$ de $\mathbf{x}_0 + \mathcal{K}_k$ tal que,

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}$$

imponiendo la condición de mínimo para,

$$\mathbf{J}(\mathbf{y}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\| \tag{3.4}$$

y como,

$$\mathbf{b} - \mathbf{A}\mathbf{x} = \mathbf{b} - \mathbf{A}\left(\mathbf{x}_0 + \mathbf{V}_k\mathbf{y}\right) = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_k\mathbf{y}$$

Teniendo en cuenta la ecuación (2.9) y $\mathbf{v}_1 = \mathbf{r}_0 / \parallel \mathbf{r}_0 \parallel$ obtenido del algoritmo de Arnoldi y llamando $\beta = \parallel \mathbf{r}_0 \parallel$, entonces,

$$\mathbf{b} - \mathbf{A}\mathbf{x} = \beta\mathbf{v}_1 - \mathbf{V}_{k+1}\overline{\mathbf{H}}_k\mathbf{y}$$

pero, $\mathbf{v}_1 = \mathbf{V}_{k+1}\mathbf{e}_1$, con $\mathbf{e}_1 \in \Re^{k+1}$, por tanto,

$$\mathbf{b} - \mathbf{A}\mathbf{x} = \mathbf{V}_{k+1}\left(\beta\mathbf{e}_1 - \overline{\mathbf{H}}_k\mathbf{y}\right) \tag{3.5}$$

y la ecuación (3.4) quedará,

$$\mathbf{J}(\mathbf{y}) = \left\|\mathbf{V}_{k+1}\left(\beta\mathbf{e}_1 - \overline{\mathbf{H}}_k\mathbf{y}\right)\right\|$$

Como las columnas de la matriz $\mathbf{V}_{k+1}$ son ortonormales por construcción, podemos simplificar la anterior expresión,

$$\mathbf{J}(\mathbf{y}) = \left\|\left(\beta\mathbf{e}_1 - \overline{\mathbf{H}}_k\mathbf{y}\right)\right\| \tag{3.6}$$

El algoritmo GMRES busca el único vector de $\mathbf{x}_0 + \mathcal{K}_k$ que minimiza la funcional $\mathbf{J}(\mathbf{y})$.

ALGORITMO GMRES
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Definir la $(k+1) \times k$ matriz $\overline{\mathbf{H}}_k = \{\mathbf{H}\}_{1 \le i \le k+1, 1 \le j \le k}$. Poner $\overline{\mathbf{H}}_k = 0$.
Desde $j = 1, ..., k$ hacer
  $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$
  Desde $i = 1, ..., j$ hacer
    $\{\mathbf{H}\}_{ij} = \langle \mathbf{w}_j, \mathbf{v}_i \rangle$;
    $\mathbf{w}_j = \mathbf{w}_j - \{\mathbf{H}\}_{ij}\mathbf{v}_i$;
  Fin
  $\{\mathbf{H}\}_{j+1,j} = \parallel \mathbf{w}_j \parallel$; Si $\{\mathbf{H}\}_{j+1,j} = 0$ poner $k = j$ y parar
  $\mathbf{v}_{j+1} = \dfrac{1}{\{\mathbf{H}\}_{j+1,j}}\mathbf{w}_j$;
Fin
Hallar $\mathbf{y}_k$ que minimiza $\left\|\left(\beta\mathbf{e}_1 - \overline{\mathbf{H}}_k\mathbf{y}\right)\right\|$;
Determinar $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\mathbf{y}_k$ siendo $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$;
Calcular $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$

Igual que ocurría en el algoritmo FOM, el algoritmo GMRES, resulta impracticable cuando k es muy grande, ya que conlleva un elevado coste computacional y de almacenamiento, por lo que se procede de forma análoga usando las técnicas *restart* y de truncamiento.

ALGORITMO RESTARTED GMRES

1) Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\beta = \| \mathbf{r}_0 \|$, y $\mathbf{v}_1 = \mathbf{r}_0/\beta$;

2) Generar la base de Arnoldi y la matriz $\overline{\mathbf{H}}_k$ usando el algoritmo de Arnoldi;

3) Iniciar con $\mathbf{v}_1$;

4) Hallar $\mathbf{y}_k$ que minimiza $\left\| \left(\beta \mathbf{e}_1 - \overline{\mathbf{H}}_k \mathbf{y}\right) \right\|$ y $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k$;

5) Si se satisface entonces parar. En caso contrario poner $\mathbf{x}_0 := \mathbf{x}_k$ y volver al paso 1.

3.1.3. Método de Lanczos para sistemas simétricos

La aplicación del método de Lanczos a los sistemas lineales de ecuaciones se realiza de la siguiente forma:

Dada una aproximación incial $\mathbf{x}_0$ del sistema de matriz simétrica, definida positiva $\mathbf{A}\mathbf{x} = \mathbf{b}$, y obtenidos los vectores $\mathbf{v}_i$ $(i = 1, 2, ...)$ del algoritmo de Lanczos, junto con la matriz tridiagonal $\mathbf{T}_k$, la solución aproximada obtenida por un método de proyección ortogonal sobre $\mathcal{K}_k$ viene dada por,

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k$$

$$\mathbf{y}_k = \mathbf{T}_k^{-1} \left(\beta \mathbf{e}_1\right)$$

ALGORITMO DE LANCZOS PARA SISTEMAS LINEALES

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\beta = \| \mathbf{r}_0 \|$, y $\mathbf{v}_1 = \mathbf{r}_0/\beta$

Desde $j = 1, ..., k$ hacer

$\mathbf{w}_j = \mathbf{A}\mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$(si $j = 1$ poner $\beta_1 \mathbf{v}_0 \equiv 0$);

$\alpha_j = \langle \mathbf{w}_j, \mathbf{v}_j \rangle$;

$\mathbf{w}_j = \mathbf{w}_j - \alpha_j \mathbf{v}_j$;

$\beta_{j+1} = \| \mathbf{w}_j \|$;

$\mathbf{v}_{j+1} = \dfrac{1}{\beta_{j+1}} \mathbf{w}_j$

Fin

Generar las matrices $\mathbf{T}_k = tridiag(\beta_i, \alpha_i, \beta_{i+1})$, y $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$

Resolver $\mathbf{T}_k \mathbf{y}_k = \beta \mathbf{e}_1$

Calcular $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k$

donde el vector residuo de la solución aproximada $\mathbf{x}_k$ es tal que,

$$\mathbf{b} - \mathbf{A}\mathbf{x}_k = -\beta_{k+1} \mathbf{e}_k^T \mathbf{y}_k \mathbf{v}_{k+1} \tag{3.7}$$

3.1.4. Método del Gradiente Conjugado (CG)

Una variante del algoritmo de Lanczos [41], se puede obtener mediante el truncamiento del proceso de ortogonalización realizando una factorización de

la matriz $\mathbf{T}_k$ en $\mathbf{L}_k\mathbf{U}_k$, esto es,

$$\mathbf{T}_k = \begin{pmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \cdot & \cdot & \cdot & \\ & & \lambda_{k-1} & 1 & \\ & & & \lambda_k & 1 \end{pmatrix} \times \begin{pmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ & & \cdot & \cdot & \cdot \\ & & & \eta_{k-1} & \beta_k \\ & & & & \eta_k \end{pmatrix} \tag{3.8}$$

entonces, la solución aproximada vendrá dada por,

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\mathbf{U}_k^{-1}\mathbf{L}_k^{-1}\left(\beta\mathbf{e}_1\right)$$

haciendo,

$$\mathbf{P}_k = \mathbf{V}_k\mathbf{U}_k^{-1}$$
$$\mathbf{z}_k = \mathbf{L}_k^{-1}\left(\beta\mathbf{e}_1\right)$$

quedará,

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{P}_k\mathbf{z}_k \tag{3.9}$$

La última columna $\mathbf{p}_k$ de la matriz $\mathbf{P}_k$, puede ser calculada a partir de las previas $\mathbf{p}_i$ y $\mathbf{v}_k$ de la expresión,

$$\mathbf{p}_k = \eta_k^{-1}\left[\mathbf{v}_k - \beta_k\mathbf{p}_{k-1}\right]$$

donde β_k es obtenido en el algoritmo de Lanczos y η_k se obtiene de la factorización de $\mathbf{T}_k$, esto es,

$$\lambda_k = \frac{\beta_k}{\eta_{k-1}} \tag{3.10}$$

$$\eta_k = \alpha_k - \lambda_k\beta_k \tag{3.11}$$

Si hacemos,

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{z}_{k-1} \\ \zeta_k \end{bmatrix}$$

donde $\zeta_k = -\lambda_k\zeta_{k-1}$, podremos escribir,

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \zeta_k\mathbf{p}_k$$

que proporciona una nueva versión del Algoritmo de Lanczos para sistemas lineales.

ALGORITMO D-LANCZOS
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\zeta_1 = \beta = \| \mathbf{r}_0 \|$, y $\mathbf{v}_1 = \mathbf{r}_0/\beta$
$\lambda_1 = \beta_1 = 0$, $\mathbf{p}_0 = \mathbf{0}$
Desde $k = 1, 2, ...$ hasta converger, hacer
$\mathbf{w} = \mathbf{A}\mathbf{v}_k - \beta_k\mathbf{v}_{k-1}$ y $\alpha_k = \langle \mathbf{w}, \mathbf{v}_k \rangle$;

Si $k > 1$ entonces $\lambda_k = \dfrac{\beta_k}{\eta_{k-1}}$ y $\zeta_k = -\lambda_k \zeta_{k-1}$;
$\eta_k = \alpha_k - \lambda_k \beta_k$
$\mathbf{p}_k = \eta_k^{-1} \left[\mathbf{v}_k - \beta_k \mathbf{p}_{k-1}\right]$
$\mathbf{x}_k = \mathbf{x}_{k-1} + \zeta_k \mathbf{p}_k$
Si $\mathbf{x}_k$ converge Fin
$\mathbf{w} = \mathbf{w} - \alpha_k \mathbf{v}_k$;
$\beta_{k+1} = \| \mathbf{w} \|$;
$\mathbf{v}_{k+1} = \dfrac{1}{\beta_{k+1}} \mathbf{w}$
Fin

Este algoritmo es matemáticamente equivalente al anterior, es decir, proporciona la misma solución aproximada, pero este último calcula la solución del sistema tridiagonal $\mathbf{T}_k \mathbf{y}_k = \beta \mathbf{e}_1$ progresivamente utilizando la eliminación Gaussiana de forma implicita.

Los vectores residuos generados son ortogonales y los vectores $\mathbf{p}_i$ son $\mathbf{A}$-ortogonales, o conjugados.

Efectivamente, cada vector residuo es tal que $\mathbf{r}_k = \sigma_k \mathbf{v}_{k+1}$, donde σ_k es un cierto escalar, por tanto como los vectores $\mathbf{v}_i$ son ortogonales, los vectores residuos también lo serán. Por otro lado como $\mathbf{P}_k = \mathbf{V}_k \mathbf{U}_k^{-1}$, entonces,

$$\begin{aligned} \mathbf{P}_k^T \mathbf{A} \mathbf{P}_k &= \left(\mathbf{V}_k \mathbf{U}_k^{-1}\right)^T \mathbf{A} \mathbf{V}_k \mathbf{U}_k^{-1} = \\ &= \mathbf{U}_k^{-T} \mathbf{V}_k^T \mathbf{A} \mathbf{V}_k \mathbf{U}_k^{-1} = \mathbf{U}_k^{-T} \mathbf{T}_k \mathbf{U}_k^{-1} = \\ &= \mathbf{U}_k^{-T} \mathbf{L}_k \end{aligned}$$

que es una matriz triangular inferior, necesariamente simétrica. Por tanto, la matriz $\mathbf{P}_k^T \mathbf{A} \mathbf{P}_k$ es diagonal y como consecuencia los vectores $\mathbf{p}_i$ son $\mathbf{A}$-ortogonales, es decir que $\langle \mathbf{A}\mathbf{p}_i, \mathbf{p}_j \rangle = 0$ si $i \neq j$.

El método del Gradiente Conjugado está basado en una técnica de proyección ortogonal sobre el subespacio de Krylov $\mathcal{K}_k(\mathbf{A}; \mathbf{r}_0)$ donde $\mathbf{r}_0$ es el residuo inicial y se fundamenta en el algoritmo D-Lanczos de la siguiente forma:

Una aproximación $\mathbf{x}_{j+1}$ puede ser expresada,

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j \tag{3.12}$$

entonces, los vectores residuos deben satisfacer que,

$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A} \mathbf{p}_j \tag{3.13}$$

estos vectores residuos son ortogonales,

$$\left\langle \mathbf{r}_j - \alpha_j \mathbf{A} \mathbf{p}_j, \mathbf{r}_j \right\rangle = 0$$

por tanto,

$$\alpha_j = \frac{\langle \mathbf{r}_j, \mathbf{r}_j \rangle}{\left\langle \mathbf{A} \mathbf{p}_j, \mathbf{r}_j \right\rangle} \tag{3.14}$$

como las siguientes direcciones $\mathbf{p}_{j+1}$ son una combinación lineal de $\mathbf{r}_{j+1}$ y $\mathbf{p}_j$, después de reescalar los vectores $\mathbf{p}$ apropiadamente,

$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j \tag{3.15}$$

con lo cual,

$$\langle \mathbf{A}\mathbf{p}_j, \mathbf{r}_j \rangle = \langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1} \rangle = \langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j \rangle$$

ya que $\mathbf{A}\mathbf{p}_j$ es ortogonal a $\mathbf{p}_{j-1}$. Entonces, teniendo en cuenta (3.14) y (3.15) obtenemos que,

$$\beta_j = -\frac{\langle \mathbf{r}_{j+1}, \mathbf{A}\mathbf{p}_j \rangle}{\langle \mathbf{p}_j, \mathbf{A}\mathbf{p}_j \rangle}$$

y como de (3.13), podemos escribir,

$$\mathbf{A}\mathbf{p}_j = -\frac{1}{\alpha_j}(\mathbf{r}_{j+1} - \mathbf{r}_j) \tag{3.16}$$

entonces,

$$\beta_j = -\frac{1}{\alpha_j}\frac{\langle \mathbf{r}_{j+1}, (\mathbf{r}_{j+1} - \mathbf{r}_j) \rangle}{\langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j \rangle} = \frac{\langle \mathbf{r}_{j+1}, \mathbf{r}_{j+1} \rangle}{\langle \mathbf{r}_j, \mathbf{r}_j \rangle}$$

ALGORITMO DEL GRADIENTE CONJUGADO (CG)
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{r}_0$
Mientras $\| \mathbf{r}_{j-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (j = 1, 2, 3, ...)$, hacer

$$\alpha_j = \frac{\langle \mathbf{r}_j, \mathbf{r}_j \rangle}{\langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j \rangle};$$
$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$$
$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$$
$$\beta_j = \frac{\langle \mathbf{r}_{j+1}, \mathbf{r}_{j+1} \rangle}{\langle \mathbf{r}_j, \mathbf{r}_j \rangle}$$
$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$$

Fin

Téngase en cuenta que los escalares α_j, β_j en este algoritmo no son los obtenidos en el algoritmo de Lanczos, pero podemos encontrar la relación entre estos y los coeficientes de la matriz obtenida en el algoritmo de Lanczos. Sea,

$$\mathbf{T}_k = tridiag(\eta_j, \delta_j, \eta_{j+1})$$

De la ecuación (3.7) podemos escribir,

$$\mathbf{r}_j = escalar \times \mathbf{v}_{j+1} \tag{3.17}$$

Como,

$$\delta_{j+1} = \frac{\langle \mathbf{A}\mathbf{v}_{j+1}, \mathbf{v}_{j+1} \rangle}{\langle \mathbf{v}_{j+1}, \mathbf{v}_{j+1} \rangle} = \frac{\langle \mathbf{A}\mathbf{r}_j, \mathbf{r}_j \rangle}{\langle \mathbf{r}_j, \mathbf{r}_j \rangle}$$

De (3.15) se obtiene,

$$\mathbf{r}_j = \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1} \tag{3.18}$$

entonces,

$$\langle \mathbf{A}\mathbf{r}_j, \mathbf{r}_j \rangle = \langle \mathbf{A}\left(\mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1}\right), \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1} \rangle$$

teniendo en cuenta que los términos $\beta_{j-1}\mathbf{p}_{j-1}$ se anulan para $j = 0$ porque los vectores $\mathbf{p}$ son $\mathbf{A}$-ortogonales,

$$\langle \mathbf{A}\mathbf{r}_j, \mathbf{r}_j \rangle = \left\langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j \right\rangle + \beta_{j-1}^2 \left\langle \mathbf{A}\mathbf{p}_{j-1}, \mathbf{p}_{j-1} \right\rangle$$

Obtenemos para $j > 0$,

$$\delta_{j+1} = \frac{\left\langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j \right\rangle}{\langle \mathbf{r}_j, \mathbf{r}_j \rangle} + \beta_{j-1}^2 \frac{\left\langle \mathbf{A}\mathbf{p}_{j-1}, \mathbf{p}_{j-1} \right\rangle}{\langle \mathbf{r}_j, \mathbf{r}_j \rangle} = \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}} \tag{3.19}$$

Por tanto, los elementos de la diagonal de $\mathbf{T}_k$ serán,

$$\delta_{j+1} = \begin{cases} \dfrac{1}{\alpha_j} \text{ para } j = 0 \\ \dfrac{1}{\alpha_j} + \dfrac{\beta_{j-1}}{\alpha_{j-1}} \text{ para } j > 0 \end{cases} \tag{3.20}$$

Para encontrar una expresión de los elementos co-diagonales η_{j+1}, de la definición del algoritmo de Lanczos,

$$\eta_{j+1} = \langle \mathbf{A}\mathbf{v}_j, \mathbf{v}_{j+1} \rangle = \frac{\langle \mathbf{A}\mathbf{r}_{j-1}, \mathbf{r}_{j-1} \rangle}{\| \mathbf{r}_{j-1} \| \| \mathbf{r}_j \|}$$

Por la relación (3.18) y teniendo en cuenta la propiedad de ortogonalidad del algoritmo del Gradiente Conjugado obtenemos,

$$\begin{aligned} \langle \mathbf{A}\mathbf{r}_{j-1}, \mathbf{r}_j \rangle &= \langle \mathbf{A}\left(\mathbf{p}_{j-1} - \beta_{j-2}\mathbf{p}_{j-2}\right), \mathbf{r}_j \rangle = \\ &= \left\langle \mathbf{A}\mathbf{p}_{j-1}, \mathbf{r}_j \right\rangle - \beta_{j-2} \left\langle \mathbf{A}\mathbf{p}_{j-2}, \mathbf{r}_j \right\rangle = \\ &= \frac{-1}{\alpha_{j-1}} \langle \mathbf{r}_j - \mathbf{r}_{j-1}, \mathbf{r}_j \rangle + \frac{\beta_{j-2}}{\alpha_{j-2}} \langle \mathbf{r}_{j-1} - \mathbf{r}_{j-2}, \mathbf{r}_j \rangle = \\ &= \frac{-1}{\alpha_{j-1}} \langle \mathbf{r}_j, \mathbf{r}_j \rangle \end{aligned}$$

Entonces,

$$\eta_{j+1} = \frac{1}{\alpha_{j-1}} \frac{\langle \mathbf{r}_j, \mathbf{r}_j \rangle}{\| \mathbf{r}_{j-1} \| \| \mathbf{r}_j \|} = \frac{1}{\alpha_{j-1}} \frac{\| \mathbf{r}_j \|}{\| \mathbf{r}_{j-1} \|} = \frac{\sqrt{\beta_{j-1}}}{\alpha_{j-1}} \tag{3.21}$$

Finalmente, podemos escribir la matriz tridiagonal de Lanczos en función de los coeficientes del Gradiente Conjugado,

$$\mathbf{T}_k = \begin{pmatrix} \frac{1}{\alpha_0} & \frac{\sqrt{\beta_0}}{\alpha_0} & & & & \\ \frac{\sqrt{\beta_0}}{\alpha_0} & \frac{1}{\alpha_0} + \frac{\beta_0}{\alpha_0} & \frac{\sqrt{\beta_1}}{\alpha_1} & & & \\ & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \frac{\sqrt{\beta_{k-2}}}{\alpha_{k-2}} \\ & & & & \frac{\sqrt{\beta_{k-2}}}{\alpha_{k-2}} & \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-2}}{\alpha_{k-2}} \end{pmatrix}$$

3.2. Métodos de biortogonalización

Estos métodos son también métodos de proyección pero intrínsecamente no ortogonales. Tienen la ventaja de que las fórmulas de recurrencia son reducidas y el almacenamiento no aumenta con el número de iteraciones, pero por contra, presentan un comportamiento irregular en la convergencia, con oscilaciones y abundantes "picos" que pueden conducir a criterios de parada con soluciones erróneas. Se fundamentan en el algoritmo de biortogonalización de Lanczos [17].

3.2.1. Método de biortogonalización de Lanczos para sistemas no simétricos

La aplicación del método de biortogonalización de Lanczos a los sistemas no simétricos de ecuaciones lineales se realiza de la siguiente forma:

ALGORITMO DE BIORTOGONALIZACIÓN DE LANCZOS PARA SISTEMAS LINEALES

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\beta = \| \mathbf{r}_0 \|$

Ejecutar k pasos del Algoritmo de biortogonalización de Lanczos,

Inciar con $\mathbf{v}_1 = \mathbf{r}_0/\beta$, y con $\mathbf{w}_1$ tal que $\langle \mathbf{v}_1, \mathbf{w}_1 \rangle = 1$

Generar los vectores $\mathbf{v}_1, ..., \mathbf{v}_k,\ \mathbf{w}_1, ..., \mathbf{w}_k$

Generar la matriz tridiagonal $\mathbf{T}_k$

Resolver $\mathbf{T}_k \mathbf{y}_k = \beta \mathbf{e}_1$ y calcular $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k$

donde el vector residuo de la solución aproximada $\mathbf{x}_k$ es tal que,

$$\| \mathbf{b} - \mathbf{A}\mathbf{x}_k \| = \left| \delta_{k+1} \mathbf{e}_k^T \mathbf{y}_k \right| \| \mathbf{v}_{k+1} \| \tag{3.22}$$

3.2.2. Método del Doble Gradiente Conjugado (Bi-CG)

El algoritmo del doble gradiente conjugado (Bi-CG) [17], se deriva del algoritmo de biortogonalización de Lanczos de la misma forma en que el Gradiente Conjugado (CG) se derivó del algoritmo de Lanczos. Implicitamente, el algoritmo resuelve, no sólo el sistema original $\mathbf{A}\mathbf{x} = \mathbf{b}$, sino también el sistema dual o sistema auxiliar $\mathbf{A}^T \mathbf{x}^* = \mathbf{b}^*$, aunque éste es ignorado en la formulación del algoritmo.

El algoritmo Bi-CG es un proceso de proyección sobre,

$$\mathcal{K}_k (\mathbf{A}; \mathbf{v}_1) = C.L. \left\{ \mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \mathbf{A}^2 \mathbf{v}_1, ..., \mathbf{A}^{k-1} \mathbf{v}_1 \right\}$$

ortogonalmente a,

$$\mathcal{L}_k \left(\mathbf{A}^T; \mathbf{w}_1 \right) = C.L. \left\{ \mathbf{w}_1, \mathbf{A}^T \mathbf{w}_1, \left(\mathbf{A}^T \right)^2 \mathbf{w}_1, ..., \left(\mathbf{A}^T \right)^{k-1} \mathbf{w}_1 \right\}$$

haciendo $\mathbf{v}_1 = \mathbf{r}_0 / \parallel \mathbf{r}_0 \parallel$. El vector $\mathbf{w}_1$ es elegido arbitrariamente tal que $\langle \mathbf{v}_1, \mathbf{w}_1 \rangle \neq 0$, generalmente igual a $\mathbf{v}_1$. El residuo inicial del sistema dual es $\mathbf{r}_0^* = \mathbf{b}^* - \mathbf{A}^T \mathbf{x}_0^*$.

Procediendo de la misma forma que en el algoritmo CG, descomponemos la matriz $\mathbf{T}_k$,

$$\mathbf{T}_k = \mathbf{L}_k \mathbf{U}_k \tag{3.23}$$

y se define,

$$\mathbf{P}_k = \mathbf{V}_k \mathbf{U}_k^{-1} \tag{3.24}$$

Entonces, la solución se puede expresar,

$$\begin{aligned} \mathbf{x}_k &= \mathbf{x}_0 + \mathbf{V}_k \mathbf{T}_k^{-1} (\beta \mathbf{e}_1) \\ &= \mathbf{x}_0 + \mathbf{V}_k \mathbf{U}_k^{-1} \mathbf{L}_k^{-1} (\beta \mathbf{e}_1) \\ &= \mathbf{x}_0 + \mathbf{P}_k \mathbf{L}_k^{-1} (\beta \mathbf{e}_1) \end{aligned}$$

Como en el algoritmo CG los vectores $\mathbf{r}_k$ y $\mathbf{r}_k^*$ tienen las mismas direcciones que los $\mathbf{v}_{k+1}$ y $\mathbf{w}_{k+1}$ respectivamente, forman una secuencia de vectores biortogonales.

Definimos de forma similar la matriz,

$$\mathbf{P}_k^* = \mathbf{W}_k \mathbf{L}_k^{-1} \tag{3.25}$$

Claramente, los vectores columnas $\mathbf{p}_i^*$ de $\mathbf{P}_k^*$ y los $\mathbf{p}_i$ de $\mathbf{P}_k$ son $\mathbf{A}$-ortogonales, ya que,

$$(\mathbf{P}_k^*)^T \mathbf{A} \mathbf{P}_k = \mathbf{L}_k^{-1} \mathbf{W}_k^T \mathbf{A} \mathbf{V}_k \mathbf{U}_k^{-1} = \mathbf{L}_k^{-1} \mathbf{T}_k \mathbf{U}_k^{-1} = \mathbf{I}$$

ALGORITMO BI-CG
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$. Elegir $\mathbf{r}_0^*$ tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$,
$\mathbf{p}_0 = \mathbf{r}_0$, $\mathbf{p}_0^* = \mathbf{r}_0^*$
Mientras $\parallel \mathbf{r}_{j-1} \parallel / \parallel \mathbf{r}_0 \parallel \geq \varepsilon \quad (j = 1, 2, 3, ...)$, hacer

$$\begin{aligned} \alpha_j &= \frac{\langle \mathbf{r}_j, \mathbf{r}_j^* \rangle}{\langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j^* \rangle}; \\ \mathbf{x}_{j+1} &= \mathbf{x}_j + \alpha_j \mathbf{p}_j \\ \mathbf{r}_{j+1} &= \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j \\ \mathbf{r}_{j+1}^* &= \mathbf{r}_j^* - \alpha_j \mathbf{A}^T \mathbf{p}_j^* \\ \beta_j &= \frac{\langle \mathbf{r}_{j+1}, \mathbf{r}_{j+1}^* \rangle}{\langle \mathbf{r}_j, \mathbf{r}_j^* \rangle} \\ \mathbf{p}_{j+1} &= \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j \\ \mathbf{p}_{j+1}^* &= \mathbf{r}_{j+1}^* + \beta_j \mathbf{p}_j^* \end{aligned}$$

Fin

Los vectores generados en el algoritmo anterior satisfacen las siguientes propiedades de ortogonalidad:

$$\langle \mathbf{r}_j, \mathbf{r}_i^* \rangle = 0 \text{ para } i \neq j \tag{3.26}$$

$$\left\langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_i^* \right\rangle = 0 \text{ para } i \neq j \tag{3.27}$$

Estas propiedades pueden desmostrarse por inducción o más facilmente a partir de las relaciones entre los vectores $\mathbf{r}_j, \mathbf{r}_j^*, \mathbf{p}_j, \mathbf{p}_j^*$ y los vectores columnas de las matrices $\mathbf{V}_k, \mathbf{W}_k, \mathbf{P}_k, \mathbf{P}_k^*$.

3.2.3. Método CGS (*Conjugate Gradient Squared*)

Desarrollado por Sonneveld en 1989 [76], es una modificación del Bi-CG. Sonneveld observa que en caso de convergencia del Doble Gradiente, los vectores $\mathbf{r}_i$ y $\mathbf{r}_j$ tienden a cero, pero la convergencia de los "residuos auxiliares" no es explotada y sólo se calculan para la valoración de los parámetros que aparecen en el algoritmo. Propone entonces, la siguiente modificación donde todos los esfuerzos se concentran en la convergencia de los vectores del sistema original $\mathbf{r}_i$.

Usando expresiones polinómicas para los vectores residuo $\mathbf{r}_j$, y la correspondiente dirección conjugada $\mathbf{p}_j$,

$$\mathbf{r}_j = \phi_j(\mathbf{A})\mathbf{r}_0 \tag{3.28}$$

$$\mathbf{p}_j = \pi_j(\mathbf{A})\mathbf{r}_0 \tag{3.29}$$

donde ϕ_j es un cierto polinomio matricial de grado j que satisface la restricción $\phi_j(\mathbf{0}) = \mathbf{I}$, y π_j es igualmente otro polinomio matricial de grado j.

De forma similar, y teniendo en cuenta que los vectores $\mathbf{r}_j^*$ y $\mathbf{p}_j^*$ se obtuvieron a la vez usando las mismas fórmulas de recurrencia que $\mathbf{r}_j$ y $\mathbf{p}_j$, sólo reemplazando la matriz $\mathbf{A}$ por $\mathbf{A}^T$, entonces,

$$\mathbf{r}_j^* = \phi_j(\mathbf{A}^T)\mathbf{r}_0^*, \;\; \mathbf{p}_j^* = \pi_j(\mathbf{A}^T)\mathbf{r}_0^*$$

Por tanto, el escalar α_j del algoritmo Bi-CG se obtendrá,

$$\alpha_j = \frac{\left\langle \phi_j(\mathbf{A})\mathbf{r}_0, \phi_j(\mathbf{A}^T)\mathbf{r}_0^* \right\rangle}{\left\langle \mathbf{A}\pi_j(\mathbf{A})\mathbf{r}_0, \pi_j(\mathbf{A}^T)\mathbf{r}_0^* \right\rangle} = \frac{\left\langle \phi_j^2(\mathbf{A})\mathbf{r}_0, \mathbf{r}_0^* \right\rangle}{\left\langle \mathbf{A}\pi_j^2(\mathbf{A})\mathbf{r}_0, \mathbf{r}_0^* \right\rangle}$$

Sonneveld, sugiere trabajar con aproximaciones $\widehat{\mathbf{x}}_j$, que satisfacen,

$$\widehat{\mathbf{r}}_j = \mathbf{b} - \mathbf{A}\widehat{\mathbf{x}}_j = \phi_j^2(\mathbf{A})\mathbf{r}_0 \tag{3.30}$$

Como la expresión del vector residuo del sistema original en el algoritmo Bi-CG viene dada por $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$, entonces sustituyendo,

$$\phi_{j+1}(\mathbf{A})\mathbf{r}_0 = \phi_j(\mathbf{A})\mathbf{r}_0 - \alpha_j \mathbf{A}\pi_j(\mathbf{A})\mathbf{r}_0 \tag{3.31}$$

y elevando al cuadrado,

$$\phi_{j+1}^2(\mathbf{A}) = \phi_j^2(\mathbf{A}) - \alpha_j \mathbf{A}\left[2\phi_j(\mathbf{A})\pi_j(\mathbf{A}) - \alpha_j \mathbf{A}\pi_j^2(\mathbf{A})\right] \tag{3.32}$$

Para calcular por recurrencia las direcciones conjugadas, como,

$$\pi_{j+1}(\mathbf{A}) = \phi_{j+1}(\mathbf{A}) + \beta_j \pi_j(\mathbf{A}) \tag{3.33}$$

elevando al cuadrado,

$$\pi_{j+1}^2(\mathbf{A}) = \phi_{j+1}^2(\mathbf{A}) + 2\beta_j \phi_{j+1}(\mathbf{A})\pi_j(\mathbf{A}) + \beta_j^2 \pi_j(\mathbf{A})^2 \tag{3.34}$$

De (3.33) se tiene que,

$$\phi_j(\mathbf{A})\pi_j(\mathbf{A}) = \phi_j^2(\mathbf{A}) + \beta_{j-1}\phi_j(\mathbf{A})\pi_{j-1}(\mathbf{A})$$

y de forma similar,

$$\phi_{j+1}(\mathbf{A})\pi_j(\mathbf{A}) = \phi_j^2(\mathbf{A}) + \beta_{j-1}\phi_j(\mathbf{A})\pi_{j-1}(\mathbf{A}) - \alpha_j \mathbf{A}\pi_j^2(\mathbf{A}) \tag{3.35}$$

Si definimos,

$$\mathbf{r}_j = \phi_j^2(\mathbf{A})\mathbf{r}_0 \tag{3.36}$$

$$\mathbf{p}_j = \pi_j^2(\mathbf{A})\mathbf{r}_0 \tag{3.37}$$

$$\mathbf{q}_j = \phi_{j+1}(\mathbf{A})\pi_j(\mathbf{A})\mathbf{r}_0 \tag{3.38}$$

Transformando estas recurrencias de polinomios, obtenemos,

$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\left[2\mathbf{r}_j + 2\beta_{j-1}\mathbf{q}_{j-1} - \alpha_j \mathbf{A}\mathbf{p}_j\right] \tag{3.39}$$

$$\mathbf{q}_j = \mathbf{r}_j + \beta_{j-1}\mathbf{q}_{j-1} - \alpha_j \mathbf{A}\mathbf{p}_j \tag{3.40}$$

$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + 2\beta_j \mathbf{q}_j + \beta_j^2 \mathbf{p}_j \tag{3.41}$$

Definimos los vectores auxiliares,

$$\mathbf{d}_j = 2\mathbf{r}_j + 2\beta_{j-1}\mathbf{q}_{j-1} - \alpha_j \mathbf{A}\mathbf{p}_j \tag{3.42}$$

$$\mathbf{u}_j = \mathbf{r}_j + \beta_{j-1}\mathbf{q}_{j-1} \tag{3.43}$$

Con estas relaciones resulta el siguiente algoritmo en el que no figuran los vectores residuos ni las direcciones conjugadas del sistema auxiliar, eliminando de esta forma los productos $\mathbf{A}^T$ por vector.

ALGORITMO CGS
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{r}_0^*$ arbitrario tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$
$\mathbf{p}_0 = \mathbf{u}_0 = \mathbf{r}_0$
Mientras $\| \mathbf{r}_{j-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \;\; (j = 1, 2, 3, ...)$, hacer

$$\alpha_j = \frac{\langle \mathbf{r}_j, \mathbf{r}_o^* \rangle}{\langle \mathbf{A}\mathbf{p}_j, \mathbf{r}_o^* \rangle}$$
$$\mathbf{q}_j = \mathbf{u}_j - \alpha_j \mathbf{A}\mathbf{p}_j$$
$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j (\mathbf{u}_j + \mathbf{q}_j)$$
$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A} (\mathbf{u}_j + \mathbf{q}_j)$$
$$\beta_j = \frac{\langle \mathbf{r}_{j+1}, \mathbf{r}_0^* \rangle}{\langle \mathbf{r}_j, \mathbf{r}_0^* \rangle}$$
$$\mathbf{u}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{q}_j$$
$$\mathbf{p}_{j+1} = \mathbf{u}_{j+1} + \beta_j (\mathbf{q}_j + \beta_j \mathbf{p}_j)$$

Fin

3.2.4. Método Bi-CGSTAB (*Biconjugate Gradient Stabilized*)

En el CGS, los vectores residuos verifican $\widehat{\mathbf{r}}_j = \phi_j^2(\mathbf{A})\mathbf{r}_0$. Van der Vorst en el Bi-CGSTAB [85, 37], propone obtener un vector residuo por aplicación sucesiva de dos polinomios reductores distintos de la forma,

$$\mathbf{r}_j^{'} = \psi_j(\mathbf{A})\phi_j(\mathbf{A})\mathbf{r}_0 \tag{3.44}$$

tal que las relaciones de recurrencia del algoritmo donde intervenga este polinomio $\psi_j(\mathbf{A})$ no deben ser excesivamente complicadas y los parámetros que figuren en su definición sean fácilmente optimizables. En este sentido sugiere para $\psi_j(\mathbf{A})$ la expresión,

$$\psi_{j+1}(\mathbf{A}) = (\mathbf{I} - w_j \mathbf{A})\psi_j(\mathbf{A}) \tag{3.45}$$

determinando w_j, por la condición de mínimo para $\mathbf{r}_j$ en la iteración j-ésima.

Las relaciones de recurrencia se derivan de forma similar que en el algoritmo CGS. Así,

$$\psi_{j+1}(\mathbf{A})\phi_{j+1}(\mathbf{A}) = (\mathbf{I} - w_j \mathbf{A})\psi_j(\mathbf{A})\phi_{j+1}(\mathbf{A}) \tag{3.46}$$
$$= (\mathbf{I} - w_j \mathbf{A}) [\psi_j(\mathbf{A})\phi_j(\mathbf{A}) - \alpha_j \mathbf{A}\psi_j(\mathbf{A})\pi_j(\mathbf{A})] \tag{3.47}$$

Necesitamos una relación de recurrencia para la correspondiente dirección conjugada. Esto es,

$$\psi_j(\mathbf{A})\pi_j(\mathbf{A}) = \psi_j(\mathbf{A}) [\phi_j(\mathbf{A}) + \beta_{j-1}\pi_{j-1}(\mathbf{A})] \tag{3.48}$$
$$= \psi_j(\mathbf{A})\phi_j(\mathbf{A}) + \beta_{j-1}(\mathbf{I} - w_{j-1}\mathbf{A})\psi_{j-1}(\mathbf{A})\pi_{j-1}(\mathbf{A}) \tag{3.49}$$

Definimos,

$$\mathbf{r}_j = \psi_j(\mathbf{A})\phi_j(\mathbf{A})\mathbf{r}_0,$$

$$\mathbf{p}_j = \psi_j(\mathbf{A})\pi_j(\mathbf{A})\mathbf{r}_0$$

Teniendo en cuenta las anteriores relaciones, podemos encontrar las correspondientes fórmulas de recurrencia en función de los escalares α_j y β_j,

$$\mathbf{r}_{j+1} = (\mathbf{I} - w_j\mathbf{A})\left(\mathbf{r}_j - \alpha_j\mathbf{A}\mathbf{p}_j\right) \tag{3.50}$$

$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j(\mathbf{I} - w_j\mathbf{A})\mathbf{p}_j \tag{3.51}$$

Para obtener el algoritmo, se toma como referencia al igual que en el CGS el algoritmo Bi-CG, efectuando las transformaciones necesarias para que las relaciones de recurrencia de la solución sean función del nuevo vector residuo.

Recordando que en el algoritmo Bi-CG, $\beta_j = \rho_{j+1}/\rho_j$ con,

$$\rho_j = \left\langle \phi_j(\mathbf{A})\mathbf{r}_0, \phi_j(\mathbf{A}^T)\mathbf{r}_0^* \right\rangle = \left\langle \phi_j^2(\mathbf{A})\mathbf{r}_0, \mathbf{r}_0^* \right\rangle$$

Pero como ρ_j no se calcula debido a que ninguno de los vectores $\phi_j(\mathbf{A})\mathbf{r}_0$, $\phi_j(\mathbf{A}^T)\mathbf{r}_0^*$ ó $\phi_j^2(\mathbf{A})\mathbf{r}_0$ están disponibles, lo podemos relacionar con el escalar,

$$\widetilde{\rho}_j = \left\langle \phi_j(\mathbf{A})\mathbf{r}_0, \psi_j(\mathbf{A}^T)\mathbf{r}_0^* \right\rangle$$

que podemos obtener como,

$$\widetilde{\rho}_j = \langle \psi_j(\mathbf{A})\phi_j(\mathbf{A})\mathbf{r}_0, \mathbf{r}_0^* \rangle = \langle \mathbf{r}_j, \mathbf{r}_0^* \rangle$$

Desarrollando $\phi_j(\mathbf{A}^T)\mathbf{r}_0^*$ explícitamente para relacionar los escalares, ρ_j y $\widetilde{\rho}_j$,

$$\widetilde{\rho}_j = \left\langle \phi_j(\mathbf{A})\mathbf{r}_0, \eta_1^j(\mathbf{A}^T)^j\mathbf{r}_0^* + \eta_2^j(\mathbf{A}^T)^{j-1}\mathbf{r}_0^* + \ldots \right\rangle$$

incorporando la ortogonalidad de $\phi_j(\mathbf{A})\mathbf{r}_0$ respecto de cada uno de los vectores $(\mathbf{A}^T)^i\mathbf{r}_0^*$, con $i < j$ y teniendo en cuenta que sólo los coeficientes principales de los polinomios son relevantes en el desarrollo del anterior producto. Si γ_1^j es el principal coeficiente para el polinomio $\phi_j(\mathbf{A})$, entonces,

$$\widetilde{\rho}_j = \left\langle \phi_j(\mathbf{A})\mathbf{r}_0, \frac{\eta_1^j}{\gamma_1^j}\phi_j(\mathbf{A}^T)\mathbf{r}_0^* \right\rangle = \frac{\eta_1^j}{\gamma_1^j}\rho_j$$

Examinando las relaciones de recurrencia para $\phi_{j+1}(\mathbf{A})$ y $\psi_{j+1}(\mathbf{A})$, los coeficientes principales para estos polinomios fueron hallados para satisfacer las relaciones,

$$\eta_1^{j+1} = -w_j\eta_1^j, \quad \gamma_1^{j+1} = -\alpha_j\gamma_1^j$$

Por tanto,

$$\frac{\widetilde{\rho}_{j+1}}{\widetilde{\rho}_j} = \frac{w_j}{\alpha_j}\frac{\rho_{j+1}}{\rho_j}$$

que nos permite encontrar la siguiente relación para β_j,

$$\beta_j = \frac{\widetilde{\rho}_{j+1}}{\widetilde{\rho}_j}\frac{\alpha_j}{w_j}$$

De forma similar, y por una simple fórmula de recurrencia podemos encontrar α_j como,

$$\alpha_j = \frac{\left\langle \phi_j(\mathbf{A})\mathbf{r}_0, \phi_j(\mathbf{A}^T)\mathbf{r}_0^* \right\rangle}{\langle \mathbf{A}\pi_j(\mathbf{A})\mathbf{r}_0, \pi_j(\mathbf{A}^T)\mathbf{r}_0^* \rangle}$$

De la misma manera que anteriormente, en los productos de polinomios, tanto en el numerador como en el denominador, sólo se consideran los términos correpondientes a sus respectivos coeficientes principales y como éstos para $\phi_j(\mathbf{A}^T)\mathbf{r}_0^*$ y $\pi_j(\mathbf{A}^T)\mathbf{r}_0^*$ son idénticos, podemos escribir,

$$\begin{aligned}
\alpha_j &= \frac{\left\langle \phi_j(\mathbf{A})\mathbf{r}_0, \phi_j(\mathbf{A}^T)\mathbf{r}_0^* \right\rangle}{\langle \mathbf{A}\pi_j(\mathbf{A})\mathbf{r}_0, \phi_j(\mathbf{A}^T)\mathbf{r}_0^* \rangle} \\
&= \frac{\left\langle \phi_j(\mathbf{A})\mathbf{r}_0, \psi_j(\mathbf{A}^T)\mathbf{r}_0^* \right\rangle}{\langle \mathbf{A}\pi_j(\mathbf{A})\mathbf{r}_0, \psi_j(\mathbf{A}^T)\mathbf{r}_0^* \rangle} \\
&= \frac{\langle \psi_j(\mathbf{A})\phi_j(\mathbf{A})\mathbf{r}_0, \mathbf{r}_0^* \rangle}{\langle \mathbf{A}\psi_j(\mathbf{A})\pi_j(\mathbf{A})\mathbf{r}_0, \mathbf{r}_0^* \rangle}
\end{aligned}$$

Y como $\mathbf{p}_j = \psi_j(\mathbf{A})\pi_j(\mathbf{A})\mathbf{r}_0$, entonces,

$$\alpha_j = \frac{\widetilde{\rho}_j}{\left\langle \mathbf{A}\mathbf{p}_j, \mathbf{r}_0^* \right\rangle} \tag{3.52}$$

A partir de la ecuación (3.50), si hacemos,

$$\mathbf{s}_j = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j \tag{3.53}$$

el valor optimo para el parámetro w_j que interviene en la construcción del polinomio reductor $\psi_j(\mathbf{A})$ y que figura en las relaciones de recurrencia del algoritmo lo obtendremos con la condición de minimizar la norma del vector residuo,

$$\mathbf{r}_{j+1} = \mathbf{s}_j - w_j \mathbf{A}\mathbf{s}_j$$

$$\|\mathbf{r}_{j+1}\|^2 = \langle \mathbf{s}_j - w_j\mathbf{A}\mathbf{s}_j, \mathbf{s}_j - w_j\mathbf{A}\mathbf{s}_j \rangle = \langle \mathbf{s}_j, \mathbf{s}_j \rangle - 2w_j \langle \mathbf{s}_j, \mathbf{A}\mathbf{s}_j \rangle + w_j^2 \langle \mathbf{A}\mathbf{s}_j, \mathbf{A}\mathbf{s}_j \rangle$$

$$\frac{\partial \|\mathbf{r}_{j+1}\|^2}{\partial w_j} = -2 \langle \mathbf{s}_j, \mathbf{A}\mathbf{s}_j \rangle + 2w_j \langle \mathbf{A}\mathbf{s}_j, \mathbf{A}\mathbf{s}_j \rangle = 0$$

con lo que,

$$w_j = \frac{\left\langle \mathbf{A}\mathbf{s}_j, \mathbf{s}_j \right\rangle}{\left\langle \mathbf{A}\mathbf{s}_j, \mathbf{A}\mathbf{s}_j \right\rangle} \tag{3.54}$$

La ecuación (3.50) ahora nos quedará,

$$\mathbf{r}_{j+1} = \mathbf{s}_j - w_j \mathbf{A}\mathbf{s}_j = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j - w_j \mathbf{A}\mathbf{s}_j \tag{3.55}$$

y la relación de recurrencia para el vector solución viene dada por,

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j + w_j \mathbf{s}_j \tag{3.56}$$

Una vez expresados todos los vectores en función del nuevo residuo y determinadas sus relaciones de recurrencia, así como los escalares que intervienen en las mismas, se puede escribir el algoritmo.

ALGORITMO BI-CGSTAB
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Elegir $\mathbf{r}_0^*$ arbitrario tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$.
$\mathbf{p}_0 = \mathbf{r}_0$
Mientras $\| \mathbf{r}_{j-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (j = 1, 2, 3, ...)$, hacer

$$\alpha_j = \frac{\langle \mathbf{r}_j, \mathbf{r}_o^* \rangle}{\langle \mathbf{A}\mathbf{p}_j, \mathbf{r}_o^* \rangle}$$
$$\mathbf{s}_j = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$$
$$w_j = \frac{\langle \mathbf{A}\mathbf{s}_j, \mathbf{s}_j \rangle}{\langle \mathbf{A}\mathbf{s}_j, \mathbf{A}\mathbf{s}_j \rangle}$$
$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j + w_j \mathbf{s}_j$$
$$\mathbf{r}_{j+1} = \mathbf{s}_j - w_j \mathbf{A}\mathbf{s}_j$$
$$\beta_j = \frac{\langle \mathbf{r}_{j+1}, \mathbf{r}_0^* \rangle}{\langle \mathbf{r}_j, \mathbf{r}_0^* \rangle} \frac{\alpha_j}{w_j}$$
$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \left(\mathbf{p}_j - w_j \mathbf{A}\mathbf{p}_j \right)$$

Fin

En el algoritmo figuran dos productos matriz por vector y cuatro productos internos, mientras que el CGS exige los mismos productos matriz por vector y sólo dos productos internos. Sin embargo en la mayoría de los casos la convergencia del Bi-CGSTAB es más rápida y uniforme e incluso necesita menor carga computacional para alcanzar una determinada tolerancia, pues la reducción del número de iteraciones compensa su mayor coste.

3.2.5. Método de Cuasi-mínimo Residuo (QMR)

Este método propuesto por Freund y Nachtigal [21, 22, 23], conserva la propiedad de los métodos tipo gradiente de utilizar relaciones de recurrencia que impliquen bajo coste computacional, con la particularidad de no usar una condición estricta de minimización para generar los vectores residuo. Plantea para ello, una técnica para cuasi-minimizar estos vectores, condición que da nombre al método, intentando suavizar las fluctuaciones que tienen lugar en la convergencia debidas a este hecho.

El resultado del algoritmo de Lanczos es una relación de la forma,

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_{k+1}\overline{\mathbf{T}}_k \tag{3.57}$$

donde $\overline{\mathbf{T}}_k$ es una matriz tridiagonal $(k+1) \times k$,

$$\overline{\mathbf{T}}_k = \begin{pmatrix} \mathbf{T}_k \\ \delta_{k+1}\mathbf{e}_k^t \end{pmatrix} \tag{3.58}$$

con,

$$\mathbf{T}_k = \begin{pmatrix} \alpha_1 & \beta_2 & & . & & & \\ \delta_2 & \alpha_2 & \beta_3 & . & & & \\ & \delta_3 & \alpha_3 & . & & & \\ . & . & . & . & . & . & . \\ & & & . & \alpha_{k-2} & \beta_{k-1} & \\ & & & . & \delta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & . & & \delta_k & \alpha_k \end{pmatrix} \tag{3.59}$$

α_i, $i = 1, ..., k$; β_j, $j = 2, ..., k$; δ_l, $l = 2, ..., k+1$, son los parámetros obtenidos durante el proceso de Lanczos.

A partir de la ecuación (3.57) y desarrollando de la misma forma que en el GMRES, si $\mathbf{v}_1$ se define en función de $\mathbf{r}_0$, por ejemplo $\gamma\mathbf{v}_1 = \mathbf{r}_0$, entonces el vector residuo asociado a una solución aproximada de la forma,

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\,\mathbf{z} \tag{3.60}$$

viene dado por,

$$\begin{aligned} \mathbf{b} - \mathbf{A}\mathbf{x}_k &= \mathbf{b} - \mathbf{A}\left(\mathbf{x}_0 + \mathbf{V}_k\,\mathbf{z}\right) \\ &= \mathbf{r}_0 - \mathbf{A}\mathbf{V}_k\,\mathbf{z} \\ &= \gamma\mathbf{v}_1 - \mathbf{V}_{k+1}\overline{\mathbf{T}}_k\mathbf{z} \\ &= \mathbf{V}_{k+1}\left(\gamma\mathbf{e}_1 - \overline{\mathbf{T}}_k\,\mathbf{z}\right) \end{aligned} \tag{3.61}$$

La norma, del vector residuo viene dada por,

$$\|\,\mathbf{r}\|_2 = \left\|\mathbf{V}_{k+1}\left(\gamma\mathbf{e}_1 - \overline{\mathbf{T}}_k\,\mathbf{z}\right)\right\|_2 \tag{3.62}$$

donde la matriz $\mathbf{V}_{k+1}$ tiene por columnas los vectores $\mathbf{v}_i$, $i = 1, ..., k+1$, obtenidos en el procedimiento de biortogonalización de Lanczos, pero no son ortonormados como los obtenidos en el proceso de Arnoldi utilizado para el GMRES, por lo que en este caso ocurre que,

$$\|\,\mathbf{r}\|_2 \neq \left\|\gamma\mathbf{e}_1 - \overline{\mathbf{T}}_k\,\mathbf{z}\right\|_2 \tag{3.63}$$

La idea del QMR es precisamente minimizar sólo este factor del residuo, y es por ello que la solución resultante es llamada *"Aproximación del cuasi-mínimo residuo"*.

Si introducimos una matriz diagonal $(k+1)\times(k+1)$ de pesos,

$$\mathbf{\Omega}_k = diag\left(\omega_1, \omega_2, \ldots, \omega_{k+1}\right), \omega_j > 0, j = 1, 2, \ldots, k+1$$

que se considera como un parámetro libre que puede ser usado para escalar el problema, entonces podemos escribir la ecuación (3.61) en la forma,

$$\begin{aligned} \mathbf{r}_k &= \mathbf{V}_{k+1}\left[\mathbf{\Omega}_k\right]^{-1}\mathbf{\Omega}_k\left(\gamma\mathbf{e}_1 - \overline{\mathbf{T}}_k\,\mathbf{z}\right) \\ &= \mathbf{V}_{k+1}\left[\mathbf{\Omega}_k\right]^{-1}\left(\mathbf{d}_k - \mathbf{\Omega}_k\overline{\mathbf{T}}_k\mathbf{z}\right) \end{aligned} \tag{3.64}$$

con $\mathbf{d}_k = \omega_1\gamma\mathbf{e}_1$.

El problema es ahora minimizar la funcional cuadrática,

$$\left\|\mathbf{d}_k - \mathbf{\Omega}_k\overline{\mathbf{T}}_k\mathbf{z}\right\|_2 \tag{3.65}$$

Este problema de mínimos cuadrados implica para su resolución, la factorización QR de $\mathbf{\Omega}_k\overline{\mathbf{T}}_k$ como,

$$\mathbf{\Omega}_k\overline{\mathbf{T}}_k = \left[\mathbf{Q}_k\right]^T\begin{bmatrix}\mathbf{R}_k\\ \mathbf{0}\end{bmatrix}$$

donde $\mathbf{Q}_k$ es una matriz ortogonal $(k+1)\times(k+1)$ y $\mathbf{R}_k$ es una matriz no-singular, $k\times k$ triangular superior.

El problema de minimización expresado en la ecuación (3.65) se transforma en,

$$\begin{aligned} \text{mín}\left\|\left(\mathbf{d}_k - \mathbf{\Omega}_k\overline{\mathbf{T}}_k\mathbf{z}\right)\right\|_2 &= \text{mín}\left\|\left[\mathbf{Q}_k\right]^T\left(\mathbf{Q}_k\mathbf{d}_k - \begin{bmatrix}\mathbf{R}_k\\ \mathbf{0}\end{bmatrix}\mathbf{z}\right)\right\|_2 = \\ &= \text{mín}\left\|\left(\mathbf{Q}_k\mathbf{d}_k - \begin{bmatrix}\mathbf{R}_k\\ \mathbf{0}\end{bmatrix}\mathbf{z}\right)\right\|_2 \end{aligned} \tag{3.66}$$

que permite obtener $\mathbf{z}_k$, el valor óptimo de z, como,

$$\mathbf{z}_k = \left[\mathbf{R}_k\right]^{-1}\mathbf{t}_k$$

donde,

$$\mathbf{t}_k = \begin{bmatrix}\tau_1\\ \vdots\\ \tau_k\end{bmatrix}, \quad \begin{bmatrix}\mathbf{t}_k\\ \widetilde{\tau}_{k+1}\end{bmatrix} = \mathbf{Q}_k\mathbf{d}_k \tag{3.67}$$

Y por tanto,

$$\text{mín}\left\|\left(\mathbf{d}_k - \mathbf{\Omega}_k\overline{\mathbf{T}}_k\mathbf{z}_k\right)\right\|_2 = |\widetilde{\tau}_{k+1}| \tag{3.68}$$

Con lo que el algoritmo QMR puede escribirse,

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\gamma = \| \mathbf{r}_0 \|$;
Inciar con $\mathbf{v}_1 = \mathbf{r}_0/\gamma$, y con $\mathbf{w}_1$ tal que $\| \mathbf{w}_1 \| = 1$;

Para $k = 1, 2, \ldots$:
1) Ejecutar las k iteraciones del algoritmo de biortogonalización de Lanczos; Generar las matrices $\mathbf{V}_k$, $\mathbf{V}_{k+1}$ y $\overline{\mathbf{T}}_k$
2) Calcular la factorización QR de la matriz $\mathbf{\Omega}_k\overline{\mathbf{T}}_k$ y el vector $\mathbf{t}_k$
3) Resolver $\mathbf{R}_k\mathbf{z}_k = \mathbf{t}_k$ y calcular $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\mathbf{z}_k$;
4) Si $\mathbf{x}_k$ converge, Fin.
Que desarrollado queda,

ALGORITMO QMR
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\rho_1 = \| \mathbf{r}_0 \|$ y ; $\mathbf{v}_1 = \mathbf{r}_0/\rho_1$;
Elegir $\mathbf{w}_1$ tal que $\| \mathbf{w}_1 \| = 1$;
$\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{d}_0 = 0$;
$c_0 = \epsilon_0 = \xi_1 = 1$;
$\nu_0 = 0$, $\eta_0 = -1$;
Mientras $\sqrt{j+1}\,|\eta_{j-1}| \,/\, \|\mathbf{r}_0\| \geq \varepsilon (j = 1, 2, ...)$, hacer:

$\delta_j = \langle \mathbf{v}_j, \mathbf{w}_j \rangle$,
$\mathbf{p}_j = \mathbf{v}_j - (\xi_j\delta_j/\epsilon_{j-1})\,\mathbf{p}_{j-1}$;
$\mathbf{q}_j = \mathbf{w}_j - (\rho_j\delta_j/\epsilon_{j-1})\,\mathbf{q}_{j-1}$,
$\epsilon_j = \langle \mathbf{A}\mathbf{p}_j, \mathbf{q}_j \rangle$;
$\beta_j = \epsilon_j/\delta_j$;
$\widehat{\mathbf{v}}_{j+1} = \mathbf{A}\mathbf{p}_j - \beta_j\mathbf{v}_j$, $\rho_{j+1} = \|\widehat{\mathbf{v}}_{j+1}\|$;
$\widehat{\mathbf{w}}_{j+1} = \mathbf{A}^T\mathbf{q}_j - \beta_j\mathbf{w}_j$, $\xi_{j+1} = \|\widehat{\mathbf{w}}_{j+1}\|$;
$\nu_j = \dfrac{\rho_{j+1}}{c_{j-1}\,|\beta_j|}$, $c_j = \dfrac{1}{\sqrt{1+\nu_j^2}}$, $\eta_j = -\eta_{j-1}\dfrac{\rho_j c_j^2}{\beta_j c_{j-1}^2}$;
$\mathbf{d}_j = \eta_j\mathbf{p}_j + (\nu_{j-1}c_j)^2\,\mathbf{d}_{j-1}$;
$\mathbf{x}_j = \mathbf{x}_{j-1} + \mathbf{d}_j$;
$\mathbf{v}_{j+1} = \widehat{\mathbf{v}}_{j+1}/\rho_{j+1}$;
$\mathbf{w}_{j+1} = \widehat{\mathbf{w}}_{j+1}/\xi_{j+1}$;

Fin

3.2.5.1. Método QMR con *Look-Ahead Lanczos*

Procediendo de la misma forma que ya se hizo en el apartado anterior, a partir de la ecuación (2.24), el algoritmo QMR usando la biortogonalización obtenida con la variante "*look-ahead*" del método de Lanczos, puede escribirse,

ALGORITMO QMR CON LAL
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\rho_0 = \| \mathbf{r}_0 \|$, $\mathbf{v}_1 = \mathbf{r}_0/\,\rho_0$;
Iniciar con $\mathbf{v}_1 = \mathbf{r}_0/\gamma_0$, y con $\mathbf{w}_1$ tal que $\| \mathbf{w}_1 \| = 1$
Para $k = 1, 2, \ldots$:
1) Ejecutar las k iteraciones del algoritmo LAL;
Generar las matrices $\mathbf{V}_k$, $\mathbf{V}_{k+1}$ y $\overline{\mathbf{T}}_k$

2) Calcular la factorización QR de la matriz $\mathbf{\Omega}_k\overline{\mathbf{T}}_k$ y el vector $\mathbf{t}_k$
3) Resolver $\mathbf{R}_k\mathbf{z}_k = \mathbf{t}_k$ y calcular $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\mathbf{z}_k$;
4) Si $\mathbf{x}_k$ converge, Fin.

3.2.6. Método TFQMR (*Transpose-Free QMR*)

Derivado del algoritmo CGS y desarrollado por Freund en 1993 [20], tiene la ventaja de no requerir productos matriz por vector con $\mathbf{A}^T$ y puede implementarse fácilmente, simplemente cambiando unas pocas líneas en el algoritmo CGS visto anteriormente. Si dividimos en dos medios pasos el cáculo de $\mathbf{x}_j$ en el algoritmo haciendo,

$$\mathbf{y}_m = \begin{cases} \mathbf{u}_{k-1} \text{ si } m = 2k-1 \text{ es impar} \\ \mathbf{q}_k \text{ si } m = 2k \text{ es par} \end{cases} \tag{3.69}$$

y,

$$\mathbf{w}_m = \begin{cases} \phi_k^2(\mathbf{A})\mathbf{r}_0 \text{ si } m = 2k+1 \text{ es impar} \\ \phi_k(\mathbf{A})\phi_{k-1}(\mathbf{A})\mathbf{r}_0 \text{ si } m = 2k \text{ es par} \end{cases} \tag{3.70}$$

siendo,

$$\mathbf{w}_{2k+1} = \mathbf{r}_k^{CGS}, \qquad k = 1, 2, \ldots, k_* \tag{3.71}$$

por lo que éste se verificará siempre que $m \in \{1, 2, \ldots, 2k_*\}$.

Teniendo en cuenta las expresiones dadas en el algoritmo CGS, (3.40), (3.43) y (3.31), se podrán relacionar los vectores $\mathbf{y}_m$ y $\mathbf{w}_m$ por,

$$\mathbf{A}\mathbf{y}_m = \frac{1}{\alpha_{[(m-1)/2]}}\left(\mathbf{w}_m - \mathbf{w}_{m+1}\right) \tag{3.72}$$

Y como en el algoritmo CGS, $\alpha_{k-1} \neq 0$ para todo k, queda garantizado que el denominador de la expresión anterior (3.72) será siempre distinto de cero.

Poniendo,

$$\mathbf{Y}_m = [\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_m] \ ,$$
$$\mathbf{W}_{m+1} = [\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_m, \mathbf{w}_{m+1}]$$

se puede expresar (3.72) en forma matricial,

$$\mathbf{A}\mathbf{Y}_m = \mathbf{W}_{m+1}\overline{\mathbf{B}}_m \tag{3.73}$$

donde,

$$\overline{\mathbf{B}}_m = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 1 \\ 0 & \cdots & 0 & -1 \end{pmatrix} \left[diag\left(\alpha_0, \alpha_0, \alpha_1, \ldots, \alpha_{[(m-1)/2]}\right)\right]^{-1} \tag{3.74}$$

es una matriz $(m+1) \times m$ bidiagonal inferior. Finalmente teniendo en cuenta que en las expresiones (3.31) y (3.33), los polinomios ϕ_k y π_k son de grado k y por (3.40), (3.43) y (3.69),

$$\mathcal{K}_m\left(\mathbf{A}; \mathbf{r}_0\right) = C.L.\left\{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_m\right\} = \left\{\mathbf{Y}_m \mathbf{z}\right\} \tag{3.75}$$

por lo que podremos expresar una aproximación a la solución del sistema por,

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{Y}_m \mathbf{z} \tag{3.76}$$

Por (3.73) y como $\mathbf{w}_1 = \mathbf{r}_0$, los correspondientes vectores residuos satisfacen,

$$\mathbf{r}_m = \mathbf{r}_0 - \mathbf{A}\mathbf{Y}_m \mathbf{z} = \mathbf{W}_{m+1}\left(\mathbf{e}_1^{m+1} - \overline{\mathbf{B}}_m \mathbf{z}\right) \tag{3.77}$$

donde $\mathbf{e}_1^{m+1} = (1 \quad 0 \quad \cdots \quad 0)^T \in \Re^{m+1}$

Introduciendo una matriz de escala,

$$\mathbf{\Omega}_{m+1} = diag\left(\omega_1, \omega_2, \ldots, \omega_{m+1}\right), \qquad \omega_N > 0, \qquad N = 1, 2, \ldots, m+1 \tag{3.78}$$

y reescribiendo la ecuación (3.77),

$$\mathbf{r}_k = \mathbf{W}_{m+1}\mathbf{\Omega}_{m+1}^{-1}\left(\mathbf{f}_{m+1} - \overline{\mathbf{T}}_m \mathbf{z}\right) \tag{3.79}$$

donde,

$$\begin{aligned} \mathbf{f}_{m+1} &= \omega_1 \mathbf{e}_1^{m+1} \qquad \text{y} \\ \overline{\mathbf{T}}_m &= \mathbf{\Omega}_{m+1}\overline{\mathbf{B}}_m \end{aligned} \tag{3.80}$$

Las sucesivas iteraciones del TFQMR, vienen dadas por,

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{Y}_m \mathbf{z}_m \tag{3.81}$$

donde $\mathbf{z}_m$ es la solución del problema de mínimos cuadrados,

$$\tau_m = \text{mín}\left\|\left(\mathbf{f}_{m+1} - \overline{\mathbf{T}}_m \mathbf{z}\right)\right\| \tag{3.82}$$

Las iteraciones $\mathbf{x}_m$ dependen de la elección de los pesos ω_N en (3.78). La estrategia estándar es tomar,

$$\omega_N = \left\|\mathbf{w}_N\right\|, \qquad N = 1, 2, \ldots, m+1 \tag{3.83}$$

lo que significa que todas las columnas de $\mathbf{W}_{m+1}\mathbf{\Omega}_{m+1}^{-1}$ en la ecuación (3.79) son tratadas igualmente y escaladas para ser vectores unitarios.

Para la implementación del método TFQMR, las iteraciones auxiliares serán,

$$\widetilde{\mathbf{x}}_m = \mathbf{x}_0 + \mathbf{Y}_m \widetilde{\mathbf{z}}_m \qquad \text{donde} \qquad \widetilde{\mathbf{z}}_m = \overline{\mathbf{T}}_m^{-1}\mathbf{f}_m \tag{3.84}$$

siendo $\mathbf{T}_m$ la matriz $m \times m$ que se obtiene al eliminar la última fila de $\overline{\mathbf{T}}_m$. Usando (3.74), (3.77) y (3.80) se comprueba que $\mathbf{T}_m$ es no-singular y que,

$$\widetilde{\mathbf{z}}_m = \left(\alpha_0, \alpha_0, \alpha_1, \ldots, \alpha_{[(m-1)/2]}\right)^T, \tag{3.85}$$

$$\omega_{m+1} = \left\| \left(\mathbf{f}_{m+1} - \overline{\mathbf{T}}_m \mathbf{z}\right) \right\| \tag{3.86}$$

Finalmente comparando (3.84) y (3.85) con la expresión para $\mathbf{x}_k^{CGS}$ en el algoritmo CGS se concluye que,

$$\widetilde{\mathbf{x}}_{2k} = \mathbf{x}_k^{CGS} \tag{3.87}$$

De la resolución del problema de mínimos cuadrados planteado en (3.82) (ver [20]) las iteraciones para el TFQMR dadas en (3.81) y (3.84) quedan relacionadas por,

$$\mathbf{x}_m = \left(1 - c_m^2\right) \mathbf{x}_{m-1} + c_m^2 \widetilde{\mathbf{x}}_m \tag{3.88}$$

con,

$$\vartheta_m = \frac{\omega_{m+1}}{\tau_{m-1}}, \qquad c_m = \frac{1}{\sqrt{1+\vartheta_m^2}} \qquad \text{y} \qquad \tau_m = \tau_{m-1}\vartheta_m c_m \tag{3.89}$$

Haciendo,

$$\mathbf{d}_m = \frac{1}{\alpha_{[(m-1)/2]}} \left(\widetilde{\mathbf{x}}_m - \mathbf{x}_{m-1}\right) \tag{3.90}$$

podemos reescribir (3.88) en la forma,

$$\mathbf{x}_m = \mathbf{x}_{m-1} + \eta_m \mathbf{d}_m, \qquad \text{donde} \qquad \eta_m = c_m^2 \alpha_{[(m-1)/2]} \tag{3.91}$$

De (3.84) y (3.85) obtenemos que,

$$\widetilde{\mathbf{x}}_m = \widetilde{\mathbf{x}}_{m-1} + \alpha_{[(m-1)/2]} \mathbf{y}_m$$

y junto con (3.90) y (3.91) (reemplazando m por $m-1$) se llega a que,

$$\mathbf{d}_m = \mathbf{y}_m + \frac{\vartheta_{m-1}^2 \eta_{m-1}}{\alpha_{[(m-1)/2]}} \mathbf{d}_{m-1}, \qquad \text{donde} \qquad \vartheta_{m-1}^2 = \frac{1 - c_{m-1}^2}{c_{m-1}^2} \tag{3.92}$$

Teniendo en cuenta que por (3.69) y (3.71) las recurrencias para $\mathbf{q}_k$ y $\mathbf{u}_k$ en el algoritmo CGS pueden reescribirse como,

$$\mathbf{y}_{2k} = \mathbf{y}_{2k-1} - \alpha_{k-1} \mathbf{v}_{k-1} \tag{3.93}$$

$$\mathbf{y}_{2k-1} = \mathbf{w}_{2k-1} + \beta_k \mathbf{y}_{2k} \tag{3.94}$$

y multiplicando la expresión para $\mathbf{p}_k$ en el algoritmo CGS por $\mathbf{A}$ obtenemos la expresión recurrente,

$$\mathbf{v}_k = \mathbf{A}\mathbf{y}_{2k+1} + \beta_k \left(\mathbf{A}\mathbf{y}_{2k} + \beta_k \mathbf{v}_{k-1}\right) \tag{3.95}$$

para $\mathbf{v}_k = \mathbf{A}\mathbf{p}_k$.

Por (3.72), los vectores $\mathbf{w}_m$ son generados como,

$$\mathbf{w}_{m+1} = \mathbf{w}_m - \alpha_{[(m-1)/2]}\mathbf{A}\mathbf{y}_m \tag{3.96}$$

ALGORITMO TFQMR

Aproximación inicial $\mathbf{x}_0$. ;
$\mathbf{w}_1 = \mathbf{y}_1 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v}_0 = \mathbf{A}\mathbf{y}_1$, $\mathbf{d}_0 = \mathbf{0}$;
$\tau_0 = \|\mathbf{r}_0\|$, $\vartheta_0 = 0$, $\eta_0 = 0$;
Elegir $\widetilde{\mathbf{r}}_0$ tal que $\rho_0 = \langle \mathbf{r}_0, \widetilde{\mathbf{r}}_0 \rangle \neq 0$
Mientras $\sqrt{k+1}\tau_{k-1} / \|\mathbf{r}_0\| \geq \varepsilon (k = 1, 2, ...)$, hacer:

(a) $\sigma_{k-1} = \langle \mathbf{v}_{k-1}, \widetilde{\mathbf{r}}_0 \rangle$, $\alpha_{k-1} = \rho_{k-1}/\sigma_{k-1}$;
$\mathbf{y}_{2k} = \mathbf{y}_{2k-1} - \alpha_{k-1}\mathbf{v}_{k-1}$;

(b) Para $m = 2k-1$,$2k$ hacer:
$\mathbf{w}_{m+1} = \mathbf{w}_m - \alpha_{k-1}\mathbf{A}\mathbf{y}_m$;
$\vartheta_m = \dfrac{\|\mathbf{w}_{m+1}\|}{\tau_{m-1}}$, $c_m = \dfrac{1}{\sqrt{1+\vartheta_m^2}}$;
$\tau_m = \tau_{m-1}\vartheta_m c_m$, $\eta_m = c_m^2 \alpha_{k-1}$;
$\mathbf{d}_m = \mathbf{y}_m + \dfrac{\vartheta_{m-1}^2 \eta_{m-1}}{\alpha_{k-1}}\mathbf{d}_{m-1}$;
$\mathbf{x}_m = \mathbf{x}_{m-1} + \eta_m \mathbf{d}_m$;
Si $\mathbf{x}_m$ converge: Fin

(c) $\rho_k = \langle \mathbf{w}_{2k+1}, \widetilde{\mathbf{r}}_0 \rangle$, $\beta_k = \dfrac{\rho_k}{\rho_{k-1}}$;
$\mathbf{y}_{2k+1} = \mathbf{w}_{2k+1} + \beta_k \mathbf{y}_{2k}$;
$\mathbf{v}_k = \mathbf{A}\mathbf{y}_{2k+1} + \beta_k \left(\mathbf{A}\mathbf{y}_{2k} + \beta_k \mathbf{v}_{k-1}\right)$

Fin

El test de convergencia en el paso (b) del algoritmo anterior está generalmente basado en la norma del vector residuo $\|\mathbf{r}_m\|$ correspondiente a $\mathbf{x}_m$. Estos vectores no son generados explícitamente en el algoritmo, no obstante, es posible acotarlo superiormente sin un coste extra de la siguiente forma,

$$\|\mathbf{r}_m\| \leq \left\|\mathbf{W}_{m+1}\mathbf{\Omega}_{m+1}^{-1}\right\| \cdot \left\|\mathbf{f}_{m+1} - \overline{\mathbf{T}}_m \mathbf{z}_m\right\| \leq \sqrt{m+1}\tau_m \tag{3.97}$$

3.2.7. Método QMRCGSTAB

Desarrollado por Chan y otros [6] está basado en la aplicación del principio de minimización usado en el algoritmo Bi-CGSTAB al método QMR, de la misma forma que el TFQMR es derivado del CGS.

Sea,

$$\mathbf{Y}_k = [\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_k] \ ,$$
$$\mathbf{W}_{k+1} = [\mathbf{w}_0, \mathbf{w}_1, \ldots \mathbf{w}_k]$$

tal que,

$$\begin{cases} \mathbf{y}_{2l-1} = \mathbf{p}_l \quad \text{para } \ l = 1, \ldots, [(k+1)/2] \\ \mathbf{y}_{2l} = \mathbf{s}_l \quad \text{para } \ l = 1, \ldots, [k/2] \end{cases}$$

y,

$$\begin{cases} \mathbf{w}_{2l-1} = \mathbf{s}_l \quad \text{para } \ l = 1, \ldots, [(k+1)/2] \\ \mathbf{w}_{2l} = \mathbf{r}_l \quad \text{para } \ l = 0, 1, \ldots, [k/2] \end{cases}$$

donde $[k/2]$ y $[(k+1)/2]$ son la parte entera de $k/2$ y $(k+1)/2$ respectivamente.

Definiendo,

$$[\delta_1, \delta_2, \ldots \delta_k] \,/ \begin{cases} \delta_{2l} = \omega_l \quad \text{para } \ l = 1, \ldots, [(k+1)/2] \\ \delta_{2l-1} = \alpha_l \quad \text{para } \ l = 1, \ldots, [(k+1)/2] \end{cases}$$

Entonces, para cada columna de $\mathbf{W}_{k+1}$ y $\mathbf{Y}_k$ las expresiones (3.53) y (3.55) pueden escribirse como,

$$\mathbf{A}\mathbf{y}_m = (\mathbf{w}_{m-1} - \mathbf{w}_m)\, \delta_m^{-1}, \qquad m = 1, \ldots, k \tag{3.98}$$

o usando notación matricial,

$$\mathbf{A}\mathbf{Y}_k = \mathbf{W}_{k+1}\mathbf{E}_{k+1}$$

donde $\mathbf{E}_{k+1}$ es una matriz bidiagonal $(k+1) \times k$ con elementos diagonales δ_m^{-1} y en la diagonal inferior $-\delta_m^{-1}$.

Esto puede ser facilmente comprobado hasta que el grado de los polinomios correspondientes a los vectores $\mathbf{r}_j$, $\mathbf{s}_j$ y $\mathbf{p}_j$ sean $2j$, $2j-1$, y $2j-2$ respectivamente. Entonces, $\mathbf{Y}_k$ y $\mathbf{W}_k$ generarán el mismo subespacio de Krylov generado por $\mathbf{r}_0$ pero de grado $k-1$.

La idea principal del QMRCGSTAB es encontrar una aproximación a la solución del sistema (2.1) usando el subespacio de Krylov $\mathcal{K}_{k-1}$ en la forma,

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{Y}_k\mathbf{g} \qquad \text{con} \qquad \mathbf{g} \in \Re^n$$

la expresión para el vector residuo queda,

$$\mathbf{r}_k = \mathbf{r}_0 - \mathbf{A}\mathbf{Y}_k\mathbf{g} = \mathbf{r}_0 - \mathbf{W}_{k+1}\mathbf{E}_{k+1}\mathbf{g}$$

Teniendo en cuenta el hecho de que el primer vector de $\mathbf{W}_{k+1}$ es justamente $\mathbf{r}_0$, entonces,

$$\mathbf{r}_k = \mathbf{W}_{k+1}\,(\mathbf{e}_1 - \mathbf{E}_{k+1}\mathbf{g})$$

Como las columnas de $\mathbf{W}_{k+1}$ no están normalizadas, se usará una matriz de escala $\mathbf{\Sigma}_{k+1} = diag\,(\sigma_1, \ldots, \sigma_{k+1})$ con $\sigma_j = \|\mathbf{w}_j\|$ para hacer unitarias las columnas de $\mathbf{W}_{k+1}$. Entonces,

$$\mathbf{r}_k = \mathbf{W}_{k+1}\mathbf{\Sigma}_{k+1}^{-1}\mathbf{\Sigma}_{k+1}\,(\mathbf{e}_1 - \mathbf{E}_{k+1}\mathbf{g}) = \mathbf{W}_{k+1}\mathbf{\Sigma}_{k+1}^{-1}\,(\sigma_1\mathbf{e}_1 - \mathbf{H}_{k+1}\mathbf{g})$$

con $\mathbf{H}_{k+1} = \mathbf{\Sigma}_{k+1}\mathbf{E}_{k+1}$.

La aproximación QMR consiste en la minimización de $\|\sigma_1 \mathbf{e}_1 - \mathbf{H}_{k+1}\mathbf{g}\|$ para obtener el óptimo $\mathbf{g_k} \in \Re^k$, donde este problema de mínimos cuadrados es resuelto usando la descomposición QR de la matriz $\mathbf{H}_{k+1}$ de forma incremental utilizando las rotaciones de Givens y como $\mathbf{H}_{k+1}$ es bidiagonal inferior, solamente es necesaria la rotación en los pasos previos.

ALGORITMO QMRCGSTAB

Aproximación inicial $\mathbf{x}_0$. ;
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Elegir $\widetilde{\mathbf{r}}_0$ tal que $\langle \mathbf{r}_0, \widetilde{\mathbf{r}}_0 \rangle \neq 0$
$\mathbf{p}_0 = \mathbf{v}_0 = \mathbf{d}_0 = \mathbf{0}$;
$\rho_0 = \alpha_0 = \omega_0 = 1$; $\tau = \|\mathbf{r}_0\|$, $\theta_0 = 0$, $\eta_0 = 0$;
Mientras $\sqrt{j+1}\,|\widetilde{\tau}| \,/\, \|\mathbf{r}_0\| \geq \varepsilon (j = 1, 2, ...)$, hacer:
$\rho_j = \langle \mathbf{r}_{j-1}, \widetilde{\mathbf{r}}_0 \rangle$; $\beta_j = (\rho_j \alpha_{j-1}) \,/ \rho_{j-1}\omega_{j-1}$;
$\mathbf{p}_j = \mathbf{r}_{j-1} + \beta_j \left(\mathbf{p}_{j-1} - \omega_j \mathbf{v}_{j-1}\right)$;
$\mathbf{v}_j = \mathbf{A}\mathbf{p}_j$;
$\alpha_j = \dfrac{\rho_j}{\langle \mathbf{v}_j, \widetilde{\mathbf{r}}_0 \rangle}$;
$\mathbf{s}_j = \mathbf{r}_{j-1} - \alpha_j \mathbf{v}_j$;
Primera cuasi-minimización
$\widetilde{\theta}_j = \|\mathbf{s}_j\| \,/ \tau$; $c = \dfrac{1}{\sqrt{1 + \widetilde{\theta}_j^2}}$; $\widetilde{\tau} = \tau\, \widetilde{\theta}_j c$;
$\widetilde{\eta}_j = c^2 \alpha_j$;
$\widetilde{\mathbf{d}}_j = \mathbf{p}_j + \dfrac{\theta_{j-1}^2 \eta_{j-1}}{\alpha_j} \mathbf{d}_{j-1}$;
$\widetilde{\mathbf{x}}_j = \mathbf{x}_{j-1} + \widetilde{\eta}_j \widetilde{\mathbf{d}}_j$;
$\mathbf{t}_j = \mathbf{A}\mathbf{s}_j$;
$\omega_j = \dfrac{\langle \mathbf{s}_j, \mathbf{t}_j \rangle}{\langle \mathbf{t}_j, \mathbf{t}_j \rangle}$;
$\mathbf{r}_j = \mathbf{s}_j - \omega_j \mathbf{t}_j$;
Segunda cuasi-minimización
$\theta_j = \|\mathbf{r}_j\| \,/ \tau$; $c = \dfrac{1}{\sqrt{1 + \theta_j^2}}$; $\tau = \widetilde{\tau}\, \theta_j c$;
$\eta_j = c^2 \omega_j$;
$\mathbf{d}_j = \mathbf{s}_j + \dfrac{\widetilde{\theta}_j^2 \widetilde{\eta}_j}{\omega_j} \widetilde{\mathbf{d}}_j$;
$\mathbf{x}_j = \widetilde{\mathbf{x}}_j + \eta_j \mathbf{d}_j$;
Fin

3.3. Métodos basados en la Ecuación Normal

La resolución del sistema de ecuaciones (2.1), donde la matriz $\mathbf{A}$ es no simétrica, es equivalente a resolver el sistema $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^\mathbf{T}\mathbf{b}$ de matriz $\mathbf{A}^T\mathbf{A}$ simétrica definida positiva, al que podremos aplicar el algoritmo del Gradiente Conjugado. La ecuación $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^\mathbf{T}\mathbf{b}$ recibe el nombre de ecuación normal.

Al igual que el CG, los métodos desarrollados a partir de la ecuación normal cumplen las dos condiciones fundamentales de minimización de la norma residual y optimización del coste computacional, sin embargo presentan el inconveniente de que el condicionamiento del nuevo sistema es el cuadrado del sistema inicial ($K_2\left(\mathbf{A}^T\mathbf{A}\right) = K_2\left(\mathbf{A}\right)^2$), lo cual, para sistemas mal condicionados puede resultar desastroso y además en cada iteración aparecen dos productos matriz por vector correspondientes a las matrices $\mathbf{A}$ y $\mathbf{A}^T$ aumentando el coste computacional.

3.3.1. Método del Gradiente Conjugado para la Ecuación Normal (CGN)

El método [39] construye una sucesión de vectores,

$$\mathbf{x}_k = \mathbf{x}_0 + \left[\mathbf{A}^T\mathbf{r}_0, \left(\mathbf{A}^T\mathbf{A}\right)\mathbf{A}^T\mathbf{r}_0, \left(\mathbf{A}^T\mathbf{A}\right)^2\mathbf{A}^T\mathbf{r}_0, ..., \left(\mathbf{A}^T\mathbf{A}\right)^{k-1}\mathbf{A}^T\mathbf{r}_0\right]$$

con residuo mínimo en cada paso, sin efectuar el cálculo explícito del producto $\mathbf{A}^T\mathbf{A}$.

ALGORITMO CGN
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{A}^T\mathbf{r}_0$
Mientras $\| \mathbf{r}_{j-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (j = 1, 2, 3, ...)$, hacer

$$\alpha_j = \frac{\left\langle \mathbf{A}^T\mathbf{r}_j, \mathbf{A}^T\mathbf{r}_j \right\rangle}{\left\langle \mathbf{A}\mathbf{p}_j, \mathbf{A}\mathbf{p}_j \right\rangle};$$
$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$$
$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j\mathbf{A}\mathbf{p}_j$$
$$\beta_j = \frac{\left\langle \mathbf{A}^T\mathbf{r}_{j+1}, \mathbf{A}^T\mathbf{r}_{j+1} \right\rangle}{\left\langle \mathbf{A}^T\mathbf{r}_j, \mathbf{A}^T\mathbf{r}_j \right\rangle}$$
$$\mathbf{p}_{j+1} = \mathbf{A}^T\mathbf{r}_{j+1} + \beta_j\mathbf{p}_j$$

Fin

3.3.2. Método LSQR (*Least-Square QR*)

Propuesto por Paige y Saunders [63], este método intenta corregir el posible empeoramiento del número de condición de los sistemas al aplicar el método de Ecuación Normal.

La idea básica del LSQR es hallar la solución del sistema simétrico,

$$\begin{pmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^T & -\lambda^2\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \tag{3.99}$$

minimizando,

$$\left\| \begin{pmatrix} \mathbf{A} \\ \lambda\mathbf{I} \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \right\|_2 \tag{3.100}$$

donde λ es un número real arbitrario.

De la aplicación del proceso de Lanczos a este sistema simétrico resultan dos formas de un proceso de bidiagonalización propuestas por Golub y Kahan [30]:

Forma bidiagonal inferior

Si comenzamos tomando:

$$\mathbf{r}_0 = \beta_1\mathbf{u}_1, \quad \alpha_1\mathbf{v}_1 = \mathbf{A}^T\,\mathbf{u}_1$$

$$\left.\begin{aligned} \beta_{j+1}\mathbf{u}_{j+1} &= \mathbf{A}\mathbf{v}_j - \alpha_j\mathbf{u}_j \\ \alpha_{j+1}\mathbf{v}_{j+1} &= \mathbf{A}^T\mathbf{u}_{j+1} - \beta_{j+1}\mathbf{v}_j \end{aligned}\right\} \quad j = 1, 2, ... \tag{3.101}$$

los escalares $\alpha_j \geq 0$ y $\beta_j \geq 0$ son elegidos tal que $\|\mathbf{u}_j\| = \|\mathbf{v}_j\| = 1$ y,

$$\mathbf{B}_k = \begin{pmatrix} \alpha_1 & & & & \\ \beta_2 & \alpha_2 & & & \\ & \beta_3 & \alpha_3 & & \\ & & \cdots & \cdots & \\ & & & \beta_k & \alpha_k \\ & & & & \beta_{k+1} \end{pmatrix}$$

Las relaciones de recurrencia dadas en (3.101) pueden reescribirse,

$$\mathbf{r}_0 = \mathbf{U}_{k+1}\left(\beta_1\mathbf{e}_1\right), \tag{3.102}$$

$$\mathbf{A}\mathbf{V}_k = \mathbf{U}_{k+1}\mathbf{B}_k\,, \tag{3.103}$$

$$\mathbf{A}^T\mathbf{U}_{k+1} = \mathbf{V}_k\mathbf{B}_k^T + \alpha_{k+1}\mathbf{v}_{k+1}\mathbf{e}_{k+1}^T \tag{3.104}$$

teniendo en cuenta que en aritmética exacta se verifica que $\mathbf{U}_{k+1}^T\mathbf{U}_{k+1} = \mathbf{I}$ y $\mathbf{V}_k^T\mathbf{V}_k = \mathbf{I}$.

Forma bidiagonal superior

En este caso se comienza a partir de:

$$\mathbf{A}^T\mathbf{r}_0 = \theta_1\mathbf{v}_1, \quad \rho_1\mathbf{p}_1 = \mathbf{A}\mathbf{v}_1$$

$$\left.\begin{array}{r} \theta_{j+1}\mathbf{v}_{j+1} = \mathbf{A}^T\mathbf{v}_j - \rho_j\mathbf{v}_j \\ \rho_{j+1}\mathbf{p}_{j+1} = \mathbf{A}\mathbf{v}_{j+1} - \theta_{j+1}\mathbf{p}_j \end{array}\right\} \quad j = 1, 2, ... \tag{3.105}$$

Los escalares $\rho_j \geq 0$ y $\theta_j \geq 0$ son elegidos tal que $\|\mathbf{p}_j\| = \|\mathbf{v}_j\| = 1$ y,

$$\mathbf{R}_k = \begin{pmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \cdots & \cdots & \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \end{pmatrix}$$

Las relaciones dadas en (3.105) pueden reescribirse,

$$\mathbf{A}^T\mathbf{r}_0 = \mathbf{V}_k\left(\theta_1\mathbf{e}_1\right), \tag{3.106}$$

$$\mathbf{A}\mathbf{V}_k = \mathbf{P}_k\mathbf{R}_k\,, \tag{3.107}$$

$$\mathbf{A}^T\mathbf{P}_k = \mathbf{V}_k\mathbf{R}_k^T + \theta_{k+1}\mathbf{v}_{k+1}\mathbf{e}_k^T \tag{3.108}$$

y también en aritmética exacta se verifica que $\mathbf{P}_k^T\mathbf{P}_k = \mathbf{I}$ y $\mathbf{V}_k^T\mathbf{V}_k = \mathbf{I}$.

Estos dos procesos de bidiagonalización, pueden relacionarse entre sí, ya que la matriz $\mathbf{V}_k$ obtenida al aplicar el algoritmo de Lanczos con $\mathbf{B} = \mathbf{A}^T\mathbf{A}$, es la misma en ambos casos y además se verifica que,

$$\mathbf{B}_k^T\mathbf{B}_k = \mathbf{R}_k^T\mathbf{R}_k \tag{3.109}$$

Por otro lado, $\mathbf{R}_k$ debe ser idéntica, salvo errores de redondeo a la matriz obtenida de la factorización QR de $\mathbf{B}_k$. Entonces,

$$\mathbf{Q}_k\mathbf{B}_k = \begin{pmatrix} \mathbf{R}_k \\ \mathbf{0} \end{pmatrix} \tag{3.110}$$

donde $\mathbf{Q}_k$ es ortogonal.

Las matrices ortogonales, $\mathbf{U}_k$ y $\mathbf{P}_k$ están relacionadas por,

$$\mathbf{U}_{k+1} = [\mathbf{U}_k\mathbf{u}_{k+1}] = \left[\mathbf{P}_k\frac{\mathbf{r}_k}{\|\mathbf{r}_k\|}\right]\mathbf{Q}_k \tag{3.111}$$

Y además también se verifica,

$$\begin{array}{r} \alpha_1^2 + \beta_2^2 = \rho_1^2\,, \quad \alpha_1\beta_1 = \theta_1 \\ \alpha_j^2 + \beta_{j+1}^2 = \rho_j^2 + \theta_j^2\,, \quad \alpha_j\beta_j = \rho_{j-1}\theta_j\,, \quad \text{para } j > 1 \end{array} \tag{3.112}$$

Cuando aplicamos estos procesos al sistema (3.99), las relaciones del algoritmo de Lanczos,

$$\begin{aligned} \mathbf{w}_j &= \mathbf{B}\mathbf{v}_j - \beta_j\mathbf{v}_{j-1} \\ \alpha_j &= \langle\mathbf{w}_j, \mathbf{v}_j\rangle \\ \beta_{j+1}\mathbf{v}_{j+1} &= \mathbf{w}_j - \alpha_j\mathbf{v}_j \end{aligned}$$

se reducen al proceso de bidiagonalización inferior, mientras que las expresiones,

$$\begin{aligned}\mathbf{y}_k &= \mathbf{T}_k^{-1}\left(\beta_1\mathbf{e}_1\right)\\ \mathbf{x}_k &= \mathbf{V}_k\mathbf{y}_k\end{aligned}$$

después de $2k+1$ iteraciones se habrán transformado en,

$$\begin{pmatrix} \mathbf{I} & \mathbf{B}_k \\ \mathbf{B}_k^T & -\lambda^2\mathbf{I} \end{pmatrix}\begin{pmatrix} \mathbf{t}_{k+1} \\ \mathbf{y}_k \end{pmatrix} = \begin{pmatrix} \beta_1\mathbf{e}_1 \\ \mathbf{0} \end{pmatrix}, \tag{3.113}$$

$$\begin{pmatrix} \mathbf{r}_k \\ \mathbf{x}_k \end{pmatrix} = \begin{pmatrix} \mathbf{U}_{k+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_k \end{pmatrix}\begin{pmatrix} \mathbf{t}_{k+1} \\ \mathbf{y}_k \end{pmatrix} \tag{3.114}$$

donde $\mathbf{B}_k$ es una matriz bidiagonal inferior $(k+1)\times k$ e $\mathbf{y}_k$ es la solución de otro problema de ecuación normal,

$$\min\left\|\begin{pmatrix} \mathbf{B}_k \\ \lambda\mathbf{I} \end{pmatrix}\mathbf{y}_k - \begin{pmatrix} \beta_1\mathbf{e}_1 \\ \mathbf{0} \end{pmatrix}\right\|_2 \tag{3.115}$$

que puede resolverse mediante transformaciones ortogonales.

Otra forma de resolver el problema de Ecuación Normal se puede derivar de forma análoga. Definiendo $\mathbf{s} = -\mathbf{A}\mathbf{x}$, podemos escribir la ecuación (3.99) como,

$$\begin{pmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^T & -\lambda^2\mathbf{I} \end{pmatrix}\begin{pmatrix} \mathbf{s} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\mathbf{A}^T\mathbf{b} \end{pmatrix} \tag{3.116}$$

Aplicando nuevamente el algoritmo de Lanczos, después de $2k$ iteraciones se habrá transformado en,

$$\begin{pmatrix} \mathbf{I} & \mathbf{R}_k \\ \mathbf{R}_k^T & -\lambda^2\mathbf{I} \end{pmatrix}\begin{pmatrix} \mathbf{q}_k \\ \mathbf{y}_k \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\theta_1\mathbf{e}_1 \end{pmatrix}, \tag{3.117}$$

$$\begin{pmatrix} \mathbf{s}_k \\ \mathbf{x}_k \end{pmatrix} = \begin{pmatrix} \mathbf{P}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_k \end{pmatrix}\begin{pmatrix} \mathbf{q}_k \\ \mathbf{y}_k \end{pmatrix} \tag{3.118}$$

donde $\mathbf{R}_k$ es una matriz bidiagonal superior $k\times k$ e $\mathbf{y}_k$ satisface,

$$\left(\mathbf{R}_k^T\mathbf{R}_k + \lambda^2\mathbf{I}\right)\mathbf{y}_k = \theta_1\mathbf{e}_1 \tag{3.119}$$

Se puede ver que las matrices generadas en ambos procesos de bidiagonalización, $\mathbf{B}_k$, $\mathbf{U}_{k+1}$, $\mathbf{R}_k$, $\mathbf{P}_k$ y $\mathbf{V}_k$ son independientes de λ, lo que supone que son generadas igualmente cuando $\lambda = 0$.

Las matrices, los vectores y parámetros generados en la primera bidiagonalización son usados para resolver el problema de Ecuación Normal,

$$\text{mín}\,\|\mathbf{b} - \mathbf{A}\mathbf{x}\|$$

Los vectores,

$$\mathbf{x}_k = \mathbf{V}_k \mathbf{y}_k \tag{3.120}$$
$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k \tag{3.121}$$
$$\mathbf{t}_{k+1} = \beta_1 \mathbf{e}_1 - \mathbf{B}_k \mathbf{y}_k \tag{3.122}$$

dependen de $\mathbf{y}_k$, y se tiene que,

$$\mathbf{r}_k = \mathbf{U}_{k+1} \mathbf{t}_{k+1} \tag{3.123}$$

Debemos minimizar $\|\mathbf{r}_k\|$ y como teóricamente las columnas de la matriz $\mathbf{U}_{k+1}$ son ortonormadas, la elección de $\mathbf{y}_k$ debe ser tal que minimice $\|\mathbf{t}_{k+1}\|$. Es decir que el problema que trata de resolverse es,

$$\text{mín} \, \|\beta_1 \mathbf{e}_1 - \mathbf{B}_k \mathbf{y}_k\| \tag{3.124}$$

La factorización QR de $\mathbf{B}_k$ es la misma que se hizo en los procesos de bidiagonalización y toma la forma,

$$\mathbf{Q}_k \begin{pmatrix} \mathbf{B}_k & \beta_1 \mathbf{e}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_k & \mathbf{f}_k \\ & \overline{\phi}_{k+1} \end{pmatrix} \equiv \begin{pmatrix} \rho_1 & \theta_2 & & & & \phi_1 \\ & \rho_2 & \theta_3 & & & \phi_2 \\ & & \cdots & \cdots & & \cdots \\ & & & \rho_{k-1} & \theta_k & \phi_{k-1} \\ & & & & \rho_k & \overline{\phi}_k \\ & & & & & \overline{\phi}_{k+1} \end{pmatrix}$$

donde $\mathbf{Q}_k \equiv \mathbf{Q}_{k,k+1} \ldots \mathbf{Q}_{2,3} \mathbf{Q}_{1,2}$ es un producto de rotación de planos y los vectores $\mathbf{y}_k$ y $\mathbf{t}_{k+1}$ pueden ser obtenidos de la forma,

$$\mathbf{R}_k \mathbf{y}_k = \mathbf{f}_k \tag{3.125}$$

$$\mathbf{t}_{k+1} = \mathbf{Q}_k^T \begin{pmatrix} \mathbf{0} \\ \overline{\phi}_{k+1} \end{pmatrix} \tag{3.126}$$

Como $\begin{pmatrix} \mathbf{R}_k & \mathbf{f}_k \end{pmatrix}$ es la misma matriz que $\begin{pmatrix} \mathbf{R}_{k-1} & \mathbf{f}_{k-1} \end{pmatrix}$ pero con una nueva fila y columna añadidas, se pueden combinar adecuadamente las ecuaciones (3.120) y (3.125) para obtener,

$$\mathbf{x}_k = \mathbf{V}_k \mathbf{R}_k^{-1} \mathbf{f}_k = \mathbf{D}_k \mathbf{f}_k \tag{3.127}$$

donde las columnas de $\mathbf{D}_k = \begin{pmatrix} \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_k \end{pmatrix}$ se obtienen sucesivamente de la resolución del sistema $\mathbf{R}_k^T \mathbf{D}_k = \mathbf{V}_k^T$. Con $\mathbf{d}_0 = \mathbf{x}_0 = \mathbf{0}$, resulta,

$$\mathbf{d}_k = \frac{1}{\rho_k} \left(\mathbf{v}_k - \theta_k \mathbf{d}_{k-1} \right), \tag{3.128}$$
$$\mathbf{x}_k = \mathbf{x}_{k-1} + \phi_k \mathbf{d}_k \tag{3.129}$$

La factorización QR anterior es deteminada para construir el k-ésimo plano de rotación $\mathbf{Q}_{k,k+1}$ que opera en las filas k y $k+1$ de $\begin{pmatrix} \mathbf{B}_k & \beta_1\mathbf{e}_1 \end{pmatrix}$ para eliminar β_{k+1}. Las relaciones de recurrencia pueden expresarse,

$$\begin{pmatrix} c_k & s_k \\ s_k & -c_k \end{pmatrix} \begin{pmatrix} \overline{\rho}_k & 0 & \overline{\phi}_k \\ \beta_{k+1} & \alpha_{k+1} & 0 \end{pmatrix} = \begin{pmatrix} \rho_k & \theta_{k+1} & \phi_k \\ 0 & \overline{\rho}_{k+1} & \overline{\phi}_{k+1} \end{pmatrix} \tag{3.130}$$

donde $\overline{\rho}_1 = \alpha_1$, $\overline{\phi}_1 = \beta_1$ y los escalares c_k y s_k son elementos no triviales de $\mathbf{Q}_{k,k+1}$. Los escalares $\overline{\rho}_k$ y $\overline{\phi}_k$ se van reemplazando secuencialmente por ρ_k y ϕ_k.

Por otro lado, en la ecuación (3.128) se puede usar el vector $\mathbf{w}_k = \rho_k\mathbf{d}_k$ en lugar de $\mathbf{d}_k$.

Para la estimación de $\|\mathbf{r}_k\|$, teniendo en cuenta las expresiones (3.123) y (3.126),

$$\mathbf{r}_k = \overline{\phi}_{k+1}\mathbf{Q}_k^T\mathbf{U}_{k+1}\mathbf{e}_{k+1} \tag{3.131}$$

y asumiendo que $\mathbf{U}_{k+1}^T\mathbf{U}_{k+1} = I$, se obtiene,

$$\|\mathbf{r}_k\| = \overline{\phi}_{k+1} = \beta_1 s_k s_{k-1} \cdots s_1 \tag{3.132}$$

ALGORITMO LSQR

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\beta_1 = \|\mathbf{z}_0\|$, $\mathbf{u}_1 = \mathbf{r}_0/\beta_1$;
$\alpha_1 = \left\|\mathbf{A}^T\mathbf{u}_1\right\|$, $\mathbf{v}_1 = \mathbf{A}^T\mathbf{u}_1/\alpha_1$, $\mathbf{w}_1 = \mathbf{v}_1$,
$\overline{\phi}_1 = \beta_1$, $\overline{\rho}_1 = \alpha_1$
Mientras $\overline{\phi}_j / \|\mathbf{r}_0\| \geq \varepsilon \quad (j = 1, 2, 3, ...)$, hacer

$$\begin{aligned}
&\beta_{j+1} = \|\mathbf{A}\mathbf{v}_j - \alpha_j\mathbf{u}_j\| \\
&\mathbf{u}_{j+1} = \frac{\mathbf{A}\mathbf{v}_j - \alpha_j\mathbf{u}_j}{\beta_{j+1}} \\
&\alpha_{j+1} = \left\|\mathbf{A}^T\mathbf{u}_{j+1} - \beta_{j+1}\mathbf{v}_j\right\| \\
&\mathbf{v}_{j+1} = \frac{\mathbf{A}^T\mathbf{u}_{j+1} - \beta_{j+1}\mathbf{v}_j}{\alpha_{j+1}} \\
&\rho_j = \left(\overline{\rho}_j^2 + \beta_{j+1}^2\right)^{\frac{1}{2}} \\
&c_j = \frac{\overline{\rho}_j}{\rho_j} \\
&s_j = \frac{\beta_{j+1}}{\rho_j} \\
&\theta_{j+1} = s_j\alpha_{j+1} \\
&\overline{\rho}_{j+1} = -c_j\alpha_{j+1} \\
&\phi_j = c_j\overline{\phi}_j \\
&\overline{\phi}_{j+1} = s_j\overline{\phi}_j \\
&\mathbf{x}_j = \mathbf{x}_{j-1} + \left(\frac{\phi_j}{\rho_j}\right)\mathbf{w}_j \\
&\mathbf{w}_{j+1} = \mathbf{v}_{j+1} - \left(\frac{\theta_{j+1}}{\rho_j}\right)\mathbf{w}_j
\end{aligned}$$

Fin

Capítulo 4

Precondicionamiento

La convergencia de los métodos basados en los subespacios de Krylov mejora con el uso de las técnicas de precondicionamiento. Éstas consisten generalmente en cambiar el sistema original $\mathbf{Ax} = \mathbf{b}$ por otro de idéntica solución, de forma que el número de condicionamiento de la matriz del nuevo sistema sea menor que el de $\mathbf{A}$, o bien que tenga una mejor distribución de autovalores.

Generalmente, se considera como matriz de precondicionamiento a $\mathbf{M}^{-1}$ y obtener $\mathbf{M}$ como una aproximación de $\mathbf{A}$, esto es,

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$$

tal que,

$$K\left(\mathbf{M^{-1}A}\right) < K\left(\mathbf{A}\right)$$

El menor valor de K corresponde a $\mathbf{M} = \mathbf{A}$, de forma que $K\left(\mathbf{A}^{-1}\mathbf{A}\right) = 1$, que es el caso ideal y el sistema convergería en una sola iteración, pero el coste computacional del cálculo de $\mathbf{A}^{-1}$ equivaldría a resolver el sistema por un método directo, por lo que se sugiere para $\mathbf{M}$ que sea una matriz lo más próxima a $\mathbf{A}$ sin que su determinación suponga un coste elevado.

Por tanto, la matriz $\mathbf{M}$ debe ser facilmente inversible para poder efectuar los productos $\mathbf{M}^{-1}$ por vector que aparecen en los algoritmos precondicionados sin excesivo coste adicional, por ejemplo en el caso en que $\mathbf{M}$ es una matriz diagonal, o está factorizada adecuadamente para efectuar dichos productos mediante remonte sin necesidad de calcular $\mathbf{M}^{-1}$.

Dependiendo de la forma de plantear el producto de $\mathbf{M}^{-1}$ por la matriz del sistema obtendremos distintas formas de precondicionamiento. Éstas son,

$$\begin{array}{ll} \mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b} & \text{(Precondicionamiento por la izquierda)} \\ \mathbf{AM}^{-1}\mathbf{Mx} = \mathbf{b} & \text{(Precondicionamiento por la derecha)} \\ \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M}_2\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} & \text{(Precondicionamiento por ambos lados)} \end{array} \tag{4.1}$$

si $\mathbf{M}$ puede ser factorizada como $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$.

Por tanto, los precondicionadores han de cumplir dos requisitos fundamentales, fácil implementación, evitando un coste computacional excesivo del producto de $\mathbf{M}^{-1}$ por cualquier vector y mejorar la convergencia del método. Por

tanto una matriz que sea una aproximación más o menos cercana de $\mathbf{A}$, obtenida con estos criterios puede ser un buen precondicionador. El campo de posibles precondicionadores es por tanto muy amplio. Algunos de los más usados son los bien conocidos Diagonal o de Jacobi, SSOR e ILU(0). En esta tésis consideraremos estos, además de la Aproximada Inversa de estructura diagonal que hemos nombrado como precondicionador Diagonal Óptimo.

4.1. Precondicionador de Jacobi

Surge comparando la fórmula de recurrencia para la solución que resulta de aplicar el método de Richardson, cuya relación de recurrencia viene dada por $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha\,(\mathbf{b} - \mathbf{A}\mathbf{x}_i)$, con $\alpha > 0$, al sistema precondicionado con la fórmula correspondiente que se obtiene aplicando el método de Jacobi al sistema sin precondicionar. De la aplicación del método de Richardson al sistema precondicionado $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$, se obtiene para el cálculo de los sucesivos valores de la solución,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha\left(\mathbf{M}^{-1}\mathbf{b} - \mathbf{M}^{-1}\mathbf{A}\mathbf{x}_i\right)$$

multiplicando por la matriz de precondicionamiento $\mathbf{M}$, queda,

$$\mathbf{M}\mathbf{x}_{i+1} = \mathbf{M}\mathbf{x}_i + \alpha\,(\mathbf{b} - \mathbf{A}\mathbf{x}_i) \tag{4.2}$$

Por otro lado, descomponiendo la matriz del sistema en $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$, ($\mathbf{D}$ matriz diagonal formada por los elementos de la diagonal de $\mathbf{A}$ y $\mathbf{E}$ y $\mathbf{F}$ matrices triangulares inferior y superior respectivamente), y utilizando el método de Jacobi para la resolución del sistema, se obtiene,

$$\mathbf{x}_{i+1} = \mathbf{D}^{-1}\,(\mathbf{E} + \mathbf{F})\,\mathbf{x}_i + \mathbf{D}^{-1}\mathbf{b}$$

que multiplicando por $\mathbf{D}$ y operando resulta,

$$\mathbf{D}\mathbf{x}_{i+1} = \mathbf{D}\mathbf{x}_i + (\mathbf{b} - \mathbf{A}\mathbf{x}_i) \tag{4.3}$$

Comparando las expresiones de recurrencia finales de ambos métodos, se observa que el método de Jacobi aplicado al sistema sin precondicionar, equivale al de Richardson, con $\alpha = 1$, menos robusto y más simple, cuando este se aplica al sistema precondicionado con la matriz diagonal $\mathbf{D}$. Resulta así un precondicionador elemental, fácil de implementar y con matriz inversa que se determina con muy bajo coste computacional, ya que la matriz de precondicionamiento es diagonal y sus entradas son las de la diagonal de $\mathbf{A}$.

4.2. Precondicionador SSOR

Si aplicamos el método SSOR al sistema sin precondicionar, considerando la descomposición de la matriz $\mathbf{A}$ en $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$ como en el caso anterior

siendo ω el parámetro de relajación, se obtiene para la solución,

$$\mathbf{x}_{i+1} = \left(\frac{\mathbf{D}}{\omega} - \mathbf{F}\right)^{-1} \left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{E}\right) \left(\frac{\mathbf{D}}{\omega} - \mathbf{E}\right)^{-1} \left(\frac{1-\omega}{\omega}\mathbf{D} + \mathbf{F}\right) \mathbf{x}_i + \\ + \left(\frac{\mathbf{D}}{\omega} - \mathbf{F}\right)^{-1} \frac{2-\omega}{\omega}\mathbf{D} \left(\frac{\mathbf{D}}{\omega} - \mathbf{E}\right)^{-1} \mathbf{b}$$

operando para expresar esta relación de forma que se pueda comparar con la solución que resulta de aplicar el método de Richardson al sistema precondicionado,

$$\frac{1}{\omega\,(2-\omega)} \left(\mathbf{D}-\omega\mathbf{E}\right) \mathbf{D}^{-1} \left(\mathbf{D}-\omega\mathbf{F}\right) \mathbf{x}_{i+1} \\ = \frac{1}{\omega\,(2-\omega)} \left(\mathbf{D}-\omega\mathbf{E}\right) \mathbf{D}^{-1} \left(\mathbf{D}-\omega\mathbf{F}\right) \mathbf{x}_i + \left(\mathbf{b} - \mathbf{A}\mathbf{x}_i\right)$$

con lo que resulta como matriz de precondicionamiento,

$$\mathbf{M} = \frac{1}{\omega\,(2-\omega)} \left(\mathbf{D}-\omega\mathbf{E}\right) \mathbf{D}^{-1} \left(\mathbf{D}-\omega\mathbf{F}\right) \tag{4.4}$$

que en el caso de sistemas simétricos, como se cumple que,

$$\left(\mathbf{D}-\omega\mathbf{F}\right) = \left(\mathbf{D}-\omega\mathbf{E}\right)^T$$

podemos expresarla como un producto de dos matrices triangulares traspuestas,

$$\mathbf{M} = \left[\frac{\left(\mathbf{D}-\omega\mathbf{E}\right) \mathbf{D}^{-1/2}}{\sqrt{\omega\,(2-\omega)}}\right] \left[\frac{\left(\mathbf{D}-\omega\mathbf{E}\right) \mathbf{D}^{-1/2}}{\sqrt{\omega\,(2-\omega)}}\right]^T \tag{4.5}$$

para el caso de sistemas no simétricos también lo podremos expresar como un producto de matrices triangulares, inferior y superior respectivamente,

$$\mathbf{M} = \left(\mathbf{I}-\omega\mathbf{E}\mathbf{D}^{-1}\right) \left(\frac{\mathbf{D}-\omega\mathbf{F}}{\omega(2-\omega)}\right) \tag{4.6}$$

4.3. Precondicionador ILUT

Resulta de la aproximación de $\mathbf{A}$ por una factorización incompleta LU en la que se establecen unas determinadas reglas para eliminar (hacer cero) algunas entradas de las matrices $\mathbf{L}$ y $\mathbf{U}$, que eran cero en la estructura original de la matriz $\mathbf{A}$. Se puede establecer la regla para todas las filas, aplicando la misma regla a todos los elementos de la fila.

Llamando w a la longitud completa de trabajo de la fila, que es usada para acumular combinaciones lineales de filas *sparse* en la eliminación gaussiana,

siendo w_k la k-ésima entrada de esta fila y denotando por a_{i*} la i-ésima fila de **A**, el algoritmo ILUT queda,

ALGORITMO ILUT
Para $i = 1, ..., n$, Hacer
$w = a_{i*}$
Para $k = 1, ..., i-1$ y cuando $w_k \neq 0$ Hacer
$w_k = w_k / a_{kk}$
Aplicando una regla de eliminación para w_k
Si $w_k \neq 0$ entonces $w = w - w_{k*} u_{k*}$
Fin
Fin
Aplicando una regla de eliminación para la fila w
$l_{i,j} = w_j$ para $j = 1, ..., i-1$
$u_{i,j} = w_j$ para $j = i, ..., n$
$w = 0$
Fin

En la factorización ILUT(p, τ) la regla de eliminación usada es:
1. Un elemento w_k, será reemplazado por cero si es menor que una tolerancia relativa τ_i obtenida multiplicando por τ la norma del vector original de la i-ésima fila.
2. Para cada vector fila w primero se hacen cero cualquier elemento cuya magnitud sea menor que una tolerancia relativa τ_i. Entonces se consideran sólo los p elementos mayores de la fila correspondiente, tanto en **L** como en **U**, aunque en ésta última además se incluye siempre el elemento de la diagonal.

4.3.1. Precondicionador ILU(0)

Es un caso particular del precondicionador ILUT que resulta de la aproximación de **A** por una factorización incompleta LU, guardando las mismas entradas nulas en las matrices triangulares **L** y **U** [48],

$$\mathbf{A} = \mathbf{LU} \approx \mathbf{ILU}(0) = \mathbf{M} \tag{4.7}$$

donde m_{ij} son las entradas de **M** tal que,

$$m_{ij} = 0 \qquad \text{if} \qquad a_{ij} = 0 \tag{4.8}$$
$$\{\mathbf{A} - \mathbf{LU}\}_{ij} = 0 \qquad \text{if} \qquad a_{ij} \neq 0 \tag{4.9}$$

Es decir que lo elementos nulos de la matriz del sistema siguen siendo nulos en las posiciones respectivas de las matrices triangulares para no incrementar el coste computacional.

4.4. Precondicionador Diagonal Óptimo

Recientemente, el uso de inversas aproximadas se ha convertido en una alternativa a los precondicionadores implícitos debido a sus propiedades de paralelización [5, 36]. Por ejemplo, en el caso de precondicionamiento por la izquierda, el precondicionador N resulta de resolver un problema de minimización [3], definido sobre la norma de Frobenius,

$$\min_{\mathbf{M}\in\mathcal{S}} \|\mathbf{MA}-\mathbf{I}\|_F = \|\mathbf{NA}-\mathbf{I}\|_F$$

donde $\mathcal{S}$ es un subespacio de las matrices cuadradas de orden n y $\mathbf{N}\in\mathcal{S}$, es tanto mejor precondicionador (aproximada inversa de $\mathbf{A}$) cuanto más pequeña sea $\|\mathbf{NA}-\mathbf{I}\|_F$, siendo la situación deseable que $\|\mathbf{NA}-\mathbf{I}\|_F < 1$.

Si $\{\mathbf{M}_i\}_{i=1}^p$ es una base de $\mathbf{S}$ tal que $\{\mathbf{M}_i\mathbf{A}\}_{i=1}^p$ es base ortogonal de $\mathbf{SA}$, Entonces, la solución a este problema, que puede verse en [31, 60], es:

$$\mathbf{N}=\sum_{i=1}^{p}\frac{tr\left(\mathbf{M}_i\mathbf{A}\right)}{\|\mathbf{M}_i\mathbf{A}\|_F^2}\mathbf{M}_i \quad \text{y} \quad \|\mathbf{NA}-\mathbf{I}\|_F^2 = n-\sum_{i=1}^{p}\frac{\left[tr\left(\mathbf{M}_i\mathbf{A}\right)\right]^2}{\|\mathbf{M}_i\mathbf{A}\|_F^2} \tag{4.10}$$

En el caso particular en que $\mathcal{S}$ es el subespacio de las matrices diagonales de orden n, el mejor precondicionador diagonal del sistema es,

$$\mathbf{N}=diag\left(\frac{a_{11}}{\|\mathbf{e}_1^T\mathbf{A}\|_2^2},\frac{a_{22}}{\|\mathbf{e}_2^T\mathbf{A}\|_2^2},\ldots,\frac{a_{nn}}{\|\mathbf{e}_n^T\mathbf{A}\|_2^2}\right) \tag{4.11}$$

$$\|\mathbf{NA}-\mathbf{I}\|_F^2 = n-\sum_{i=1}^{n}\frac{a_{ii}}{\|\mathbf{e}_i^T\mathbf{A}\|_2^2} \tag{4.12}$$

Capítulo 5

Esquemas de almacenamiento

Las matrices de los sistemas lineales que estamos resolviendo tienen estructura *sparse* y el número de elementos no nulos es mucho menor que el de los nulos. El esquema de almacenamiento de estas matrices pues, debe fundamentalmente reducir el coste computacional y el requerimiento de memoria en el ordenador.

Existen distintas formas de almacenamiento [68, 72], para estas matrices *sparse*, que datan de las primeras experiencias numéricas en el campo de la Ingeniería y que han ido mejorándose a medida que lo ha permitido el desarrollo de la tecnología informática. Una técnica más reciente, que tuvo su auge con la aparición de ordenadores paralelos es la de almacenamiento elemento a elemento (ver por ejemplo Montero y otros [53]). La efectividad de estos métodos está directamente relacionada con la utilización del paralelismo masivo en la computación.

5.1. Almacenamiento de la matriz del sistema

El esquema de almacenamiento que utilizamos en este trabajo, para una matriz *sparse* $\mathbf{A}$ de dimensión n es una versión del formato Ellpack-Itpack [72]. Se requieren dos matrices rectangulares de dimensión $n \times n_d$, una de reales y otra de enteros, donde n_d es el máximo número de términos no nulo por fila y $n_d << n$. La primera columna de la matriz de reales contiene a los términos de la diagonal de $\mathbf{A}$, y a continuación, de forma ordenada se coloca el resto de términos no nulos de la fila. En la primera columna de la matriz de enteros se coloca el número de términos no nulos de la fila. Seguidamente, en cada fila se almacena la posición de columna de cada elemento no nulo de $\mathbf{A}$. En ambas matrices las filas son completadas con tantos ceros como sea necesario.

Por ejemplo, para la matriz,

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 & 0 & 0 & a_{17} \\ 0 & a_{22} & 0 & a_{24} & a_{25} & a_{26} & 0 \\ a_{31} & 0 & a_{33} & 0 & 0 & 0 & 0 \\ 0 & a_{42} & 0 & a_{44} & a_{45} & 0 & 0 \\ 0 & a_{52} & 0 & 0 & a_{55} & 0 & a_{57} \\ 0 & a_{62} & 0 & 0 & 0 & a_{66} & 0 \\ a_{71} & 0 & 0 & a_{74} & a_{75} & 0 & a_{77} \end{bmatrix}$$

las matrices de elementos no nulos y de posiciones serían,

$$\mathbf{V_A} = \begin{bmatrix} a_{11} & a_{13} & a_{17} & 0 \\ a_{22} & a_{24} & a_{25} & a_{26} \\ a_{33} & a_{31} & 0 & 0 \\ a_{44} & a_{42} & a_{45} & 0 \\ a_{55} & a_{52} & a_{57} & 0 \\ a_{66} & a_{62} & 0 & 0 \\ a_{77} & a_{71} & a_{74} & a_{75} \end{bmatrix} \qquad \mathbf{P_A} = \begin{bmatrix} 3 & 3 & 7 & 0 \\ 4 & 4 & 5 & 6 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 5 & 0 \\ 3 & 2 & 7 & 0 \\ 2 & 2 & 0 & 0 \\ 4 & 1 & 4 & 5 \end{bmatrix}$$

5.2. Almacenamiento de la matriz de precondicionamiento

El esquema de almacenamiento descrito anteriormente tiene interesantes ventajas para la computación del producto matriz por vector en máquinas vectoriales/paralelas, pero además permite utilizar la misma matriz de posiciones de **A** para los precondicionadores ILU(0) y SSOR.

5.2.1. Almacenamiento de la matriz de precondicionamiento factorizada ILU(0)

En el ejemplo anterior, la factorización ILU(0) de la matriz **A** quedaría,

$$\begin{bmatrix} a_{11} & 0 & a_{13} & 0 & 0 & 0 & a_{17} \\ 0 & a_{22} & 0 & a_{24} & a_{25} & a_{26} & 0 \\ a_{31} & 0 & a_{33} & 0 & 0 & 0 & 0 \\ 0 & a_{42} & 0 & a_{44} & a_{45} & 0 & 0 \\ 0 & a_{52} & 0 & 0 & a_{55} & 0 & a_{57} \\ 0 & a_{62} & 0 & 0 & 0 & a_{66} & 0 \\ a_{71} & 0 & 0 & a_{74} & a_{75} & 0 & a_{77} \end{bmatrix} \approx$$

$$\approx \begin{bmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ l_{31} & 0 & 1 & & & & \\ 0 & l_{42} & 0 & 1 & & & \\ 0 & l_{52} & 0 & 0 & 1 & & \\ 0 & l_{62} & 0 & 0 & 0 & 1 & \\ l_{71} & 0 & 0 & l_{74} & l_{75} & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{11} & 0 & u_{13} & 0 & 0 & 0 & u_{17} \\ & u_{22} & 0 & u_{24} & u_{25} & u_{26} & 0 \\ & & u_{33} & 0 & 0 & 0 & 0 \\ & & & u_{44} & u_{45} & 0 & 0 \\ & & & & u_{55} & 0 & u_{57} \\ & & & & & u_{66} & 0 \\ & & & & & & u_{77} \end{bmatrix}$$

Las dos matrices **L** y **U** se almacenarían en la matriz única cuyos elementos no nulos ocupan idénticos lugares que en la matriz del sistema y por tanto, definida por la misma matriz de posiciones,

$$\begin{bmatrix} u_{11} & 0 & u_{13} & 0 & 0 & 0 & u_{17} \\ 0 & u_{22} & 0 & u_{24} & u_{25} & u_{26} & 0 \\ l_{31} & 0 & u_{33} & 0 & 0 & 0 & 0 \\ 0 & l_{42} & 0 & u_{44} & u_{45} & 0 & 0 \\ 0 & l_{52} & 0 & 0 & u_{55} & 0 & u_{57} \\ 0 & l_{62} & 0 & 0 & 0 & u_{66} & 0 \\ l_{71} & 0 & 0 & l_{74} & l_{75} & 0 & u_{77} \end{bmatrix}$$

5.2.2. Almacenamiento de la matriz de precondicionamiento factorizada SSOR

En la factorización de M obtenida para el precondicionador SSOR, las matrices serían,

$$\left(\mathbf{I}-\omega\mathbf{E}\mathbf{D}^{-1}\right) = \begin{bmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ s_{31} & 0 & 1 & & & & \\ 0 & s_{42} & 0 & 1 & & & \\ 0 & s_{52} & 0 & 0 & 1 & & \\ 0 & s_{62} & 0 & 0 & 0 & 1 & \\ s_{71} & 0 & 0 & s_{74} & s_{75} & 0 & 1 \end{bmatrix}$$

$$\left(\frac{\mathbf{D}-\omega\mathbf{F}}{\omega(2-\omega)}\right) = \begin{bmatrix} r_{11} & 0 & r_{13} & 0 & 0 & 0 & r_{17} \\ & r_{22} & 0 & r_{24} & r_{25} & r_{26} & 0 \\ & & r_{33} & 0 & 0 & 0 & 0 \\ & & & r_{44} & r_{45} & 0 & 0 \\ & & & & r_{55} & 0 & r_{57} \\ & & & & & r_{66} & 0 \\ & & & & & & r_{77} \end{bmatrix}$$

Y de la misma forma que para el precondicionador ILU(0), bastará una matriz de elementos no nulos para almacenar conjuntamente las dos matrices,

$$\left[\begin{array}{ccccccc} r_{11} & 0 & r_{13} & 0 & 0 & 0 & r_{17} \\ 0 & r_{22} & 0 & r_{24} & r_{25} & r_{26} & 0 \\ s_{31} & 0 & r_{33} & 0 & 0 & 0 & 0 \\ 0 & s_{42} & 0 & r_{44} & r_{45} & 0 & 0 \\ 0 & s_{52} & 0 & 0 & r_{55} & 0 & r_{57} \\ 0 & s_{62} & 0 & 0 & 0 & r_{66} & 0 \\ s_{71} & 0 & 0 & s_{74} & s_{75} & 0 & r_{77} \end{array}\right]$$

Capítulo 6

Reordenación

Las técnicas de reordenación, que se aplican fundamentalmente en la resolución de sistemas de ecuaciones lineales por métodos directos, están basadas en la teoría de grafos y proporcionan una nueva matriz con un ancho de banda o pérfil menor, reduciendo el coste de almacenamiento y donde el efecto de relleno que se produce en la factorización LU queda disminuido. El objetivo que se persigue al aplicar estas técnicas a los algoritmos que hemos estudiado es conseguir que la factorización ILU(0) sea más cercana a la factorización completa LU, para que la matriz de precondicionamiento se asemeje lo más posible a la matriz del sistema inicial $\mathbf{A}$, lo que supone teóricamente, un mejor precondicionador, mejora que se ve reflejada en los resultados obtenidos para las aplicaciones prácticas. Efectos similares se pueden extender al precondicionador SSOR ya que es otra aproximación de la factorización de $\mathbf{A}$.

La reordenación no afecta al almacenamiento de la matriz, ya que el número de elementos no nulos, que son los almacenados en la forma compacta que usamos, se sigue conservando aunque ocupen posiciones distinstas.

En los experimentos númericos, las técnicas que se han adaptado al esquema de almacenamiento descrito, son el algoritmo de grado mínimo (MDG, *Mínimum Degree*) propuesto por George y Liu [29], el algoritmo inverso de Cuthill-Mckee (RCM, *Reverse Cuthill-Mckee*) [28] propuesto por George para mejorar el algoritmo de Cuthill-Mckee [8] y el algoritmo del mínimo vecino (MN, *Minimum Neighboring*) [13], que es una variante del MDG.

6.1. Algoritmo de Grado Mínimo

Este algoritmo se usa para matrices con estructura *sparse* pero simétrica. Consiste en hacer una renumeración de los nodos en orden creciente de sus grados respectivos (el grado de un nodo es el número de aristas o conexiones de dicho nodo en el grafo asociado). Los nodos de mayor grado originarán en teoría, un mayor *fill-in*. La idea es renumerarlos al final para evitar el incremento del *fill-in* durante el proceso.

ALGORITMO MDG
1 - Construir el grafo asociado a la matriz **A**, $g(x) = \langle V, E \rangle$, donde V es el conjunto de nodos y $E = \{\{a, b\} : a \neq b \ / \ a, b \in V\}$.
2 - Mientras $V \neq \emptyset$:
2.1- Elegir un nodo v de grado mínimo en $g(x) = \langle V, E \rangle$ y reordenar como nodo siguiente.
2.2 - Definir:
$V_v = V - \{v\}$,
$E_v = \{\{a, b\} \in E \ / \ a, b \in V_v\} \cup \{\{a, b\} \, a \neq b \ / \ a, b \in Adj_g(v)\}$.
Siendo, $Adj_g(v)$ el conjunto de nodos conectados a v en el grafo $g(x)$.
y hacer
$V = V_v, \ \ E = E_v$ y $g(x) = \langle V, E \rangle$.
3 - Fin

6.2. Algoritmo de Cuthill-McKee Inverso

El algoritmo de Cuthill-Mckee proporciona un método sencillo para reordenar una matriz *sparse* con objeto de reducir los costes de almacenamiento (conservar la *sparsidad*, es decir reducir el efecto *fill-in*) transformando la matriz en una matriz banda. La ordenación resultante al invertir el algoritmo de Cuthill-Mckee resulta frecuentemente mejor que el original, en términos de reducción de perfil, aunque la anchura de banda permanece sin mejorarse. Este algoritmo se denomina de Cuthill-Mckee Inverso.

ALGORITMO RCM
1 - Construir el grafo asociado a la matriz **A**, $g(x) = \langle V, E \rangle$, siendo V el conjunto de nodos y $E = \{\{a, b\} : a \neq b \ / \ a, b \in V\}$.
2 - Determinar un nodo inicial (pseudo-periférico) y renumerarlo como x_1.
3 - Renumerar los nodos conectados a x_i en orden ascendente de grado.
4 - Efectuar el ordenamiento inverso.
5 - Fin

6.3. Algoritmo del Mínimo Vecino

Este algoritmo es una variante del algoritmo de grado mínimo que opera eliminando los nodos seleccionados en la estructura del grafo asociado a la matriz y de forma que no se define ni se inserta en el grafo ninguna nueva conexión. Este algoritmo selecciona el nodo que tiene el menor número de vecinos. Es especialmente útil cuando se hace una factorización incompleta con el mismo

patrón de *sparsidad* que la matriz del sistema, por ejemplo cuando se utilizan precondicionadores como el ILU(0).

ALGORITMO MN
1 - Construir el grafo asociado a la matriz $\mathbf{A}$, $g(x) = \langle V, E\rangle$, donde V es el conjunto de nodos y $E = \{\{a, b\} : a \neq b \ / \ a, b \in V\}$.
2 - Mientras $V \neq \emptyset$:
  2.1- Elegir un nodo v de grado mínimo en $g(x) = \langle V, E\rangle$ y reordenar como nodo siguiente.
  2.2 - Definir:
    $V_v = V - \{v\}\,, \ \ E_v = \{\{a, b\} \in E \ / \ a, b \in V_v\}$.
    y hacer
    $V = V_v, \ \ E = E_v$ y $g(x) = \langle V, E\rangle$.
3 - Fin

6.4. Algoritmo de George

La elección del nodo inicial en el segundo paso de los algoritmos anteriores se ha determinado usando el algoritmo de George [29, 65], lo que nos permite comenzar desde un nodo pseudo-periférico.

Si definimos la distancia $d(x, y)$ entre dos nodos x e y en un grafo $g(x) = \langle V, E\rangle$, como la longitud de la trayectoria más corta que une ambos nodos, y la excentricidad de un nodo x por $\varepsilon(x) = Max\,\{d(x, y)/x, y \in V\}$, el algoritmo se escribe de la siguiente forma,

ALGORITMO DE GEORGE PARA LA BÚSQUEDA
DE NODOS PSEUDO-PERIFÉRICOS
1- Elegir un nodo arbitrario r de V.
2 . Generar una estructura con niveles enraizada en r,

$$\left\{L_0(r), L_1(r), \ldots, L_{\varepsilon(r)}(r)\right\}.$$

siendo $L_i(r) = \{x/d(x, r) = i\}$.
3 - Elegir un nodo x de grado mínimo en $L_{\varepsilon(r)}(r)$.
4 . Generar una estructura con niveles enraizada en x,

$$\left\{L_0(x), L_1(x), \ldots, L_{\varepsilon(x)}(x)\right\}$$

5 - Si $\varepsilon(x) > \varepsilon(r)$, establecer $x \to r$ y volver al paso 3.
6 - Caso contrario tomamos x como nodo inicial.
7 - Fin

Capítulo 7

Algoritmos precondicionados

En este capítulo se presenta la versión precondicionada de los algoritmos basados en los subespacios de Krylov desarrollados en el capítulo 3.

Cada sistema es transformado en otro precondicionado para ser resuelto mediante un método de Krylov [81]. A medida que se desarrolla el algoritmo, se efectúan transformaciones, introduciendo nuevos vectores y se aprovechan las características de la matriz de precondicionamiento que debe cumplir unos requisitos generales que garanticen su efectividad. Esto es,

- La matriz de precondicionamiento nunca es multiplicada explícitamente por la matriz del sistema $\mathbf{A}$.

- A pesar de estar resolviendo el sistema precondicionado $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$, el algoritmo debe proporcionar la solución del sistema original $\mathbf{x}$.

- El criterio de parada se define a partir de los vectores residuos del sistema original sin precondicionar $(\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x})$.

7.1. Algoritmo CG

La aplicación del algoritmo del Gradiente Conjugado requiere que la matriz del sistema a resolver sea simétrica y definida positiva, pero aunque estemos resolviendo un sistema como el (2.1) que cumpla estas condiciones, al aplicar una técnica de precondicionamiento, el nuevo sistema no sabemos si las cumplirá. Es decir, que si por ejemplo vamos a resolver el sistema $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$, la matriz $\mathbf{M}^{-1}\mathbf{A}$ debe ser simétrica y definida positiva para poder usar el algoritmo CG.

Una alternativa es reemplazar el producto escalar euclídeo usual en el algoritmo por otro producto interno de matriz $\mathbf{M}$, ya que como,

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{M}} = \langle \mathbf{M}\mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{M}\mathbf{y} \rangle$$

entonces,

$$\begin{aligned}\left\langle \mathbf{M}^{-1}\mathbf{A}\mathbf{x},\mathbf{y}\right\rangle_{\mathbf{M}} &= \left\langle \mathbf{M}\left(\mathbf{M}^{-1}\mathbf{A}\right)\mathbf{x},\mathbf{y}\right\rangle = \left\langle \mathbf{A}\mathbf{x},\mathbf{y}\right\rangle = \left\langle \mathbf{x},\mathbf{A}\mathbf{y}\right\rangle = \\ &= \left\langle \mathbf{x},\mathbf{M}\left(\mathbf{M}^{-1}\mathbf{A}\right)\mathbf{y}\right\rangle = \left\langle \mathbf{x},\mathbf{M}^{-1}\mathbf{A}\mathbf{y}\right\rangle_{\mathbf{M}}\end{aligned}$$

Reescribiendo el algoritmo CG con este nuevo producto interno, siendo el residuo del sistema original $\mathbf{r}_j = \mathbf{b} - \mathbf{A}\mathbf{x}_j$, y llamando $\mathbf{z}_j = \mathbf{M}^{-1}\mathbf{r}_j$ al residuo del sistema precondicionado obtendremos,

1. $\alpha_j = \dfrac{\left\langle \mathbf{z}_j,\mathbf{z}_j\right\rangle_{\mathbf{M}}}{\left\langle \mathbf{M}^{-1}\mathbf{A}\mathbf{p}_j,\mathbf{p}_j\right\rangle_{\mathbf{M}}}$
2. $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$
3. $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j\mathbf{A}\mathbf{p}_j$ y $\mathbf{z}_{j+1} = \mathbf{M}^{-1}\mathbf{r}_{j+1}$
4. $\beta_j = \dfrac{\left\langle \mathbf{z}_{j+1},\mathbf{z}_{j+1}\right\rangle_{\mathbf{M}}}{\left\langle \mathbf{z}_j,\mathbf{z}_j\right\rangle_{\mathbf{M}}}$
5. $\mathbf{p}_{j+1} = \mathbf{z}_{j+1} + \beta_j\mathbf{p}_j$

Y como $\left\langle \mathbf{z}_j,\mathbf{z}_j\right\rangle_{\mathbf{M}} = \left\langle \mathbf{r}_j,\mathbf{z}_j\right\rangle$ y $\left\langle \mathbf{M}^{-1}\mathbf{A}\mathbf{p}_j,\mathbf{p}_j\right\rangle_{\mathbf{M}} = \left\langle \mathbf{A}\mathbf{p}_j,\mathbf{p}_j\right\rangle$, el algoritmo precondicionado resulta,

ALGORITMO GRADIENTE CONJUGADO PRECONDICIONADO (PCG)
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$, $\mathbf{p}_0 = \mathbf{z}_0$;
Mientras $\| \mathbf{r}_j \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (j = 0, 1, 2, 3, ...)$, hacer
$\alpha_j = \dfrac{\left\langle \mathbf{r}_j,\mathbf{z}_j\right\rangle}{\left\langle \mathbf{A}\mathbf{p}_j,\mathbf{p}_j\right\rangle}$;
$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$;
$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j\mathbf{A}\mathbf{p}_j$;
Resolver $\mathbf{M}\mathbf{z}_{j+1} = \mathbf{r}_{j+1}$;
$\beta_j = \dfrac{\left\langle \mathbf{r}_{j+1},\mathbf{z}_{j+1}\right\rangle}{\left\langle \mathbf{r}_j,\mathbf{z}_j\right\rangle}$;
$\mathbf{p}_{j+1} = \mathbf{z}_{j+1} + \beta_j\mathbf{p}_j$;
Fin

Si utilizamos como matriz de precondicionamiento $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, obtenida por una factorización incompleta de Cholesky de la matriz $\mathbf{A}$ del sistema original, entonces,

$$\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}\mathbf{u} = \mathbf{L}^{-1}\mathbf{b}, \quad \mathbf{x} = \mathbf{L}^{-T}\mathbf{u} \tag{7.1}$$

Definiendo las siguientes matrices y vectores auxiliares,

$$\begin{aligned}\widehat{\mathbf{p}}_j &= \mathbf{L}^T\mathbf{p}_j \\ \mathbf{u}_j &= \mathbf{L}^T\mathbf{x}_j \\ \widehat{\mathbf{r}}_j &= \mathbf{L}^T\mathbf{z}_j = \mathbf{L}^{-1}\mathbf{r}_j \\ \widehat{\mathbf{A}} &= \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}\end{aligned}$$

y como,

$$\langle \mathbf{r}_j, \mathbf{z}_j \rangle = \left\langle \mathbf{r}_j, \mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{z}_j \right\rangle = \left\langle \mathbf{L}^{-1}\mathbf{r}_j, \mathbf{L}^{-1}\mathbf{r}_j \right\rangle = \langle \widehat{\mathbf{r}}_j, \widehat{\mathbf{r}}_j \rangle$$

$$\left\langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j \right\rangle = \left\langle \mathbf{A}\mathbf{L}^{-T}\widehat{\mathbf{p}}_j, \mathbf{L}^{-T}\widehat{\mathbf{p}}_j \right\rangle \left\langle \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}\widehat{\mathbf{p}}_j, \widehat{\mathbf{p}}_j \right\rangle = \left\langle \widehat{\mathbf{A}}\,\widehat{\mathbf{p}}_j, \widehat{\mathbf{p}}_j \right\rangle$$

todos los pasos del algoritmo anterior pueden ser reescritos con estas nuevas variables,

1. $\alpha_j = \dfrac{\langle \widehat{\mathbf{r}}_j, \widehat{\mathbf{r}}_j \rangle}{\left\langle \widehat{\mathbf{A}}\widehat{\mathbf{p}}_j, \widehat{\mathbf{p}}_j \right\rangle}$
2. $\mathbf{u}_{j+1} = \mathbf{u}_j + \alpha_j \widehat{\mathbf{p}}_j$
3. $\widehat{\mathbf{r}}_{j+1} = \widehat{\mathbf{r}}_j - \alpha_j \widehat{\mathbf{A}}\widehat{\mathbf{p}}_j$
4. $\beta_j = \dfrac{\langle \widehat{\mathbf{r}}_{j+1}, \widehat{\mathbf{r}}_{j+1} \rangle}{\langle \widehat{\mathbf{r}}_j, \widehat{\mathbf{r}}_j \rangle}$
5. $\widehat{\mathbf{p}}_{j+1} = \widehat{\mathbf{r}}_{j+1} + \beta_j \widehat{\mathbf{p}}_j$

que corresponde al algoritmo del Gradiente Conjugado aplicado al sistema precondicionado $\widehat{\mathbf{A}}\mathbf{u} = \mathbf{L}^{-1}\mathbf{b}$, donde $\mathbf{u} = \mathbf{L}^T\mathbf{x}$. El algoritmo puede escribirse referido a $\mathbf{x}$ y las direcciones conjugadas originales, de la siguiente forma,

ALGORITMO GRADIENTE CONJUGADO CON PRECONDICIONADOR FACTORIZADO (SPLIT-PCG)

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Resolver $\mathbf{L}\widehat{\mathbf{r}}_0 = \mathbf{r}_0$;
Resolver $\mathbf{L}^T\mathbf{p}_0 = \widehat{\mathbf{r}}_0$;
Mientras $\| \widehat{\mathbf{r}}_j \| \, / \, \| \mathbf{r}_0 \| \geq \varepsilon \quad (j = 0, 1, 2, 3, ...)$, hacer
$\alpha_j = \dfrac{\langle \widehat{\mathbf{r}}_j, \widehat{\mathbf{r}}_j \rangle}{\left\langle \mathbf{A}\mathbf{p}_j, \mathbf{p}_j \right\rangle}$;
$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
Resolver $\mathbf{L}\mathbf{u}_j = \mathbf{A}\mathbf{p}_j$;
$\widehat{\mathbf{r}}_{j+1} = \widehat{\mathbf{r}}_j - \alpha_j \mathbf{u}_j$;
$\beta_j = \dfrac{\langle \widehat{\mathbf{r}}_{j+1}, \widehat{\mathbf{r}}_{j+1} \rangle}{\langle \widehat{\mathbf{r}}_j, \widehat{\mathbf{r}}_j \rangle}$;
Resolver $\mathbf{L}^T\mathbf{v}_{j+1} = \widehat{\mathbf{r}}_{j+1}$;
$\mathbf{p}_{j+1} = \mathbf{v}_{j+1} + \beta_j \mathbf{p}_j$;
Fin

Analogamente se puede obtener un algoritmo equivalente para el CG precondicionado por la derecha sin más que considerar los productos internos referidos a la matriz $\mathbf{M}^{-1}$.

7.2. Algoritmo Flexible GMRES (FGMRES)

El Flexible GMRES [71] es una versión que admite varios precondicionadores en cada paso, tal que si $\mathbf{M}_j,\ j = 1, ..., k$ es un conjunto de matrices no singulares de precondicionamiento, el algoritmo se puede escribir como,

ALGORITMO FGMRES
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; Elegir k
Mientras $\| \mathbf{r}_{i-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer
  Resolver $\mathbf{M}_0 \overline{\mathbf{r}}_{i-1} = \mathbf{r}_{i-1}$;
  $\beta_{i-1} = \| \overline{\mathbf{r}}_{i-1} \|$;
  $\mathbf{v}_1 = \frac{1}{\beta_{i-1}} \overline{\mathbf{r}}_{i-1}$;
  Desde $j = 1, ..., k$ hacer
    Resolver $\mathbf{M}_j \mathbf{z}_j = \mathbf{v}_j$;
    $\mathbf{w} = \mathbf{A}\mathbf{z}_j$;
    Desde $n = 1, ..., j$ hacer
      $\{\mathbf{H}\}_{nj} = \langle \mathbf{w}, \mathbf{v}_n \rangle$;
      $\mathbf{w} = \mathbf{w} - \{\mathbf{H}\}_{nj} \mathbf{v}_n$;
    Fin
    $\{\mathbf{H}\}_{j+1j} = \| \mathbf{w} \|$;
    $\mathbf{v}_{j+1} = \frac{1}{\{\mathbf{H}\}_{j+1j}} \mathbf{w}$;
  Fin
Encontrar $\mathbf{u}_k$, tal que $\|\beta_{i-1}\mathbf{e}_1 - \mathbf{H}_k \mathbf{u}\|_2$ sea mínimo, donde $\mathbf{e}_1$ es el primer vector de la base canónica en R^{k+1}, y $\mathbf{H}_k$ es la matriz $(k+1) \times k$ cuyos elementos son los $\{\mathbf{H}\}_{nj}$ definidos en el algoritmo.
  $\mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{V}_k \mathbf{u}_k$; siendo $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$;
  $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i$;
Fin

7.3. Algoritmo FGMRES Modificado

Esta modificación del FGMRES consiste en utilizar un método directo para la resolución del problema de mínimos cuadrados que aparece en cada iteración [25] en lugar de la tradicional factorización QR [72] u otra implementación basada en la transformación de Householder [90].

Considérese la proyección ortogonal sobre el subespacio de soluciones mediante la multiplicación por la matriz $\mathbf{H}_k^t$:

$$\mathbf{H}_k^t \, \mathbf{H}_k \, \mathbf{u} = \mathbf{H}_k^t \, \beta_{i-1} \mathbf{e}_1 \tag{7.2}$$

La forma de $\mathbf{H}_k$ sugiere descomponer el producto $\mathbf{H}_k^t \, \mathbf{H}_k$ en una suma. En efecto, se advierte fácilmente que $\mathbf{H}_k$ es una matriz $(k+1) \times k$ con la siguiente

estructura:

$$\mathbf{H}_k = \begin{pmatrix} \boxed{\mathbf{d}_k^t} \\ \boxed{\begin{array}{c} \mathbf{U}_k \\ \\ \mathbf{0} \end{array}} \end{pmatrix}$$

La primera fila está definida por un vector de dimensión k ($\mathbf{d}_k^t$) y el resto forma una matriz cuadrada triangular superior $\mathbf{U}_k$:

$$\{\mathbf{d}_k\}_i = d_i = \{\mathbf{H}\}_{1i} \qquad i = 1, ..., k \tag{7.3}$$

$$\{\mathbf{U}_k\}_{ij} = u_{ij} = \begin{cases} \{\mathbf{H}\}_{i+1\,j} & 1 \leq i \leq j \leq k \\ 0 & \text{en el resto} \end{cases} \tag{7.4}$$

Teorema. Sean $\mathbf{d}_k$ y $\mathbf{U}_k$ definidos por (7.3) y (7.4), y $\bar{\mathbf{p}}_k$, $\mathbf{p}_k$, respectivamente, las soluciones de los sistemas triangulares $\mathbf{U}_k^t \bar{\mathbf{p}}_k = \mathbf{d}_k$ y $\mathbf{U}_k \mathbf{p}_k = \bar{\mathbf{p}}_k$. Si se define,

$$\lambda_i = \frac{\beta_{i-1}}{1 + \langle \mathbf{d}_k,\, \mathbf{p}_k \rangle}; \qquad \mathbf{u}_k = \lambda_i\, \mathbf{p}_k$$

entonces $\mathbf{u}_k$ minimiza la forma cuadrática $\mathbf{J}(\mathbf{u}) = \|\beta_{i-1}\mathbf{e}_1 - \mathbf{H}_k\, \mathbf{u}\|_2$ sobre R^k.

Demostración: Obsérvese en primer lugar que $1 + \langle \mathbf{d}_k,\, \mathbf{p}_k \rangle \neq 0$, ya que

$$\langle \mathbf{d}_k,\, \mathbf{p}_k \rangle = \left\langle \mathbf{U}_k^t \mathbf{U}_k \mathbf{p}_k,\, \mathbf{p}_k \right\rangle = \|\mathbf{U}_k \mathbf{p}_k\|_2^2 \geq 0$$

y, por tanto, λ_i nunca puede degenerar.

Consideremos ahora el sistema lineal dado en (7.2). El (i, j)-ésimo elemento de $\mathbf{H}_k^t\, \mathbf{H}_k$ es:

$$\left\{\mathbf{H}_k^t\, \mathbf{H}_k\right\}_{ij} = d_i d_j + \sum_{m=1}^{k} u_{mi} u_{mj} \tag{7.5}$$

En definitiva, se puede expresar,

$$\left(\mathbf{d}_k \mathbf{d}_k^t + \mathbf{U}_k^t \mathbf{U}_k\right) \mathbf{u} = \mathbf{H}_k^t\, \beta_{i-1} \mathbf{e}_1 \tag{7.6}$$

Teniendo en cuenta que $\mathbf{H}_k^t \mathbf{e}_1 = \mathbf{d}_k$, se obtiene la siguiente formulación:

$$\left(\mathbf{d}_k \mathbf{d}_k^t + \mathbf{U}_k^t \mathbf{U}_k\right) \mathbf{u} = \beta_{i-1} \mathbf{d}_k \tag{7.7}$$

Si pasamos al segundo miembro el producto externo y aplicamos la propiedad asociativa de la multiplicación de matrices, se obtiene,

$$\mathbf{U}_k^t \mathbf{U}_k \mathbf{u} = \mathbf{d}_k \left(\beta_{i-1} - \langle \mathbf{d}_k, \mathbf{u} \rangle\right) \tag{7.8}$$

Si definimos,

$$\lambda_i = \beta_{i-1} - \langle \mathbf{d}_k, \mathbf{u} \rangle \tag{7.9}$$

$$\mathbf{u} = \lambda_i \mathbf{p}_k \tag{7.10}$$

entonces se tiene que,

$$\mathbf{U}_k^t \mathbf{U}_k \mathbf{p}_k = \mathbf{d}_k \tag{7.11}$$

Esto supone resolver sólo dos sistemas triangulares, ya que $\mathbf{U}_k^t$ y $\mathbf{U}_k$ son matrices triangulares inferiores y superiores, respectivamente. Una vez resuelto este sistema, se necesita calcular λ_i para obtener $\mathbf{u}$ en la ecuación (7.10). Sustituyendo (7.10) en (7.9) se llega a,

$$\lambda_i = \beta_{i-1} - \langle \mathbf{d}_k, \mathbf{u} \rangle = \beta_{i-1} - \lambda_i \langle \mathbf{d}_k, \mathbf{p}_k \rangle$$

y por tanto, se demuestra que

$$\lambda_i = \frac{\beta_{i-1}}{1 + \langle \mathbf{d}_k,\, \mathbf{p}_k \rangle} \tag{7.12}$$

El cálculo del nuevo residuo se basa en el siguiente resultado [71],

$$\mathbf{r}_i = \mathbf{V}_{k+1} \left(\beta_{i-1} \mathbf{e}_1 - \mathbf{H}_k \, \mathbf{u} \right) \tag{7.13}$$

donde se sabe que $\mathbf{V}_{k+1} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{k+1}]$ y es unitaria. Así, se puede escribir,

$$\mathbf{r}_i = \mathbf{V}_{k+1} \hat{\mathbf{r}}_i \tag{7.14}$$

representando $\hat{\mathbf{r}}_i$ el siguiente vector de dimensión $(k+1)$,

$$\hat{\mathbf{r}}_i = \beta_{i-1} \mathbf{e}_1 - \mathbf{H}_k \, \mathbf{u} \tag{7.15}$$

Además, es evidente que,

$$\|\mathbf{r}_i\|_2 = \|\hat{\mathbf{r}}_i\|_2 \tag{7.16}$$

A partir de la descomposición de $\mathbf{H}_k$, la primera componente del vector $(\mathbf{H}_k \, \mathbf{u})$ de dimensión $(k+1)$ es el producto escalar $\langle \mathbf{d}_k, \mathbf{u} \rangle$, y el resto de las componentes vienen dadas por el vector $(\mathbf{U}_k \mathbf{u})$ de dimensión k. Por tanto, la primera componente de $\hat{\mathbf{r}}_i$ es λ_i, y las otras,

$$-\mathbf{U}_k \mathbf{u} = -\lambda_i \mathbf{U}_k \mathbf{p}_k = -\lambda_i \bar{\mathbf{p}}_k \tag{7.17}$$

siendo posible almacenar el vector $\bar{\mathbf{p}}_k$ después del primer remonte al resolver (7.11).

Con estos cambios, el algoritmo FGMRES Modificado se formula como sigue:

ALGORITMO FGMRES-MODIFICADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; Elegir k
Mientras $\| \mathbf{r}_{i-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer
Resolver $\mathbf{M}_0 \overline{\mathbf{r}}_{i-1} = \mathbf{r}_{i-1}$;
$\beta_{i-1} = \| \overline{\mathbf{r}}_{i-1} \|$;
$\mathbf{v}_1 = \frac{1}{\beta_{i-1}} \overline{\mathbf{r}}_{i-1}$;
Desde $j = 1, ..., k$ hacer
$\mathbf{z}_j = \mathbf{A}\mathbf{v}_j$;
Resolver $\mathbf{M}_j \mathbf{w} = \mathbf{z}_j$;
Desde $n = 1, ..., j$ hacer
$\{\mathbf{H}\}_{nj} = \langle \mathbf{w}, \mathbf{v}_n \rangle$;
$\mathbf{w} = \mathbf{w} - \{\mathbf{H}\}_{nj} \mathbf{v}_n$;
Fin
$\{\mathbf{H}\}_{j+1j} = \| \mathbf{w} \|$;
$\mathbf{v}_{j+1} = \frac{1}{\{\mathbf{H}\}_{j+1j}} \mathbf{w}$;
Fin
Resolver $\mathbf{U}_k^t \bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k \mathbf{p} = \bar{\mathbf{p}}$;
donde $\begin{matrix} \{\mathbf{d}_k\}_m = \{\mathbf{H}\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \{\mathbf{H}\}_{l+1m} \end{matrix} \quad l, m = 1, ..., k$;
$\lambda_i = \dfrac{\beta_{i-1}}{1 + \langle \mathbf{d}_k, \mathbf{p} \rangle}$;
$\mathbf{u}_k = \lambda_i \mathbf{p}$;
$\mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{V}_k \mathbf{u}_k$; siendo $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$;
$\mathbf{r}_i = \mathbf{V}_{k+1} \hat{\mathbf{r}}_i$; siendo $\mathbf{V}_{k+1} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{k+1}]$;
donde $\begin{matrix} \{\hat{\mathbf{r}}_i\}_1 = \lambda_i \\ \{\hat{\mathbf{r}}_i\}_{l+1} = -\lambda_i \{\bar{\mathbf{p}}\}_l \end{matrix} \quad l = 1, ..., k$;
Fin

7.4. Algoritmo Variable FGMRES (VFGMRES)

El algoritmo Variable GMRES es una variante del FGMRES-Modificado. Además de la resolución directa del problema de mínimos cuadrados [25] la idea básica es usar el *full* GMRES al principio del algoritmo hasta que satisface una cierta subtolerancia δ que puede ser función de ε, la tolerancia exigida a la solución, y se incrementa la dimensión del subespacio de Krylov k una unidad en cada paso mientras que la norma del vector residuo sea mayor o igual a δ y k sea menor que la dimensión máxima permitida (por ejemplo, por exigencias de memoria) del subespacio de Krylov k_{top}. A partir de este punto, con ese último valor de k, comenzar el algoritmo como si se tratase del estándar *restarted* GMRES hasta alcanzar la tolerancia ε dada (ver Galán y otros [24]).

ALGORITMO VFGMRES
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Elegir k_{init}, k_{top}, $\delta \in [0,1]$, $k = k_{init}$
Mientras $\| \widehat{\mathbf{r}}_{i-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \;\; (i = 1, 2, 3, ...)$,
$\beta_{i-1} = \| \mathbf{r}_{i-1} \|$, $\mathbf{v}_i = \mathbf{r}_{i-1}/\beta_{i-1}$;
Si $\| \mathbf{r}_{i-1} \| / \| \mathbf{r}_0 \| \geq \delta$ y $k < k_{top}$ hacer $k = k + 1$;
Para $j = 1, ..., k$ hacer
Resolver $\mathbf{M}\mathbf{z}_j = \mathbf{v}_j$;
$\mathbf{w} = \mathbf{A}\mathbf{z}_j$;
Para $n = 1, ..., j$ hacer
$\{\mathbf{H}\}_{nj} = \mathbf{w}^t \mathbf{v}_n$;
$\mathbf{w} = \mathbf{w} - \{\mathbf{H}\}_{nj} \mathbf{v}_n$;
Fin
$\{\mathbf{H}\}_{j+1,j} = \| \mathbf{w} \|$;
$\mathbf{v}_{j+1} = \mathbf{w} / \{\mathbf{H}\}_{j+1,j}$:
Fin
Resolver $\mathbf{U}_k^t \bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k \mathbf{p} = \bar{\mathbf{p}}$;
con $\begin{matrix} \{\mathbf{d}_k\}_m = \{\mathbf{H}\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \{\mathbf{H}\}_{l+1,m} \end{matrix} \quad l, m = 1, ..., k$;
$\lambda_i = \dfrac{\beta_{i-1}}{1 + \mathbf{d}_k^t \mathbf{p}}$;
$\mathbf{u}_k = \lambda_i \mathbf{p}$;
$\mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{Z}_k \mathbf{u}_k$; siendo $\mathbf{Z}_k = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k]$;
$\mathbf{r}_i = \mathbf{Z}_{k+1} \hat{\mathbf{r}}_i$; con $\begin{matrix} \{\hat{\mathbf{r}}_i\}_1 = \lambda_i \\ \{\hat{\mathbf{r}}_i\}_{l+1} = -\lambda_i \{\bar{\mathbf{p}}\}_l \end{matrix} \quad l = 1, ..., k$;
Fin

Este algoritmo, al igual que el FGMRES, permite cambiar el precondicionador en cada iteración.

7.5. Algoritmo Bi-CG

Precondicionando por la izquierda el algoritmo Bi-CG obtenemos,

ALGORITMO BI-CG PRECONDICIONADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Elegir $\mathbf{r}_0^*$ tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
$\mathbf{p}_0 = \mathbf{z}_0$, $\mathbf{p}_0^* = \mathbf{r}_0^*$
Mientras $\| \mathbf{r}_{j-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \;\; (j = 1, 2, 3, ...)$, hacer
Resolver $\mathbf{M}\mathbf{z}_j = \mathbf{r}_j$;
$\mathbf{t}_j = \mathbf{A}\mathbf{p}_j$;
Resolver $\mathbf{M}\mathbf{v}_j = \mathbf{t}_j$;

$\alpha_j = \dfrac{\langle \mathbf{z}_j, \mathbf{r}_j^* \rangle}{\langle \mathbf{v}_j, \mathbf{p}_j^* \rangle}$;
$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$;
$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$;
Resolver $\mathbf{M}\mathbf{z}_{j+1} = \mathbf{r}_{j+1}$;
$\mathbf{r}_{j+1}^* = \mathbf{r}_j^* - \alpha_j \mathbf{A}^T \mathbf{p}_j^*$;
$\beta_j = \dfrac{\langle \mathbf{z}_{j+1}, \mathbf{r}_{j+1}^* \rangle}{\langle \mathbf{z}_j, \mathbf{r}_j^* \rangle}$;
$\mathbf{p}_{j+1} = \mathbf{z}_{j+1} + \beta_j \mathbf{p}_j$;
$\mathbf{p}_{j+1}^* = \mathbf{r}_{j+1}^* + \beta_j \mathbf{p}_j^*$;
Fin

7.6. Algoritmo CGS

Si se aplica el método CGS a los sistemas precondicionados [54], se obtienen tres algoritmos que se corresponden con cada una de las formas de precondicionamiento. Sin embargo la única diferencia entre ellos estriba en el vector residuo inicial $\mathbf{r}_0^*$ que influye en cada iteración, siendo la relación con el vector residuo inicial del sistema auxiliar precondicionado,

$$\left.\begin{array}{ll} \mathbf{r}_0^* = \tilde{\mathbf{r}}_0^* & \text{precondicionamiento por la derecha} \\ \mathbf{r}_0^* = \mathbf{M}^{-T}\tilde{\mathbf{r}}_0^* & \text{precondicionamiento por la izquierda} \\ \mathbf{r}_0^* = \mathbf{L}^{-T}\tilde{\mathbf{r}}_0^* & \text{precondicionamiento por ambos lados} \end{array}\right\} \tag{7.18}$$

Al ser $\tilde{\mathbf{r}}_0^*$ arbitrario, se puede elegir de un modo adecuado para cada forma de precondicionamiento tal que un algoritmo se transforma en otro. La conclusión es que las diferentes formas de precondicionamiento son equivalentes a determinadas elecciones del vector inicial $\mathbf{r}_0^*$ [82].

ALGORITMO CGS PRECONDICIONADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Elegir $\mathbf{r}_0^*$ arbitrario tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
$\mathbf{p}_0 = \mathbf{u}_0 = \mathbf{z}_0$;
Mientras $\| \mathbf{r}_{j-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (j = 1, 2, 3, ...)$, hacer
Resolver $\mathbf{M}\mathbf{z}_j = \mathbf{r}_j$;
$\mathbf{t}_j = \mathbf{A}\mathbf{p}_j$;
Resolver $\mathbf{M}\mathbf{v}_j = \mathbf{t}_j$;
$\alpha_j = \dfrac{\langle \mathbf{z}_j, \mathbf{r}_0^* \rangle}{\langle \mathbf{v}_j, \mathbf{r}_0^* \rangle}$;
$\mathbf{q}_j = \mathbf{u}_j - \alpha_j \mathbf{v}_j$;
$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j (\mathbf{u}_j + \mathbf{q}_j)$;
$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A} (\mathbf{u}_j + \mathbf{q}_j)$;

Resolver $\mathbf{M}\mathbf{z}_{j+1} = \mathbf{r}_{j+1}$;
$\beta_j = \dfrac{\langle \mathbf{z}_{j+1}, \mathbf{r}_0^* \rangle}{\langle \mathbf{z}_j, \mathbf{r}_0^* \rangle}$;
$\mathbf{u}_{j+1} = \mathbf{z}_{j+1} + \beta_j \mathbf{q}_j$;
$\mathbf{p}_{j+1} = \mathbf{u}_{j+1} + \beta_j (\mathbf{q}_j + \beta_j \mathbf{p}_j)$;
Fin

7.7. Algoritmo BICGSTAB

El método BICGSTAB es una variante del CGS. Introduce un nuevo parámetro $\widetilde{\omega}$ en cada iteración para minimizar el residuo (ver [85]). Para cada forma de precondicionamiento [54], el valor del parámetro $\widetilde{\omega}$ resulta:

$$\left.\begin{array}{ll} \widetilde{\omega} = \dfrac{(\mathbf{A}\mathbf{s})^T \mathbf{s}}{(\mathbf{A}\mathbf{s})^T \mathbf{A}\mathbf{s}} & \text{precondicionamiento por la derecha} \\ \widetilde{\omega} = \dfrac{(\mathbf{M}^{-1}\mathbf{A}\mathbf{s})^T (\mathbf{M}^{-1}\mathbf{s})}{(\mathbf{M}^{-1}\mathbf{A}\mathbf{s})^T (\mathbf{M}^{-1}\mathbf{A}\mathbf{s})} & \text{precondicionamiento por la izquierda} \\ \widetilde{\omega} = \dfrac{(\mathbf{L}^{-1}\mathbf{A}\mathbf{s})^T (\mathbf{L}^{-1}\mathbf{s})}{(\mathbf{L}^{-1}\mathbf{A}\mathbf{s})^T (\mathbf{L}^{-1}\mathbf{A}\mathbf{s})} & \text{precondicionamiento por ambos lados} \end{array}\right\} \quad (7.19)$$

De este modo, el cálculo de $\widetilde{\omega}$ incluye un proceso de sustitución por iteración para el precondicionamiento por la izquierda y dos para el precondicionamiento por ambos lados. No obstante, si obtenemos $\widetilde{\omega}$ a partir de la minimización del residuo del sistema original sin precondicionar, todos los valores dados en la ecuación (7.19) coinciden con el del precondicionamiento por la derecha. En este caso se obtiene también un único algoritmo,

ALGORITMO BICGSTAB PRECONDICIONADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Elegir $\mathbf{r}_0^*$ arbitrario tal $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
$\mathbf{p}_0 = \mathbf{z}_0$;
Mientras $\| \mathbf{r}_{j-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (j = 1, 2, 3, ...)$, hacer
Resolver $\mathbf{M}\mathbf{z}_j = \mathbf{r}_j$;
$\mathbf{y}_j = \mathbf{A}\mathbf{p}_j$;
Resolver $\mathbf{M}\mathbf{v}_j = \mathbf{y}_j$;
$\alpha_j = \dfrac{\langle \mathbf{z}_j, \mathbf{r}_0^* \rangle}{\langle \mathbf{v}_j, \mathbf{r}_0^* \rangle}$;
$\mathbf{s}_j = \mathbf{r}_j - \alpha_j \mathbf{y}_j$;
$\mathbf{u}_j = \mathbf{A}\mathbf{s}_j$;
Resolver $\mathbf{M}\mathbf{t}_j = \mathbf{u}_j$;
$\widetilde{\omega}_j = \dfrac{\langle \mathbf{t}_j, \mathbf{s}_j \rangle}{\langle \mathbf{t}_j, \mathbf{t}_j \rangle}$;

$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j + \widetilde{\omega}_j \mathbf{u}_j$;
$\mathbf{z}_{j+1} = \mathbf{s}_j - \widetilde{\omega}_j \mathbf{t}_j$;
$\beta_j = \dfrac{\langle \mathbf{z}_{j+1}, \mathbf{r}_0^* \rangle}{\langle \mathbf{z}_j, \mathbf{r}_0^* \rangle} \times \dfrac{\alpha_j}{\widetilde{\omega}_j}$;
$\mathbf{p}_{j+1} = \mathbf{z}_{j+1} + \beta_j \left(\mathbf{p}_j - \widetilde{\omega}_j \mathbf{v}_j \right)$;
Fin

Cualquier otra elección del residuo inicial conducirá a una nueva forma de precondicionamiento (ver [82]).

7.8. Algoritmo QMR

Precondicionando por la izquierda el algoritmo QMR, obtenemos,

ALGORITMO QMR
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
$\rho_1 = \| \mathbf{z}_0 \|$ y ; $\mathbf{v}_1 = \mathbf{z}_0 / \rho_1$;
Elegir $\mathbf{w}_1$ tal que $\| \mathbf{w}_1 \| = 1$;
$\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{d}_0 = 0$;
$c_0 = \epsilon_0 = \xi_1 = 1$;
$\nu_0 = 0$, $\eta_0 = -1$;
Mientras $\sqrt{j+1}\,|\eta_{j-1}| \,/\, \|\mathbf{r}_0\| \geq \varepsilon (j = 1, 2, ...)$, hacer:
$\delta_j = \langle \mathbf{v}_j, \mathbf{w}_j \rangle$,
$\mathbf{p}_j = \mathbf{v}_j - (\xi_j \delta_j / \epsilon_{j-1})\, \mathbf{p}_{j-1}$;
$\mathbf{q}_j = \mathbf{w}_j - (\rho_j \delta_j / \epsilon_{j-1})\, \mathbf{q}_{j-1}$,
$\mathbf{u}_j = \mathbf{A}\mathbf{p}_j$;
Resolver $\mathbf{M}\mathbf{s}_j = \mathbf{u}_j$;
$\epsilon_j = \langle \mathbf{s}_j, \mathbf{q}_j \rangle$;
$\beta_j = \epsilon_j / \delta_j$;
$\widehat{\mathbf{v}}_{j+1} = \mathbf{s}_j - \beta_j \mathbf{v}_j$, $\rho_{j+1} = \|\widehat{\mathbf{v}}_{j+1}\|$;
Resolver $\mathbf{M}^T \mathbf{t}_j = \mathbf{q}_j$;
$\widehat{\mathbf{w}}_{j+1} = \mathbf{A}^T \mathbf{t}_j - \beta_j \mathbf{w}_j$, $\xi_{j+1} = \|\widehat{\mathbf{w}}_{j+1}\|$;
$\nu_j = \dfrac{\rho_{j+1}}{c_{j-1}\,|\beta_j|}$, $c_j = \dfrac{1}{\sqrt{1+\nu_j^2}}$, $\eta_j = -\eta_{j-1} \dfrac{\rho_j c_j^2}{\beta_j c_{j-1}^2}$;
$\mathbf{d}_j = \eta_j \mathbf{p}_j + (\nu_{j-1} c_j)^2\, \mathbf{d}_{j-1}$;
$\mathbf{x}_j = \mathbf{x}_{j-1} + \mathbf{d}_j$;
$\mathbf{v}_{j+1} = \widehat{\mathbf{v}}_{j+1} / \rho_{j+1}$;
$\mathbf{w}_{j+1} = \widehat{\mathbf{w}}_{j+1} / \xi_{j+1}$;
Fin

7.9. Algoritmo TFQMR

Aplicando el mismo principio para el precondicionamiento del algoritmo CGS obtenemos el correspondiente algoritmo TFQMR precondicionado.

ALGORITMO TFQMR PRECONDICIONADO

Aproximación inicial $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
$\mathbf{w}_1 = \mathbf{y}_1 = \mathbf{z}_0$;
$\mathbf{u}_1 = \mathbf{A}\mathbf{y}_1$;
Resolver $\mathbf{M}\mathbf{v}_0 = \mathbf{u}_1$
$\mathbf{d}_0 = \mathbf{0}$;
$\tau_0 = \|\mathbf{z}_0\|$, $\vartheta_0 = 0$, $\eta_0 = 0$;
Elegir $\widetilde{\mathbf{r}}_0$ tal que $\rho_0 = \langle \mathbf{z}_0, \widetilde{\mathbf{r}}_0 \rangle \neq 0$;
Mientras $\sqrt{k+1}\tau_{k-1} / \|\mathbf{r}_0\| \geq \varepsilon (k = 1, 2, ...)$, hacer:

(a) $\sigma_{k-1} = \langle \mathbf{v}_{k-1}, \widetilde{\mathbf{r}}_0 \rangle$, $\alpha_{k-1} = \rho_{k-1} / \sigma_{k-1}$;
$\mathbf{y}_{2k} = \mathbf{y}_{2k-1} - \alpha_{k-1}\mathbf{p}_{k-1}$;

(b) Para $m = 2k-1$,$2k$ hacer:
$\mathbf{u}_m = \mathbf{A}\mathbf{y}_m$;
Resolver $\mathbf{M}\mathbf{q}_m = \mathbf{u}_m$
$\mathbf{w}_{m+1} = \mathbf{w}_m - \alpha_{k-1}\mathbf{q}_m$;
$\vartheta_m = \dfrac{\|\mathbf{w}_{m+1}\|}{\tau_{m-1}}$, $c_m = \dfrac{1}{\sqrt{1+\vartheta_m^2}}$;
$\tau_m = \tau_{m-1}\vartheta_m c_m$, $\eta_m = c_m^2 \alpha_{k-1}$;
$\mathbf{d}_m = \mathbf{y}_m + \dfrac{\vartheta_{m-1}^2 \eta_{m-1}}{\alpha_{k-1}} \mathbf{d}_{m-1}$;
$\mathbf{x}_m = \mathbf{x}_{m-1} + \eta_m \mathbf{d}_m$;
Si $\mathbf{x}_m$ converge: Fin

(c) $\rho_k = \langle \mathbf{w}_{2k+1}, \widetilde{\mathbf{r}}_0 \rangle$, $\beta_k = \dfrac{\rho_k}{\rho_{k-1}}$;
$\mathbf{y}_{2k+1} = \mathbf{w}_{2k+1} + \beta_k \mathbf{y}_{2k}$;
$\mathbf{u}_{2k+1} = \mathbf{A}\mathbf{y}_{2k+1}$;
Resolver $\mathbf{M}\mathbf{q}_{2k+1} = \mathbf{u}_{2k+1}$
$\mathbf{v}_k = \mathbf{q}_{2k+1} + \beta_k (\mathbf{q}_{2k} + \beta_k \mathbf{v}_{k-1})$;

Fin

7.10. Algoritmo QMRCGSTAB

Este método aplica el principio de cuasi-minimización al BICGSTAG. La minimización por mínimos cuadrados incluida en el proceso es resuelta usando la descomposición QR de la matriz de Hessemberg de forma incremental con

las rotaciones de Givens. En el algoritmo QMRCGSTAB precondicionado dado a continuación, las rotaciones de Givens son escritas explícitamente como propone originalmente Chan y otros [6].

ALGORITMO QMRCGSTAB PRECONDICIONADO

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;

Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;

Elegir $\widetilde{\mathbf{r}}_0$ tal que $\langle \mathbf{z}_0, \widetilde{\mathbf{r}}_0 \rangle \neq 0$

$\mathbf{p}_0 = \mathbf{v}_0 = \mathbf{d}_0 = \mathbf{0}$;

$\rho_0 = \alpha_0 = \widetilde{\omega}_0 = 1$, $\tau_0 = \|\mathbf{z}_0\|$, $\theta_0 = 0$, $\eta_0 = 0$;

Mientras $\sqrt{j+1}\,|\widetilde{\tau}| \,/\, \|\mathbf{r}_0\| \geq \varepsilon (j = 1, 2, ...)$, hacer:

$\rho_j = \langle \mathbf{z}_{j-1}, \widetilde{\mathbf{r}}_0 \rangle$, $\beta_j = (\rho_j / \rho_{j-1})(\alpha_{j-1} / \tilde{\omega}_{j-1})$;

$\mathbf{p}_j = \mathbf{z}_{j-1} + \beta_j (\mathbf{p}_{j-1} - \tilde{\omega}_{j-1} \mathbf{v}_{j-1})$;

$\mathbf{y}_j = \mathbf{A}\mathbf{p}_j$;

Resolver $\mathbf{M}\mathbf{v}_j = \mathbf{y}_j$;

$\alpha_j = \rho_j / \langle \mathbf{v}_j, \widetilde{\mathbf{r}}_0 \rangle$;

$\mathbf{s}_j = \mathbf{z}_{j-1} - \alpha_j \mathbf{v}_j$;

Primera cuasi-minimización

$\widetilde{\theta}_j = \|\mathbf{s}_j\| / \tau$, $c = \dfrac{1}{\sqrt{1 + \widetilde{\theta}_j^2}}$; $\widetilde{\tau} = \tau\, \widetilde{\theta}_j c$;

$\widetilde{\eta}_j = c_j^2 \alpha_j$;

$\widetilde{\mathbf{d}}_j = \mathbf{p}_j + \dfrac{\theta_{j-1}^2 \eta_{j-1}}{\alpha_j} \mathbf{d}_{j-1}$;

$\widetilde{\mathbf{x}}_j = \mathbf{x}_{j-1} + \widetilde{\eta}_j \widetilde{\mathbf{d}}_j$;

$\mathbf{u}_j = \mathbf{A}\mathbf{s}_j$;

Resolver $\mathbf{M}\mathbf{t}_j = \mathbf{u}_j$;

$\widetilde{\omega}_j = \dfrac{\langle \mathbf{s}_j, \mathbf{t}_j \rangle}{\langle \mathbf{t}_j, \mathbf{t}_j \rangle}$;

$\mathbf{z}_j = \mathbf{s}_j - \widetilde{\omega}_j \mathbf{t}_j$;

Segunda cuasi-minimización

$\theta_j = \|\mathbf{z}_j\| / \widetilde{\tau}$, $c = \dfrac{1}{\sqrt{1 + \theta_j^2}}$; $\tau = \widetilde{\tau}\, \theta_j c$;

$\eta_j = c^2 \widetilde{\omega}_j$;

$\mathbf{d}_j = \mathbf{s}_j + \dfrac{\widetilde{\theta}_j^2 \widetilde{\eta}_j}{\widetilde{\omega}_j} \widetilde{\mathbf{d}}_j$;

$\mathbf{x}_j = \widetilde{\mathbf{x}}_j + \eta_j \mathbf{d}_j$;

Fin

7.11. Algoritmo CGN

Si aplicamos la técnica de precondicionamiento por la izquierda al algoritmo CGN desarrollado en el capítulo 3 obtenemos,

ALGORITMO CGN PRECONDICIONADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$,
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
Resolver $\mathbf{M}^T\mathbf{s}_0 = \mathbf{z}_0$;
$\mathbf{p}_0 = \mathbf{A}^T\mathbf{s}_0$;
Mientras $\| \mathbf{r}_j \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (j = 0, 1, 2, 3, ...)$, hacer
$\mathbf{t}_j = \mathbf{A}\mathbf{p}_j$
Resolver $\mathbf{M}\mathbf{v}_j = \mathbf{t}_j$;
$\alpha_j = \dfrac{\langle \mathbf{A}^T\mathbf{s}_j, \mathbf{A}^T\mathbf{s}_j \rangle}{\langle \mathbf{v}_j, \mathbf{v}_j \rangle}$;
$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$;
$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j\mathbf{t}_j$;
Resolver $\mathbf{M}\mathbf{z}_{j+1} = \mathbf{r}_{j+1}$;
Resolver $\mathbf{M}^T\mathbf{s}_{j+1} = \mathbf{z}_{j+1}$;
$\beta_j = \dfrac{\langle \mathbf{A}^T\mathbf{s}_{j+1}, \mathbf{A}^T\mathbf{s}_{j+1} \rangle}{\langle \mathbf{A}^T\mathbf{s}_j, \mathbf{A}^T\mathbf{s}_j \rangle}$;
$\mathbf{p}_{j+1} = \mathbf{A}^T\mathbf{s}_{j+1} + \beta_j\mathbf{p}_j$;
Fin

7.12. Algoritmo LSQR

Si aplicamos la técnica de precondicionamiento por la izquierda al algoritmo LSQR desarrollado en el capítulo 3 obtenemos,

ALGORITMO LSQR PRECONDICIONADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
$\beta_1 = \|\mathbf{z}_0\|$, $\mathbf{u}_1 = \mathbf{z}_0/\beta_1$;
Resolver $\mathbf{M}^T\mathbf{s}_1 = \mathbf{u}_1$;
$\alpha_1 = \left\|\mathbf{A}^T\mathbf{s}_1\right\|$, $\mathbf{v}_1 = \mathbf{A}^T\mathbf{s}_1/\alpha_1$, $\mathbf{w}_1 = \mathbf{v}_1$;
$\overline{\phi}_1 = \beta_1$, $\overline{\rho}_1 = \alpha_1$;
Mientras $\overline{\phi}_j / \| \mathbf{r}_0 \| \geq \varepsilon$ ($j = 1, ...$), hacer
$\mathbf{p}_j = \mathbf{A}\mathbf{v}_j$;
Resolver $\mathbf{M}\mathbf{q}_j = \mathbf{p}_j$;
$\beta_{j+1} = \|\mathbf{q}_j - \alpha_j\mathbf{u}_j\|$;
$\mathbf{u}_{j+1} = \dfrac{\mathbf{q}_j - \alpha_j\mathbf{u}_j}{\beta_{j+1}}$;
Resolver $\mathbf{M}\mathbf{s}_{j+1} = \mathbf{u}_{j+1}$;

$$\alpha_{j+1} = \left\| \mathbf{A}^T \mathbf{s}_{j+1} - \beta_{j+1} \mathbf{v}_j \right\|;$$
$$\mathbf{v}_{j+1} = \frac{\mathbf{A}^T \mathbf{s}_{j+1} - \beta_{j+1} \mathbf{v}_j}{\alpha_{j+1}};$$
$$\rho_j = \left(\overline{\rho}_j^2 + \beta_{j+1}^2 \right)^{\frac{1}{2}};$$
$$c_j = \frac{\overline{\rho}_j}{\rho_j};$$
$$s_j = \frac{\beta_{j+1}}{\rho_j};$$
$$\theta_{j+1} = s_j \alpha_{j+1};$$
$$\overline{\rho}_{j+1} = -c_j \alpha_{j+1};$$
$$\phi_j = c_j \overline{\phi}_j;$$
$$\overline{\phi}_{j+1} = s_j \overline{\phi}_j;$$
$$\mathbf{x}_j = \mathbf{x}_{j-1} + \left(\frac{\phi_j}{\rho_j} \right) \mathbf{w}_j;$$
$$\mathbf{w}_{j+1} = \mathbf{v}_{j+1} - \left(\frac{\theta_{j+1}}{\rho_j} \right) \mathbf{w}_j;$$

Fin

Capítulo 8

Métodos de Cuasi-Mínimo Residuo Modificados

Como hemos visto en el capítulo 2, la solución aproximada del QMR para el subespacio de Krylov de orden k, se puede escribir como,

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k\,\mathbf{u} \tag{8.1}$$

donde $\mathbf{u}$ minimiza la norma,

$$\left\|\gamma\mathbf{e}_1 - \overline{\mathbf{T}}_k\,\mathbf{u}\right\|_2 \tag{8.2}$$

que es una simplificación de la norma del residuo,

$$\|\,\mathbf{r}\|_2 = \left\|\mathbf{V}_{k+1}\left(\gamma\mathbf{e}_1 - \overline{\mathbf{T}}_k\,\mathbf{u}\right)\right\|_2 \tag{8.3}$$

siendo $\mathbf{V}_k$ la matriz que tiene por columnas los vectores $\mathbf{v}_i$, $i = 1, ..., k$, obtenidos en el procedimiento de biortogonalización de Lanczos, $\gamma = \|\mathbf{r}_0\|_2$, y la matriz $\overline{\mathbf{T}}_k$ se define de la forma,

$$\overline{\mathbf{T}}_k = \begin{pmatrix} \mathbf{T}_k \\ \delta_{k+1}\mathbf{e}_k^t \end{pmatrix} \tag{8.4}$$

con,

$$\mathbf{T}_k = \begin{pmatrix} \alpha_1 & \beta_2 & & . & & & \\ \delta_2 & \alpha_2 & \beta_3 & . & & & \\ & \delta_3 & \alpha_3 & . & & & \\ . & . & . & . & . & . & . \\ & & & . & \alpha_{k-2} & \beta_{k-1} & \\ & & & . & \delta_{k-1} & \alpha_{k-1} & \beta_k \\ & & & . & & \delta_k & \alpha_k \end{pmatrix} \tag{8.5}$$

α_i, $i = 1, ..., k$; β_j, $j = 2, ..., k$; δ_l, $l = 2, ..., k+1$, son los parámetros obtenidos durante el proceso de Lanczos (ver apartado 2.2.1).

En este capítulo queremos abordar la resolución del problema de mínimos cuadrados, que supone la minimización de la funcional cuadrática dada en (8.2) de forma directa sin tener que recurrir a la factorización QR de la matriz $\overline{\mathbf{T}}_k$ [58].

8.1. Resolución directa del problema de Cuasi-minimización

Consideremos la proyección ortogonal sobre el subespacio de soluciones del problema de cuasiminimización dado en (8.2). Multiplicando por la matriz $\overline{\mathbf{T}}_k^T$ obtenemos,

$$\overline{\mathbf{T}}_k^T \overline{\mathbf{T}}_k \mathbf{u} = \overline{\mathbf{T}}_k^T \gamma \mathbf{e}_1 \tag{8.6}$$

La estructura de $\overline{\mathbf{T}}_k$ que es una matriz $(k+1) \times k$ es,

$$\overline{\mathbf{T}}_k = \begin{pmatrix} \boxed{\mathbf{d}_k^T} \\ \boxed{\begin{matrix} & \mathbf{U}_k \\ \\ \mathbf{0} & \end{matrix}} \end{pmatrix}$$

donde la primera fila de $\overline{\mathbf{T}}_k$ es un vector $\mathbf{d}_k^t$, de dimension k y el resto forma una matriz triangular superior $\mathbf{U}_k$,

Esto es,

$$\mathbf{d}_k = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & . & . & . & 0 \end{pmatrix}$$

$$\mathbf{U}_k = \begin{pmatrix} \delta_2 & \alpha_2 & \beta_3 & . & & & \\ & \delta_3 & \alpha_3 & \beta_4 & & & \\ & & & . & & & \\ . & . & . & . & . & . & . \\ & & & . & \delta_{k-1} & \alpha_{k-1} & \beta_k \\ & (0) & & . & & \delta_k & \alpha_k \\ & & & . & & & \delta_{k+1} \end{pmatrix}$$

donde,

$$\{\mathbf{d}_k\}_i = d_i = \left\{\overline{\mathbf{T}}\right\}_{1i} \qquad i = 1, ..., k \tag{8.7}$$

$$\{\mathbf{U}_k\}_{ij} = u_{ij} = \begin{cases} \left\{\overline{\mathbf{T}}\right\}_{i+1,\,j} & 1 \le i \le j \le k \\ 0 & \text{en el resto} \end{cases} \tag{8.8}$$

lo que sugiere la descomposición del producto $\overline{\mathbf{T}}_k^T \overline{\mathbf{T}}_k$ que aparece en la ecuación (8.6) como una suma, de la forma,

$$\left\{\overline{\mathbf{T}}_k^T \overline{\mathbf{T}}_k\right\}_{ij} = d_i d_j + \sum_{m=1}^{k} u_{mi} u_{mj} \tag{8.9}$$

La ecuación (8.6), teniendo en cuenta esta descomposición de $\overline{\mathbf{T}}_k^T \overline{\mathbf{T}}_k$, se puede escribir como,

$$\left(\mathbf{d}_k \mathbf{d}_k^T + \mathbf{U}_k^T \mathbf{U}_k\right) \mathbf{u} = \overline{\mathbf{T}}_k^T \gamma \mathbf{e}_1 \tag{8.10}$$

Y teniendo en cuenta que $\overline{\mathbf{T}}_k^T \mathbf{e}_1 = \mathbf{d}_k$, obtenemos la siguiente expresión,

$$\left(\mathbf{d}_k \mathbf{d}_k^T + \mathbf{U}_k^T \mathbf{U}_k\right) \mathbf{u} = \gamma \mathbf{d}_k \tag{8.11}$$

Si hacemos uso de las propiedades asociativa y distributiva de las matrices, entonces la ecuación anterior quedará,

$$\mathbf{U}_k^T \mathbf{U}_k \mathbf{u} = \mathbf{d}_k \left(\gamma - \langle \mathbf{d}_k, \mathbf{u} \rangle\right) \tag{8.12}$$

y haciendo,

$$\lambda_i = \gamma - \langle \mathbf{d}_k, \mathbf{u} \rangle \tag{8.13}$$

$$\mathbf{u} = \lambda_i \mathbf{p}_k \tag{8.14}$$

obtenemos,

$$\mathbf{U}_k^T \mathbf{U}_k \mathbf{p}_k = \mathbf{d}_k \tag{8.15}$$

que es un doble sistema triangular, ya que $\mathbf{U}_k^T$ y $\mathbf{U}_k$ son matrices triangulares y sólo requiere dos procesos de sustitución, uno por descenso y otro por remonte, para su resolución.

Después de resolver (8.15), calculamos λ_i para obtener finalmente $\mathbf{u}$ a partir de la ecuación (8.14),

$$\lambda_i = \gamma - \langle \mathbf{d}_k, \mathbf{u} \rangle = \gamma - \lambda_i \langle \mathbf{d}_k, \mathbf{p}_k \rangle \tag{8.16}$$

y de este modo,

$$\lambda_i = \frac{\gamma}{1 + \langle \mathbf{d}_k,\ \mathbf{p}_k \rangle} \tag{8.17}$$

Nótese que $1 + \langle \mathbf{d}_k, \mathbf{p}_k \rangle \neq 0,$ ya que,

$$\langle \mathbf{d}_k, \mathbf{p}_k \rangle = \left\langle \mathbf{U}_k^T \mathbf{U}_k \mathbf{p}_k, \mathbf{p}_k \right\rangle = \|\mathbf{U}_k \mathbf{p}_k\|_2^2 \geq 0 \tag{8.18}$$

y por tanto λ_i nunca degenera.

En resumen, el método propuesto requiere:

1. Dados $\mathbf{d}_k$ y $\mathbf{U}_k$ definidos en (8.7) y (8.8), resolver en doble sistema triangular dado en (8.15) haciendo,

$$\mathbf{U}_k^T \bar{\mathbf{p}}_k = \mathbf{d}_k \tag{8.19}$$

$$\mathbf{U}_k \mathbf{p}_k = \bar{\mathbf{p}}_k \tag{8.20}$$

2. Calcular λ_i en la expresión (8.17).

3. Obtener $\mathbf{u}$ resolviendo la ecuación (8.14)

El vector residuo cuya norma viene definida en (8.3) se puede obtener de,

$$\mathbf{r}_i = \mathbf{V}_{k+1} \widehat{\mathbf{r}}_i \tag{8.21}$$

siendo $\widehat{\mathbf{r}}_i$ el $(k+1)$-vector,

$$\widehat{\mathbf{r}}_i = \gamma \mathbf{e}_1 - \overline{\mathbf{T}}_k \mathbf{u} \tag{8.22}$$

y sus componentes se pueden calcular de la siguiente forma,

$$\{\widehat{\mathbf{r}}_i\}_j = \begin{cases} \lambda_i & \text{if} \quad j = 1 \\ -\lambda_i \bar{\mathbf{p}}_k & \text{if} \quad j = 2, ..., k+1 \end{cases} \tag{8.23}$$

Ya que de la partición de $\overline{\mathbf{T}}_k$, la primera componente del $(k+1)$-vector $(\overline{\mathbf{T}}_k\, \mathbf{u})$ es $\langle \mathbf{d}_k, \mathbf{u} \rangle$, y el resto de las componentes vienen dadas por el k-vector $(\mathbf{U}_k \mathbf{u})$. De este modo, la primera componente de $\hat{\mathbf{r}}_i$ es λ_i, y las otras,

$$-\mathbf{U}_k \mathbf{u} = -\lambda_i \mathbf{U}_k \mathbf{p}_k = -\lambda_i \bar{\mathbf{p}}_k \tag{8.24}$$

donde $\bar{\mathbf{p}}_k$ se puede conservar en la resolución del primer sistema triangular dado en (8.19).

Téngase en cuenta que ahora los residuos no son equivalentes (como ocurría en el GMRES), ya que los vectores $\mathbf{v}_i$ no son ortonormados. Es decir que,

$$\|\mathbf{r}_i\|_2 \neq \|\widehat{\mathbf{r}}_i\|_2 \tag{8.25}$$

8.1.1. Método QMR Modificado

La aplicación de la resolución directa del problema de cuasiminimización planteado en el QMR, da como resultado el siguiente algoritmo,

ALGORITMO QMR MODIFICADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\beta_1 = \delta_1 = 0$;
$\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$;
$\gamma = \|\mathbf{r}_0\|$;
$\mathbf{v}_1 = \mathbf{w}_1 = \dfrac{1}{\gamma}\mathbf{r}_0$;
Mientras $\sqrt{k+1}\, \| \widehat{\mathbf{r}}_{k-1} \| \, / \, \| \mathbf{r}_0 \| \geq \varepsilon \quad (k = 1, 2, 3, ...)$, hacer
$\alpha_k = \langle \mathbf{A}\mathbf{v}_k, \mathbf{w}_k \rangle$;
$\widehat{\mathbf{v}}_{k+1} = \mathbf{A}\mathbf{v}_k - \alpha_k \mathbf{v}_k - \beta_k \mathbf{v}_{k-1}$;
$\widehat{\mathbf{w}}_{k+1} = \mathbf{A}^T \mathbf{w}_k - \alpha_k \mathbf{w}_k - \delta_k \mathbf{w}_{k-1}$;
$\delta_{k+1} = |\langle \widehat{\mathbf{v}}_{k+1}, \widehat{\mathbf{w}}_{k+1} \rangle|^{1/2}$;
$\beta_{k+1} = \langle \widehat{\mathbf{v}}_{k+1}, \widehat{\mathbf{w}}_{k+1} \rangle / \delta_{k+1}$;
$\mathbf{v}_{k+1} = \widehat{\mathbf{v}}_{k+1} / \delta_{k+1}$;
$\mathbf{w}_{k+1} = \widehat{\mathbf{w}}_{k+1} / \beta_{k+1}$;
Resolver $\mathbf{U}_k^T \bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k \mathbf{p} = \bar{\mathbf{p}}$;

$$\text{donde} \begin{cases} \{\mathbf{d}_k\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$$

$\lambda_k = \dfrac{\gamma}{1 + \langle \mathbf{d}_k,\, \mathbf{p} \rangle}$;
$\mathbf{u}_k = \lambda_k\, \mathbf{p}$;
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{u}_k$; siendo $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$;

$\mathbf{r}_k = \mathbf{V}_{k+1}\widehat{\mathbf{r}}_k$; siendo $\mathbf{V}_{k+1} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{k+1}]$;

$$\text{donde} \begin{cases} \{\widehat{\mathbf{r}}_k\}_1 = \lambda_k \\ \{\widehat{\mathbf{r}}_k\}_{l+1} = -\lambda_k \{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., k;$$

Fin

Se debe tener en cuenta que el criterio de convergencia se formula ahora en función de $\widehat{\mathbf{r}}_k$, que representa el residuo calculado del QMR-Modificado.

8.1.2. Método TFQMR Modificado

Análogamente, la aproximación obtenida con el TFQMR en un subespacio de Krylov de dimensión k, es de la forma,

$$\mathbf{x}_0 + \mathbf{Y}_k \mathbf{u}_k \tag{8.26}$$

donde $\mathbf{Y}_k = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k]$, $\mathbf{y}_k = \mathbf{t}_{i-1}$ si $k = 2i-1$ es impar, e $\mathbf{y}_k = \mathbf{q}_i$ si $k = 2i$ es par, y $\mathbf{u}_k$ minimiza la norma $\left\| \left(\delta_1 \mathbf{e}_1 - \bar{\mathbf{T}}_k \mathbf{u}\right) \right\|_2$, lo que representa un cuasi-mínimo de la norma del residuo (ver p.e. Saad [72]),

$$\| \mathbf{r}_k \|_2 = \left\| \mathbf{W}_{k+1} \boldsymbol{\Delta}_{k+1}^{-1} \left(\delta_1 \mathbf{e}_1 - \boldsymbol{\Delta}_{k+1} \overline{\mathbf{B}}_k \mathbf{u}_k\right) \right\|_2 \tag{8.27}$$

siendo,

$$\overline{\mathbf{T}}_k = \boldsymbol{\Delta}_{k+1} \overline{\mathbf{B}}_k \tag{8.28}$$

Donde $\mathbf{W}_{k+1}$ es la matrix que tiene por columnas los vectores,

$$\mathbf{W}_{k+1} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{k+1}] \tag{8.29}$$

Y $\boldsymbol{\Delta}_{k+1}$ es una matriz diagonal, tal que $\mathbf{W}_{k+1}$ queda escalada ($\delta_k = \| \mathbf{r}_i \|$, si $k = 2i+1$ es impar, o $\delta_k = \sqrt{\| \mathbf{r}_{i-1} \| \ \| \mathbf{r}_i \|}$, si $k = 2i$ es par),

$$\boldsymbol{\Delta}_{k+1} = \begin{pmatrix} \delta_1 & & \cdot & & \\ & \delta_2 & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ & & \cdot & \delta_k & \\ & & \cdot & & \delta_{k+1} \end{pmatrix} \tag{8.30}$$

y $\overline{\mathbf{B}}_k$ es la matriz de orden $(k+1) \times k$,

$$\overline{\mathbf{B}}_k = \begin{pmatrix} \alpha_0^{-1} & & & \cdot & & \\ -\alpha_0^{-1} & \alpha_0^{-1} & & \cdot & & \\ & -\alpha_0^{-1} & \alpha_1^{-1} & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & \cdot & \alpha_{(k-1)/2}^{-1} & \\ & & & \cdot & -\alpha_{(k-1)/2}^{-1} & \alpha_{(k-1)/2}^{-1} \\ & & & \cdot & & -\alpha_{(k-1)/2}^{-1} \end{pmatrix} \tag{8.31}$$

El algoritmo TFQMR Modificado que se ha obtenido es,

ALGORITMO TFQMR MODIFICADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\mathbf{r}_0^*$ es arbitrario, tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
$\mathbf{s}_0 = \mathbf{t}_0 = \mathbf{r}_0$;
$\mathbf{v}_0 = \mathbf{A}\mathbf{s}_0$;
$\rho_0 = \langle \mathbf{r}_0, \mathbf{r}_0^* \rangle$;
$\delta_1 = \| \mathbf{r}_0 \|$;
Mientras $\sqrt{i+1} \parallel \widehat{\mathbf{r}}_{i-1} \parallel / \parallel \mathbf{r}_0 \parallel \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer
$\sigma_{i-1} = \langle \mathbf{v}_{i-1}, \mathbf{r}_0^* \rangle$;
$\alpha_{i-1} = \rho_{i-1} / \sigma_{i-1}$;
$\mathbf{q}_i = \mathbf{t}_{i-1} - \alpha_{i-1}\mathbf{v}_{i-1}$;
$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1}\mathbf{A}\left(\mathbf{t}_{i-1} + \mathbf{q}_i\right)$;
Desde $k = 2i-1, 2i$ hacer
Si k es impar hacer
$\delta_{k+1} = \sqrt{\| \mathbf{r}_{i-1} \| \ \| \mathbf{r}_i \|}$; $\mathbf{y}_k = \mathbf{t}_{i-1}$;
En caso contrario hacer
$\delta_{k+1} = \| \mathbf{r}_i \|$; $\mathbf{y}_k = \mathbf{q}_i$;
Fin
Fin
Resolver $\mathbf{U}_k^T \bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k \mathbf{p} = \bar{\mathbf{p}}$;
donde $\begin{cases} \{\mathbf{d}_k\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k$;
$\lambda_k = \dfrac{\delta_1}{1 + \langle \mathbf{d}_k, \ \mathbf{p} \rangle}$;
$\mathbf{u}_k = \lambda_k \, \mathbf{p}$;
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{Y}_k \mathbf{u}_k$; con $\mathbf{Y}_k = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k]$;
$\begin{cases} \{\widehat{\mathbf{r}}_i\}_1 = \lambda_{2i} \\ \{\widehat{\mathbf{r}}_i\}_{l+1} = -\lambda_{2i} \{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., 2i$;
$\rho_i = \langle \mathbf{r}_i, \mathbf{r}_0^* \rangle$;
$\beta_i = \rho_i / \rho_{i-1}$;
$\mathbf{t}_i = \mathbf{r}_i + \beta_i \mathbf{q}_i$;
$\mathbf{s}_i = \mathbf{t}_i + \beta_i (\mathbf{q}_i + \beta_i \mathbf{s}_{i-1})$;
$\mathbf{v}_i = \mathbf{A}\mathbf{s}_i$;
Fin

Al igual que en QMR Modificado, se debe tener en cuenta que el criterio de convergencia se formula ahora en función de $\widehat{\mathbf{r}}_k$, que representa el residuo calculado del TFQMR-Modificado.

8.1.3. Método QMRCGSTAB Modificado

El algoritmo QMRCGSTAB propuesto por Chan y otros [6], realiza dos cuasi-minimizaciones por iteración. Si definimos $\mathbf{Y}_k = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k]$, siendo $\mathbf{y}_{2l-1} = \mathbf{g}_l$ para $l = 1, ..., [k+1/2]$ ($[k+1/2]$ es la parte entera de $k+1/2$) y $\mathbf{y}_{2l} = \mathbf{s}_l$ para $l = 1, ..., [k/2]$($[k/2]$ es la parte entera de $k/2$). La solución aproximada del sistema $\mathbf{Ax} = \mathbf{b}$, a partir del k-ésimo subespacio de Krylov, se construye como $\mathbf{x}_0 + \mathbf{Y}_k\mathbf{u}_k$, donde $\mathbf{u}_k$ minimiza la norma $\left\|\left(\delta_1\mathbf{e}_1 - \overline{\mathbf{T}}_k\,\mathbf{u}\right)\right\|_2$, que es otra vez un cuasi-mínimo de la norma del residuo,

$$\| \mathbf{r}_k\|_2 = \left\|\mathbf{W}_{k+1}\boldsymbol{\Delta}_{k+1}^{-1}\left(\delta_1\mathbf{e}_1 - \boldsymbol{\Delta}_{k+1}\overline{\mathbf{B}}_k\,\mathbf{u}_k\right)\right\|_2 \tag{8.32}$$

siendo,

$$\overline{\mathbf{T}}_k = \boldsymbol{\Delta}_{k+1}\overline{\mathbf{B}}_k \tag{8.33}$$

$\mathbf{W}_{k+1}$ es nuevamente la matriz cuyas columnas son los vectores residuos,

$$\mathbf{W}_{k+1} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{k+1}] \tag{8.34}$$

con $\mathbf{w}_{2l-1} = \mathbf{s}_l$ para $l = 1, ..., [(k+1)/2]$ y $\mathbf{w}_{2l} = \mathbf{r}_l$ para $l = 1, ..., [k/2]$; y $\boldsymbol{\Delta}_{k+1}$ es una matriz diagonal, tal que $\mathbf{W}_{k+1}$ es escalada ($\delta_i = \| \mathbf{w}_i\|$),

$$\boldsymbol{\Delta}_{k+1} = \begin{pmatrix} \delta_1 & & \cdot & & \\ & \delta_2 & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ & & \cdot & \delta_k & \\ & & \cdot & & \delta_{k+1} \end{pmatrix} \tag{8.35}$$

$\overline{\mathbf{B}}_k$ es la matriz de orden $(k+1) \times k$,

$$\overline{\mathbf{B}}_k = \begin{pmatrix} \sigma_1^{-1} & & & \cdot & & \\ -\sigma_1^{-1} & \sigma_2^{-1} & & \cdot & & \\ & -\sigma_2^{-1} & \sigma_3^{-1} & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & \cdot & \sigma_{k-1}^{-1} & \\ & & & \cdot & -\sigma_{k-1}^{-1} & \sigma_k^{-1} \\ & & & \cdot & & -\sigma_k^{-1} \end{pmatrix} \tag{8.36}$$

con $\sigma_{2l} = \omega_l$ para $l = 1, ..., [(k+1)/2]$, y $\sigma_{2l-1} = \alpha_l$ para $l = 1, ..., [(k+1)/2]$.

ALGORITMO QMRCGSTAB MODIFICADO
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$;
$\mathbf{r}_0^*$ es arbitrario, tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
$\rho_0 = \alpha_0 = \omega_0 = 1$;
$\mathbf{g}_0 = \mathbf{v}_0 = \mathbf{0}$;
Mientras $\sqrt{2i+1}\ \| \widehat{\mathbf{r}}_{i-1} \| \ / \ \| \mathbf{r}_0 \| \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer:

$\rho_i = \langle \mathbf{r}_0^*, \mathbf{r}_{i-1} \rangle$;
$\beta_i = (\rho_i / \rho_{i-1})(\alpha_{i-1} / \omega_{i-1})$;
$\mathbf{g}_i = \mathbf{r}_{i-1} + \beta_i(\mathbf{g}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1})$;
$\mathbf{v}_i = \mathbf{A}\mathbf{g}_i$;
$\alpha_i = \dfrac{\rho_i}{\langle \mathbf{v}_i, \mathbf{r}_0^* \rangle}$;
$\mathbf{s}_i = \mathbf{r}_{i-1} - \alpha_i \mathbf{v}_i$;
$\delta_{2i-1} = \|\mathbf{s}_i\|$; $\mathbf{y}_{2i-1} = \mathbf{g}_i$;
$\mathbf{t}_i = \mathbf{A}\mathbf{s}_i$;
$\omega_i = \dfrac{\langle \mathbf{t}_i, \mathbf{s}_i \rangle}{\langle \mathbf{t}_i, \mathbf{t}_i \rangle}$;
$\mathbf{r}_i = \mathbf{s}_i - \omega_i \mathbf{t}_i$;
$\delta_{2i} = \|\mathbf{r}_i\|$; $\mathbf{y}_{2i} = \mathbf{s}_i$;
Resolver $\mathbf{U}_{2i}^t \bar{\mathbf{p}} = \mathbf{d}_{2i}$ y $\mathbf{U}_{2i}\mathbf{p} = \bar{\mathbf{p}}$;

$$\text{donde} \begin{cases} \{\mathbf{d}_{2i}\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_{2i}\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., 2i;$$

$\lambda_{2i} = \dfrac{\delta_1}{1 + \langle \mathbf{d}_{2i},\ \mathbf{p} \rangle}$;
$\mathbf{u}_{2i} = \lambda_{2i}\, \mathbf{p}$;
$\mathbf{x}_i = \mathbf{x}_0 + \mathbf{Y}_{2i}\mathbf{u}_{2i}$; con $\mathbf{Y}_{2i} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{2i}]$;

$$\begin{cases} \{\widehat{\mathbf{r}}_i\}_1 = \lambda_{2i} \\ \{\widehat{\mathbf{r}}_i\}_{l+1} = -\lambda_{2i}\, \{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., 2i;$$

Fin

Asimismo, se debe tener en cuenta que el criterio de convergencia se formula ahora en función de $\widehat{\mathbf{r}}_k$, que representa el residuo calculado del QMRCGSTAB-Modificado.

Capítulo 9

Métodos tipo QMR Modificado precondicionados

En este capítulo, se obtendrán los algoritmos precondicionados de los métodos tipo QMR-Modificado propuestos en el capítulo anterior, estableciendo comparaciones en cada uno para las distintas formas de precondicionamiento y extrayendo las conclusiones que se deriven de las mismas.

Se establecerán las relaciones entre las matrices, vectores incógnitas y segundo miembro, del sistema original y del precondicionado para las formas de precondicionamiento por la izquierda, por la derecha y por ambos lados. Las expresiones resultantes, junto con las de los vectores residuos se sustituirán en los algoritmos correspondientes.

9.1. Método QMR Modificado

9.1.1. Precondicionamiento por la izquierda

El sistema precondicionado sería,

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}} \begin{cases} \widetilde{\mathbf{A}} = \mathbf{M}^{-1}\mathbf{A} \\ \widetilde{\mathbf{x}} = \mathbf{x} \\ \widetilde{\mathbf{b}} = \mathbf{M}^{-1}\mathbf{b} \end{cases}$$

y la relación para el vector residuo del sistema con y sin precondicionamiento quedaría,

$$\widetilde{\mathbf{r}} = \widetilde{\mathbf{b}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{b} - \mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\left(\mathbf{b} - \mathbf{A}\mathbf{x}\right) = \mathbf{M}^{-1}\mathbf{r}$$

Aplicando el algoritmo QMR-Modificado a este sistema precondicionado y estableciendo las nuevas expresiones para los distintos parámetros y vectores:

$\widetilde{\beta}_1 = \widetilde{\delta}_1 = 0$;

$\widetilde{\mathbf{v}}_0 = \widetilde{\mathbf{w}}_0 = \mathbf{0}$;

$\widetilde{\gamma} = \|\widetilde{\mathbf{r}}_0\| = \|\mathbf{M}^{-1}\mathbf{r}_0\| = \|\mathbf{z}_0\|$; con $\mathbf{M}^{-1}\mathbf{r}_0 = \mathbf{z}_0$
$\widetilde{\mathbf{v}}_1 = \widetilde{\mathbf{w}}_1 = \dfrac{1}{\widetilde{\gamma}}\widetilde{\mathbf{r}}_0 = \dfrac{1}{\widetilde{\gamma}}\mathbf{z}_0$;
$\widetilde{\alpha}_k = \left\langle \widetilde{\mathbf{A}}\widetilde{\mathbf{v}}_k, \widetilde{\mathbf{w}}_k \right\rangle = \langle \mathbf{M}^{-1}\mathbf{A}\widetilde{\mathbf{v}}_k, \widetilde{\mathbf{w}}_k \rangle$; llamando $\mathbf{A}\widetilde{\mathbf{v}}_k = \mathbf{y}_k$ y haciendo,
$\mathbf{M}\mathbf{s}_k = \mathbf{y}_k$, queda $\widetilde{\alpha}_k = \langle \mathbf{s}_k, \widetilde{\mathbf{w}}_k \rangle$;
$\widetilde{\widetilde{\mathbf{v}}}_{k+1} = \widetilde{\mathbf{A}}\widetilde{\mathbf{v}}_k - \widetilde{\alpha}_k\widetilde{\mathbf{v}}_k - \widetilde{\beta}_k\widetilde{\mathbf{v}}_{k-1} = \mathbf{s}_k - \widetilde{\alpha}_k\widetilde{\mathbf{v}}_k - \widetilde{\beta}_k\widetilde{\mathbf{v}}_{k-1}$;
$\widetilde{\widetilde{\mathbf{w}}}_{k+1} = \widetilde{\mathbf{A}}^T\widetilde{\mathbf{w}}_k - \widetilde{\alpha}_k\widetilde{\mathbf{w}}_k - \widetilde{\delta}_k\widetilde{\mathbf{w}}_{k-1} = \mathbf{A}^T\mathbf{M}^{-T}\widetilde{\mathbf{w}}_k - \widetilde{\alpha}_k\widetilde{\mathbf{w}}_k - \widetilde{\delta}_k\widetilde{\mathbf{w}}_{k-1}$;
haciendo $\mathbf{M}^T\mathbf{f}_k = \widetilde{\mathbf{w}}_k$ y llamando $\mathbf{A}^T\mathbf{f}_k = \mathbf{t}_k$ resulta,
$\widetilde{\widetilde{\mathbf{w}}}_{k+1} = \mathbf{t}_k - \widetilde{\alpha}_k\widetilde{\mathbf{w}}_k - \widetilde{\delta}_k\widetilde{\mathbf{w}}_{k-1}$;
$\widetilde{\delta}_{k+1} = \left| \left\langle \widetilde{\widetilde{\mathbf{v}}}_{k+1}, \widetilde{\widetilde{\mathbf{w}}}_{k+1} \right\rangle \right|^{1/2}$;
$\widetilde{\beta}_{k+1} = \left\langle \widetilde{\widetilde{\mathbf{v}}}_{k+1}, \widetilde{\widetilde{\mathbf{w}}}_{k+1} \right\rangle / \widetilde{\delta}_{k+1}$;
$\widetilde{\mathbf{v}}_{k+1} = \widetilde{\widetilde{\mathbf{v}}}_{k+1}/\widetilde{\delta}_{k+1}$;
$\widetilde{\mathbf{w}}_{k+1} = \widetilde{\widetilde{\mathbf{w}}}_{k+1}/\widetilde{\beta}_{k+1}$;
Resolver $\widetilde{\mathbf{U}}_k^T\widetilde{\mathbf{p}}=\widetilde{\mathbf{d}}_k$ y $\widetilde{\mathbf{U}}_k\widetilde{\mathbf{p}}=\widetilde{\widetilde{\mathbf{p}}}$;

$$\text{donde} \begin{cases} \left\{\widetilde{\mathbf{d}}_k\right\}_m = \left\{\widetilde{\widetilde{\mathbf{T}}}\right\}_{1m} \\ \left\{\widetilde{\mathbf{U}}_k\right\}_{lm} = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{l+1m} \end{cases} \qquad l, m = 1, ..., k;$$

$$\widetilde{\lambda}_k = \frac{\widetilde{\gamma}}{1 + \left\langle \widetilde{\mathbf{d}}_k, \widetilde{\mathbf{p}} \right\rangle};$$

$\widetilde{\mathbf{u}}_k = \widetilde{\lambda}_k\widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_k = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{V}}_k\mathbf{u}_k$; entonces, $\mathbf{x}_k = \mathbf{x}_0 + \widetilde{\mathbf{V}}_k\widetilde{\mathbf{u}}_k$, siendo $\widetilde{\mathbf{V}}_k = [\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, ..., \widetilde{\mathbf{v}}_k]$;
$\widetilde{\mathbf{r}}_k = \widetilde{\mathbf{V}}_{k+1}\widetilde{\widetilde{\mathbf{r}}}_k$, siendo $\widetilde{\mathbf{V}}_{k+1} = [\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, ..., \widetilde{\mathbf{v}}_{k+1}]$; entonces,
$\mathbf{z}_k = \widetilde{\mathbf{V}}_{k+1}\widetilde{\widetilde{\mathbf{r}}}_k$, siendo $\widetilde{\mathbf{V}}_{k+1} = [\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, ..., \widetilde{\mathbf{v}}_{k+1}]$ y,
$\mathbf{r}_k = \mathbf{M}\mathbf{z}_k$;

$$\text{donde} \begin{cases} \left\{\widetilde{\widetilde{\mathbf{r}}}_k\right\}_1 = \widetilde{\lambda}_k \\ \left\{\widetilde{\widetilde{\mathbf{r}}}_k\right\}_{l+1} = -\widetilde{\lambda}_k \left\{\widetilde{\widetilde{\mathbf{p}}}\right\}_l \end{cases} \quad l = 1, ..., k;$$

resulta entonces,

ALGORITMO QMR MODIFICADO PRECONDICIONADO POR LA IZQUIERDA
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\beta_1 = \delta_1 = 0$;
$\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$
$\gamma = \|\mathbf{z}_0\|$;
$\mathbf{v}_1 = \mathbf{w}_1 = \dfrac{1}{\gamma}\mathbf{z}_0$;
Mientras $\sqrt{k+1}\ \|\ \widehat{\mathbf{r}}_{k-1}\ \|\ /\ \|\ \mathbf{r}_0\ \| \geq \varepsilon \quad (k = 1, 2, 3, ...)$, hacer
    $\mathbf{A}\mathbf{v}_k = \mathbf{y}_k$;
    Resolver $\mathbf{M}\mathbf{s}_k = \mathbf{y}_k$;

Resolver $\mathbf{M}^T\mathbf{f}_k = \mathbf{w}_k$;
$\mathbf{A}^T\mathbf{f}_k = \mathbf{t}_k$;
$\alpha_k = \langle \mathbf{s}_k, \mathbf{w}_k \rangle$;
$\widehat{\mathbf{v}}_{k+1} = \mathbf{s}_k - \alpha_k \mathbf{v}_k - \beta_k \mathbf{v}_{k-1}$;
$\widehat{\mathbf{w}}_{k+1} = \mathbf{t}_k - \alpha_k \mathbf{w}_k - \delta_k \mathbf{w}_{k-1}$;
$\delta_{k+1} = \left| \langle \widehat{\mathbf{v}}_{k+1}, \widehat{\mathbf{w}}_{k+1} \rangle \right|^{1/2}$;
$\beta_{k+1} = \langle \widehat{\mathbf{v}}_{k+1}, \widehat{\mathbf{w}}_{k+1} \rangle / \delta_{k+1}$;
$\mathbf{v}_{k+1} = \widehat{\mathbf{v}}_{k+1} / \delta_{k+1}$;
$\mathbf{w}_{k+1} = \widehat{\mathbf{w}}_{k+1} / \beta_{k+1}$;
Resolver $\mathbf{U}_k^T \bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k \mathbf{p} = \bar{\mathbf{p}}$;

$$\text{donde} \begin{cases} \{\mathbf{d}_k\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$$

$$\lambda_k = \frac{\gamma}{1 + \langle \mathbf{d}_k, \mathbf{p} \rangle};$$
$\mathbf{u}_k = \lambda_k \, \mathbf{p}$;
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{u}_k$, con $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$;
$\mathbf{z}_k = \mathbf{V}_{k+1} \widehat{\mathbf{r}}_k$, con $\mathbf{V}_{k+1} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{k+1}]$;
$\mathbf{r}_k = \mathbf{M}\mathbf{z}_k$;

$$\text{donde} \begin{cases} \{\widehat{\mathbf{r}}_k\}_1 = \lambda_k \\ \{\widehat{\mathbf{r}}_k\}_{l+1} = -\lambda_k \{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., k;$$

Fin

9.1.2. Precondicionamiento por la derecha

Precondicionando por la derecha con la matriz M quedaría,

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}} \begin{cases} \widetilde{\mathbf{A}} = \mathbf{A}\mathbf{M}^{-1} \\ \widetilde{\mathbf{x}} = \mathbf{M}\mathbf{x} \\ \widetilde{\mathbf{b}} = \mathbf{b} \end{cases}$$

y el nuevo vector residuo sería,

$$\widetilde{\mathbf{r}} = \widetilde{\mathbf{b}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \mathbf{b} - \mathbf{A}\mathbf{M}^{-1}\mathbf{M}\mathbf{x} = \mathbf{b} - \mathbf{A}\mathbf{x} = \mathbf{r}$$

Aplicando el algoritmo QMR-Modificado a este sistema precondicionado y estableciendo las nuevas expresiones para los distintos parámetros y vectores:
$\widetilde{\beta}_1 = \widetilde{\delta}_1 = 0$;
$\widetilde{\mathbf{v}}_0 = \widetilde{\mathbf{w}}_0 = \mathbf{0}$;
$\widetilde{\gamma} = \|\widetilde{\mathbf{r}}_0\| = \|\mathbf{r}_0\|$;
$\widetilde{\mathbf{v}}_1 = \widetilde{\mathbf{w}}_1 = \frac{1}{\widetilde{\gamma}} \mathbf{r}_0$;
$\widetilde{\alpha}_k = \left\langle \widetilde{\mathbf{A}}\widetilde{\mathbf{v}}_k, \widetilde{\mathbf{w}}_k \right\rangle = \left\langle \mathbf{A}\mathbf{M}^{-1}\widetilde{\mathbf{v}}_k, \widetilde{\mathbf{w}}_k \right\rangle$; haciendo, $\mathbf{M}\mathbf{y}_k = \widetilde{\mathbf{v}}_k$, resulta,
$\widetilde{\alpha}_k = \langle \mathbf{A}\mathbf{y}_k, \widetilde{\mathbf{w}}_k \rangle$;
$\widetilde{\widehat{\mathbf{v}}}_{k+1} = \widetilde{\mathbf{A}}\widetilde{\mathbf{v}}_k - \widetilde{\alpha}_k \widetilde{\mathbf{v}}_k - \widetilde{\beta}_k \widetilde{\mathbf{v}}_{k-1} = \mathbf{A}\mathbf{y}_k - \widetilde{\alpha}_k \widetilde{\mathbf{v}}_k - \widetilde{\beta}_k \widetilde{\mathbf{v}}_{k-1}$;

$\widetilde{\widehat{\mathbf{w}}}_{k+1} = \widetilde{\mathbf{A}}^T\widetilde{\mathbf{w}}_k - \widetilde{\alpha}_k\widetilde{\mathbf{w}}_k - \widetilde{\delta}_k\widetilde{\mathbf{w}}_{k-1} = \mathbf{M}^{-T}\mathbf{A}^T\widetilde{\mathbf{w}}_k - \widetilde{\alpha}_k\widetilde{\mathbf{w}}_k - \widetilde{\delta}_k\widetilde{\mathbf{w}}_{k-1}$;
llamando $\mathbf{A}^T\widetilde{\mathbf{w}}_k = \mathbf{f}_k$ y haciendo $\mathbf{M}^T\mathbf{t}_k = \mathbf{f}_k$, resulta,
$\widetilde{\widehat{\mathbf{w}}}_{k+1} = \mathbf{t}_k - \widetilde{\alpha}_k\widetilde{\mathbf{w}}_k - \widetilde{\delta}_k\widetilde{\mathbf{w}}_{k-1}$;
$\widetilde{\delta}_{k+1} = \left|\left\langle\widetilde{\widehat{\mathbf{v}}}_{k+1}, \widetilde{\widehat{\mathbf{w}}}_{k+1}\right\rangle\right|^{1/2}$;
$\widetilde{\beta}_{k+1} = \left\langle\widetilde{\widehat{\mathbf{v}}}_{k+1}, \widetilde{\widehat{\mathbf{w}}}_{k+1}\right\rangle / \widetilde{\delta}_{k+1}$;
$\widetilde{\mathbf{v}}_{k+1} = \widetilde{\widehat{\mathbf{v}}}_{k+1}/\widetilde{\delta}_{k+1}$;
$\widetilde{\mathbf{w}}_{k+1} = \widetilde{\widehat{\mathbf{w}}}_{k+1}/\widetilde{\beta}_{k+1}$;
Resolver $\widetilde{\mathbf{U}}_k^T\widetilde{\mathbf{p}}=\widetilde{\mathbf{d}}_k$ y $\widetilde{\mathbf{U}}_k\widetilde{\mathbf{p}}=\widetilde{\widetilde{\mathbf{p}}}$;

$$\text{donde}\begin{cases} \left\{\widetilde{\mathbf{d}}_k\right\}_m = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{1m} \\ \left\{\widetilde{\mathbf{U}}_k\right\}_{lm} = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$$

$$\widetilde{\lambda}_k = \frac{\widetilde{\gamma}}{1 + \left\langle\widetilde{\mathbf{d}}_k, \widetilde{\mathbf{p}}\right\rangle};$$

$\widetilde{\mathbf{u}}_k = \widetilde{\lambda}_k\,\widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_k = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{V}}_k\widetilde{\mathbf{u}}_k$, siendo $\widetilde{\mathbf{V}}_k = [\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, ..., \widetilde{\mathbf{v}}_k]$; multiplicando por $\mathbf{M}^{-1}$,
resulta, $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{M}^{-1}\widetilde{\mathbf{V}}_k\widetilde{\mathbf{u}}_k$; llamando $\widetilde{\mathbf{V}}_k\widetilde{\mathbf{u}}_k = \mathbf{s}_k$ y haciendo
$\mathbf{M}\mathbf{q}_k = \mathbf{s}_k$, se obtiene $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{q}_k$;
$\widetilde{\mathbf{r}}_k = \widetilde{\mathbf{V}}_{k+1}\widetilde{\widehat{\mathbf{r}}}_k$, entonces,
$\mathbf{r}_k = \widetilde{\mathbf{V}}_{k+1}\widetilde{\widehat{\mathbf{r}}}_k$, siendo $\widetilde{\mathbf{V}}_{k+1} = [\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, ..., \widetilde{\mathbf{v}}_{k+1}]$;

$$\text{donde}\begin{cases} \left\{\widetilde{\widehat{\mathbf{r}}}_k\right\}_1 = \widetilde{\lambda}_k \\ \left\{\widetilde{\widehat{\mathbf{r}}}_k\right\}_{l+1} = -\widetilde{\lambda}_k\left\{\widetilde{\widetilde{\mathbf{p}}}\right\}_l \end{cases} \quad l = 1, ..., k;$$

se obtiene,

ALGORITMO QMR MODIFICADO PRECONDICIONADO POR LA DERECHA
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\beta_1 = \delta_1 = 0$;
$\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$;
$\gamma = \|\mathbf{r}_0\|$;
$\mathbf{v}_1 = \mathbf{w}_1 = \dfrac{1}{\gamma}\mathbf{r}_0$;
Mientras $\sqrt{k+1}\,\|\,\widehat{\mathbf{r}}_{k-1}\,\|\,/\,\|\,\mathbf{r}_0\,\| \geq \varepsilon \quad (k = 1, 2, 3, ...)$, hacer
Resolver $\mathbf{M}\mathbf{y}_k = \mathbf{v}_k$;
$\alpha_k = \langle\mathbf{A}\mathbf{y}_k, \mathbf{w}_k\rangle$;
Hacer $\mathbf{A}_k^T\mathbf{w}_k = \mathbf{f}_k$;
Resolver $\mathbf{M}^T\mathbf{t}_k = \mathbf{f}_k$;
$\widehat{\mathbf{v}}_{k+1} = \mathbf{A}\mathbf{y}_k - \alpha_k\mathbf{v}_k - \beta_k\mathbf{v}_{k-1}$;
$\widehat{\mathbf{w}}_{k+1} = \mathbf{t}_k - \alpha_k\mathbf{w}_k - \delta_k\mathbf{w}_{k-1}$;
$\delta_{k+1} = |\langle\widehat{\mathbf{v}}_{k+1}, \widehat{\mathbf{w}}_{k+1}\rangle|^{1/2}$;
$\beta_{k+1} = \langle\widehat{\mathbf{v}}_{k+1}, \widehat{\mathbf{w}}_{k+1}\rangle / \delta_{k+1}$;

$\mathbf{v}_{k+1} = \widehat{\mathbf{v}}_{k+1}/\delta_{k+1}$;
$\mathbf{w}_{k+1} = \widehat{\mathbf{w}}_{k+1}/\beta_{k+1}$;
Resolver $\mathbf{U}_k^T\bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k\mathbf{p} = \bar{\mathbf{p}}$;

$$\text{donde} \begin{cases} \{\mathbf{d}_k\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$$

$$\lambda_k = \frac{\gamma}{1 + \langle \mathbf{d}_k, \mathbf{p} \rangle};$$

$\mathbf{u}_k = \lambda_k\, \mathbf{p}$;
Hacer $\mathbf{V}_k\mathbf{u}_k = \mathbf{s}_k$; con $\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$;
Resolver $\mathbf{M}\mathbf{q}_k = \mathbf{s}_k$
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{q}_k$
$\mathbf{r}_k = \mathbf{V}_{k+1}\widehat{\mathbf{r}}_k$; con $\mathbf{V}_{k+1} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{k+1}]$;

$$\text{donde} \begin{matrix} \{\widehat{\mathbf{r}}_k\}_1 = \lambda_k \\ \{\widehat{\mathbf{r}}_k\}_{l+1} = -\lambda_k \{\bar{\mathbf{p}}\}_l \end{matrix} \quad l = 1, ..., k;$$

Fin

9.1.3. Precondicionamiento por ambos lados

Precondicionando con la matriz factorizada $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$ por ambos lados, quedaría,

$$\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}} \begin{cases} \widetilde{\mathbf{A}} = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1} \\ \widetilde{\mathbf{x}} = \mathbf{M}_2\mathbf{x} \\ \widetilde{\mathbf{b}} = \mathbf{M}_1^{-1}\mathbf{b} \end{cases}$$

Las relación para el vector residuo del sistema con y sin precondicionamiento sería,

$$\widetilde{\mathbf{r}} = \widetilde{\mathbf{b}} - \widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \mathbf{M}_1^{-1}\mathbf{b} - \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M_2}\mathbf{x} = \mathbf{M}_1^{-1}\left(\mathbf{b} - \mathbf{A}\mathbf{x}\right) = \mathbf{M}_1^{-1}\mathbf{r}$$

Aplicando el algoritmo QMR-Modificado a este sistema precondicionado y estableciendo las nuevas expresiones para los distintos parámetros y vectores:
$\widetilde{\beta}_1 = \widetilde{\delta}_1 = 0$;
$\widetilde{\mathbf{v}}_0 = \widetilde{\mathbf{w}}_0 = \mathbf{0}$;
$\widetilde{\gamma} = \|\widetilde{\mathbf{r}}_0\| = \|\mathbf{M}_1^{-1}\mathbf{r}_0\| = \|\mathbf{z}_0\|$;
$\widetilde{\mathbf{v}}_1 = \frac{1}{\widetilde{\gamma}}\mathbf{z}_0$;
Elegir $\widetilde{\mathbf{w}}_1$ tal que $\|\widetilde{\mathbf{w}}_1\| = \|\mathbf{M}_2^{-T}\mathbf{w}_1\| = 1$;
$\widetilde{\alpha}_k = \left\langle \widetilde{\mathbf{A}}\widetilde{\mathbf{v}}_k, \widetilde{\mathbf{w}}_k \right\rangle = \left\langle \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\mathbf{v}_k, \mathbf{M}_2^{-T}\mathbf{w}_k \right\rangle$;
$\widetilde{\alpha}_k = \mathbf{v}_k^T\mathbf{M}_1^{-T}\mathbf{M}_2^{-T}\mathbf{A}^T\mathbf{M}_1^{-T}\mathbf{M}_2^{-T}\mathbf{w}_k = \mathbf{v}_k^T\mathbf{M}^{-T}\mathbf{A}^T\mathbf{M}^{-T}\mathbf{w}_k$, entonces,
$\widetilde{\alpha}_k = \left\langle \mathbf{A}\mathbf{M}^{-1}\mathbf{v}_k, \mathbf{M}^{-T}\mathbf{w}_k \right\rangle$;
y llamando $\mathbf{v}_k^* = \mathbf{M}^{-1}\mathbf{v}_k$ y $\mathbf{w}_k^* = \mathbf{M}^{-T}\mathbf{w}_k$ queda,
$\widetilde{\alpha}_k = \langle \mathbf{A}\mathbf{v}_k^*, \mathbf{w}_k^* \rangle$;
$\widetilde{\widehat{\mathbf{v}}}_{k+1} = \widetilde{\mathbf{A}}\widetilde{\mathbf{v}}_k - \widetilde{\alpha}_k\widetilde{\mathbf{v}}_k - \widetilde{\beta}_k\widetilde{\mathbf{v}}_{k-1} = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\mathbf{v}_k - \widetilde{\alpha}_k\mathbf{M}_1^{-1}\mathbf{v}_k - \widetilde{\beta}_k\mathbf{M}_1^{-1}\mathbf{v}_{k-1}$,
que multiplicando por $\mathbf{M}_1$ y operando resulta,

$\widehat{\mathbf{v}}_{k+1} = \mathbf{A}\mathbf{M}^{-1}\mathbf{v}_k - \widetilde{\alpha}_k\mathbf{v}_k - \widetilde{\beta}_k\mathbf{v}_{k-1} = \mathbf{A}\mathbf{v}_k^* - \widetilde{\alpha}_k\mathbf{v}_k - \widetilde{\beta}_k\mathbf{v}_{k-1}$;
$\widehat{\widetilde{\mathbf{w}}}_{k+1} = \widetilde{\mathbf{A}}^T\widetilde{\mathbf{w}}_k - \widetilde{\alpha}_k\widetilde{\mathbf{w}}_k - \widetilde{\delta}_k\widetilde{\mathbf{w}}_{k-1} =$
$= \mathbf{M}_2^{-T}\mathbf{A}^T\mathbf{M}_1^{-T}\mathbf{M}_2^{-T}\mathbf{w}_k - \widetilde{\alpha}_k\mathbf{M}_2^{-T}\mathbf{w}_k - \widetilde{\delta}_k\mathbf{M}_2^{-T}\mathbf{w}_{k-1}$,
que multiplicando por $\mathbf{M}_2^T$ y operando resulta,
$\widehat{\mathbf{w}}_{k+1} = \mathbf{A}^T\mathbf{M}^{-T}\mathbf{w}_k - \widetilde{\alpha}_k\mathbf{w}_k - \widetilde{\delta}_k\mathbf{w}_{k-1} = \mathbf{A}^T\mathbf{w}_k^* - \widetilde{\alpha}_k\mathbf{w}_k - \widetilde{\delta}_k\mathbf{w}_{k-1}$;
$\left\langle \widetilde{\widehat{\mathbf{v}}}_{k+1}, \widetilde{\widehat{\mathbf{w}}}_{k+1} \right\rangle = \left\langle \mathbf{M}_1^{-1}\widehat{\mathbf{v}}_{k+1}, \mathbf{M}_2^{-T}\widehat{\mathbf{w}}_{k+1} \right\rangle = \widehat{\mathbf{v}}_{k+1}^T\mathbf{M}_1^{-T}\mathbf{M}_2^{-T}\widehat{\mathbf{w}}_{k+1} =$
$= \widehat{\mathbf{v}}_{k+1}^T\mathbf{M}^{-T}\widehat{\mathbf{w}}_{k+1}$, con lo que,
$\left\langle \widetilde{\widehat{\mathbf{v}}}_{k+1}, \widetilde{\widehat{\mathbf{w}}}_{k+1} \right\rangle = \langle \mathbf{M}^{-1}\widehat{\mathbf{v}}_{k+1}, \widehat{\mathbf{w}}_{k+1} \rangle$;
y llamando $\widehat{\mathbf{v}}_k^* = \mathbf{M}^{-1}\widehat{\mathbf{v}}_{k+1}$ resulta,
$\left\langle \widetilde{\widehat{\mathbf{v}}}_{k+1}, \widetilde{\widehat{\mathbf{w}}}_{k+1} \right\rangle = \left\langle \widehat{\mathbf{v}}_{k+1}^*, \widehat{\mathbf{w}}_{k+1} \right\rangle$;
$\widetilde{\delta}_{k+1} = \left| \left\langle \widehat{\mathbf{v}}_{k+1}^*, \widehat{\mathbf{w}}_{k+1} \right\rangle \right|^{1/2}$;
$\widetilde{\beta}_{k+1} = \left\langle \widehat{\mathbf{v}}_{k+1}^*, \widehat{\mathbf{w}}_{k+1} \right\rangle / \widetilde{\delta}_{k+1}$;
$\widetilde{\mathbf{v}}_{k+1} = \widetilde{\widehat{\mathbf{v}}}_{k+1}/\widetilde{\delta}_{k+1}$ que multiplicando por $\mathbf{M}_1$ se obtiene,
$\mathbf{v}_{k+1} = \widehat{\mathbf{v}}_{k+1}/\widetilde{\delta}_{k+1}$
$\widetilde{\mathbf{w}}_{k+1} = \widetilde{\widehat{\mathbf{w}}}_{k+1}/\widetilde{\beta}_{k+1}$ que multiplicando por $\mathbf{M}_2^T$ se obtiene,
$\mathbf{w}_{k+1} = \widehat{\mathbf{w}}_{k+1}/\widetilde{\beta}_{k+1}$;
Resolver $\widetilde{\mathbf{U}}_k^T\widetilde{\mathbf{p}}=\widetilde{\mathbf{d}}_k$ y $\widetilde{\mathbf{U}}_k\widetilde{\mathbf{p}}=\widetilde{\widetilde{\mathbf{p}}}$;

$$\text{donde} \begin{cases} \left\{\widetilde{\mathbf{d}}_k\right\}_m = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{1m} \\ \left\{\widetilde{\mathbf{U}}_k\right\}_{lm} = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$$

$$\widetilde{\lambda}_k = \frac{\widetilde{\gamma}}{1 + \left\langle \widetilde{\mathbf{d}}_k, \widetilde{\mathbf{p}} \right\rangle};$$

$\widetilde{\mathbf{u}}_k = \widetilde{\lambda}_k\widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_k = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{V}}_k\widetilde{\mathbf{u}}_k$, siendo $\widetilde{\mathbf{V}}_k = [\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, ..., \widetilde{\mathbf{v}}_k]$;
entonces, $\mathbf{M}_2\mathbf{x}_k = \mathbf{M}_2\mathbf{x}_0 + \widetilde{\mathbf{V}}_k\widetilde{\mathbf{u}}_k$ y multiplicando por $\mathbf{M}_2^{-1}$,
queda, $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{M}_2^{-1}\widetilde{\mathbf{V}}_k\widetilde{\mathbf{u}}_k = \mathbf{x}_0 + \mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\mathbf{V}_k\widetilde{\mathbf{u}}_k$;
llamando $\mathbf{V}_k\widetilde{\mathbf{u}}_k = \mathbf{q}_k$ queda $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{M}^{-1}\mathbf{V}_k\widetilde{\mathbf{u}}_k$;
y haciendo $\mathbf{M}\mathbf{j}_k = \mathbf{q}_k$, queda,
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{j}_k$;
$\widetilde{\mathbf{r}}_k = \widetilde{\mathbf{V}}_{k+1}\widetilde{\widehat{\mathbf{r}}}_k$, siendo $\widetilde{\mathbf{V}}_{k+1} = [\widetilde{\mathbf{v}}_1, \widetilde{\mathbf{v}}_2, ..., \widetilde{\mathbf{v}}_{k+1}]$; entonces,
$\mathbf{M}_1^{-1}\mathbf{r}_k = \mathbf{M}_1^{-1}\mathbf{V}_{k+1}\widetilde{\widehat{\mathbf{r}}}_k$, que multiplicando por $\mathbf{M}_1$ queda,
$\mathbf{r}_k = \mathbf{V}_{k+1}\widetilde{\widehat{\mathbf{r}}}_k$, siendo $\mathbf{V}_{k+1} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{k+1}]$;

$$\text{donde} \begin{cases} \left\{\widetilde{\widehat{\mathbf{r}}}_k\right\}_1 = \widetilde{\lambda}_k \\ \left\{\widetilde{\widehat{\mathbf{r}}}_k\right\}_{l+1} = -\widetilde{\lambda}_k \left\{\widetilde{\widetilde{\mathbf{p}}}\right\}_l \end{cases} \quad l = 1, ..., k;$$

Con lo que obtenemos,

ALGORITMO QMR MODIFICADO PRECONDICIONADO POR AMBOS LADOS

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\beta_1 = \delta_1 = 0$;
$\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}$;
Resolver $\mathbf{M}_1\mathbf{z}_0 = \mathbf{r}_0$
$\gamma = \|\mathbf{z}_0\|$;
$\mathbf{v}_1 = \frac{1}{\gamma}\mathbf{z}_0$;
Elegir $\mathbf{w}_1$ tal que $\|\mathbf{M}_2^{-T}\mathbf{w}_1\| = 1$;
Mientras $\sqrt{k+1}\ \|\ \widehat{\mathbf{r}}_{k-1}\ \|\ /\ \|\ \mathbf{r}_0\ \| \geq \varepsilon \quad (k = 1, 2, 3, ...)$, hacer:

Resolver $\mathbf{M}\mathbf{v}_k^* = \mathbf{v}_k$;
Resolver $\mathbf{M}^T\mathbf{w}_k^* = \mathbf{w}_k$;
$\alpha_k = \langle \mathbf{A}\mathbf{v}_k^*, \mathbf{w}_k^* \rangle$;
$\widehat{\mathbf{v}}_{k+1} = \mathbf{A}\mathbf{v}_k^* - \alpha_k\mathbf{v}_k - \beta_k\mathbf{v}_{k-1}$;
$\widehat{\mathbf{w}}_{k+1} = \mathbf{A}^T\mathbf{w}_k^* - \alpha_k\mathbf{w}_k - \delta_k\mathbf{w}_{k-1}$;
Resolver $\mathbf{M}\widehat{\mathbf{v}}_{k+1}^* = \widehat{\mathbf{v}}_{k+1}$;
$\delta_{k+1} = \left|\left\langle \widehat{\mathbf{v}}_{k+1}^*, \widehat{\mathbf{w}}_{k+1} \right\rangle\right|^{1/2}$;
$\beta_{k+1} = \left\langle \widehat{\mathbf{v}}_{k+1}^*, \widehat{\mathbf{w}}_{k+1} \right\rangle / \delta_{k+1}$;
$\mathbf{v}_{k+1} = \widehat{\mathbf{v}}_{k+1}/\delta_{k+1}$;
$\mathbf{w}_{k+1} = \widehat{\mathbf{w}}_{k+1}/\beta_{k+1}$;
Resolver $\mathbf{U}_k^T\bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k\mathbf{p} = \bar{\mathbf{p}}$;

$$\text{donde} \begin{cases} \{\mathbf{d}_k\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$$

$\lambda_k = \dfrac{\gamma}{1 + \langle \mathbf{d}_k, \mathbf{p} \rangle}$;
$\mathbf{u}_k = \lambda_k\,\mathbf{p}$;
$\mathbf{q}_k = \mathbf{V}_k\widetilde{\mathbf{u}}_k$ con$\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$;
Resolver $\mathbf{M}\mathbf{j}_k = \mathbf{q}_k$;
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{j}_k$;
$\mathbf{r}_k = \mathbf{V}_{k+1}\widehat{\mathbf{r}}_k$, siendo $\mathbf{V}_{k+1} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_{k+1}]$;

$$\text{donde} \begin{cases} \{\widehat{\mathbf{r}}_k\}_1 = \lambda_k \\ \{\widehat{\mathbf{r}}_k\}_{l+1} = -\lambda_k\,\{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., k;$$

Fin

Atendiendo a los requisitos que hemos estimado al principio del capítulo, en los tres esquemas figuran relaciones de recurrencia respectivas para la solución $\mathbf{x}$ del sistema $\mathbf{A}\mathbf{x} = \mathbf{b}$, a pesar de estar resolviendo el sistema precondicionado $\widetilde{\mathbf{A}}\widetilde{\mathbf{x}}=\widetilde{\mathbf{b}}$. Se establece el criterio de parada en función de los vectores residuo calculado, $\widehat{\mathbf{r}}_k$, del algoritmo propuesto, que figuran en dichas relaciones. En la inicialización del algoritmo, además hay que elegir el valor inicial, $\mathbf{x}_0$, para las iteraciones de las soluciones aproximadas del sistema, que determinará el vector residuo inicial, $\mathbf{r}_0$, del sistema original, $\mathbf{A}\mathbf{x} = \mathbf{b}$.

En los algoritmos correspondientes, en la forma de precondicionamiento por la izquierda, figuran un producto añadido matriz inversa por vector en la inicialización y tres más por iteración, en la forma de precondicionamiento por la derecha sólo aparecen tres productos añadidos matriz inversa por vector en cada iteración y en el precondicionamiento por ambos lados se realizan cuatro por iteración y dos en la inicialización.

La elección de la matriz de precondicionamiento, M, es fundamental para que el coste computacional de estas operaciones no sea excesivo. Para una matriz genérica, el cálculo de su inversa equivale a resolver el sistema por un método directo. En el precondicionamiento por ambos lados, una forma conveniente de efectuar el producto es factorizando M como como producto de dos matrices triángulares inferior y superior respectivamente, con lo que la operación matriz inversa por vector se convierte en la resolución de un sistema lineal de ecuaciones por un proceso de remonte y otro de descenso.

9.2. Método TFQMR Modificado

9.2.1. Precondicionamiento por la izquierda

Aplicando el algoritmo TFQMR-Modificado al sistema precondicionado por la izquierda, expuesto en el apartado anterior, y estableciendo las nuevas expresiones para los distintos parámetros y vectores:

$\widetilde{\mathbf{s}}_0 = \widetilde{\mathbf{t}}_0 = \widetilde{\mathbf{r}}_0 = \mathbf{M}^{-1}\mathbf{r}_0 = \mathbf{z}_0$**;**
$\widetilde{\mathbf{v}}_0 = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_0 = \mathbf{M}^{-1}\mathbf{A}\widetilde{\mathbf{s}}_0$; haciendo $\mathbf{f}_0 = \mathbf{A}\widetilde{\mathbf{s}}_0$, queda, $\widetilde{\mathbf{v}}_0 = \mathbf{M}^{-1}\mathbf{f}_0$;
$\widetilde{\rho}_0 = \langle \widetilde{\mathbf{r}}_0, \widetilde{\mathbf{r}}_0^* \rangle = \langle \mathbf{M}^{-1}\mathbf{r}_0, \widetilde{\mathbf{r}}_0^* \rangle = \langle \mathbf{z}_0, \widetilde{\mathbf{r}}_0^* \rangle$**;**
$\widetilde{\delta}_1 = \|\widetilde{\mathbf{r}}_0\| = \|\mathbf{z}_0\|$;
$\widetilde{\sigma}_{i-1} = \langle \widetilde{\mathbf{v}}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle$
$\widetilde{\alpha}_{i-1} = \widetilde{\rho}_{i-1}/\widetilde{\sigma}_{i-1}$;
$\widetilde{\mathbf{q}}_i = \widetilde{\mathbf{t}}_{i-1} - \widetilde{\alpha}_{i-1}\widetilde{\mathbf{v}}_{i-1}$;
$\widetilde{\mathbf{r}}_i = \widetilde{\mathbf{r}}_{i-1} - \widetilde{\alpha}_{i-1}\widetilde{\mathbf{A}}(\widetilde{\mathbf{t}}_{i-1} + \widetilde{\mathbf{q}}_i)$, con loque,
$\mathbf{M}^{-1}\mathbf{r}_i = \mathbf{M}^{-1}\mathbf{r}_{i-1} - \widetilde{\alpha}_{i-1}\mathbf{M}^{-1}\mathbf{A}\left(\widetilde{\mathbf{t}}_{i-1} + \widetilde{\mathbf{q}}_i\right)$, con lo que,
$\mathbf{z}_i = \mathbf{z}_{i-1} - \widetilde{\alpha}_{i-1}\mathbf{A}\left(\widetilde{\mathbf{t}}_{i-1} + \widetilde{\mathbf{q}}_i\right)$ con $\mathbf{M}^{-1}\mathbf{r}_i = \mathbf{z}_i$;
Desde $k = 2i-1, 2i$ hacer
  Si k es impar $\delta_{k+1} = \sqrt{\|\widetilde{\mathbf{r}}_{i-1}\|\,\|\widetilde{\mathbf{r}}_i\|} = \sqrt{\|\mathbf{z}_{i-1}\|\,\|\mathbf{z}_i\|}$; $\widetilde{\mathbf{y}}_k = \widetilde{\mathbf{t}}_{i-1}$;
  En caso contrario $\delta_{k+1} = \|\widetilde{\mathbf{r}}_i\| = \|\mathbf{z}_i\|$; $\widetilde{\mathbf{y}}_k = \widetilde{\mathbf{q}}_i$;
Resolver $\widetilde{\mathbf{U}}_k^T\widetilde{\mathbf{p}}=\widetilde{\mathbf{d}}_k$ y $\widetilde{\mathbf{U}}_k\widetilde{\mathbf{p}}=\widetilde{\widetilde{\mathbf{p}}}$;

$$\text{donde}\begin{cases} \left\{\widetilde{\mathbf{d}}_k\right\}_m = \left\{\widetilde{\widetilde{\mathbf{T}}}\right\}^{1m} \\ \left\{\widetilde{\mathbf{U}}_k\right\}_{lm} = \left\{\widetilde{\widetilde{\mathbf{T}}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$$

$$\widetilde{\lambda}_k = \frac{\widetilde{\gamma}}{1 + \left\langle \widetilde{\mathbf{d}}_k, \widetilde{\mathbf{p}} \right\rangle};$$

$\widetilde{\mathbf{u}}_k = \widetilde{\lambda}_k\,\widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_k = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{Y}}_k\widetilde{\mathbf{u}}_k$, entonces,
$\mathbf{x}_k = \mathbf{x}_0 + \widetilde{\mathbf{Y}}_k\widetilde{\mathbf{u}}_k$; con $\widetilde{\mathbf{Y}}_k = [\widetilde{\mathbf{y}}_1, \widetilde{\mathbf{y}}_2, ..., \widetilde{\mathbf{y}}_k]$

$$\begin{cases} \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_1 = \widetilde{\lambda}_{2i} \\ \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_{l+1} = -\widetilde{\lambda}_{2i}\left\{\bar{\mathbf{p}}\right\}_l \end{cases} \qquad l = 1, ..., 2i;$$

$\widetilde{\rho}_i = \langle\widetilde{\mathbf{r}}_i, \widetilde{\mathbf{r}}_0^*\rangle = \langle\mathbf{z}_i, \widetilde{\mathbf{r}}_0^*\rangle$;
$\widetilde{\beta}_i = \widetilde{\rho}_i/\widetilde{\rho}_{i-1}$;
$\widetilde{\mathbf{t}}_i = \widetilde{\mathbf{r}}_i + \widetilde{\beta}_i\widetilde{\mathbf{q}}_i = \mathbf{M}^{-1}\mathbf{r}_i + \widetilde{\beta}_i\widetilde{\mathbf{q}}_i$, por tanto,
$\widetilde{\mathbf{t}}_i = \mathbf{z}_i + \widetilde{\beta}_i\widetilde{\mathbf{q}}_i$;
$\widetilde{\mathbf{s}}_i = \widetilde{\mathbf{t}}_i + \widetilde{\beta}_i(\widetilde{\mathbf{q}}_i + \widetilde{\beta}_i\widetilde{\mathbf{s}}_{i-1})$;
$\widetilde{\mathbf{v}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_i = \mathbf{M}^{-1}\mathbf{A}\widetilde{\mathbf{s}}_i$; haciendo $\mathbf{f}_i = \mathbf{A}\widetilde{\mathbf{s}}_i$, queda, $\mathbf{M}^{-1}\widetilde{\mathbf{v}}_i = \mathbf{f}_i$

ALGORITMO TFQMR MODIFICADO PRECONDICIONADO POR LA IZQUIERDA

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\mathbf{r}_0^*$ es arbitrario, tal que $\langle\mathbf{r}_0, \mathbf{r}_0^*\rangle \neq 0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
$\mathbf{s}_0 = \mathbf{t}_0 = \mathbf{z}_0$;
$\mathbf{f}_0 = \mathbf{A}\mathbf{s}_0$;
Resolver $\mathbf{M}\mathbf{v}_0 = \mathbf{f}_0$;
$\rho_0 = \langle\mathbf{z}_0, \mathbf{r}_0^*\rangle$;
$\delta_1 = \|\mathbf{z}_0\|$;
Mientras $\sqrt{i+1}\,\|\,\widehat{\mathbf{r}}_{i-1}\,\|\,/\,\|\,\mathbf{r}_0\,\| \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer:
  $\sigma_{i-1} = \langle\mathbf{v}_{i-1}, \mathbf{r}_0^*\rangle$;
  $\alpha_{i-1} = \rho_{i-1}/\sigma_{i-1}$;
  $\mathbf{q}_i = \mathbf{t}_{i-1} - \alpha_{i-1}\mathbf{v}_{i-1}$;
  $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1}\mathbf{A}\,(\mathbf{t}_{i-1} + \mathbf{q}_i)$;
  Resolver $\mathbf{M}\mathbf{z}_i = \mathbf{r}_i$;
  Desde $k = 2i-1, 2i$ hacer
    Si k es impar hacer
      $\delta_{k+1} = \sqrt{\|\,\mathbf{z}_{i-1}\|\,\|\mathbf{z}_i\|}$; $\mathbf{y}_k = \mathbf{t}_{i-1}$;
    En caso contrario hacer
      $\delta_{k+1} = \|\mathbf{z}_i\|$; $\mathbf{y}_k = \mathbf{q}_i$;
    Fin
  Fin
  Resolver $\mathbf{U}_k^T\bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k\mathbf{p} = \bar{\mathbf{p}}$;
    donde $\begin{cases} \{\mathbf{d}_k\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \qquad l, m = 1, ..., k;$
  $\lambda_k = \dfrac{\delta_1}{1 + \langle\mathbf{d}_k,\,\mathbf{p}\rangle}$;
  $\mathbf{u}_k = \lambda_k\,\mathbf{p}$;
  $\mathbf{x}_k = \mathbf{x}_0 + \mathbf{Y}_k\mathbf{u}_k$; con $\mathbf{Y}_k = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k]$;

$$\begin{cases} \{\widehat{\mathbf{r}}_i\}_1 = \lambda_{2i} \\ \{\widehat{\mathbf{r}}_i\}_{l+1} = -\lambda_{2i}\{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., 2i;$$

$\rho_i = \langle \mathbf{z}_i, \mathbf{r}_0^* \rangle$;
$\beta_i = \rho_i / \rho_{i-1}$;
$\mathbf{t}_i = \mathbf{z}_i + \beta_i \mathbf{q}_i$;
$\mathbf{s}_i = \mathbf{t}_i + \beta_i(\mathbf{q}_i + \beta_i \mathbf{s}_{i-1})$;
$\mathbf{f}_i = \mathbf{A}\mathbf{s}_i$
Resolver $\mathbf{M}\mathbf{v}_i = \mathbf{f}_i$;
Fin

9.2.2. Precondicionamiento por la derecha

Aplicando el algoritmo TFQMR-Modificado al sistema precondicionado por la derecha, expuesto en el apartado anterior, y estableciendo las nuevas expresiones para los distintos parámetros y vectores:

$\widetilde{\mathbf{s}}_0 = \widetilde{\mathbf{t}}_0 = \widetilde{\mathbf{r}}_0 = \mathbf{r}_0$;
$\widetilde{\mathbf{v}}_0 = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_0 = \mathbf{A}\mathbf{M}^{-1}\widetilde{\mathbf{s}}_0$; haciendo $\mathbf{M}\mathbf{f}_0 = \widetilde{\mathbf{s}}_0$ queda $\widetilde{\mathbf{v}}_0 = \mathbf{A}\mathbf{f}_0$;
$\widetilde{\rho}_0 = \langle \widetilde{\mathbf{r}}_0, \widetilde{\mathbf{r}}_0^* \rangle = \langle \mathbf{r}_0, \widetilde{\mathbf{r}}_0^* \rangle$;
$\widetilde{\delta}_1 = \|\widetilde{\mathbf{r}}_0\| = \|\mathbf{r}_0\|$;
$\widetilde{\sigma}_{i-1} = \langle \widetilde{\mathbf{v}}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle$;
$\widetilde{\alpha}_{i-1} = \widetilde{\rho}_{i-1} / \widetilde{\sigma}_{i-1}$;
$\widetilde{\mathbf{q}}_i = \widetilde{\mathbf{t}}_{i-1} - \widetilde{\alpha}_{i-1}\widetilde{\mathbf{v}}_{i-1}$;
$\widetilde{\mathbf{r}}_i = \widetilde{\mathbf{r}}_{i-1} - \widetilde{\alpha}_{i-1}\widetilde{\mathbf{A}}\left(\widetilde{\mathbf{t}}_{i-1} + \widetilde{\mathbf{q}}_i\right)$; $\mathbf{r}_i = \mathbf{r}_{i-1} - \widetilde{\alpha}_{i-1}\mathbf{A}\left(\mathbf{M}^{-1}\widetilde{\mathbf{t}}_{i-1} + \mathbf{M}^{-1}\widetilde{\mathbf{q}}_i\right)$;
haciendo $\mathbf{M}\mathbf{g}_{i-1} = \widetilde{\mathbf{t}}_{i-1}$, y, $\mathbf{M}\mathbf{h}_i = \widetilde{\mathbf{q}}_i$, queda,
$\mathbf{r}_i = \mathbf{r}_{i-1} - \widetilde{\alpha}_{i-1}\mathbf{A}\left(\mathbf{g}_{i-1} + \mathbf{h}_i\right)$;
Desde $k = 2i-1, 2i$ hacer
Si k es impar $\widetilde{\delta}_{k+1} = \sqrt{\|\widetilde{\mathbf{r}}_{i-1}\|\,\|\widetilde{\mathbf{r}}_i\|} = \sqrt{\|\mathbf{r}_{i-1}\|\,\|\mathbf{r}_i\|}$; $\widetilde{\mathbf{y}}_k = \widetilde{\mathbf{t}}_{i-1}$;
En caso contrario $\widetilde{\delta}_{k+1} = \|\widetilde{\mathbf{r}}_i\| = \|\mathbf{r}_i\|$; $\widetilde{\mathbf{y}}_k = \widetilde{\mathbf{q}}_i$;
Resolver $\widetilde{\mathbf{U}}_k^T\widetilde{\mathbf{p}} = \widetilde{\mathbf{d}}_k$ y $\widetilde{\mathbf{U}}_k\widetilde{\mathbf{p}} = \widetilde{\bar{\mathbf{p}}}$;

$$\text{donde} \begin{cases} \left\{\widetilde{\mathbf{d}}_k\right\}_m = \left\{\widetilde{\overline{\mathbf{T}}}\right\} \\ \left\{\widetilde{\mathbf{U}}_k\right\}_{lm} = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$$

$$\widetilde{\lambda}_k = \frac{\widetilde{\gamma}}{1 + \left\langle \widetilde{\mathbf{d}}_k, \widetilde{\mathbf{p}} \right\rangle};$$

$\widetilde{\mathbf{u}}_k = \widetilde{\lambda}_k \widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_k = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{Y}}_k\widetilde{\mathbf{u}}_k$; esto es, $\mathbf{M}\mathbf{x}_k = \mathbf{M}\mathbf{x}_0 + \widetilde{\mathbf{Y}}_k\widetilde{\mathbf{u}}_k$;
Multiplicando por $\mathbf{M}^{-1}$ queda,
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{M}_k^{-1}\widetilde{\mathbf{Y}}\widetilde{\mathbf{u}}_k$ con $\widetilde{\mathbf{Y}}_k = [\widetilde{\mathbf{y}}_1, \widetilde{\mathbf{y}}_2, ..., \widetilde{\mathbf{y}}_k]$
y llamando a $\widetilde{\mathbf{Y}}_k\widetilde{\mathbf{u}}_k = \mathbf{c}_k$ y haciendo $\mathbf{M}\mathbf{j}_k = \mathbf{c}_k$ obtenemos,
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{j}_k$;

$$\begin{cases} \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_1 = \widetilde{\lambda}_{2i} \\ \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_{l+1} = -\widetilde{\lambda}_{2i}\left\{\bar{\mathbf{p}}\right\}_l \end{cases} \quad l = 1, ..., 2i;$$

$\widetilde{\rho}_i = \langle \widetilde{\mathbf{r}}_i, \widetilde{\mathbf{r}}_0^* \rangle = \langle \mathbf{r}_i, \widetilde{\mathbf{r}}_0^* \rangle$;
$\widetilde{\beta}_i = \widetilde{\rho}_i / \widetilde{\rho}_{i-1}$;
$\widetilde{\mathbf{t}}_i = \widetilde{\mathbf{r}}_i + \widetilde{\beta}_i \widetilde{\mathbf{q}}_i = \mathbf{r}_i + \widetilde{\beta}_i \widetilde{\mathbf{q}}_i$;
$\widetilde{\mathbf{s}}_i = \widetilde{\mathbf{t}}_i + \widetilde{\beta}_i(\widetilde{\mathbf{q}}_i + \widetilde{\beta}_i \widetilde{\mathbf{s}}_{i-1})$;
$\widetilde{\mathbf{v}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_i = \mathbf{A}\mathbf{M}^{-1}\widetilde{\mathbf{s}}_i$; haciendo $\mathbf{M}\mathbf{f}_i = \widetilde{\mathbf{s}}_i$ queda,
$\widetilde{\mathbf{v}}_i = \mathbf{A}\mathbf{f}_i$;

ALGORITMO TFQMR MODIFICADO PRECONDICIONADO POR LA DERECHA
Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\mathbf{r}_0^*$ es arbitrario, tal que $\langle \mathbf{r}_0^*, \mathbf{r}_0 \rangle \neq 0$;
$\mathbf{s}_0 = \mathbf{t}_0 = \mathbf{r}_0$;
Resolver $\mathbf{M}\mathbf{f}_0 = \mathbf{s}_0$
$\mathbf{v}_0 = \mathbf{A}\mathbf{f}_0$;
$\rho_0 = \langle \mathbf{r}_0, \mathbf{r}_0^* \rangle$;
$\delta_1 = \| \mathbf{r}_0 \|$;
Mientras $\sqrt{i+1} \| \widehat{\mathbf{r}}_{i-1} \| / \| \mathbf{r}_0 \| \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer
$\sigma_{i-1} = \langle \mathbf{v}_{i-1}, \mathbf{r}_0^* \rangle$;
$\alpha_{i-1} = \rho_{i-1} / \sigma_{i-1}$;
$\mathbf{q}_i = \mathbf{t}_{i-1} - \alpha_{i-1}\mathbf{v}_{i-1}$;
Resolver $\mathbf{M}\mathbf{g}_{i-1} = \mathbf{t}_{i-1}$;
Resolver $\mathbf{M}\mathbf{h}_i = \mathbf{q}_i$;
$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1}\mathbf{A}(\mathbf{g}_{i-1} + \mathbf{h}_i)$;
Desde $k = 2i-1, 2i$ hacer
Si k es impar hacer
$\delta_{k+1} = \sqrt{\| \mathbf{r}_{i-1} \| \ \| \mathbf{r}_i \|}$; $\mathbf{y}_k = \mathbf{t}_{i-1}$;
En caso contrario hacer
$\delta_{k+1} = \| \mathbf{r}_i \|$; $\mathbf{y}_k = \mathbf{q}_i$;
Fin
Fin
Resolver $\mathbf{U}_k^T \bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k \mathbf{p} = \bar{\mathbf{p}}$;
donde $\begin{cases} \{\mathbf{d}_k\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k;$
$\lambda_k = \dfrac{\delta_1}{1 + \langle \mathbf{d}_k, \mathbf{p} \rangle}$;
$\mathbf{u}_k = \lambda_k \mathbf{p}$;
$\mathbf{c}_k = \mathbf{Y}_k \mathbf{u}_k$; con $\mathbf{Y}_k = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k]$;
Resolver $\mathbf{M}\mathbf{j}_k = \mathbf{c}_k$
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{j}_k$;
$\begin{cases} \{\widehat{\mathbf{r}}_i\}_1 = \lambda_{2i} \\ \{\widehat{\mathbf{r}}_i\}_{l+1} = -\lambda_{2i}\{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., 2i;$

$\rho_i = \langle \mathbf{r}_i, \mathbf{r}_0^* \rangle$;
$\beta_i = \rho_i / \rho_{i-1}$;
$\mathbf{t}_i = \mathbf{r}_i + \beta_i \mathbf{q}_i$;
$\mathbf{s}_i = \mathbf{t}_i + \beta_i(\mathbf{q}_i + \beta_i \mathbf{s}_{i-1})$;
Resolver $\mathbf{M}\mathbf{f}_i = \mathbf{s}_i$;
$\mathbf{v}_i = \mathbf{A}\mathbf{f}_i$;
Fin

9.2.3. Precondicionamiento por ambos lados

Aplicando el algoritmo TFQMR-Modificado al sistema precondicionado por ambos lados, expuesto en el apartado anterioir, y estableciendo las nuevas expresiones para los distintos parámetros y vectores:
$\widetilde{\mathbf{s}}_0 = \widetilde{\mathbf{t}}_0 = \widetilde{\mathbf{r}}_0 = \mathbf{M}_1^{-1}\mathbf{r}_0$, que multiplicando por $\mathbf{M}_1$ queda,
$\mathbf{s}_0 = \mathbf{t}_0 = \mathbf{r}_0$
$\widetilde{\mathbf{v}}_0 = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_0 = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\mathbf{s}_0 = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}^{-1}\mathbf{s}_0$;
Multiplicando por $\mathbf{M}_1$, y haciendo $\mathbf{M}\mathbf{f}_0 = \mathbf{s}_0$, queda, $\mathbf{v}_0 = \mathbf{A}\mathbf{f}_0$;
$\widetilde{\rho}_0 = \langle \widetilde{\mathbf{r}}_0, \widetilde{\mathbf{r}}_0^* \rangle = \left\langle \mathbf{M}_1^{-1}\mathbf{r}_0, \widetilde{\mathbf{r}}_0^* \right\rangle = \mathbf{r}_0^T\mathbf{M}_1^{-T}\widetilde{\mathbf{r}}_0^* = \mathbf{r}_0^T\mathbf{M}_1^{-T}\mathbf{M}_2^{-T}\mathbf{r}_0^* = \mathbf{r}_0^T\mathbf{M}^{-T}\mathbf{r}_0^*$
entonces, $\widetilde{\rho}_0 = \langle \mathbf{M}^{-1}\mathbf{r}_0, \mathbf{r}_0^* \rangle$, con $\widetilde{\mathbf{r}}_0^* = \mathbf{M}_2^{-T}\mathbf{r}_0^*$,
y haciendo $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$ queda, $\widetilde{\rho}_0 = \langle \mathbf{z}_0, \mathbf{r}_0^* \rangle$;
$\widetilde{\delta}_1 = \|\mathbf{z}_0\|$;
$\widetilde{\sigma}_{i-1} = \langle \widetilde{\mathbf{v}}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle = \left\langle \mathbf{M}_1^{-1}\mathbf{v}_{i-1}, \widetilde{\mathbf{r}}_0^* \right\rangle = \left\langle \mathbf{M}_1^{-1}\mathbf{v}_{i-1}, \mathbf{M}_2^{-T}\mathbf{r}_0^* \right\rangle$ esto es,
$\widetilde{\sigma}_{i-1} = \mathbf{v}_{i-1}^T\mathbf{M}_1^{-T}\mathbf{M}_2^{-T}\mathbf{r}_0^* = \mathbf{v}_{i-1}^T\mathbf{M}^{-T}\mathbf{r}_0^*$,
entonces, $\widetilde{\sigma}_{i-1} = \langle \mathbf{M}^{-1}\mathbf{v}_{i-1}, \mathbf{r}_0^* \rangle$ y haciendo $\mathbf{M}\mathbf{c}_{i-1} = \mathbf{v}_{i-1}$ resulta,
$\widetilde{\sigma}_{i-1} = \langle \mathbf{c}_{i-1}, \mathbf{r}_0^* \rangle$;
$\widetilde{\alpha}_{i-1} = \widetilde{\rho}_{i-1} / \widetilde{\sigma}_{i-1}$;
$\widetilde{\mathbf{q}}_i = \widetilde{\mathbf{t}}_{i-1} - \widetilde{\alpha}_{i-1}\widetilde{\mathbf{v}}_{i-1}$; multiplicando por $\mathbf{M}_1$ resulta,
$\mathbf{q}_i = \mathbf{t}_{i-1} - \widetilde{\alpha}_{i-1}\mathbf{v}_{i-1}$;
$\widetilde{\mathbf{r}}_i = \widetilde{\mathbf{r}}_{i-1} - \widetilde{\alpha}_{i-1}\widetilde{\mathbf{A}}\left(\widetilde{\mathbf{t}}_{i-1} + \widetilde{\mathbf{q}}_i\right) = \widetilde{\mathbf{r}}_{i-1} - \widetilde{\alpha}_{i-1}\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\left(\mathbf{t}_{i-1} + \mathbf{q}_i\right)$;
Multiplicando por $\mathbf{M}_1$ queda, $\mathbf{r}_i = \mathbf{r}_{i-1} - \widetilde{\alpha}_{i-1}\mathbf{A}\left(\mathbf{M}^{-1}\mathbf{t}_{i-1} + \mathbf{M}^{-1}\mathbf{q}_i\right)$;
Haciendo $\mathbf{M}\mathbf{g}_{i-1} = \mathbf{t}_{i-1}$ y $\mathbf{M}\mathbf{h}_i = \mathbf{q}_i$ queda,
$\mathbf{r}_i = \mathbf{r}_{i-1} - \widetilde{\alpha}_{i-1}\mathbf{A}\left(\mathbf{g}_{i-1} + \mathbf{h}_i\right)$;
Desde $k = 2i-1, 2i$ hacer
Si k es impar $\widetilde{\delta}_{k+1} = \sqrt{\|\mathbf{z}_{i-1}\|\,\|\mathbf{z}_i\|}$ con $\mathbf{M}\mathbf{z}_i = \mathbf{r}_i$;
$\widetilde{\mathbf{y}}_k = \widetilde{\mathbf{t}}_{i-1}$, que multiplicando por $\mathbf{M}_1$ queda, $\mathbf{y}_k = \mathbf{t}_{i-1}$;
En caso contrario $\widetilde{\delta}_{k+1} = \|\mathbf{z}_i\|$;
$\widetilde{\mathbf{y}}_k = \widetilde{\mathbf{q}}_i$, que multiplicando por $\mathbf{M}_1$ queda, $\mathbf{y}_k = \mathbf{q}_i$;
Resolver $\widetilde{\mathbf{U}}_k^T\widetilde{\mathbf{p}}=\widetilde{\mathbf{d}}_k$ y $\widetilde{\mathbf{U}}_k\widetilde{\mathbf{p}}=\widetilde{\widetilde{\mathbf{p}}}$;

$$\text{donde} \begin{cases} \left\{\widetilde{\mathbf{d}}_k\right\}_m = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{1m} \\ \left\{\widetilde{\mathbf{U}}_k\right\}_{lm} = \left\{\overline{\overline{\mathbf{T}}}\right\}_{l+1m} \end{cases} \qquad l, m = 1, ..., k;$$

$\widetilde{\lambda}_k = \dfrac{\widetilde{\gamma}}{1 + \left\langle \widetilde{\mathbf{d}}_k, \widetilde{\mathbf{p}} \right\rangle}$;
$\widetilde{\mathbf{u}}_k = \widetilde{\lambda}_k \widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_k = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{Y}}_k \widetilde{\mathbf{u}}_k$, con $\widetilde{\mathbf{Y}}_k = [\widetilde{\mathbf{y}}_1, \widetilde{\mathbf{y}}_2, ..., \widetilde{\mathbf{y}}_k]$;
entonces, $\mathbf{M}_2\mathbf{x}_k = \mathbf{M}_2\mathbf{x}_0 + \widetilde{\mathbf{Y}}_k \widetilde{\mathbf{u}}_k$;
que multiplicando por $\mathbf{M}_2^{-1}$ queda,
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{M}_2^{-1}\widetilde{\mathbf{Y}}_k \widetilde{\mathbf{u}}_k = \mathbf{x}_0 + \mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\mathbf{Y}_k \widetilde{\mathbf{u}}_k$, con $\mathbf{Y}_k = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k]$;
con lo que, $\mathbf{x}_0 + \mathbf{M}^{-1}\mathbf{Y}_k \widetilde{\mathbf{u}}_k$;
Llamando a $\mathbf{Y}_k \widetilde{\mathbf{u}}_k = \mathbf{u}_k^*$ y haciendo $\mathbf{M}\mathbf{j}_k = \mathbf{u}_k^*$, obtenemos,
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{j}_k$;

$$\begin{cases} \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_1 = \widetilde{\lambda}_{2i} \\ \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_{l+1} = -\widetilde{\lambda}_{2i} \left\{\bar{\mathbf{p}}\right\}_l \end{cases} \qquad l = 1, ..., 2i;$$

$\widetilde{\rho}_i = \langle \widetilde{\mathbf{r}}_i, \widetilde{\mathbf{r}}_0^* \rangle = \langle \mathbf{z}_i, \widetilde{\mathbf{r}}_0^* \rangle$;
$\widetilde{\beta}_i = \widetilde{\rho}_i / \widetilde{\rho}_{i-1}$;
$\widetilde{\mathbf{t}}_i = \widetilde{\mathbf{r}}_i + \widetilde{\beta}_i \widetilde{\mathbf{q}}_i$; multiplicando por $\mathbf{M}_1$ resulta,
$\mathbf{t}_i = \mathbf{r}_i + \widetilde{\beta}_i \mathbf{q}_i$
$\widetilde{\mathbf{s}}_i = \widetilde{\mathbf{t}}_i + \widetilde{\beta}_i(\widetilde{\mathbf{q}}_i + \widetilde{\beta}_i \widetilde{\mathbf{s}}_{i-1})$; multiplicando por $\mathbf{M}_1$ resulta,
$\mathbf{s}_i = \mathbf{t}_i + \widetilde{\beta}_i(\mathbf{q}_i + \widetilde{\beta}_i \mathbf{s}_{i-1})$;
$\widetilde{\mathbf{v}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_i = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\mathbf{s}_i$; multiplicando por $\mathbf{M}_1$ resulta,
$\mathbf{v}_i = \mathbf{A}\mathbf{M}^{-1}\mathbf{s}_i$ y haciendo $\mathbf{M}\mathbf{f}_i = \mathbf{s}_i$, resulta,
$\mathbf{v}_i = \mathbf{A}\mathbf{f}_i$;

ALGORITMO TFQMR MODIFICADO PRECONDICIONADO POR AMBOS LADOS

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\mathbf{r}_0^*$ es arbitrario, tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
$\mathbf{s}_0 = \mathbf{t}_0 = \mathbf{r}_0$;
Resolver $\mathbf{M}\mathbf{f}_0 = \mathbf{s}_0$;
$\mathbf{v}_0 = \mathbf{A}\mathbf{f}_0$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
$\rho_0 = \langle \mathbf{z}_0, \mathbf{r}_0^* \rangle$;
$\delta_1 = \| \mathbf{z}_0 \|$;
Mientras $\sqrt{i+1} \parallel \widehat{\mathbf{r}}_{i-1} \parallel / \parallel \mathbf{r}_0 \parallel \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer
    Resolver $\mathbf{M}\mathbf{c}_{i-1} = \mathbf{v}_{i-1}$
    $\sigma_{i-1} = \langle \mathbf{c}_{i-1}, \mathbf{r}_0^* \rangle$;
    $\alpha_{i-1} = \rho_{i-1} / \sigma_{i-1}$;
    $\mathbf{q}_i = \mathbf{t}_{i-1} - \alpha_{i-1}\mathbf{v}_{i-1}$;
    Resolver $\mathbf{M}\mathbf{g}_{i-1} = \mathbf{t}_{i-1}$;
    Resolver $\mathbf{M}\mathbf{h}_i = \mathbf{q}_i$;
    $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1}\mathbf{A}\,(\mathbf{g}_{i-1} + \mathbf{h}_i)$;
    Resolver $\mathbf{M}\mathbf{z}_i = \mathbf{r}_i$

Desde $k = 2i-1, 2i$ hacer
  Si k es impar hacer
    $\delta_{k+1} = \sqrt{\|\mathbf{z}_{i-1}\| \ \|\mathbf{z}_i\|}$; $\mathbf{y}_k = \mathbf{t}_{i-1}$;
  En caso contrario hacer
    $\delta_{k+1} = \|\mathbf{z}_i\|$; $\mathbf{y}_k = \mathbf{q}_i$;
  Fin
Fin
Resolver $\mathbf{U}_k^T \bar{\mathbf{p}} = \mathbf{d}_k$ y $\mathbf{U}_k \mathbf{p} = \bar{\mathbf{p}}$;
  donde $\begin{cases} \{\mathbf{d}_k\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_k\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., k$;
$\lambda_k = \dfrac{\delta_1}{1 + \langle \mathbf{d}_k, \ \mathbf{p} \rangle}$;
$\mathbf{u}_k = \lambda_k \, \mathbf{p}$;
$\mathbf{u}_k^* = \mathbf{Y}_k \mathbf{u}_k$; con $\mathbf{Y}_k = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k]$;
Resolver $\mathbf{M}\mathbf{j}_k = \mathbf{u}_k^*$;
$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{j}_k$;
$\begin{cases} \{\widehat{\mathbf{r}}_i\}_1 = \lambda_{2i} \\ \{\widehat{\mathbf{r}}_i\}_{l+1} = -\lambda_{2i} \, \{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., 2i$;
$\rho_i = \langle \mathbf{z}_i, \mathbf{r}_0^* \rangle$;
$\beta_i = \rho_i / \rho_{i-1}$;
$\mathbf{t}_i = \mathbf{r}_i + \beta_i \mathbf{q}_i$;
$\mathbf{s}_i = \mathbf{t}_i + \beta_i (\mathbf{q}_i + \beta_i \mathbf{s}_{i-1})$;
Resolver $\mathbf{M}\mathbf{f}_i = \mathbf{s}_i$;
$\mathbf{v}_i = \mathbf{A}\mathbf{f}_i$;
Fin

El algoritmo correspondiente a la forma de precondicionamiento por la izquierda, requiere dos productos añadidos matriz inversa por vector en la inicialización y dos más por iteración; en el precondicionamiento por la derecha se requieren, un producto añadido matriz inversa por vector en la inicialización y cuatro por cada iteración y en el precondicionamiento por ambos lados se realizan dos productos añadidos matriz inversa por vector en la inicialización y seis por iteración.

9.3. Método QMRCGSTAB Modificado

9.3.1. Precondicionamiento por la izquierda

Aplicando el algoritmo QMRCGSTAB-Modificado al sistema precondicionado por la izquierda y estableciendo las nuevas expresiones para los distintos parámetros y vectores:

$\widetilde{\rho}_0 = \widetilde{\alpha}_0 = \widetilde{\omega}_0 = 1$;

$\widetilde{\mathbf{g}}_0 = \widetilde{\mathbf{v}}_0 = \mathbf{0}$;
$\widetilde{\rho}_i = \langle \widetilde{\mathbf{r}}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle = \langle \mathbf{M}^{-1}\mathbf{r}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle = \langle \mathbf{z}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle$ con $\mathbf{M}^{-1}\mathbf{r}_{i-1} = \mathbf{z}_{i-1}$;
$\widetilde{\beta}_i = (\widetilde{\rho}_i / \widetilde{\rho}_{i-1})(\widetilde{\alpha}_{i-1} / \widetilde{\omega}_{i-1})$;
$\widetilde{\mathbf{g}}_i = \widetilde{\mathbf{r}}_{i-1} + \widetilde{\beta}_i(\widetilde{\mathbf{g}}_{i-1} - \widetilde{\omega}_{i-1}\widetilde{\mathbf{v}}_{i-1}) = \mathbf{M}^{-1}\mathbf{r}_{i-1} + \widetilde{\beta}_i(\widetilde{\mathbf{g}}_{i-1} - \widetilde{\omega}_{i-1}\widetilde{\mathbf{v}}_{i-1})$ entonces,
$\widetilde{\mathbf{g}}_i = \mathbf{z}_{i-1} + \widetilde{\beta}_i(\widetilde{\mathbf{g}}_{i-1} - \widetilde{\omega}_{i-1}\widetilde{\mathbf{v}}_{i-1})$;
$\widetilde{\mathbf{v}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{g}}_i = \mathbf{M}^{-1}\mathbf{A}\widetilde{\mathbf{g}}_i$, haciendo $\mathbf{A}\widetilde{\mathbf{g}}_i = \mathbf{f}_i$, resulta, $\widetilde{\mathbf{v}}_i = \mathbf{M}^{-1}\mathbf{f}_i$;
$\widetilde{\alpha}_i = \dfrac{\widetilde{\rho}_i}{\langle \widetilde{\mathbf{v}}_i, \widetilde{\mathbf{r}}_0^* \rangle}$;
$\widetilde{\mathbf{s}}_i = \widetilde{\mathbf{r}}_{i-1} - \widetilde{\alpha}_i\widetilde{\mathbf{v}}_i = \mathbf{M}^{-1}\mathbf{r}_{i-1} - \widetilde{\alpha}_i\widetilde{\mathbf{v}}_i$ con lo que,
$\widetilde{\mathbf{s}}_i = \mathbf{z}_{i-1} - \widetilde{\alpha}_i\widetilde{\mathbf{v}}_i$;
$\widetilde{\delta}_{2i} = \|\widetilde{\mathbf{s}}_i\|$;
$\widetilde{\mathbf{y}}_{2i-1} = \widetilde{\mathbf{g}}_i$;
$\widetilde{\mathbf{t}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_i = \mathbf{M}^{-1}\mathbf{A}\widetilde{\mathbf{s}}_i$ haciendo, $\mathbf{A}\widetilde{\mathbf{s}}_i = \mathbf{h}_i$, resulta $\widetilde{\mathbf{t}}_i = \mathbf{M}^{-1}\mathbf{h}_i$;
$\widetilde{\omega}_i = \dfrac{\left\langle \widetilde{\mathbf{t}}_i, \widetilde{\mathbf{s}}_i \right\rangle}{\left\langle \widetilde{\mathbf{t}}_i, \widetilde{\mathbf{t}}_i \right\rangle}$;
$\widetilde{\mathbf{r}}_i = \widetilde{\mathbf{s}}_i - \widetilde{\omega}_i\widetilde{\mathbf{t}}_i$ esto es, $\mathbf{M}^{-1}\mathbf{r}_i = \widetilde{\mathbf{s}}_i - \widetilde{\omega}_i\widetilde{\mathbf{t}}_i$, entonces
$\mathbf{z}_i = \widetilde{\mathbf{s}}_i - \widetilde{\omega}_i\widetilde{\mathbf{t}}_i$, con $\mathbf{M}^{-1}\mathbf{r}_i = \mathbf{z}_i$;
$\widetilde{\delta}_{2i+1} = \|\widetilde{\mathbf{r}}_i\| = \|\mathbf{z}_i\|$;
$\widetilde{\mathbf{y}}_{2i} = \widetilde{\mathbf{s}}_i$;
Resolver $\widetilde{\mathbf{U}}_{2i}^t\widetilde{\bar{\mathbf{p}}} = \widetilde{\mathbf{d}}_{2i}$ y $\widetilde{\mathbf{U}}_{2i}\widetilde{\mathbf{p}} = \widetilde{\bar{\mathbf{p}}}$;

$$\text{donde} \begin{cases} \left\{\widetilde{\mathbf{d}}_{2i}\right\}_m = \left\{\widetilde{\overline{\mathbf{T}}}\right\} \\ \left\{\widetilde{\mathbf{U}}_{2i}\right\}_{lm} = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{l+1m}^{1m} \end{cases} \qquad l, m = 1, ..., 2i;$$

$\widetilde{\lambda}_{2i} = \dfrac{\widetilde{\delta}_1}{1 + \left\langle \widetilde{\mathbf{d}}_{2i}, \widetilde{\mathbf{p}} \right\rangle}$;
$\widetilde{\mathbf{u}}_{2i} = \widetilde{\lambda}_{2i}\,\widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_i = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{Y}}_{2i}\widetilde{\mathbf{u}}_{2i}$ con $\widetilde{\mathbf{Y}}_{2i} = [\widetilde{\mathbf{y}}_1, \widetilde{\mathbf{y}}_2, ..., \widetilde{\mathbf{y}}_{2i}]$; entonces,
$\mathbf{x}_i = \mathbf{x}_0 + \widetilde{\mathbf{Y}}_{2i}\widetilde{\mathbf{u}}_{2i}$;

$$\begin{cases} \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_1 = \widetilde{\lambda}_{2i} \\ \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_{l+1} = -\widetilde{\lambda}_{2i}\left\{\bar{\mathbf{p}}\right\}_l \end{cases} \qquad l = 1, ..., 2i;$$

ALGORITMO QMRCGSTAB MODIFICADO PRECONDICIONADO POR LA IZQUIERDA

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\mathbf{r}_0^*$ es arbitrario, tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
$\rho_0 = \alpha_0 = \omega_0 = 1$;
$\mathbf{g}_0 = \mathbf{v}_0 = \mathbf{0}$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$;
Mientras $\sqrt{2i+1}\ \|\ \widehat{\mathbf{r}}_{i-1}\ \|\ /\ \|\ \mathbf{r}_0\ \| \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer
$\rho_i = \langle \mathbf{z}_{i-1}, \mathbf{r}_0^* \rangle$;

$\beta_i = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$;
$\mathbf{g}_i = \mathbf{z}_{i-1} + \beta_i(\mathbf{g}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1})$;
$\mathbf{f}_i = \mathbf{A}\mathbf{g}_i$;
Resolver $\mathbf{M}\mathbf{v}_i = \mathbf{f}_i$
$\alpha_i = \dfrac{\rho_i}{\langle \mathbf{v}_i, \mathbf{r}_0^* \rangle}$;
$\mathbf{s}_i = \mathbf{z}_{i-1} - \alpha_i \mathbf{v}_i$;
$\delta_{2i} = \|\mathbf{s}_i\|$; $\mathbf{y}_{2i-1} = \mathbf{g}_i$;
$\mathbf{h}_i = \mathbf{A}\mathbf{s}_i$;
Resolver $\mathbf{M}\mathbf{t}_i = \mathbf{h}_i$;
$\omega_i = \dfrac{\langle \mathbf{t}_i, \mathbf{s}_i \rangle}{\langle \mathbf{t}_i, \mathbf{t}_i \rangle}$;
$\mathbf{z}_i = \mathbf{s}_i - \omega_i \mathbf{t}_i$;
$\delta_{2i+1} = \|\mathbf{z}_i\|$; $\mathbf{y}_{2i} = \mathbf{s}_i$;
Resolver $\mathbf{U}_{2i}^t \bar{\mathbf{p}} = \mathbf{d}_{2i}$ y $\mathbf{U}_{2i}\mathbf{p} = \bar{\mathbf{p}}$;

$$\text{donde} \begin{cases} \{\mathbf{d}_{2i}\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_{2i}\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., 2i;$$

$\lambda_{2i} = \dfrac{\delta_1}{1 + \langle \mathbf{d}_{2i},\ \mathbf{p} \rangle}$;
$\mathbf{u}_{2i} = \lambda_{2i}\, \mathbf{p}$;
$\mathbf{x}_i = \mathbf{x}_0 + \mathbf{Y}_{2i}\mathbf{u}_{2i}$; con $\mathbf{Y}_{2i} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{2i}]$;

$$\begin{cases} \{\widehat{\mathbf{r}}_i\}_1 = \lambda_{2i} \\ \{\widehat{\mathbf{r}}_i\}_{l+1} = -\lambda_{2i}\, \{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., 2i;$$

Fin

9.3.2. Precondicionamiento por la derecha

Aplicando el algoritmo QMRCGSTAB-Modificado al sistema precondicionado por la derecha y estableciendo las nuevas expresiones para los distintos parámetros y vectores:

$\widetilde{\rho}_0 = \widetilde{\alpha}_0 = \widetilde{\omega}_0 = 1$;
$\widetilde{\mathbf{g}}_0 = \widetilde{\mathbf{v}}_0 = \mathbf{0}$;
$\widetilde{\rho}_i = \langle \widetilde{\mathbf{r}}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle = \langle \mathbf{r}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle$;
$\widetilde{\beta}_i = (\widetilde{\rho}_i/\widetilde{\rho}_{i-1})(\widetilde{\alpha}_{i-1}/\widetilde{\omega}_{i-1})$;
$\widetilde{\mathbf{g}}_i = \widetilde{\mathbf{r}}_{i-1} + \widetilde{\beta}_i(\widetilde{\mathbf{g}}_{i-1} - \widetilde{\omega}_{i-1}\widetilde{\mathbf{v}}_{i-1}) = \mathbf{r}_{i-1} + \widetilde{\beta}_i(\widetilde{\mathbf{g}}_{i-1} - \widetilde{\omega}_{i-1}\widetilde{\mathbf{v}}_{i-1})$;
$\widetilde{\mathbf{v}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{g}}_i = \mathbf{A}\mathbf{M}^{-1}\widetilde{\mathbf{g}}_i$ y haciendo $\mathbf{M}\mathbf{f}_i = \widetilde{\mathbf{g}}_i$, resulta, $\widetilde{\mathbf{v}}_i = \mathbf{A}\mathbf{f}_i$;
$\widetilde{\alpha}_i = \dfrac{\widetilde{\rho}_i}{\langle \widetilde{\mathbf{v}}_i, \widetilde{\mathbf{r}}_0^* \rangle}$;
$\widetilde{\mathbf{s}}_i = \widetilde{\mathbf{r}}_{i-1} - \widetilde{\alpha}_i\widetilde{\mathbf{v}}_i = \mathbf{r}_{i-1} - \widetilde{\alpha}_i\widetilde{\mathbf{v}}_i$;
$\widetilde{\delta}_{2i-1} = \|\widetilde{\mathbf{s}}_i\|$; $\widetilde{\mathbf{y}}_{2i-1} = \widetilde{\mathbf{g}}_i$;
$\widetilde{\mathbf{t}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_i = \mathbf{A}\mathbf{M}^{-1}\widetilde{\mathbf{s}}_i$ y haciendo $\mathbf{M}\mathbf{h}_i = \widetilde{\mathbf{s}}_i$ resulta, $\widetilde{\mathbf{t}}_i = \mathbf{A}\mathbf{h}_i$;
$\widetilde{\omega}_i = \dfrac{\left\langle \widetilde{\mathbf{t}}_i, \widetilde{\mathbf{s}}_i \right\rangle}{\left\langle \widetilde{\mathbf{t}}_i, \widetilde{\mathbf{t}}_i \right\rangle}$;

$\widetilde{\mathbf{r}}_i = \widetilde{\mathbf{s}}_i - \widetilde{\omega}_i \widetilde{\mathbf{t}}_i$ que es $\mathbf{r}_i = \widetilde{\mathbf{s}}_i - \widetilde{\omega}_i \widetilde{\mathbf{t}}_i$;
$\widetilde{\delta}_{2i} = \|\widetilde{\mathbf{r}}_i\| = \|\mathbf{r}_i\|$; $\widetilde{\mathbf{y}}_{2i} = \widetilde{\mathbf{s}}_i$;
Resolver $\widetilde{\mathbf{U}}_{2i}^t \widetilde{\bar{\mathbf{p}}} = \widetilde{\mathbf{d}}_{2i}$ y $\widetilde{\mathbf{U}}_{2i} \widetilde{\mathbf{p}} = \widetilde{\bar{\mathbf{p}}}$;

$$\text{donde} \begin{cases} \left\{\widetilde{\mathbf{d}}_{2i}\right\}_m = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{1m} \\ \left\{\widetilde{\mathbf{U}}_{2i}\right\}_{lm} = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., 2i;$$

$$\widetilde{\lambda}_{2i} = \frac{\widetilde{\delta}_1}{1 + \left\langle \widetilde{\mathbf{d}}_{2i}, \widetilde{\mathbf{p}} \right\rangle};$$

$\widetilde{\mathbf{u}}_{2i} = \widetilde{\lambda}_{2i}\, \widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_i = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{Y}}_{2i} \widetilde{\mathbf{u}}_{2i}$ con $\widetilde{\mathbf{Y}}_{2i} = [\widetilde{\mathbf{y}}_1, \widetilde{\mathbf{y}}_2, ..., \widetilde{\mathbf{y}}_{2i}]$; entonces,
$\mathbf{M}\mathbf{x}_i = \mathbf{M}\mathbf{x}_0 + \widetilde{\mathbf{Y}}_{2i} \widetilde{\mathbf{u}}_{2i}$; que multiplicando por $\mathbf{M}^{-1}$ resulta,
$\mathbf{x}_i = \mathbf{x}_0 + \mathbf{M}^{-1} \widetilde{\mathbf{Y}}_{2i} \widetilde{\mathbf{u}}_{2i}$; que llamando $\widetilde{\mathbf{Y}}_{2i} \widetilde{\mathbf{u}}_{2i} = \mathbf{c}_{2i}$,
y haciendo $\mathbf{M}\mathbf{j}_{2i} = \mathbf{c}_{2i}$ resulta $\mathbf{x}_i = \mathbf{x}_0 + \mathbf{j}_{2i}$;

$$\begin{cases} \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_1 = \widetilde{\lambda}_{2i} \\ \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_{l+1} = -\widetilde{\lambda}_{2i} \left\{\bar{\mathbf{p}}\right\}_l \end{cases} \quad l = 1, ..., 2i;$$

ALGORITMO QMRCGSTAB MODIFICADO PRECONDICIONADO POR LA DERECHA

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\mathbf{r}_0^*$ es arbitrario, tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
$\rho_0 = \alpha_0 = \omega_0 = 1$;
$\mathbf{g}_0 = \mathbf{v}_0 = \mathbf{0}$;
Mientras $\sqrt{2i+1} \parallel \widehat{\mathbf{r}}_{i-1} \parallel / \parallel \mathbf{r}_0 \parallel \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer

$\rho_i = \langle \mathbf{r}_{i-1}, \mathbf{r}_0^* \rangle$;
$\beta_i = (\rho_i / \rho_{i-1})(\alpha_{i-1} / \omega_{i-1})$;
$\mathbf{g}_i = \mathbf{r}_{i-1} + \beta_i (\mathbf{g}_{i-1} - \omega_{i-1} \mathbf{v}_{i-1})$;
Resolver $\mathbf{M}\mathbf{f}_i = \mathbf{g}_i$
$\mathbf{v}_i = \mathbf{A}\mathbf{f}_i$;

$$\alpha_i = \frac{\rho_i}{\langle \mathbf{v}_i, \mathbf{r}_0^* \rangle};$$

$\mathbf{s}_i = \mathbf{r}_{i-1} - \alpha_i \mathbf{v}_i$;
$\delta_{2i-1} = \|\mathbf{s}_i\|$; $\mathbf{y}_{2i-1} = \mathbf{g}_i$;
Resolver $\mathbf{M}\mathbf{h}_i = \mathbf{s}_i$;
$\mathbf{t}_i = \mathbf{A}\mathbf{h}_i$;

$$\omega_i = \frac{\langle \mathbf{t}_i, \mathbf{s}_i \rangle}{\langle \mathbf{t}_i, \mathbf{t}_i \rangle};$$

$\mathbf{r}_i = \mathbf{s}_i - \omega_i \mathbf{t}_i$;
$\delta_{2i} = \|\mathbf{r}_i\|$; $\mathbf{y}_{2i} = \mathbf{s}_i$;
Resolver $\mathbf{U}_{2i}^t \bar{\mathbf{p}} = \mathbf{d}_{2i}$ y $\mathbf{U}_{2i} \mathbf{p} = \bar{\mathbf{p}}$;

$$\text{donde} \begin{cases} \left\{\mathbf{d}_{2i}\right\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \left\{\mathbf{U}_{2i}\right\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., 2i;$$

$\lambda_{2i} = \dfrac{\delta_1}{1 + \langle \mathbf{d}_{2i},\, \mathbf{p}\rangle}$;
$\mathbf{u}_{2i} = \lambda_{2i}\, \mathbf{p}$;
$\mathbf{c}_{2i} = \mathbf{Y}_{2i}\mathbf{u}_{2i}$, con $\mathbf{Y}_{2i} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{2i}]$;
Resolver $\mathbf{M}\mathbf{j}_{2i} = \mathbf{c}_{2i}$;
$\mathbf{x}_i = \mathbf{x}_0 + \mathbf{j}_{2i}$;
$\begin{cases} \{\widehat{\mathbf{r}}_i\}_1 = \lambda_{2i} \\ \{\widehat{\mathbf{r}}_i\}_{l+1} = -\lambda_{2i}\, \{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., 2i$;
Fin

9.3.3. Precondicionamiento por ambos lados

Aplicando el algoritmo QMRCGSTAB-Modificado al sistema precondicionado por ambos lados y estableciendo las nuevas expresiones para los distintos parámetros y vectores:
$\widetilde{\rho}_0 = \widetilde{\alpha}_0 = \widetilde{\omega}_0 = 1$;
$\widetilde{\mathbf{g}}_0 = \widetilde{\mathbf{v}}_0 = \mathbf{0}$, que multiplicando por $\mathbf{M}_1$ resulta,
$\mathbf{g}_0 = \mathbf{v}_0 = \mathbf{0}$;
$\widetilde{\rho}_i = \langle \widetilde{\mathbf{r}}_{i-1}, \widetilde{\mathbf{r}}_0^* \rangle = \left\langle \mathbf{M}_1^{-1}\mathbf{r}_{i-1}, \widetilde{\mathbf{r}}_0^* \right\rangle = \mathbf{r}_{i-1}^T \mathbf{M}_1^{-T} \widetilde{\mathbf{r}}_0^* = \mathbf{r}_{i-1}^T \mathbf{M}_1^{-T}\mathbf{M}_2^{-T}\mathbf{r}_0^* =$
$= \mathbf{r}_{i-1}^T \mathbf{M}^{-T}\mathbf{r}_0^*$, entonces,
$\widetilde{\rho}_i = \langle \mathbf{M}^{-1}\mathbf{r}_{i-1}, \mathbf{r}_0^* \rangle$, con $\widetilde{\mathbf{r}}_0^* = \mathbf{M}_2^{-T}\mathbf{r}_0^*$;
Haciendo $\mathbf{M}\mathbf{z}_{i-1} = \mathbf{r}_{i-1}$ resulta,
$\widetilde{\rho}_i = \langle \mathbf{z}_{i-1}, \mathbf{r}_0^* \rangle$;
$\widetilde{\beta}_i = (\widetilde{\rho}_i / \widetilde{\rho}_{i-1})(\widetilde{\alpha}_{i-1} / \widetilde{\omega}_{i-1})$;
$\widetilde{\mathbf{g}}_i = \widetilde{\mathbf{r}}_{i-1} + \widetilde{\beta}_i(\widetilde{\mathbf{g}}_{i-1} - \widetilde{\omega}_{i-1}\widetilde{\mathbf{v}}_{i-1})$ que multiplicando por $\mathbf{M}_1$ resulta,
$\mathbf{g}_i = \mathbf{r}_{i-1} + \widetilde{\beta}_i(\mathbf{g}_{i-1} - \widetilde{\omega}_{i-1}\mathbf{v}_{i-1})$
$\widetilde{\mathbf{v}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{g}}_i = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\mathbf{g}_i$ que multiplicando por $\mathbf{M}_1$ resulta,
$\mathbf{v}_i = \mathbf{A}\mathbf{M}^{-1}\mathbf{g}_i$ y haciendo $\mathbf{M}\mathbf{f}_i = \mathbf{g}_i$ resulta, $\mathbf{v}_i = \mathbf{A}\mathbf{f}_i$;
$\widetilde{\alpha}_i = \dfrac{\widetilde{\rho}_i}{\langle \widetilde{\mathbf{v}}_i, \mathbf{r}_0^* \rangle} = \dfrac{\widetilde{\rho}_i}{\left\langle \mathbf{M}_1^{-1}\mathbf{v}_i, \mathbf{r}_0^* \right\rangle} = \dfrac{\widetilde{\rho}_i}{\langle \mathbf{M}^{-1}\mathbf{v}_i, \mathbf{r}_0^* \rangle}$, y haciendo,
$\mathbf{M}\mathbf{q}_i = \mathbf{v}_i$ resulta, $\widetilde{\alpha}_i = \dfrac{\widetilde{\rho}_i}{\langle \mathbf{q}_i, \mathbf{r}_0^* \rangle}$;
$\widetilde{\mathbf{s}}_i = \widetilde{\mathbf{r}}_{i-1} - \widetilde{\alpha}_i\widetilde{\mathbf{v}}_i$ que multiplicando por $\mathbf{M}_1$ resulta, $\mathbf{s}_i = \mathbf{r}_{i-1} - \widetilde{\alpha}_i\mathbf{v}_i$;
$\widetilde{\delta}_{2i-1} = \|\widetilde{\mathbf{s}}_i\| = \|\mathbf{M}_1^{-1}\mathbf{s}_i\|$ y haciendo $\mathbf{M}_1\mathbf{s}_i^* = \mathbf{s}_i$ resulta $\widetilde{\delta}_{2i-1} = \|\mathbf{s}_i^*\|$;
$\widetilde{\mathbf{y}}_{2i-1} = \widetilde{\mathbf{g}}_i$ que multiplicando por $\mathbf{M}_1$ resulta, $\mathbf{y}_{2i-1} = \mathbf{g}_i$;
$\widetilde{\mathbf{t}}_i = \widetilde{\mathbf{A}}\widetilde{\mathbf{s}}_i = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{M}_1^{-1}\mathbf{s}_i$ multiplicando por $\mathbf{M}_1$ resulta,
$\mathbf{t}_i = \mathbf{A}\mathbf{M}^{-1}\mathbf{s}_i$; haciendo $\mathbf{M}\mathbf{h}_i = \mathbf{s}_i$ resulta, $\mathbf{t}_i = \mathbf{A}\mathbf{h}_i$;
$\widetilde{\omega}_i = \dfrac{\left\langle \widetilde{\mathbf{t}}_i, \widetilde{\mathbf{s}}_i \right\rangle}{\left\langle \widetilde{\mathbf{t}}_i, \widetilde{\mathbf{t}}_i \right\rangle} = \dfrac{\left\langle \mathbf{M}_1^{-1}\mathbf{t}_i, \mathbf{M}_1^{-1}\mathbf{s}_i \right\rangle}{\left\langle \mathbf{M}_1^{-1}\mathbf{t}_i, \mathbf{M}_1^{-1}\mathbf{t}_i \right\rangle}$; haciendo $\mathbf{M}_1\mathbf{t}_i^* = \mathbf{t}_i$ resulta,
$\widetilde{\omega}_i = \dfrac{\langle \mathbf{t}_i^*, \mathbf{s}_i^* \rangle}{\langle \mathbf{t}_i^*, \mathbf{t}_i^* \rangle}$;
$\widetilde{\mathbf{r}}_i = \widetilde{\mathbf{s}}_i - \widetilde{\omega}_i\widetilde{\mathbf{t}}_i$ que multiplicando por $\mathbf{M}_1$ resulta, $\mathbf{r}_i = \mathbf{s}_i - \widetilde{\omega}_i\mathbf{t}_i$;

$\widetilde{\delta}_{2i} = \|\widetilde{\mathbf{r}}_i\| = \|\mathbf{M}_1^{-1}\mathbf{s}_i\|$ y haciendo $\mathbf{M}_1\mathbf{z}_i^* = \mathbf{z}_i$ resulta,
$\widetilde{\delta}_{2i} == \|\mathbf{z}_i^*\|$;
$\widetilde{\mathbf{y}}_{2i} = \widetilde{\mathbf{s}}_i$ que multiplicando por $\mathbf{M}_1$ resulta, $\mathbf{y}_{2i} = \mathbf{s}_i$;
Resolver $\widetilde{\mathbf{U}}_{2i}^t\widetilde{\overline{\mathbf{p}}}=\widetilde{\mathbf{d}}_{2i}$ y $\widetilde{\mathbf{U}}_{2i}\widetilde{\mathbf{p}}=\widetilde{\overline{\mathbf{p}}}$;

$$\text{donde}\begin{cases} \left\{\widetilde{\mathbf{d}}_{2i}\right\}_m = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{1m} \\ \left\{\widetilde{\mathbf{U}}_{2i}\right\}_{lm} = \left\{\widetilde{\overline{\mathbf{T}}}\right\}_{l+1m} \end{cases} \qquad l, m = 1, ..., 2i;$$

$$\widetilde{\lambda}_{2i} = \frac{\widetilde{\delta}_1}{1 + \left\langle \widetilde{\mathbf{d}}_{2i}, \widetilde{\mathbf{p}} \right\rangle};$$

$\widetilde{\mathbf{u}}_{2i} = \widetilde{\lambda}_{2i}\,\widetilde{\mathbf{p}}$;
$\widetilde{\mathbf{x}}_i = \widetilde{\mathbf{x}}_0 + \widetilde{\mathbf{Y}}_{2i}\widetilde{\mathbf{u}}_{2i}$ con $\widetilde{\mathbf{Y}}_{2i} = [\widetilde{\mathbf{y}}_1, \widetilde{\mathbf{y}}_2, ..., \widetilde{\mathbf{y}}_{2i}]$; entonces,
$\mathbf{M}_2\mathbf{x}_i = \mathbf{M}_2\mathbf{x}_0 + \mathbf{M}_1^{-1}\mathbf{Y}_{2i}\widetilde{\mathbf{u}}_{2i}$, con $\mathbf{Y}_{2i} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{2i}]$;
Multiplicando por $\mathbf{M}_2^{-1}$ resulta,
$\mathbf{x}_i = \mathbf{x}_0 + \mathbf{M}^{-1}\mathbf{Y}_{2i}\mathbf{u}_{2i}$; llamando $\mathbf{Y}_{2i}\mathbf{u}_{2i} = \mathbf{c}_{2i}$,
y haciendo $\mathbf{M}\mathbf{j}_{2i} = \mathbf{c}_{2i}$ resulta, $\mathbf{x}_i = \mathbf{x}_0 + \mathbf{j}_{2i}$;

$$\begin{cases} \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_1 = \widetilde{\lambda}_{2i} \\ \left\{\widetilde{\widehat{\mathbf{r}}}_i\right\}_{l+1} = -\widetilde{\lambda}_{2i}\left\{\bar{\mathbf{p}}\right\}_l \end{cases} \qquad l = 1, ..., 2i;$$

ALGORITMO QMRCGSTAB MODIFICADO PRECONDICIONADO POR AMBOS LADOS

Aproximación inicial $\mathbf{x}_0$. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
$\mathbf{r}_0^*$ es arbitrario, tal que $\langle \mathbf{r}_0, \mathbf{r}_0^* \rangle \neq 0$;
$\rho_0 = \alpha_0 = \omega_0 = 1$;
$\mathbf{g}_0 = \mathbf{v}_0 = \mathbf{0}$;
Resolver $\mathbf{M}\mathbf{z}_0 = \mathbf{r}_0$
Mientras $\sqrt{2i+1}\,\|\,\widehat{\mathbf{r}}_{i-1}\,\|\,/\,\|\,\mathbf{r}_0\,\| \geq \varepsilon \quad (i = 1, 2, 3, ...)$, hacer
$\rho_i = \langle \mathbf{z}_{i-1}, \mathbf{r}_0^* \rangle$;
$\beta_i = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$;
$\mathbf{g}_i = \mathbf{r}_{i-1} + \beta_i(\mathbf{g}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1})$;
Resolver $\mathbf{M}\mathbf{f}_i = \mathbf{g}_i$;
$\mathbf{v}_i = \mathbf{A}\mathbf{f}_i$;
Resolver $\mathbf{M}\mathbf{q}_i = \mathbf{v}_i$;
$\alpha_i = \dfrac{\rho_i}{\langle \mathbf{q}_i, \mathbf{r}_0^* \rangle}$;
$\mathbf{s}_i = \mathbf{r}_{i-1} - \alpha_i\mathbf{v}_i$;
Resolver $\mathbf{M}_1\mathbf{s}_i^* = \mathbf{s}_i$;
$\delta_{2i-1} = \|\mathbf{s}_i^*\|$; $\mathbf{y}_{2i-1} = \mathbf{g}_i$;
Resolver $\mathbf{M}\mathbf{h}_i = \mathbf{s}_i$;
$\mathbf{t}_i = \mathbf{A}\mathbf{h}_i$;
Resolver $\mathbf{M}_1\mathbf{t}_i^* = \mathbf{t}_i$
$\omega_i = \dfrac{\langle \mathbf{t}_i^*, \mathbf{s}_i^* \rangle}{\langle \mathbf{t}_i^*, \mathbf{t}_i^* \rangle}$;

$\mathbf{r}_i = \mathbf{s}_i - \omega_i \mathbf{t}_i$;
Resolver $\mathbf{M}_1 \mathbf{z}_i^* = \mathbf{r}_i$;
$\delta_{2i} = \|\mathbf{z}_i^*\|$; $\mathbf{y}_{2i} = \mathbf{s}_i$;
Resolver $\mathbf{U}_{2i}^t \bar{\mathbf{p}} = \mathbf{d}_{2i}$ y $\mathbf{U}_{2i}\mathbf{p} = \bar{\mathbf{p}}$;

$$\text{donde} \begin{cases} \{\mathbf{d}_{2i}\}_m = \left\{\overline{\mathbf{T}}\right\}_{1m} \\ \{\mathbf{U}_{2i}\}_{lm} = \left\{\overline{\mathbf{T}}\right\}_{l+1m} \end{cases} \quad l, m = 1, ..., 2i;$$

$$\lambda_{2i} = \frac{\delta_1}{1 + \langle \mathbf{d}_{2i},\, \mathbf{p} \rangle};$$

$\mathbf{u}_{2i} = \lambda_{2i}\, \mathbf{p}$;
$\mathbf{c}_{2i} = \mathbf{Y}_{2i}\mathbf{u}_{2i}$, con $\mathbf{Y}_{2i} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{2i}]$;
Resolver $\mathbf{M}\mathbf{j}_{2i} = \mathbf{c}_{2i}$;
$\mathbf{x}_i = \mathbf{x}_0 + \mathbf{j}_{2i}$;

$$\begin{cases} \{\widehat{\mathbf{r}}_i\}_1 = \lambda_{2i} \\ \{\widehat{\mathbf{r}}_i\}_{l+1} = -\lambda_{2i} \{\bar{\mathbf{p}}\}_l \end{cases} \quad l = 1, ..., 2i;$$

Fin

El algoritmo correspondiente a la forma de precondicionamiento por la izquierda, requiere un producto añadido matriz inversa por vector en la inicialización y dos más por iteración; en el precondicionamiento por la derecha se requieren, tres productos añadidos matriz inversa por vector por cada iteración y en el precondicionamiento por ambos lados se realiza un producto añadido matriz inversa por vector en la inicialización y siete por iteración.

Capítulo 10

Experimentos numéricos

Este capítulo está dedicado a la aplicación de algunos de los algoritmos y técnicas de precondicionamiento, reordenación y almacenamiento desarrollados anteriormente. Los problemas escogidos para estos experimentos corresponden a:

- Matrices test, extraídas de *User's Guide for the Harwell-Boeing Sparse Matrix Collection (Release I)* [12] (Ejemplos 1, 2, 6, 7 y 8).

- Simulación de un filtro de carbón activo desarrollado por el grupo LaCàN de la U.P.C. [15] (Ejemplo 5).

- Distintos problemas de ingeniería previamente analizados por Montero y otros [53] (Ejemplos 3, 4, 9, 10 y 11).

Esta elección obedece al interés planteado en esta tesis en la resolución de sistemas de ecuaciones lineales con matriz tipo *sparse*. Concretamente, hemos prestado especial atención a sistemas resultantes de la discretización de problemas de convección-difusión, en 2-D y 3-D. Asimismo, se consideró interesante el análisis de diferentes sistemas correspondientes a un mismo problema, que surgen bien de un proceso evolutivo, bien debido a un proceso de refinamiento de la malla. Todo ello convenía ser estudiado tanto en algunos casos extraídos de la colección de sistemas elaborada por el grupo de investigación *Álgebra Numérica Avanzada* de la U.L.P.G.C, como en otros sistemas de colecciones reconocidas internacionalmente y utilizadas como test por muchos investigadores. Por otro lado, se ha incluído algún ejemplo de sistemas derivados de problemas estudiados en el proyecto de investigación en que está inmerso esta tesis.

El objetivo que se persigue en este capítulo es comprobar la efectividad y realizar un estudio comparativo de los algoritmos y técnicas utilizadas en cada problema. En todos los casos, las matrices se almacenan en la forma compacta descrita en el capítulo 4 y las operaciones matriciales que figuran en los algoritmos se han implementado en doble precisión y aprovechando este tipo de almacenamiento.

Además se presentan, para cada uno de los ejemplos estudiados el perfil de la matriz del sistema y en algunos de los casos los perfiles correspondiente a las diferentes reordenaciones.

En las diferentes resoluciones, realizadas con un XEON Precision 530, se ha fijado un valor de tolerancia igual a 10^{-10} en todos los casos, que en ausencia de "reinicialización" por diferencia entre el residuo calculado por la relación de recurrencia, o estimado, de cada algoritmo y el residuo exacto, no afecta a las curvas de convergencia, aunque sí tiene relación directa con el número de iteraciones o el tiempo de CPU necesarios para alcanzarla.

Los ejemplos 1 a 5 corresponden a sistemas simétricos que se han resuelto con el método del Gradiente Conjugado (CG), utilizando diferentes precondicionadores y, en determinados casos, aplicando además alguna técnica de reordenación. Los resultados son expuestos mediante curvas de convergencia, en las que se representa el número de iteraciones y/o tiempo de CPU necesarios para alcanzar la tolerancia predeterminada.

Los sistemas no simétricos se estudian en los ejemplos 6 a 11, exponiéndose los resultados, igual que en los casos simétricos, mediante las correspondientes curvas de convergencia y recurriendo a tablas que reflejan el número de iteraciones y tiempo de CPU necesarios para alcanzar la tolerancia exigida con los diferentes métodos de resolución estudiados, con y sin precondicionamiento, descartando aquellos casos en que la convergencia es muy lenta y/o la tolerancia deseada no se alcanza o lo hace para un número de iteraciones mayor que el número de ecuaciones del sistema analizado, o bien cuando el tiempo de cálculo es muy elevado respecto a otros métodos o técnicas. Para aquellos ejemplos en que se ha utilizado alguna técnica de reordenación, se presenta en la misma tabla el número de iteraciones y el tiempo de CPU.

Las curvas de convergencia resultantes en de los diferentes problemas dan información sobre la evolución más o menos suave de la norma residual con el número de iteraciones y/o el tiempo de computación necesario para alcanzar la tolerancia exigida. El vector inicial en todos los algoritmos es el vector residuo del sistema original sin precondicionar. El valor del parámetro que define al precondicionador SSOR utilizado en todos los casos es $w = 1$. Las gráficas además van a permitir:

- Establecer comparaciones en la eficacia de los algoritmos con o sin precondicionamiento.
- Contrastar los distintos precondicionadores.
- Comprobar el efecto que producen las técnicas de reordenación.

10.1. Ejemplo 1 (SAYLOR)

Estas matrices fueron proporcionadas por P. Saylor para ser usadas como test en su trabajo sobre la iteración de Richardson con parámetros dinámicos [74]. Las matrices surgen de una simulación en $3D$ de reservas petrolíferas con una malla de $n_x \times n_y \times n_z$ nodos, tal que la pizarra produce barreras verticales en el fluido y crea una heterogeneidad casi aleatoria en la matriz de coeficientes. Además, existen contrastes locales enormes en los coeficientes de trasmisión de la ecuación diferencial. Estas propiedades dan lugar a sistemas de difícil solución. De estas matrices, hemos seleccionado los casos SAYLOR3 y SAYLOR4.

La simulación SAYLOR3 ha dado lugar a un sistema simétrico de 1000 $(10 \times 10 \times 10)$ ecuaciones con 3750 entradas no nulas y la SAYLOR4 produce otro sistema simétrico de 3540 $(33 \times 6 \times 18)$ ecuaciones con 22316 entradas no nulas. Las figuras 10.1 y 10.3 muestran las estructuras de las matrices de estos dos sistemas, respectivamente.

Las figuras 10.2 y 10.4 representan las curvas de convergencia relativas al número de iteraciones de los problemas SAYLOR3 y SAYLOR4, respectivamente. En ambos casos la resolución con el algoritmo del Gradiente Conjugado precondicionado SSOR acelera considerablemente la convergencia. En este ejemplo, el precondicionador ILLT no es factible al no ser la matriz del sistema definida positiva. Tampoco convergen en ninguno de los dos casos cuando es precondicionada con ILU(0) (precondicionador no simétrico).

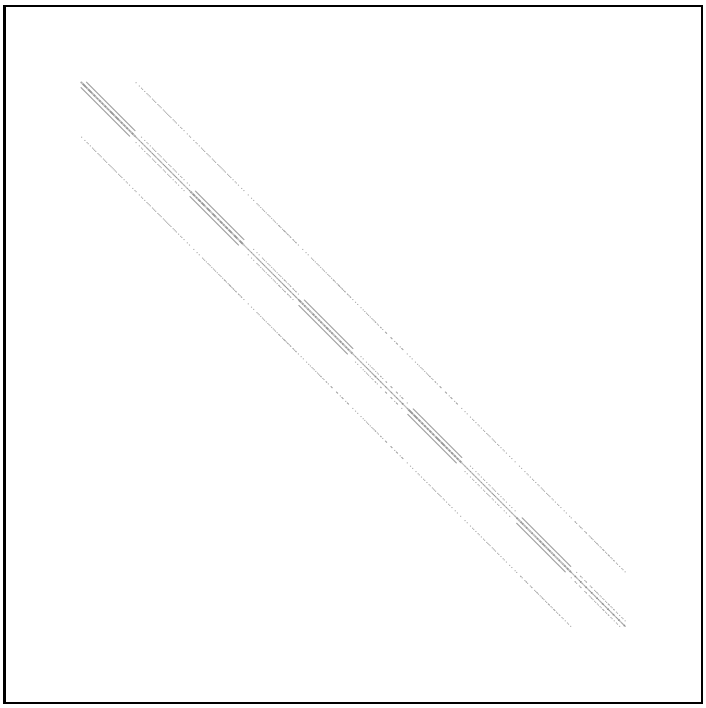

Figura 10.1: *Estructura sparse de la matriz SAYLOR3*

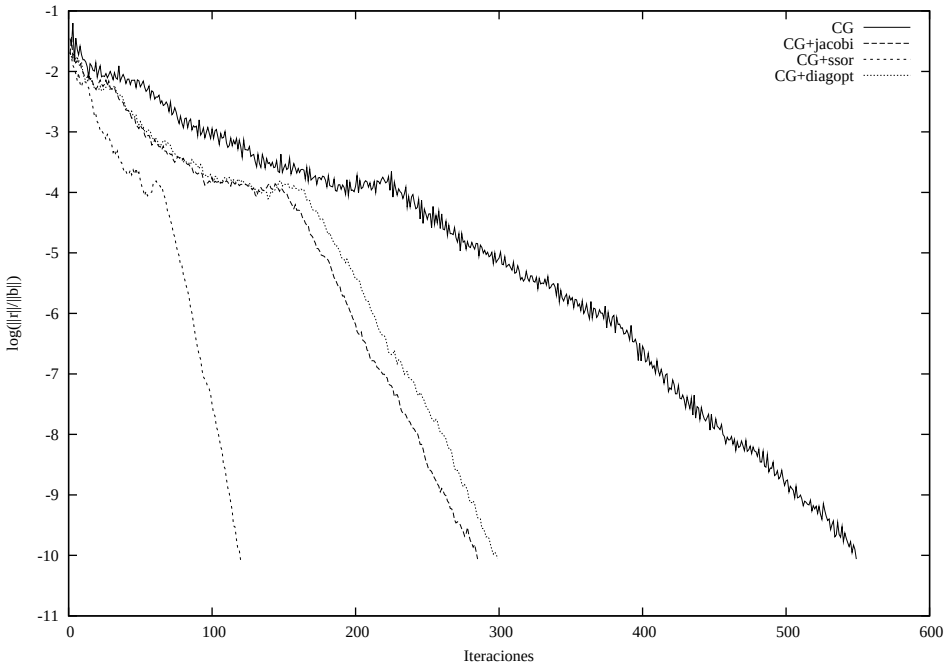

Figura 10.2: *Convergencia del CG con distintos precondicionadores para SAYLOR3*

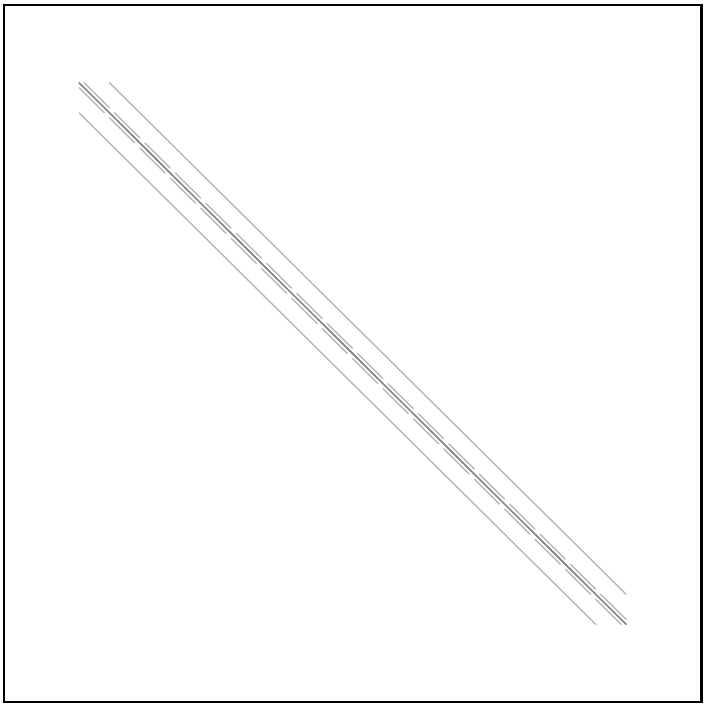

Figura 10.3: *Estructura sparse de la matriz SAYLOR4*

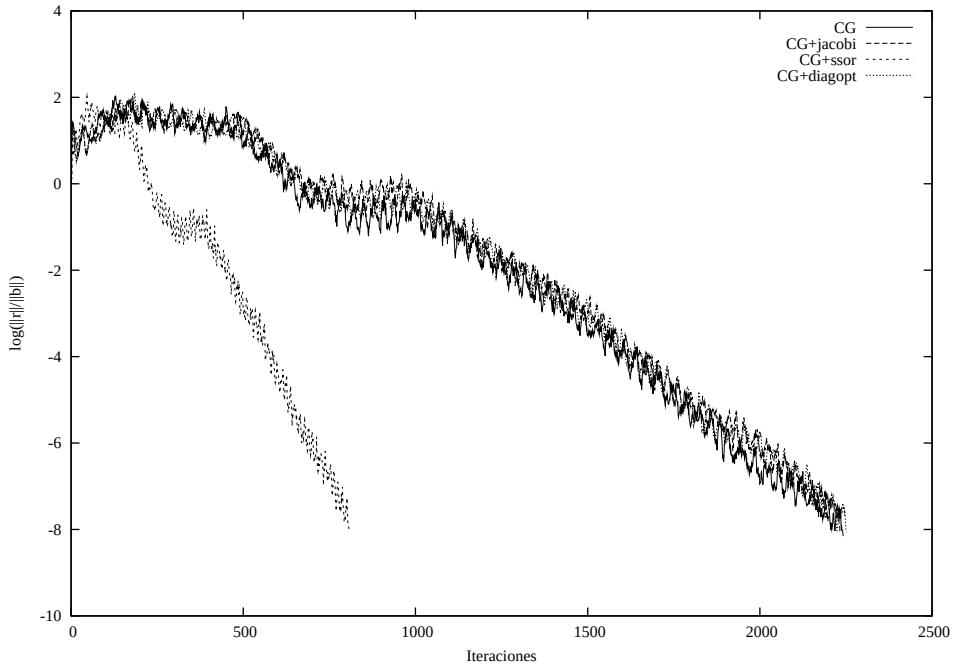

Figura 10.4: *Convergencia del CG con distintos precondicionadores para SAYLOR4*

10.2. Ejemplo 2 (SHERMAN)

En 1984, A. Sherman lanzó un reto a la industria del petróleo y a la comunidad de análisis numérico para encontrar la solución más rápida a un conjunto de cinco sistemas de ecuaciones lineales extraídos de programas de modelización de reservas petrolíferas. De estos , hemos elegido los cuatro primeros como ejemplos: SHERMAN1, SHERMAN2, SHERMAN3 y SHERMAN4. Cada matriz proviene de un modelo de simulación tridimensional con una malla $n_x \times n_y \times n_z$, usando un esquema en diferencias finitas de siete puntos y un número de ecuaciones e incognitas por nodo de la malla n_c. El correspondiente vector segundo miembro también es suministrado para cada caso.

En SHERMAN1 se usa una malla de $10 \times 10 \times 10$ nodos, siendo el número de ecuaciones e incógnitas por cada nodo de la malla $n_c = 1$. La simulación da lugar a un sistema simétrico de 1000 ecuaciones con 3750 entradas no nulas.

En SHERMAN2 se usa una malla de $6 \times 6 \times 5$ nodos, siendo el número de ecuaciones e incógnitas por cada nodo de la malla $n_c = 6$. La simulación da lugar a un sistema simétrico de 1080 ecuaciones con 23094 entradas no nulas.

En SHERMAN3 se usa una malla de $35 \times 11 \times 13$ nodos, siendo el número de ecuaciones e incógnitas por cada nodo de la malla $n_c = 1$. La simulación da lugar a un sistema simétrico de 5005 ecuaciones con 20033 entradas no nulas.

En SHERMAN4 se usa una malla de $16 \times 23 \times 3$ nodos, siendo el número de ecuaciones e incógnitas por cada nodo de la malla $n_c = 1$. La simulación da lugar a un sistema simétrico de 1104 ecuaciones con 3786 entradas no nulas.

Las figuras 10.5, 10.7, 10.10 y 10.12 muestran las estructuras de las matrices de estos cuatro sistemas, respectivamente.

Las curvas de convergencia correspondientes a los problemas SHERMAN1, SHERMAN3 y SHERMAN4 se presentan en las figuras 10.6, 10.11 y 10.13, respectivamente. En todos los casos se ha utilizado el algoritmo del Gradiente Conjugado sin precondicionar, además de aplicar los distintos precondicionadores analizados en este trabajo. El método converge en todos los casos, aunque el precondicionador ILU(0) lo hace más rápida y suavemente. El precondicionamiento con ILLT tampoco es factible en este problema.

El problema SHERMAN2, no converge cuando se aplica el método del Gradiente Conjugado, ni siquiera cuando el algoritmo es precondicionado, como puede verse en la figura 10.8. Este resultado parece indicar que no se trata de una matriz simétrica definida positiva. En la figura 10.9, se presentan las curvas de convergencia cuando se aplican otros métodos de Krylov precondicionados con ILU(0) con los que se ha resuelto este caso particular, siendo el algoritmo VGMRES el que proporciona la curva más suave, aunque con una carga más elevada en iteraciones y en tiempo de computación.

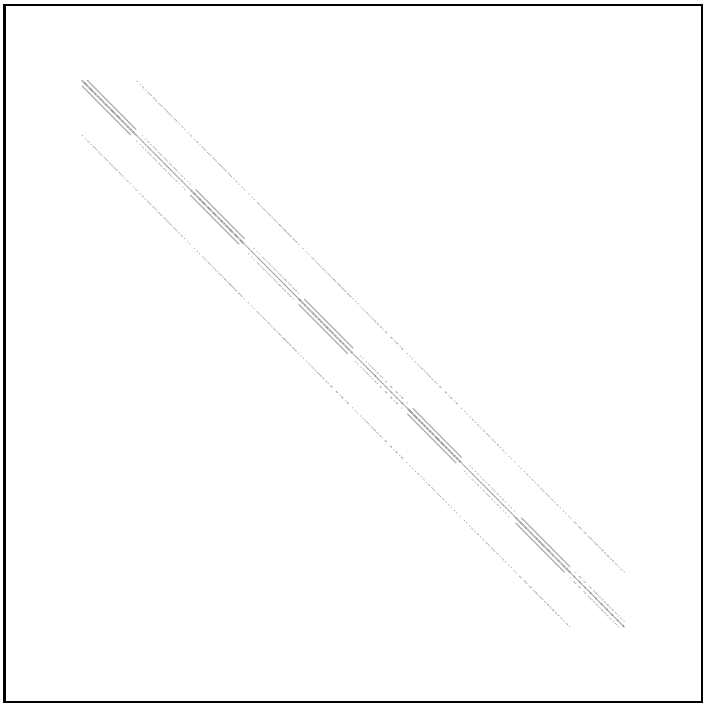

Figura 10.5: *Estructura sparse de la matriz SHERMAN1*

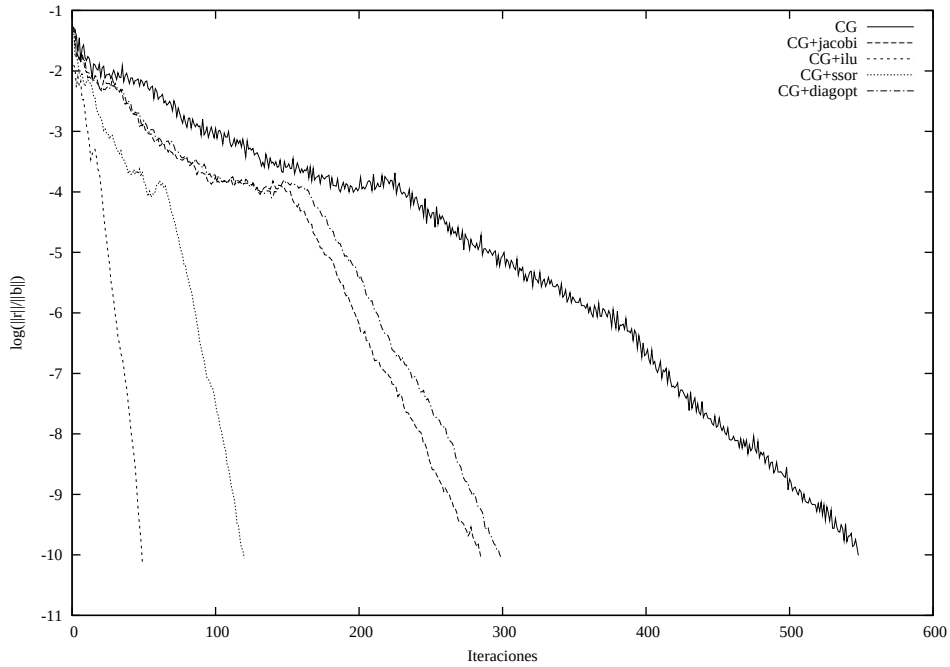

Figura 10.6: *Convergencia del CG con distintos precondicionadores para SHERMAN1*

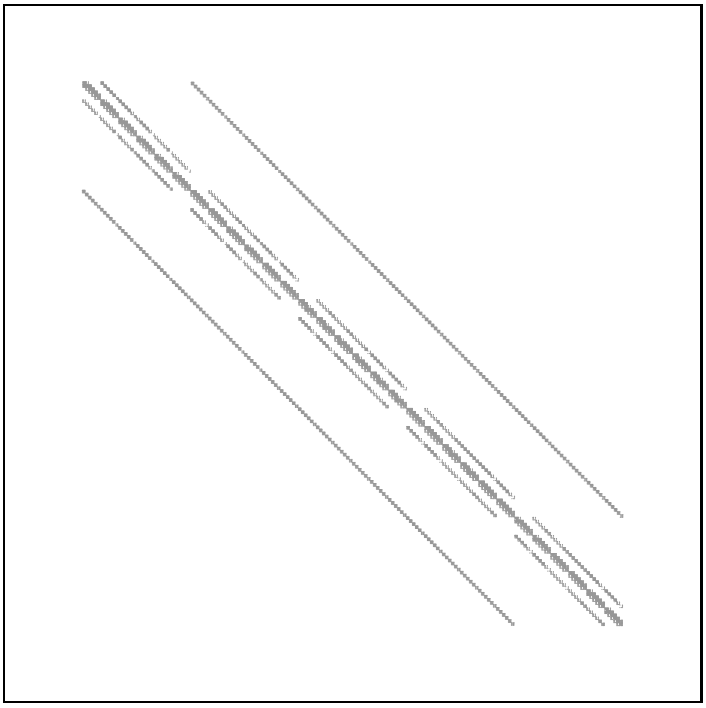

Figura 10.7: *Estructura sparse de la matriz SHERMAN2*

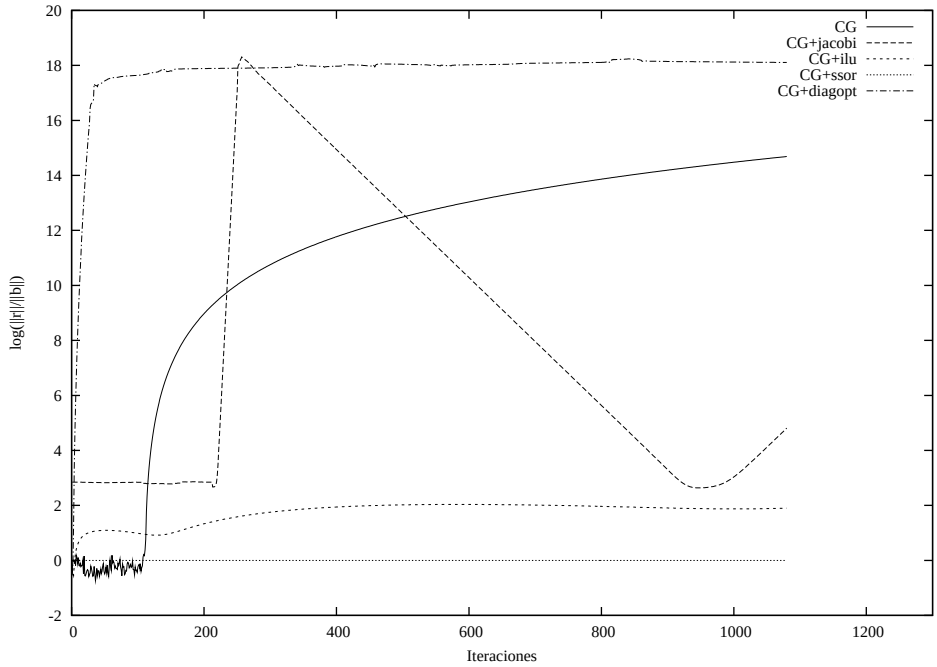

Figura 10.8: *Convergencia del CG con distintos precondicionadores para SHERMAN2*

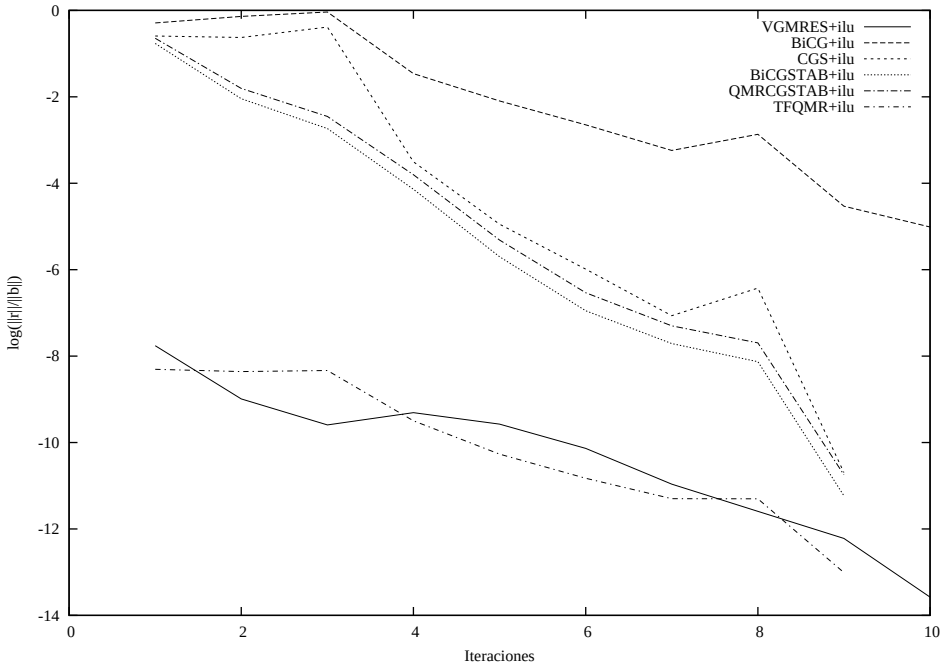

Figura 10.9: *Convergencia de distintos métodos de Krylov precondicionados con ILU(0) para SHERMAN2*

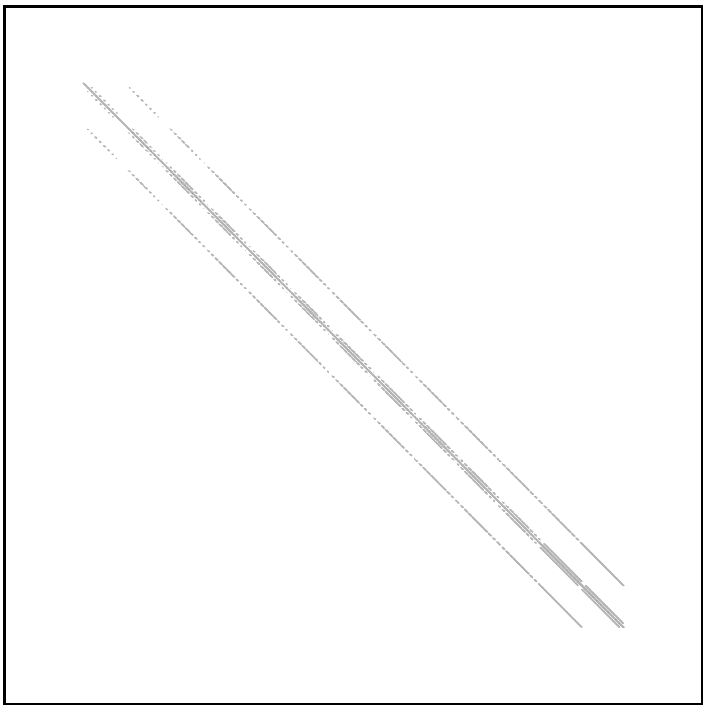

Figura 10.10: *Estructura sparse de la matriz SHERMAN3*

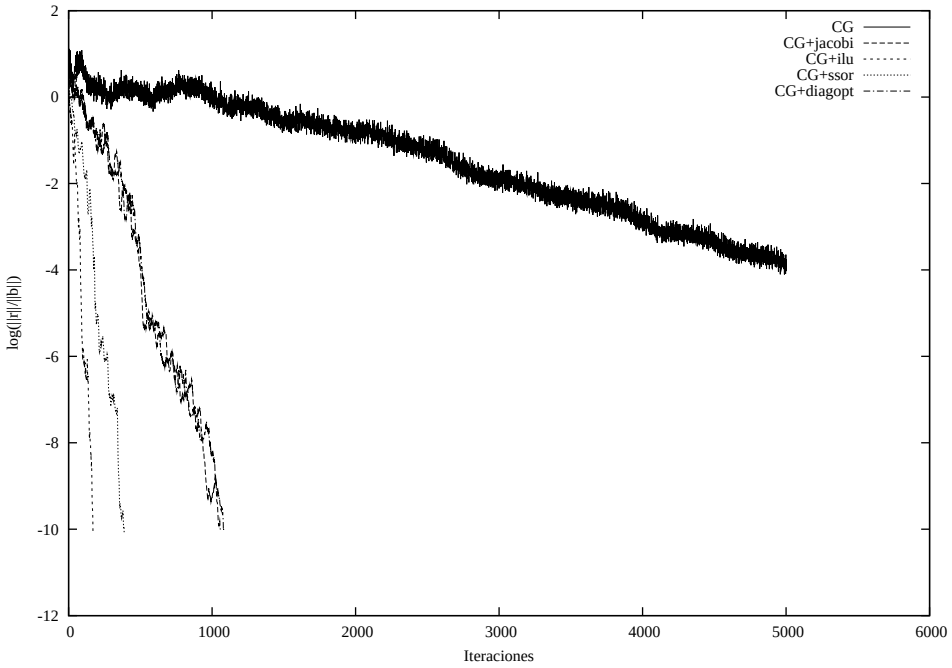

Figura 10.11: *Convergencia del CG con distintos precondicionadores para SHERMAN3*

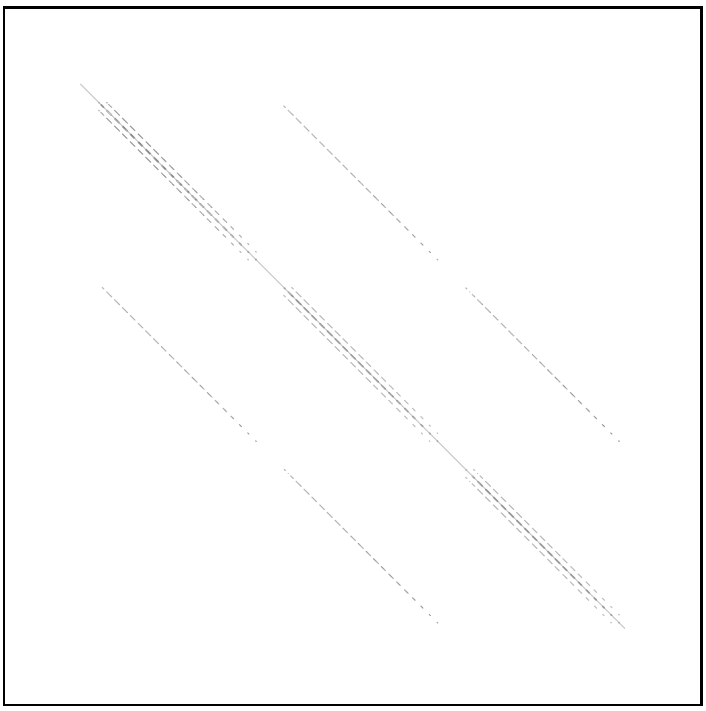

Figura 10.12: *Estructura sparse de la matriz SHERMAN4*

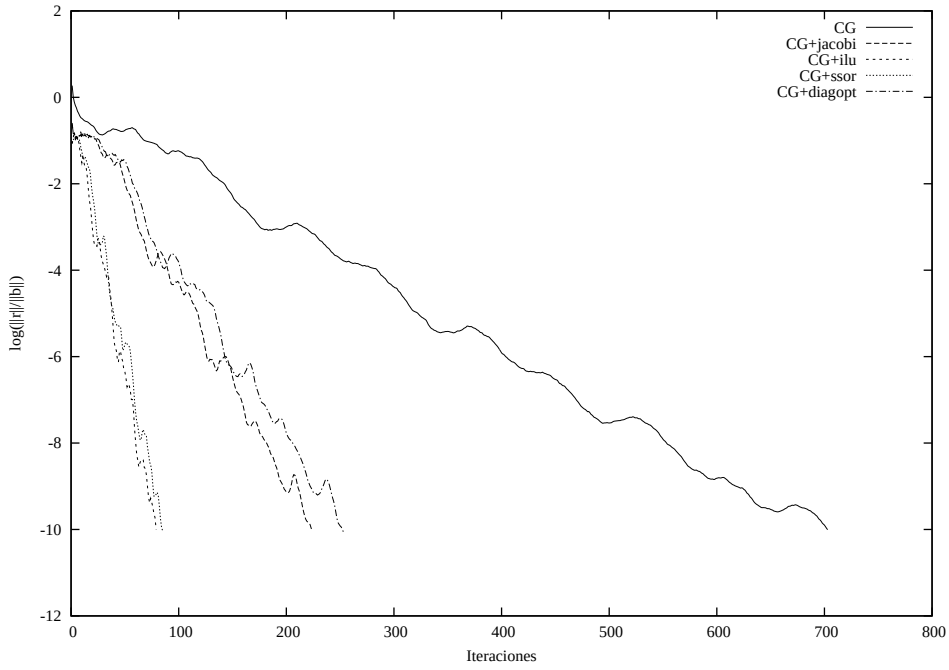

Figura 10.13: *Convergencia del CG con distintos precondicionadores para SHERMAN4*

10.3. Ejemplo 3 (calorlib)

Se trata de un problema de trasmisión de calor en un dominio cuadrado 1×1, dado por la ecuación $-\lambda \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = f$, con condiciones de contorno de Dirichlet $u = u_1$ en $y = 0$ y $x = 0$, $u = u_2$ en $x = 1$ y condición mixta en $y = 1$, $-\lambda \frac{\partial u}{\partial x} = H\,(u - u_\infty)$, siendo λ, la conductividad térmica dada en $\left[\frac{k_j}{h.m.^oC} \right]$, f, las fuentes volumétricas externas dadas en $\left[\frac{k_j}{h.m^3} \right]$ y H, el coeficiente de convección en $\left[\frac{k_j}{h.m^2.^oC} \right]$.

Para la aplicación práctica se ha tomado en las ecuaciones los valores $\lambda = 10^{-5}$, $f = 1$, $H = 20$, y como condiciones de temperatura $u_1 = 0$, $u_2 = 100$, $u_\infty = 1000$.

Se han considerado dos etapas de refinamiento que han dado lugar respectivamente, a sistemas simétricos de 4124 y 16412 ecuaciones. También se ha estudiado un caso diferente correspondiente a otra malla que ha originado un sistema con 16340 ecuaciones.

Las figuras 10.14-10.16 muestran las estructuras de las matrices del sistema de 4124 ecuaciones correspondientes a la ordenación inicial, reordenación con Grado Mínimo y Cuthill-McKee Inverso, respectivamente. De forma similar se ilustran las estructuras de las matrices del sistema de 16412 ecuaciones en las figuras 10.19-10.21 y del sistema de 16340 ecuaciones en las figuras 10.24-10.26.

Las figuras 10.17 y 10.18, correspondientes a la discretización que da lugar al sistema de 4124 ecuaciones, representan la convergencia respecto del número de iteraciones para la resolución con Gradiente Conjugado sin precondicionar y utilizando las distintas técnicas de precondicionamiento, y la convergencia respecto al tiempo de CPU cuando el sistema es renumerado utilizando los algoritmos del Grado Mínimo y Cuthill-McKee inverso. Observamos que la curva correspondiente al algoritmo CG sin precondicionar presenta muchas irregularidades y un número de iteraciones muy grande respecto a las curvas que reflejan la aplicación de algún precondicionador, siendo los precondicionadores ILLT y SSOR los más efectivos con curvas de convergencia muy suaves. Por otro lado, la convergencia cuando el sistema es reordenado resulta mejorada, tanto por reducir el número de iteraciones como el tiempo de computación (ver figura 10.18) y es, en este caso, la reordenación de Cuthill-McKee inverso la más efectiva.

Las discretizaciones que dan lugar a los sistemas de 16412 ecuaciones (figuras 10.22 y 10.23) y 16340 (figuras 10.28 y 10.28), presentan un comportamiento similar al anterior.

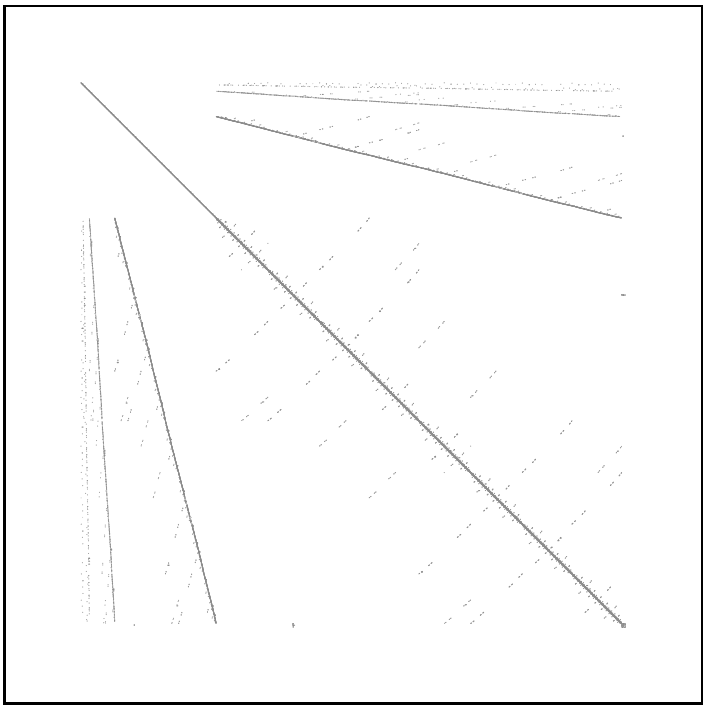

Figura 10.14: *Estructura sparse de la matriz calorlib para* $n = 4124$

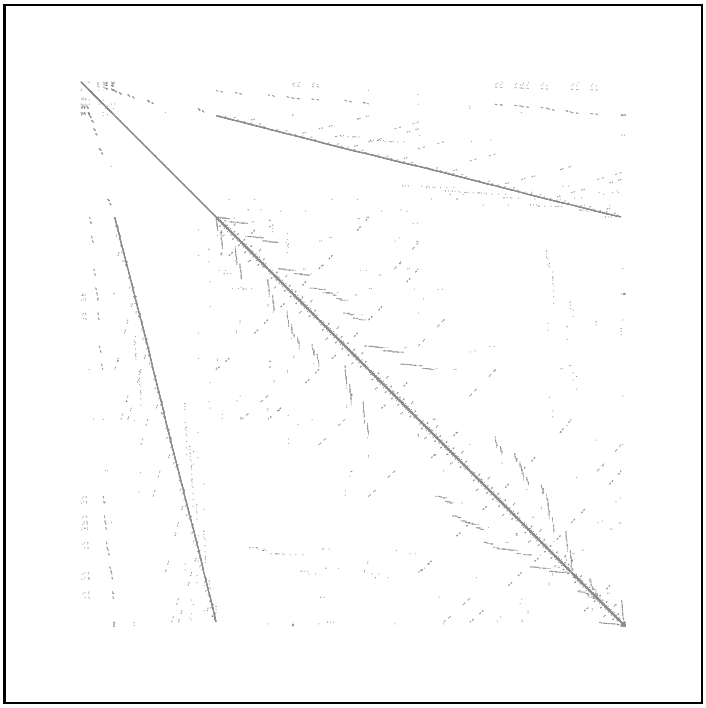

Figura 10.15: *Estructura sparse de la matriz calorlib para* $n = 4124$ *reordenada con Grado Mínimo*

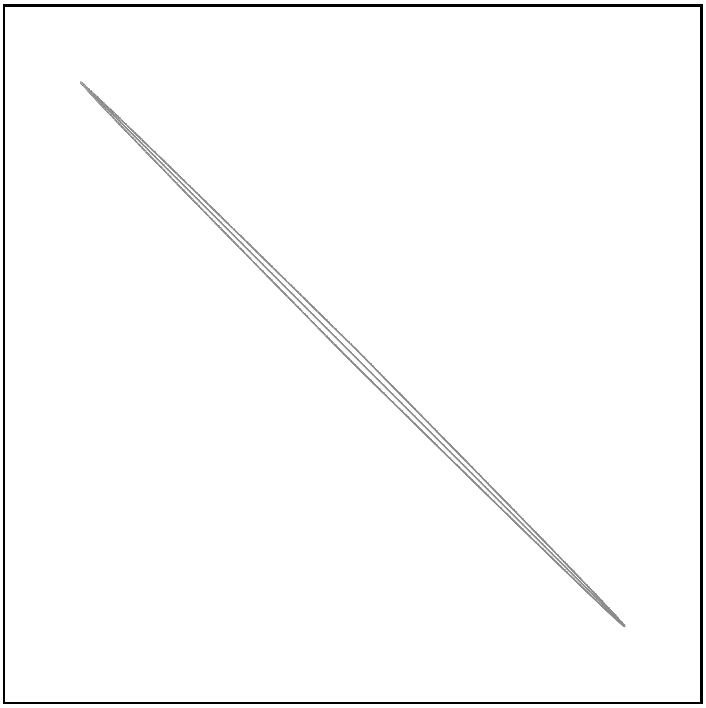

Figura 10.16: *Estructura sparse de la matriz calorlib para* $n = 4124$ *reordenada con Cuthill-McKee Inverso*

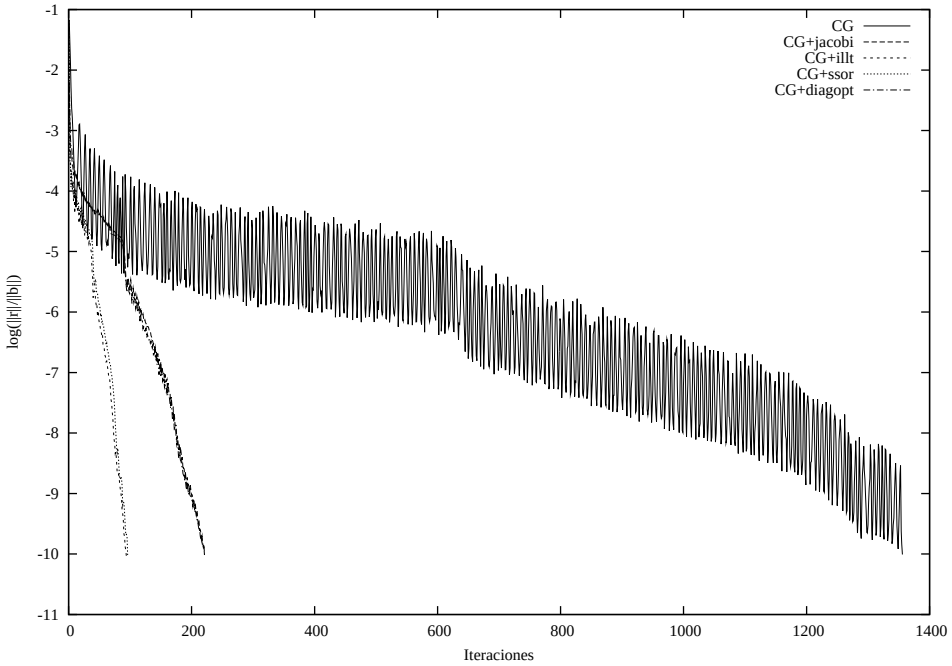

Figura 10.17: *Convergencia del CG con distintos precondicionadores para calorlib (4124 ecuaciones)*

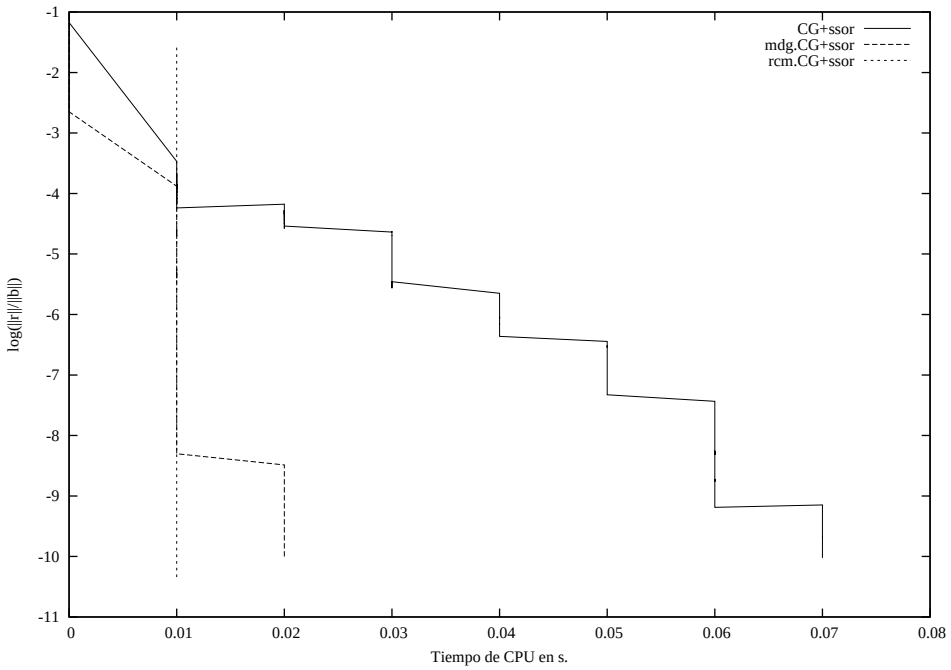

Figura 10.18: *Convergencia del CG+SSOR con diferentes reordenaciones para calorlib (4124 ecuaciones)*

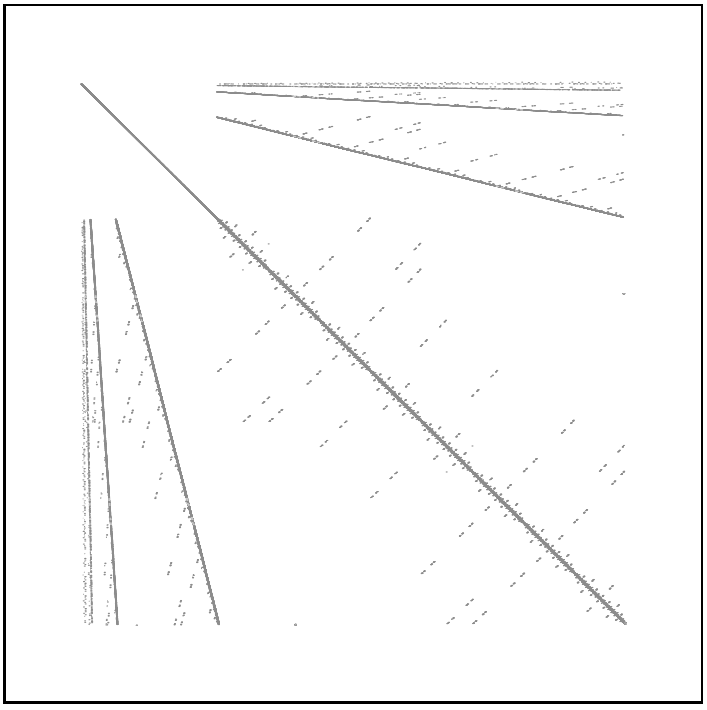

Figura 10.19: *Estructura sparse de la matriz calorlib para* $n = 16412$

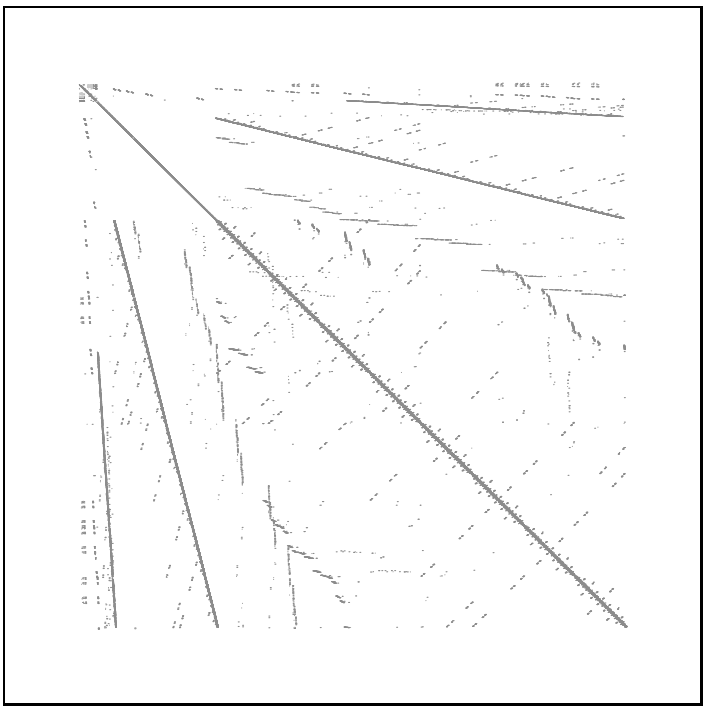

Figura 10.20: *Estructura sparse de la matriz calorlib para* $n = 16412$ *reordenada con Grado Mínimo*

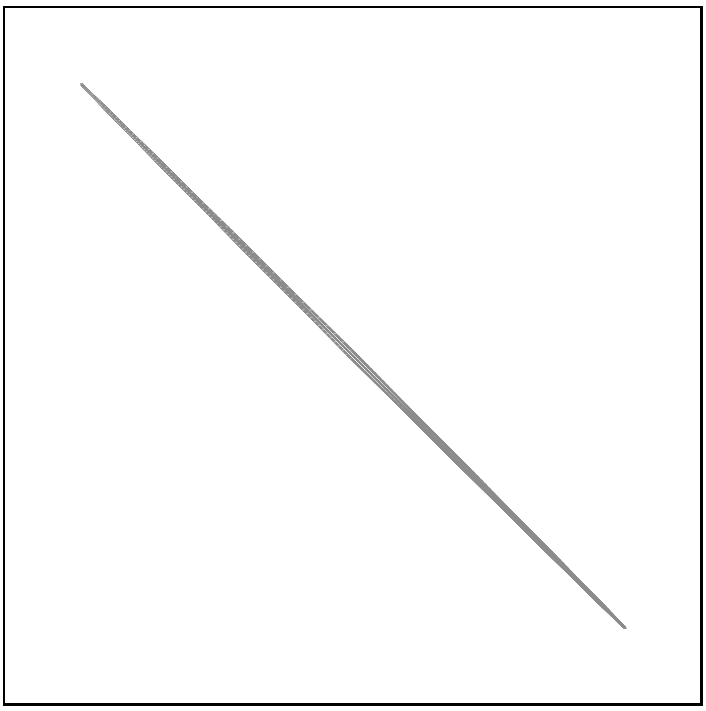

Figura 10.21: *Estructura sparse de la matriz calorlib para* $n = 16412$ *reordenada con Cuthill-McKee Inverso*

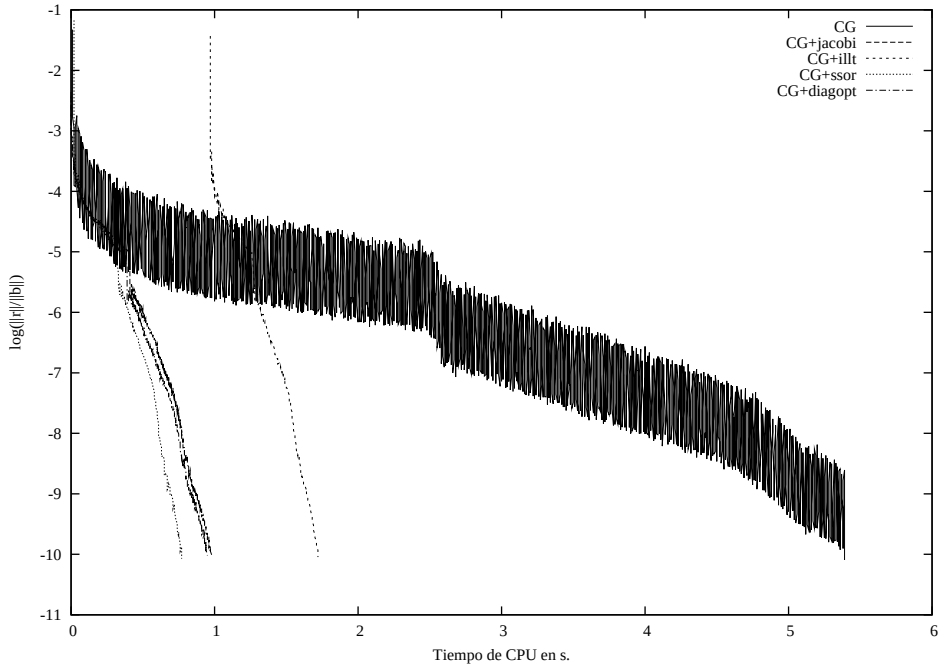

Figura 10.22: *Convergencia del CG con distintos precondicionadores para calorlib (16412 ecuaciones)*

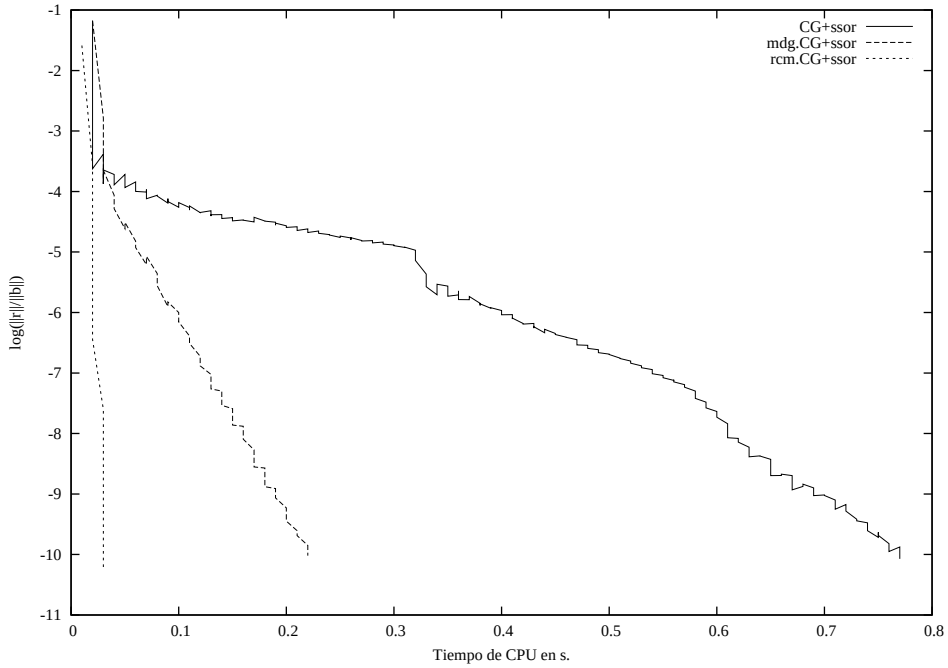

Figura 10.23: *Convergencia del CG+SSOR con diferentes reordenaciones para calorlib (16412 ecuaciones)*

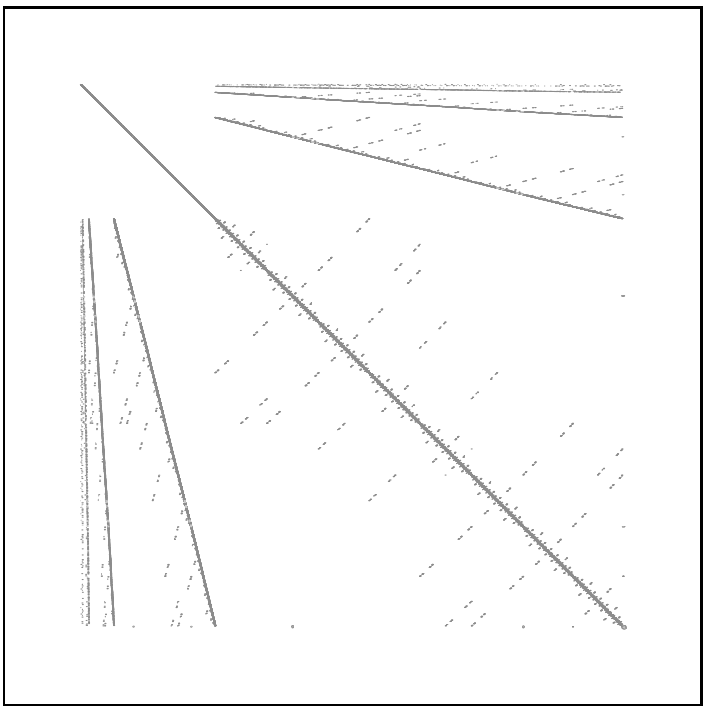

Figura 10.24: *Estructura sparse de la matriz calorlib para* $n = 16340$

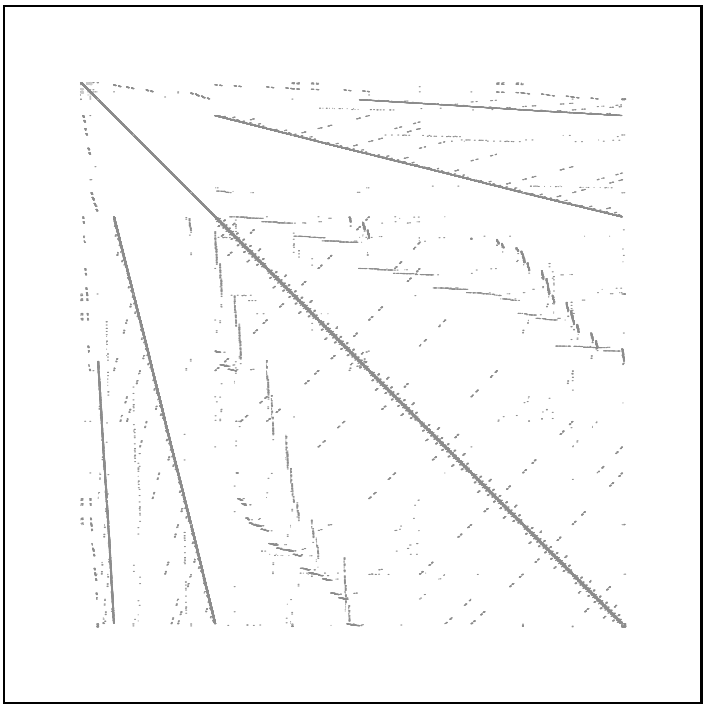

Figura 10.25: *Estructura sparse de la matriz calorlib para* $n = 16340$ *reordenada con Grado Mínimo*

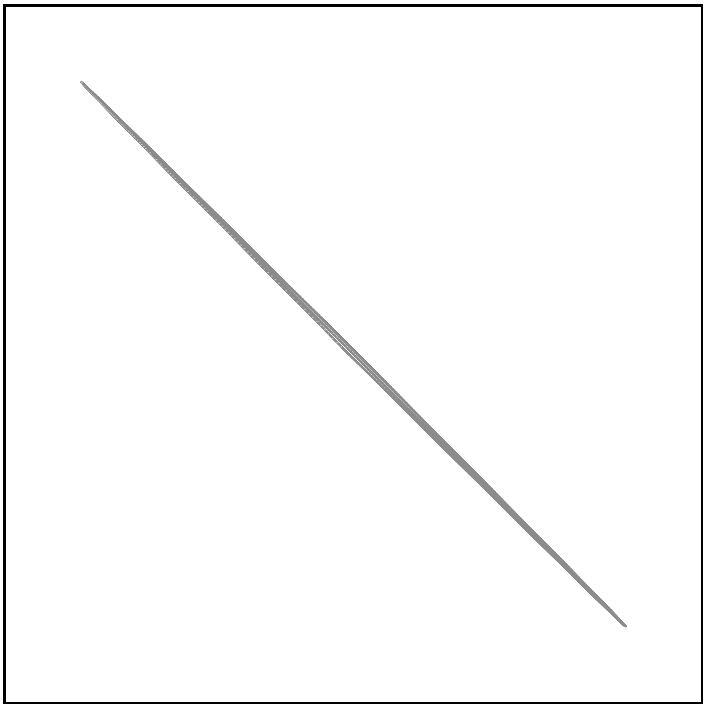

Figura 10.26: *Estructura sparse de la matriz calorlib para* $n = 16340$ *reordenada con Cuthill-McKee Inverso*

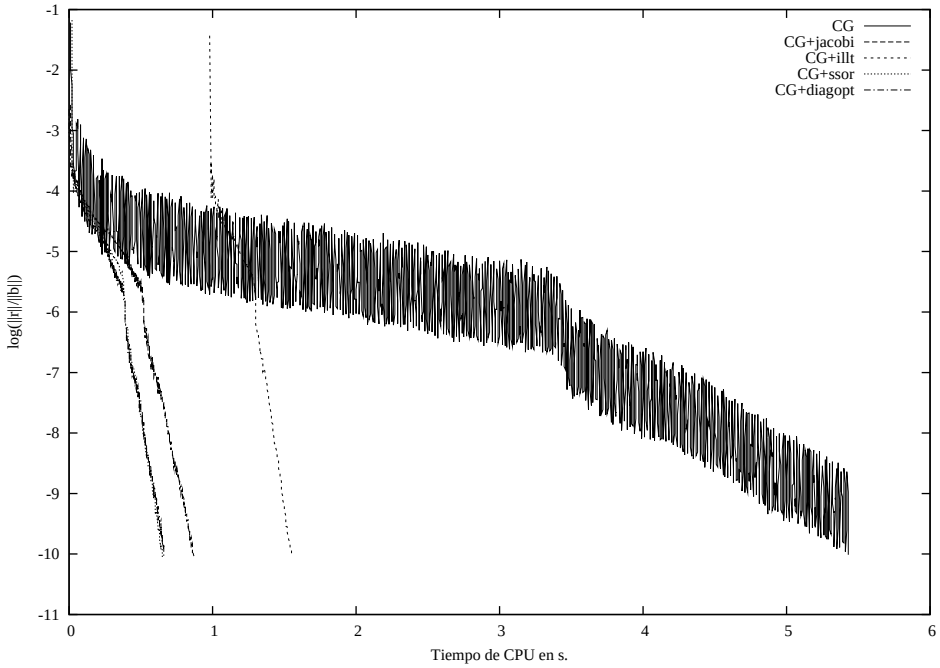

Figura 10.27: *Convergencia del CG con distintos precondicionadores para calorlib (16340 ecuaciones)*

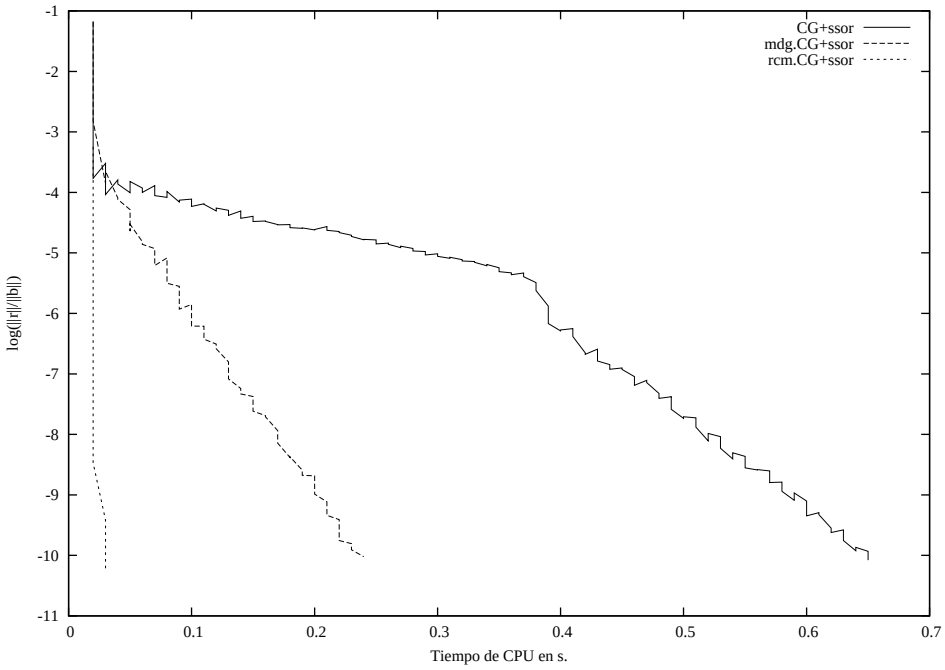

Figura 10.28: *Convergencia del CG+SSOR con diferentes reordenaciones para calorlib (16340 ecuaciones)*

10.4. Ejemplo 4 (elasticidad)

Se estudia la deformación de una laja elástica delgada Ω, sometida a una carga constante $L_2 = -1kg/m$ y a otra que varía linealmente $L_3 = -x/25kg/m$, en parte de su frontera. Se impone la condición de inmovilidad en lado vertical derecho de la laja y en el punto $(1, 1)$ (para más detalles ver [79]). Se considerarán nulas las fuerzas de volumen. El último refinamiento ha conducido a un sistema simétrico de 8434 ecuaciones.

Las figuras 10.29-10.32 muestran las estructuras de las matrices de este problema para la ordenación inicial, reordenación con Grado Mínimo, con Mínimo Vecino y con Cuthill-McKee Inverso, respectivamente.

Las figuras 10.33 y 10.34 representan la convergencia respecto al tiempo de CPU para la resolución con Gradiente conjugado sin y con precondicionamiento y cuando el sistema es reordenado utilizando los algoritmos del Grado Mínimo, Mínimo Vecino y Cuthill-McKee Inverso, respectivamente.

La figura 10.33 revela que el tiempo de computación es ligeramente menor cuando el sistema no es precondicionado, esto se debe al coste que supone la aplicación de los algoritmos de precondicionamiento, ya que la convergencia para este problema en concreto es muy rápida en todos los casos (el número de condición de la matriz *elasticidad* no es muy elevado).

La figura 10.34 muestra que la reordenación del sistema acelera la convergencia cuando se utilizan los algoritmos del Mínimo Vecino y Cuthill-McKee Inverso, siendo el primero ligeramente más rápido, y sin embargo empeora cuando se reordena el sistema con Grado Mínimo. Sin embargo, como cabía esperar, cuando el sistema no está mal condicionado, el efecto beneficioso de la reordenación no es tan espectacular.

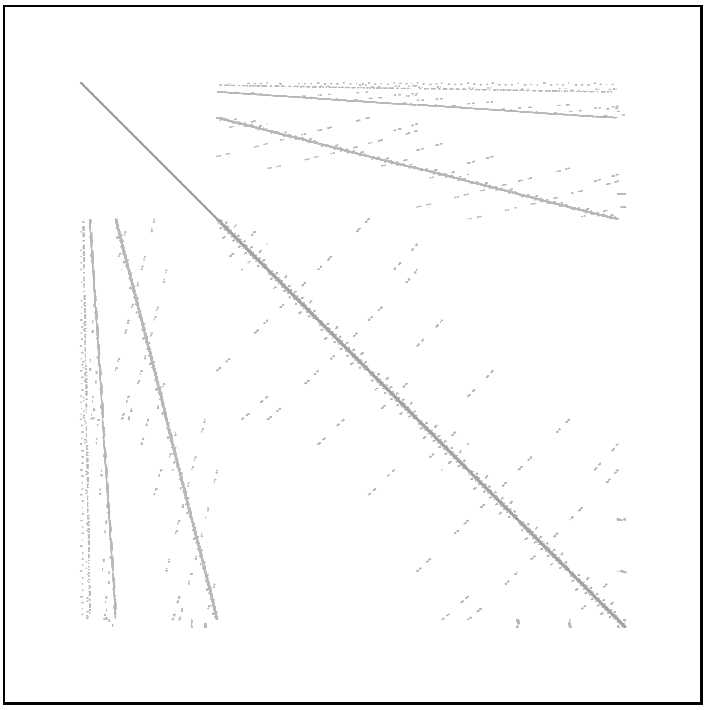

Figura 10.29: *Estructura sparse de la matriz elasticidad*

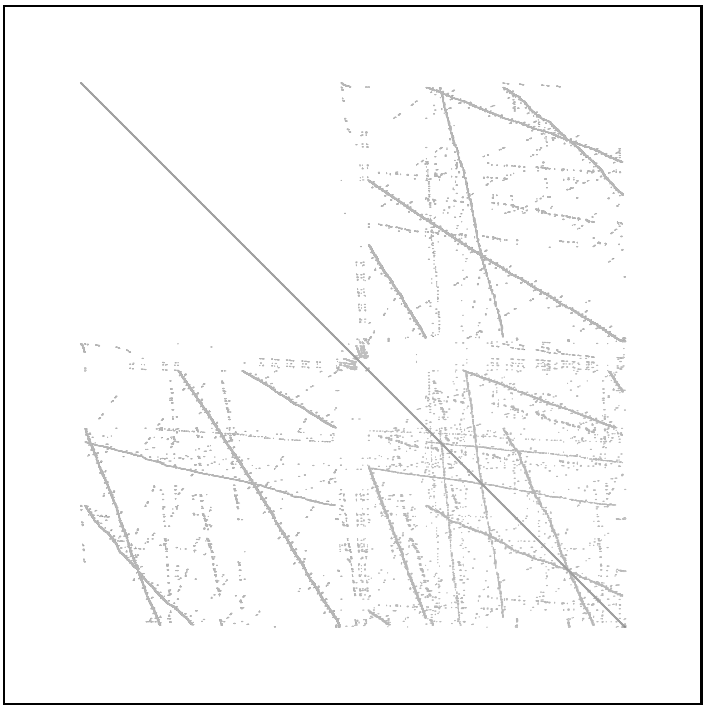

Figura 10.30: *Estructura sparse de la matriz elasticidad reordenada con Grado Mínimo*

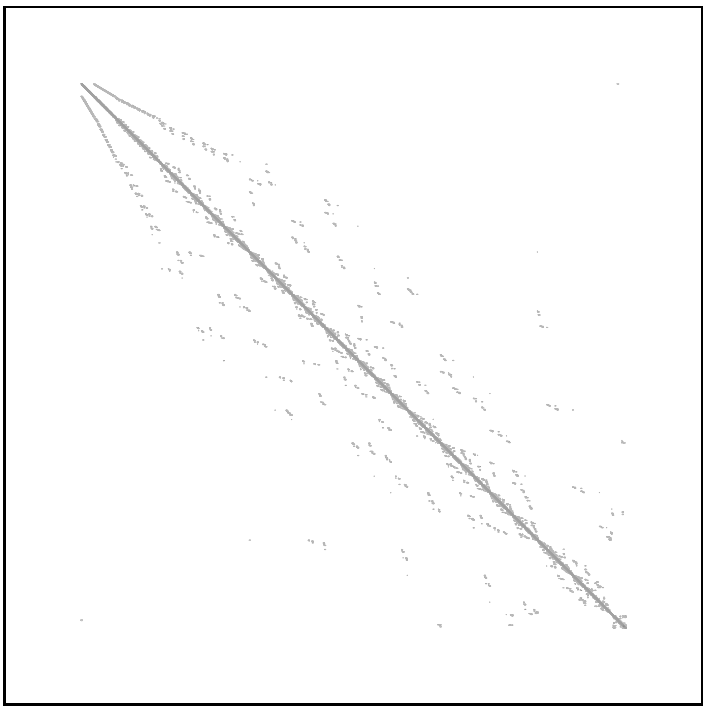

Figura 10.31: *Estructura sparse de la matriz elasticidad reordenada con Mínimo Vecino*

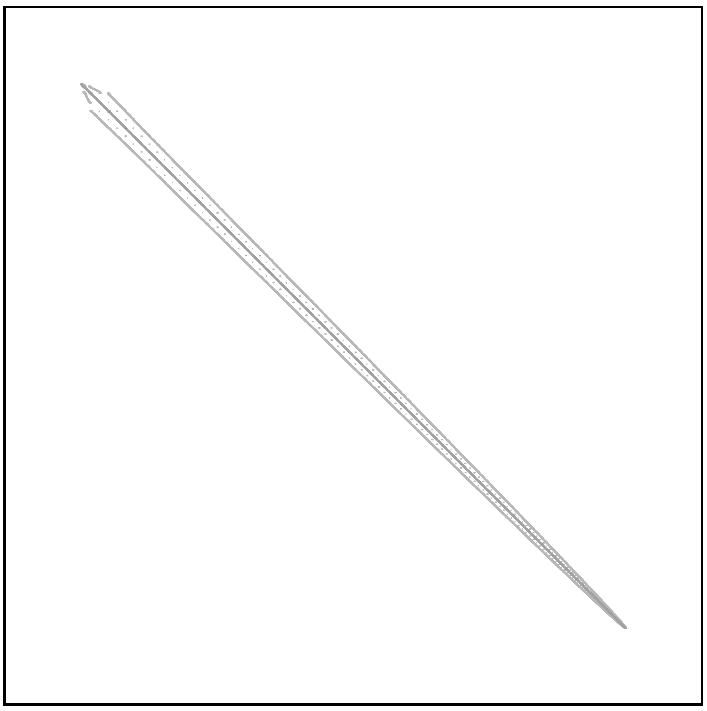

Figura 10.32: *Estructura sparse de la matriz elasticidad reordenada con Cuthill-McKee Inverso*

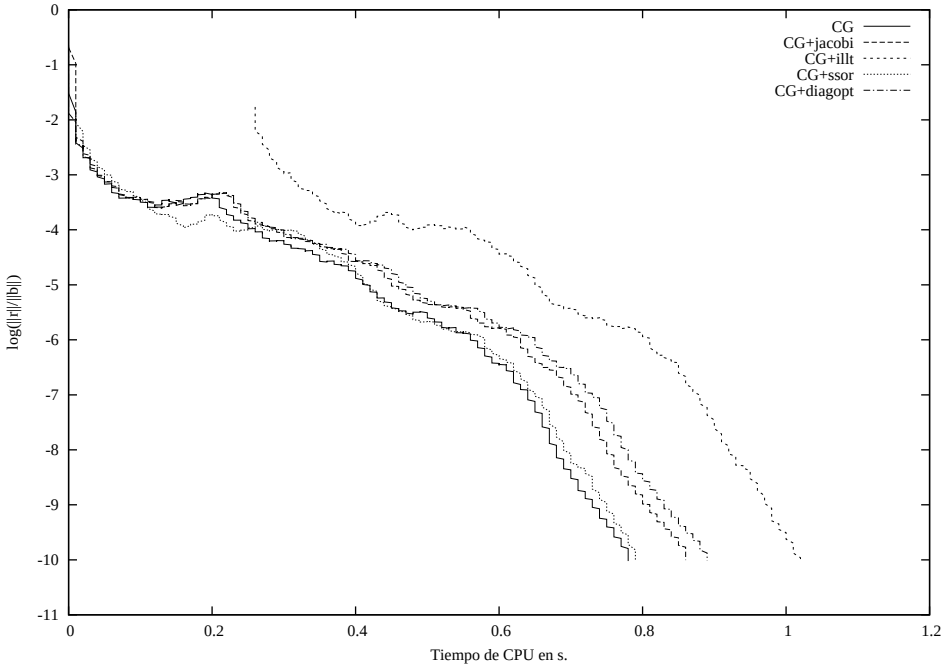

Figura 10.33: *Convergencia del CG con distintos precondicionadores para elasticidad (8434 ecuaciones)*

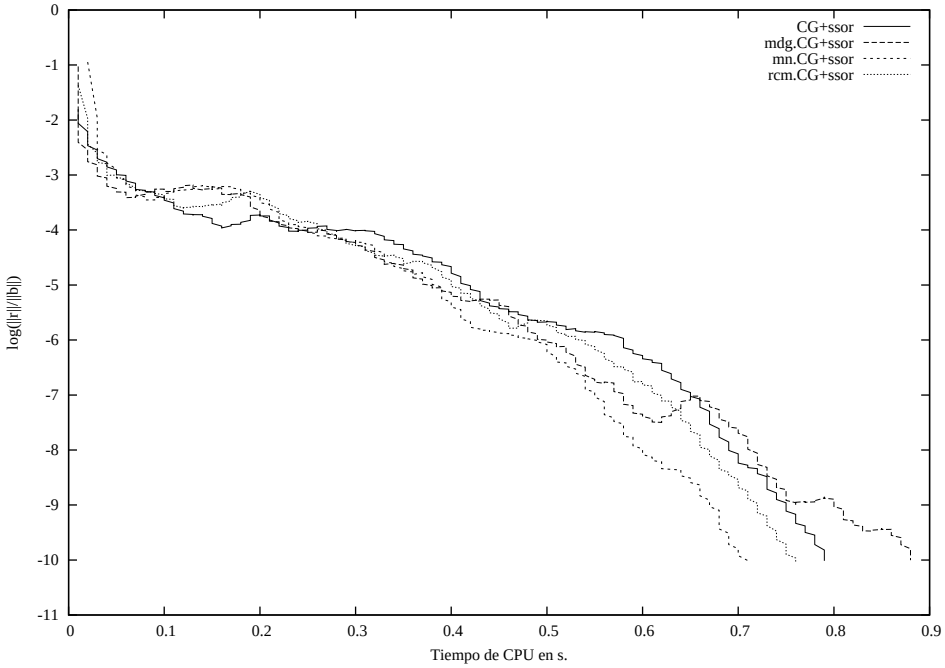

Figura 10.34: *Convergencia del CG+SSOR con diferentes reordenaciones para elasticidad (8434 ecuaciones)*

10.5. Ejemplo 5 (LightTruck)

Este problema corresponde a una simulación numérica de un filtro de carbón activo en $3D$, desarrollado por el grupo LaCàN (Laboratorio de Cálculo Numérico de la Universidad Politécnica de Cataluña). El fenómeno físico dominante en estos dispositivos es el transporte de hidrocarburos. Huerta y otros [40], han desarrollado un modelo de convección-difusión-reacción que responde a la ecuación,

$$\frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u - \nu \Delta u + \sigma\left(u\right)\, u = f\left(u\right)$$

donde u es la concentración de hidrocarburos en el aire, $\mathbf{v}$ representa el campo de velocidades del aire, que se calcula previamente resolviendo un problema de flujo potencial y ν es el coeficiente de difusión. El término de reacción $\sigma\left(u\right)\, u$ y el término fuente $f\left(u\right)$ son fuertemente no lineales.

Se han considerado tres sistemas de ecuaciones correspondientes a la discretización por elementos finitos para tres pasos de tiempo diferentes (4030, 10044 y 16802) del proceso evolutivo. La matriz de coeficientes de orden 17914, es la misma para los tres sistemas, cambiando únicamente el segundo miembro en cada caso.

En la figura 10.35 se representa la estructura de la matriz de estos sistemas.

Figura 10.35: *Estructura sparse de la matriz LightTruck*

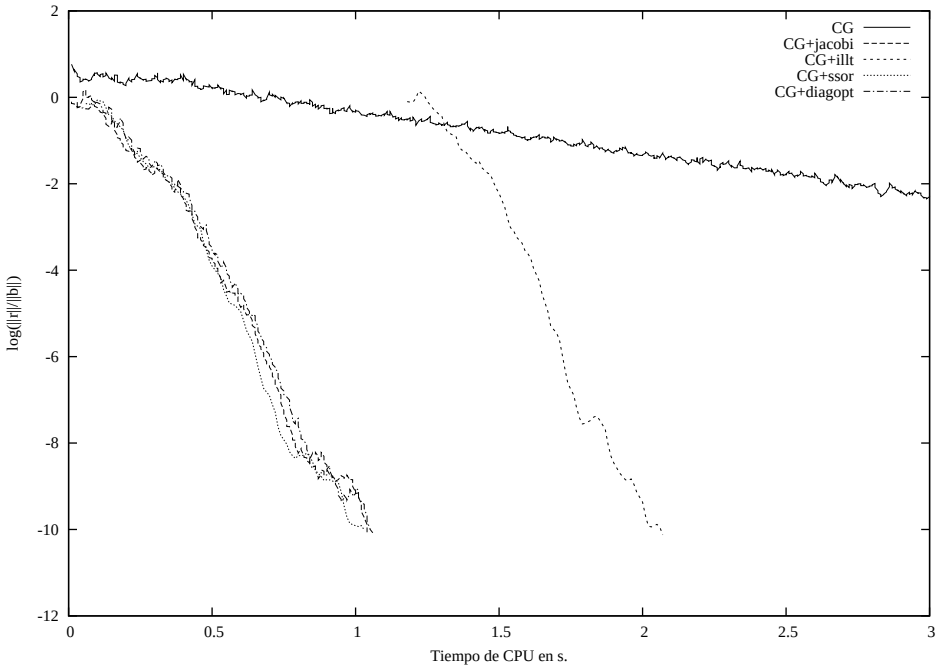

Figura 10.36: *Convergencia del CG con distintos precondicionadores para LightTruck (paso de tiempo 4030)*

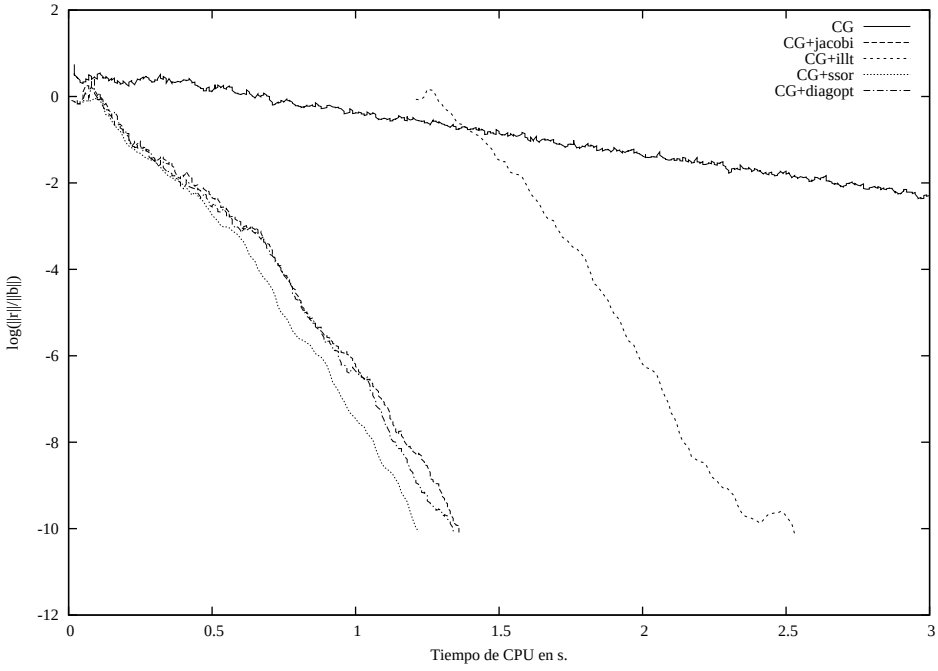

Figura 10.37: *Convergencia del CG con distintos precondicionadores para LightTruck (paso de tiempo 10044)*

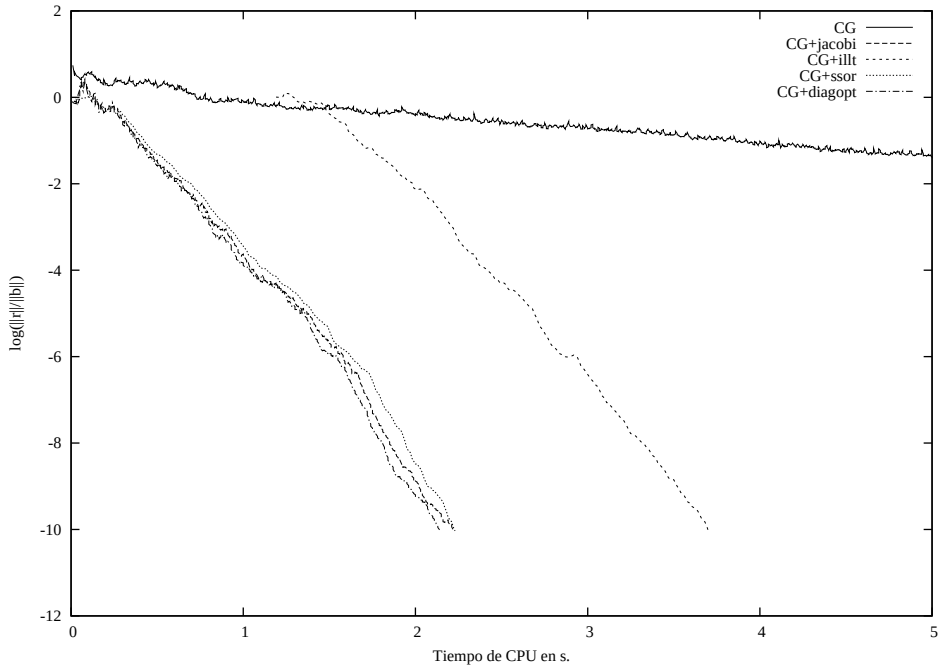

Figura 10.38: *Convergencia del CG con distintos precondicionadores para LightTruck (paso de tiempo 16802)*

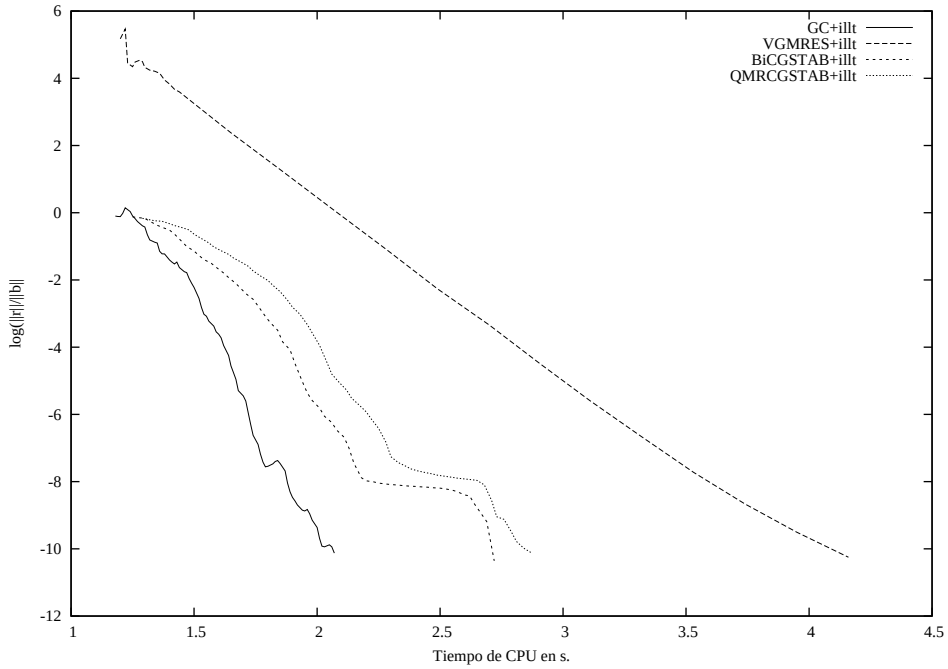

Figura 10.39: *Convergencia de distintos métodos de Krylov para LightTruck (paso de tiempo 4030)*

Las curvas de convergencia respecto al tiempo de CPU, para los distintos pasos de tiempo, del problema evolutivo objeto de estudio, son presentadas en las figuras 10.36-10.38. En los tres casos, se analiza la convergencia del algoritmo del Gradiente Conjugado. En este problema, observamos que la utilización de precondicionadores, acelera notablemente la convergencia, ya que, al contrario de lo que ocurría en el anterior, es muy lenta cuando el sistema está sin precondicionar.

En la figura 10.39 se compara la velocidad de convergencia del algoritmo del Gradiente Conjugado con la de otros métodos de Krylov, todos ellos con precondicionador ILLT(0). Se observa que, como cabía esperar, el Gradiente Conjugado tiene el mejor comportamiento ya que su coste por iteración es menor que el del resto al realizar sólo un producto matriz-vector por iteración frente a los dos de los demás métodos .

Por otro lado, debe tenerse en cuenta aquí que, aunque examinando cada sistema de forma aislada la mejor estrategia en cuanto a tiempo de computación fue el gradiente precondicionado con SSOR, se trata de un problema evolutivo que en cada paso de tiempo genera un sistema de ecuaciones con idéntica matriz de coeficientes. Por ello, el coste de la construcción del precondicionador sólo se debe considerar una vez en el cómputo total, y por tanto, a lo largo del proceso el precondicionamiento con ILLT(0) puede ser, aunque en este caso no lo sea, más eficiente que con SSOR, ya que el número de iteraciones para cada sistema puede ser menor con ILLT(0). En este problema, esto sólo ocurrió la primera vez para el tiempo 4030. En cambio, en las otras dos etapas de tiempo, el SSOR resultó más ventajoso en iteraciones y tiempo que el ILLT(0) para la resolución de los correspondientes sistemas de ecuaciones. En cualquier caso, la diferencia de iteraciones realizadas al aplicar cada uno de estos dos precondicionadores no fue tan elevada comparada con el resto de estrategias de precondicionamiento ensayadas. Evidentemente, en este tipo de problemas habría que reconsiderar los resultados obtenidos si partiéramos de la solución obtenida en el tiempo anterior como aproximación inicial de solución del sistema correspondiente al tiempo actual.

10.6. Ejemplo 6 (OILGEN)

Este ejemplo corresponde a la simulación de una reserva petrolífera orientada hacia la discretización de los depósitos con mallas en $3D$ que proporcionan una estructura regular que puede explotarse para reducir costes y velocidad de procesamiento vectorial. La restricción a una malla puede ocasionar un aumento sustancial en el número de celdas cuando se utiliza un mayor refinamiento en la vecindad de los pozos y fallas. Uniendo las celdas innecesarias para formar una malla menos fina en las áreas del depósito fuera de los pozos y fallas, se reduce considerablemente el número de celdas y, por tanto el número de incógnitas, a la mitad.

Grimes desarrolló un programa informático para generar matrices similares a las encontradas en la simulación tridimensional de reservas petrolíferas. El programa toma como entrada el depósito definido por subregiones rectangulares en $3D$ que pueden solaparse. Cada subregión tiene asociados parámetros de los materiales y mallados específicos.

De las tres matrices generadas con este programa, hemos elegido dos de ellas como ejemplos: ORSREG1 y ORSIRR1, que corresponden a un problema de un modelo de reserva petrolífera en un acuífero con tres pozos. El depósito fue subdividido en una malla de $8 \times 10 \times 5$. Las subregiones alrededor de estos tres pozos se subdividieron en mallas de $8 \times 8 \times 5$, $4 \times 4 \times 5$ y $4 \times 4 \times 5$, respectivamente. El acuífero fue subdividido en una malla de $8 \times 7 \times 5$. La malla completa resultante fue de $21 \times 21 \times 5$. La matriz ORSREG1 fue la Jacobiana obtenida que dió lugar a un sistema no simétrico de 2205 ecuaciones con 14133 entradas no nulas. Las celdas innecesarias en el depósito y en el acuífero, causadas por un mayor refinamiento alrededor de los pozos se unieron para formar ORSIRR1, dando lugar a otro sistema de 1030 ecuaciones con 6558 entradas no nulas.

La figura 10.40 y 10.42 representan las estructuras de las matrices ORSIRR1 y ORSREG1, respectivamente.

En la tabla 10.1 se exponen los resultados obtenidos para ORSIRR1 al utilizar los metodos de Krylov para la resolución del problema con y sin precondicionamiento. Se observa que la convergencia sin precondicionamiento, sólo se consigue en un número de iteraciones menor que la dimensión del sistema para el método VGMRES, que además converge precondicionado con ILU(0) pero no lo hace con el resto de los precondicionadores. Para los métodos BiCG y MQMR el precondicionador SSOR no converge en un número de iteraciones menor que la dimensión del sistema y además el precondicionador de Jacobi es sensiblemente mejor que el Diagonal Óptimo, como también ocurre para los algoritmos CGS y MQMRCGSTAB. Para todos los métodos, excepto para QMR, CGN y LSQR, con los que no converge en ningún caso, el precondicionamiento con ILU(0) acelera la convergencia tanto en número de iteraciones como en tiempo.

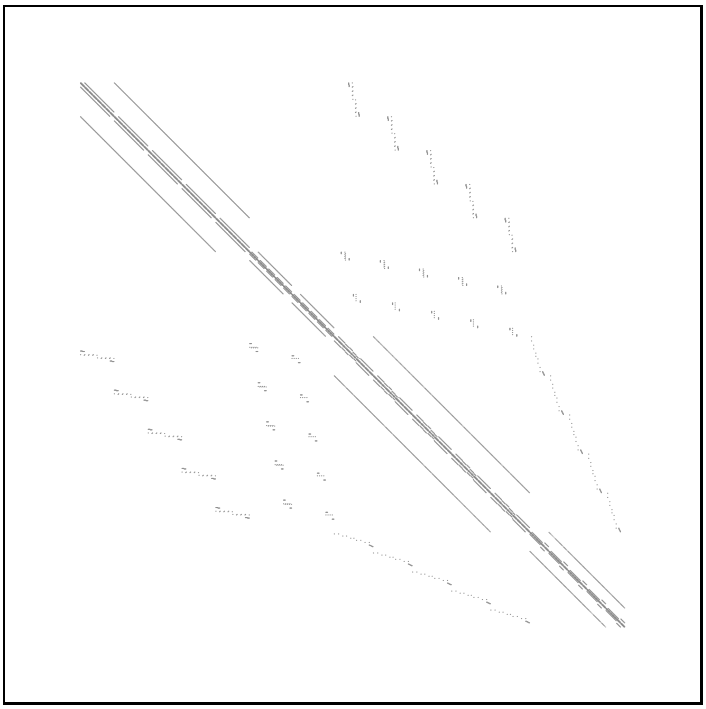

Figura 10.40: *Estructura sparse de la matriz ORSIRR1*

Tabla 10.1: *Ejemplo 6 (1030 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos sin precondicionar y con distintos precondicionadores*

ORSIRR1		SP	Jacobi	ILU	SSOR	Diagopt
VGMRES	k_{init}	1	no conv.	1	no conv.	no conv.
	k_{top}	50		100		
	n^{o}Iter.	117		65		
	t(seg.)	0.989		0.060		
BiCG	n^{o}Iter.	>1030	396	73	>1030	411
	t(seg.)	-	0.070	2.900	-	0.070
CGS	n^{o}Iter.	>1030	280	39	166	288
	t(seg.)	-	0.039	0.019	0.060	0.390
BiCGSTAB	n^{o}Iter.	>1030	635	42	239	415
	t(seg.)	-	0.100	0.019	0.089	0.070
TFQMR	n^{o}Iter.	>1030	495	55	193	411
	t(seg.)	-	0.109	0.029	0.099	0.089
QMRCGSTAB	n^{o}Iter.	>1030	547	44	236	438
	t(seg.)	-	0.140	0.030	0.110	0.110
MQMR	n^{o}Iter.	>1030	456	75	>1030	476
	t(seg.)	-	2.480	3.039	-	2.680
MTFQMR	n^{o}Iter.	>1030	500	38	162	476
	t(seg.)	-	7.069	0.039	0.639	2.680
MQMRCGSTAB	n^{o}Iter.	>1030	394	40	150	481
	t(seg.)	-	4.349	0.079	0.730	6.469

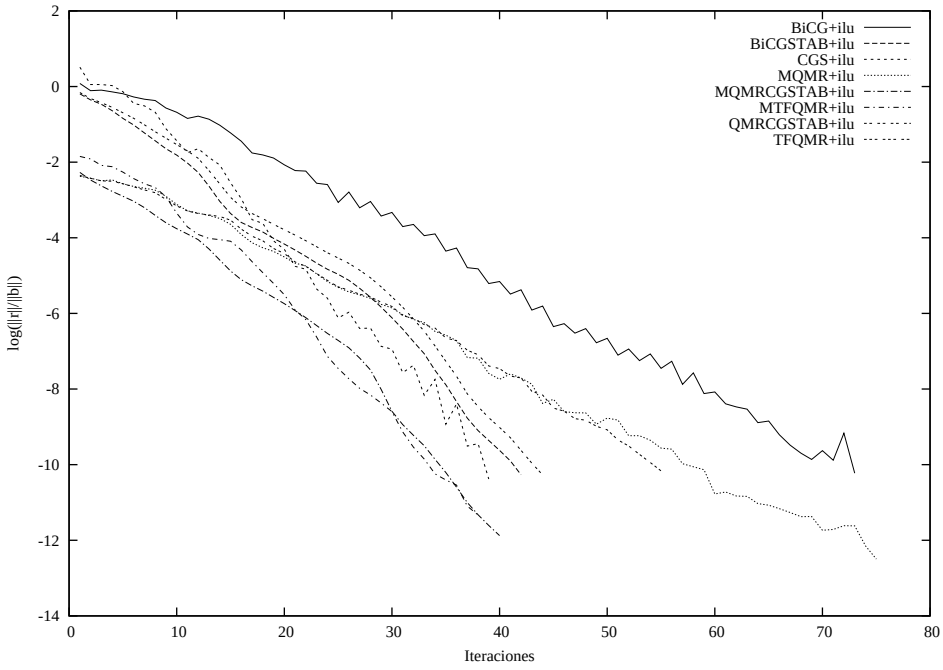

Figura 10.41: *Convergencia de distintos métodos de Krylov precondicionados con ILU(0) para ORSIRR1*

En la figura 10.41, correspondiente a esta estrategia de precondicionamiento, se puede comprobar la suavidad de las curvas de convergencia de los métodos de cuasi-mínimo residuo frente a otros como el BiCG o CGS. Se puede observar también que la modificación propuesta en esta tesis para los métodos de tipo QMR, puede reducir el número de iteraciones de las implementanciones originales, e incluso, como ocurre aquí con el QMR, hacer que puedan converger cuando no lo hacían.

La tabla 10.2 muestra las distintas estrategias para ORSREG1. Igual que en el caso anterior, la convergencia para el método VGMRES sólo se consigue sin precondicionar y con precondicionador ILU(0). Sin embargo, algunos métodos convergen sin utilizar técnicas de precondicionamiento debido al mejor condicionamiento de este sistema frente al anterior obtenido con una malla de peor calidad. Extrañamente, para el método QMR sólo se consigue la convergencia sin precondicionar. Igualmente, para algunos métodos el precondicionador de Jacobi es sensiblemente mejor que el Diagonal Óptimo.

Para los métodos CGN y LSQR, la convergencia no se consigue en ningún caso en un número razonable de iteraciones. Los métodos CGS, TFQMR y MTFQMR sólo convergen con el precondicionador ILU(0) mientras que el método MQMR no converge con los precondicionadores ILU(0) y SSOR en un número de iteraciones menor que la dimensión del sistema.

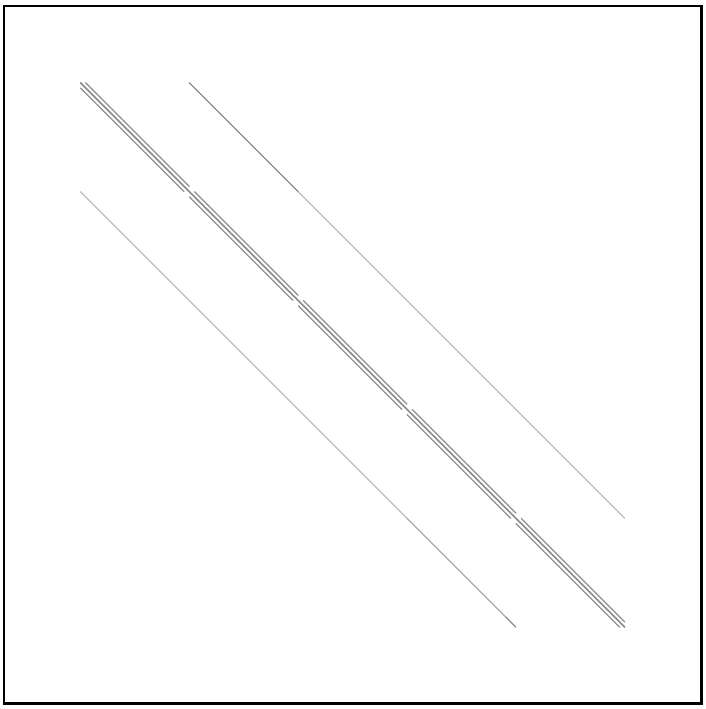

Figura 10.42: *Estructura sparse de la matriz ORSREG1*

Tabla 10.2: *Ejemplo 6 (2205 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos sin precondicionar y con distintos precondicionadores*

ORSREG1		SP	Jacobi	ILU	SSOR	Diagopt
VGMRES	k_{init}	1	no conv.	1	no conv.	no conv.
	k_{top}	50		50		
	n^{o}Iter.	59		51		
	t(seg.)	0.490		0.180		
BiCG	n^{o}Iter.	260	342	>2205	>2205	329
	t(seg.)	0.079	0.130	-	-	0.129
CGS	n^{o}Iter.	>2205	>2205	49	>2205	>2205
	t(seg.)	-	-	0.059	-	-
BiCGSTAB	n^{o}Iter.	1090	601	50	131	400
	t(seg.)	0.259	0.189	0.050	0.090	0.120
QMR	n^{o}Iter.	258	>2205	>2205	>2205	>2205
	t(seg.)	0.119	-	-	-	-
TFQMR	n^{o}Iter.	>2205	>2205	67	>2205	>2205
	t(seg.)	-	-	0.089	-	-
QMRCGSTAB	n^{o}Iter.	648	397	52	162	610
	t(seg.)	0.290	0.220	0.069	0.149	0.349
MQMR	n^{o}Iter.	348	353	>2205	>2205	375
	t(seg.)	2.950	3.09	-	-	3.47
MTFQMR	n^{o}Iter.	>2205	no conv.	44	no conv.	no conv.
	t(seg.)	-		0.120		
MQMRCGSTAB	n^{o}Iter.	826	306	50	156	300
	t(seg.)	42.880	4.710	0.149	1.27	4.409

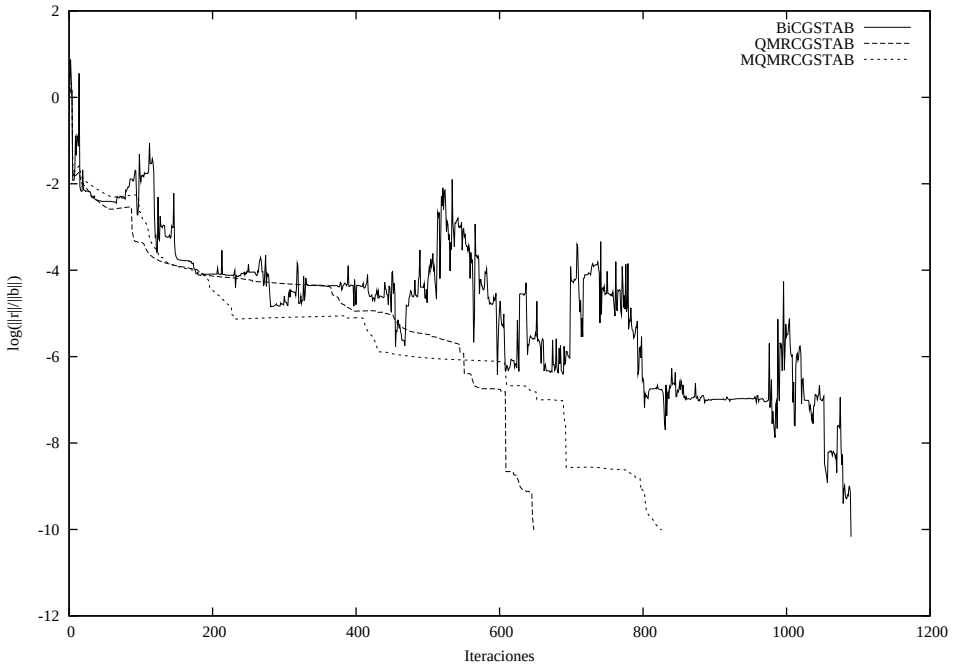

Figura 10.43: *Convergencia de distintos métodos de Krylov sin precondicionar para ORSREG1*

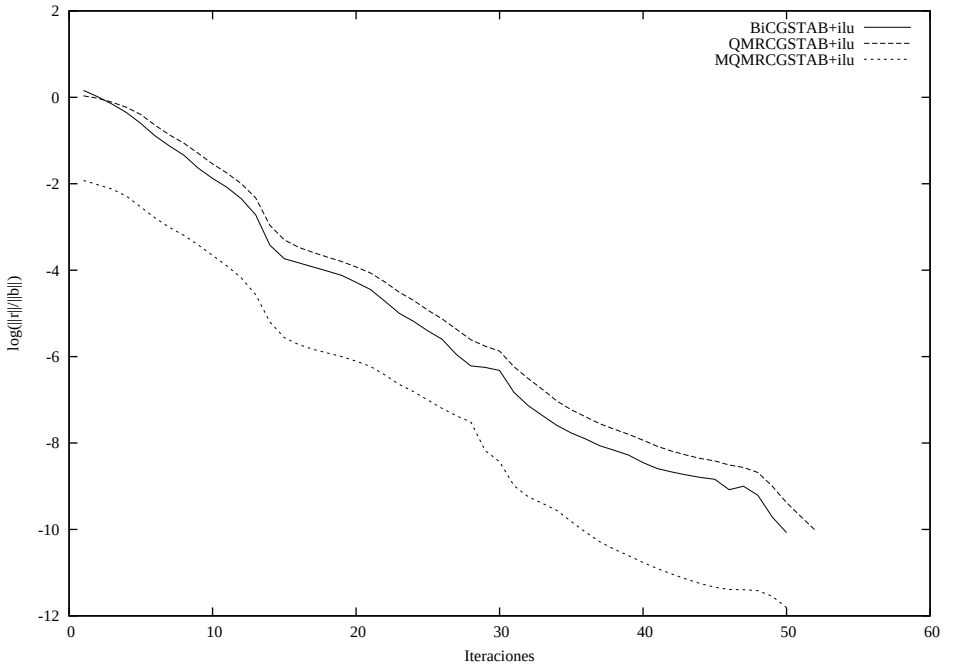

Figura 10.44: *Convergencia de distintos métodos de Krylov precondicionados con ILU(0) para ORSREG1*

La figura 10.43 muestran las curvas de convergencia correspondientes a las distintas estrategias para los métodos BiCGSTAB, QMRCGSTAB y MQMRCGSTAB sin precondicionar y la figura 10.44, las mismas estrategias precondicionadas con ILU(0). Podemos ver cómo se reduce considerablemente el número de iteraciones y se suavizan las curvas, especialmente para el algoritmo BICGSTAB que ahora converge en menos iteraciones que el QMRCGSTAB. El número de iteraciones de MQMRCGSTAB necesarias para obtener la solución con la tolerancia prefijada se mantienen por debajo no sólo de las de QMRCGSTAB sino incluso de las de BiCGSTAB.

10.7. Ejemplo 7 (WATT)

Este nuevo ejemplo extraído de la *Harwell-Boeing Sparse Matrix Colection*, consiste en un problema de ingeniería del petróleo en el que los valores de los coeficientes de la matriz pueden variar tanto como 10^{14}. El uso de subrutinas de escalado automático producen matrices singulares. El efecto de relleno (*fill-in*) es también mayor del esperado.

La simulación dió lugar a dos matrices no simétricas, con una estructura cercana a 5 diagonales de ancho de banda. Se ha tomado el problema WATT1 que corresponde a un sistema de 1856 ecuaciones con 11360 entradas no nulas.

En la figura 10.45 se ilustra la estructura *sparse* de la matriz WATT1.

La tabla 10.3 refleja los resultados obtenidos para este ejemplo. Se observa que la convergencia para el método QMR sólo se consigue sin precondicionar. Por otro lado, el algoritmo LSQR sólo converge en un número de iteraciones inferior al número de ecuaciones del sistema cuando es precondicionado con Jacobi. Asimismo, el método CGN no converge en ningún caso. El precondicionador ILU(0) vuelve a ser el más efectivo en aquellos casos en que funciona.

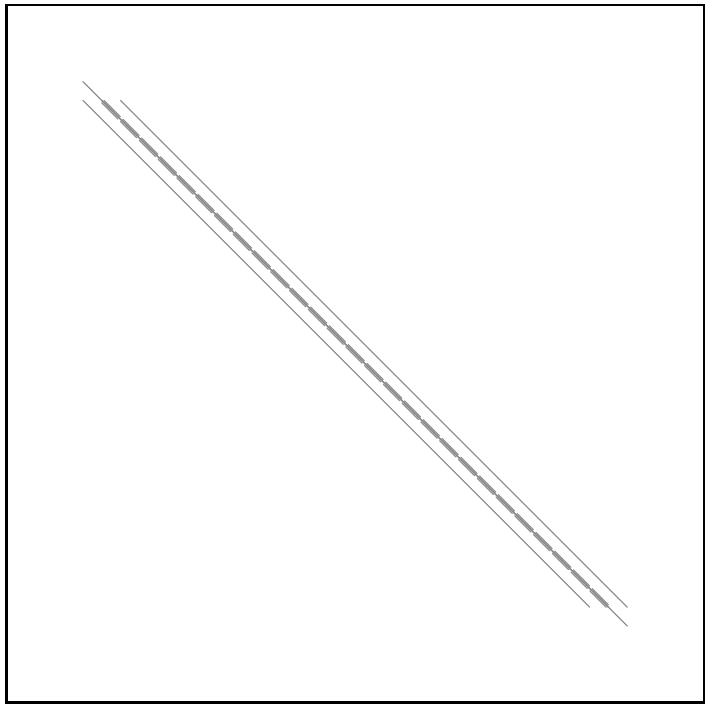

Figura 10.45: *Estructura sparse de la matriz WATT1*

Tabla 10.3: *Ejemplo 7: (1856 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos con y sin precondicionamiento*

WATT1		SP	Jacobi	ILU	SSOR	Diagopt
VGMRES	k_{init}	1	1	1	1	1
	k_{top}	100	100	100	100	100
	n^{o}Iter.	90	125	44	55	130
	t(seg.)	0.560	1.050	0.119	0.140	1.160
BiCG	n^{o}Iter.	283	91	30	37	70
	t(seg.)	0.069	0.029	3.049	3.740	0.019
CGS	n^{o}Iter.	221	54	16	23	54
	t(seg.)	0.040	0.019	0.019	0.019	0.010
BiCGSTAB	n^{o}Iter.	61	39	12	15	39
	t(seg.)	0.009	0.010	0.010	0.009	0.010
QMR	n^{o}Iter.	258	>1856	>1856	>1856	>1856
	t(seg.)	0.099	-	-	-	-
TFQMR	n^{o}Iter.	527	196	69	84	221
	t(seg.)	0.140	0.079	0.069	0.070	0.090
QMRCGSTAB	n^{o}Iter.	66	41	12	17	42
	t(seg.)	0.030	0.019	0.019	0.010	0.019
LSQR	n^{o}Iter.	>1856	1831	>1856	>1856	>1856
	t(seg.)	-	0.670	-	-	-
MQMR	n^{o}Iter.	241	133	>1856	>1856	131
	t(seg.)	1.240	0.369	-	-	0.349
MTFQMR	n^{o}Iter.	274	111	38	45	117
	t(seg.)	3.11	0.500	0.070	0.090	0.569
MQMRCGSTAB	n^{o}Iter.	68	102	33	43	104
	t(seg.)	0.189	0.449	0.070	0.079	0.439

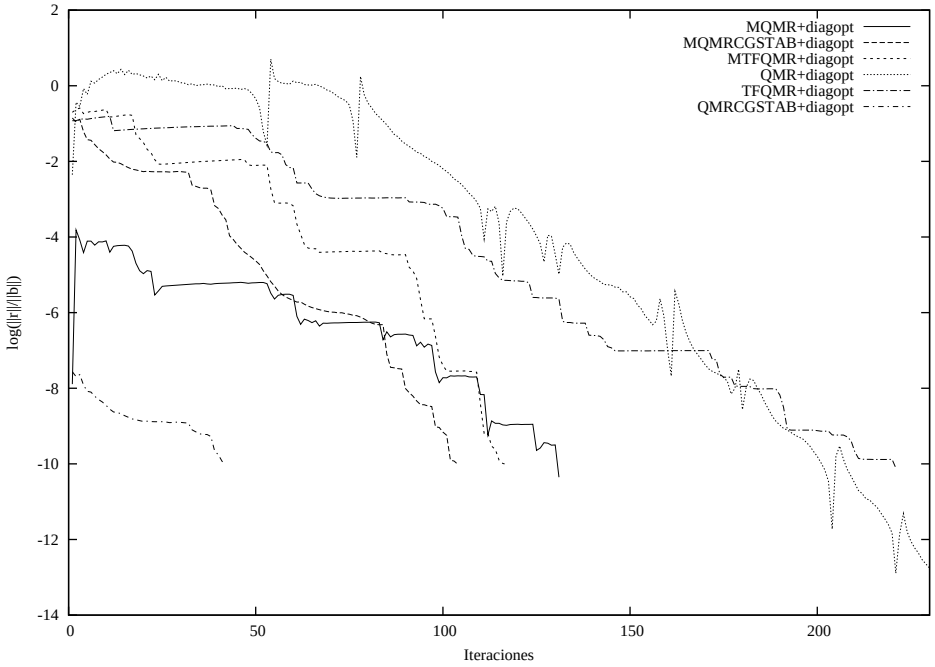

Figura 10.46: *Convergencia de distintos métodos tipo QMR con precondicionador Diagonal Óptimo para WATT1*

La figura 10.46 muestra las curvas de convergencia correspondiente a las distintas estrategias para los métodos de cuasi-mínimo residuo clásicos y modificados con precondicionador Diagonal Óptimo. Los Modificados QMR y TFQMR convergen en menos iteraciones que los correspondientes implementados clasicamente, aunque en el caso del Modificado QMRCGSTAB esto no ocurre.

10.8. Ejemplo 8 (PORES)

Se trata de un problema de modelización de depósitos, compuesto por tres matrices extraídas del paquete PORES para simulación de depósitos. Estas matrices no son simétricas aunque sí su estructura. Se ha elegido para este estudio el caso PORES2, que contiene 1224 ecuaciones con 9613 entradas no nulas.

En la figura 10.47 se ilustra la estructura *sparse* de la matriz correspondiente a este sistema de ecuaciones.

En la tabla 10.4 se presentan los resultados obtenidos para este problema, donde la convergencia sólo se logra con el precondicionador ILU(0) en seis de los métodos.

Se presenta la gráfica de convergencia correspondiente a esta estrategia en la figura 10.48. Prácticamente todos los métodos convergen en un número de iteraciones y tiempo de computación similares, sin embargo la convergencia más suave correponde a los Modificado TFQMR y QMRCGSTAB.

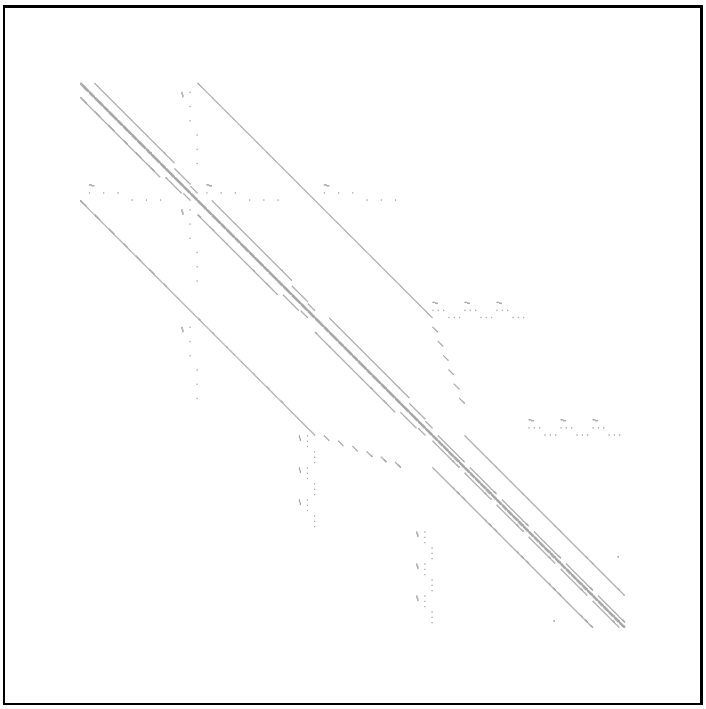

Figura 10.47: *Estructura sparse de la matriz PORES2*

Tabla 10.4: *Ejemplo 8 (1224 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos sin precondicionar y con distintos precondicionadores*

PORES2		ILU
CGS	n^oIter.	32
	t(seg.)	0.029
BiCGSTAB	n^oIter.	28
	t(seg.)	0.019
TFQMR	n^oIter.	32
	t(seg.)	0.029
QMRCGSTAB	n^oIter.	30
	t(seg.)	0.019
MTFQMR	n^oIter.	31
	t(seg.)	0.039
MQMRCGSTAB	n^oIter.	28
	t(seg.)	0.039

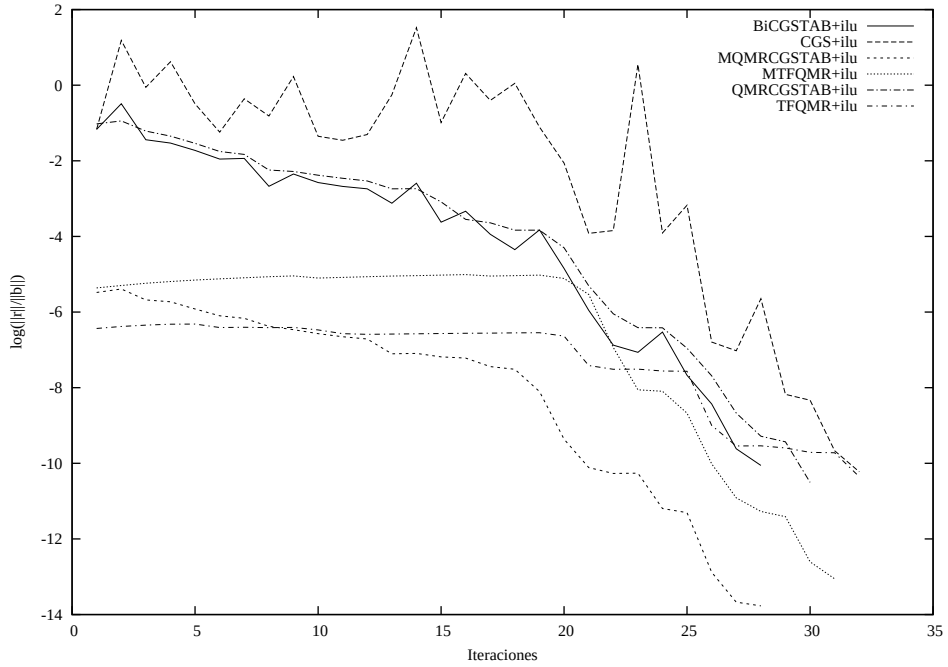

Figura 10.48: *Convergencia de distintos métodos de Krylov precondicionados con ILU(0) para PORES2*

10.9. Ejemplo 9 (convdifhor)

Consideramos el problema de convección-difusión en $2-D$ definido en un dominio cuadradro Ω por la ecuación,

$$v_1 \frac{\partial u}{\partial x} - K\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = F$$

con un campo de velocidades horizontal dado por,

$$v_1 = 10^4 \left(y - 1/2\right)\left(x - x^2\right)\left(1/2 - x\right)$$

siendo K, el coeficiente de difusión dado en $\left[\frac{m^2}{h}\right]$ y $F = \frac{f}{c\rho}$, el término fuente que depende de las fuentes volumétricas y de las características del medio, $\left[\frac{^oC}{h}\right]$.

Los valores de K varían en Ω desde 10^{-5} a 10^2, y los de F desde 10^3 a 1. Las condiciones de contorno son Dirichlet $u = 0$ en $x = 1$ y $u = 1$ en $x = 0$, y Neumann, $\frac{\partial u}{\partial n} = 0$ en el resto de la frontera.

Durante el proceso adaptativo de refinamiento de mallas, se han obtenido tres mallas que han dado lugar a sistemas no simétricos, con 1234, 3423 y 13190 ecuaciones, respectivamente. Las figuras 10.49-10.51, 10.55-10.58 y 10.61-10.63 muestran las estructuras *sparse* de las matrices con diferentes reordenaciones para cada uno de estos sistemas de ecuaciones, respectivamente.

Los resultados para los distintos métodos de Krylov con y sin precondicionamiento son mostrados en las tablas 10.5-10.7, que además incluyen los resultados obtenidos para los mismos sistemas cuando son reordenados previamente con los algoritmos de Grado Mínimo, Cuthill-McKee Inverso y, para la discretización que da lugar al sistema de 3423 ecuaciones, también con el algoritmo del Mínimo Vecino.

El sistema de 1234 ecuaciones no converge en ningún caso en un número menor de iteraciones que la dimensión del sistema sin utilizar una técnica de precondicionamiento. Para los métodos QMR, CGN y LSQR, la convergencia no se consigue en ningún caso en un número razonable de iteraciones. El método VGMRES no converge cuando se precondiciona con Jacobi, mientras que el método BiCG no converge precondicionado con SSOR y el MQMR sólo lo hace con los precondicionadores de tipo diagonal: Jacobi y Diagonal Óptimo. En los casos en que se consigue la convergencia, el mejor precondicionador vuelve a ser el ILU(0) (ver figura 10.52). Se ha analizado la convergencia de los distintos métodos precondicionados con ILU(0) cuando el sistema es reordenado y podemos ver que la reordenación con Cuthill-McKee Inverso hace posible y mejora la convergencia en todos los casos, excepto para los métodos QMR y CGN, (ver figura 10.54). En cambio, reordenando con Grado Mínimo (ver figura 10.53) no se consigue convergencia para todos los métodos, aunque en los casos en que funciona también supone una mejoría.

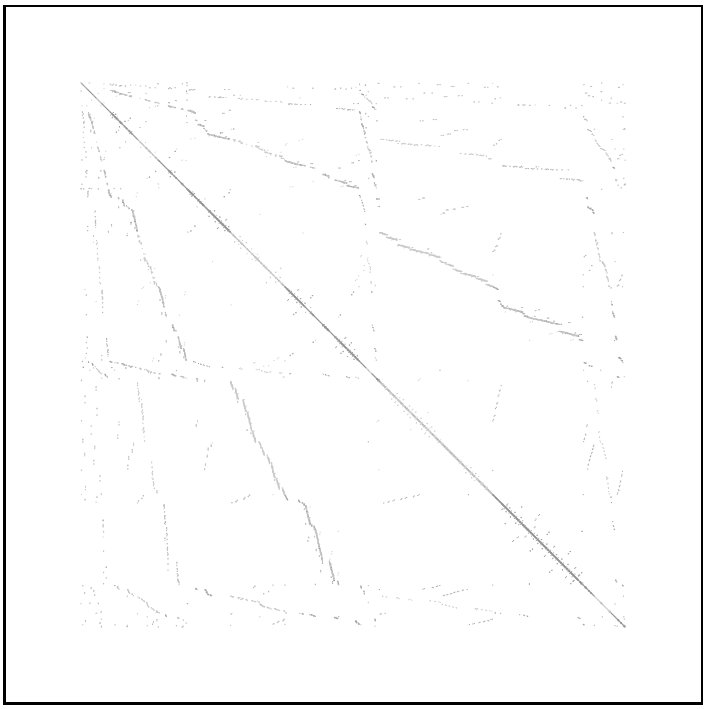

Figura 10.49: *Estructura sparse de la matriz convdifhor para* $n = 1234$

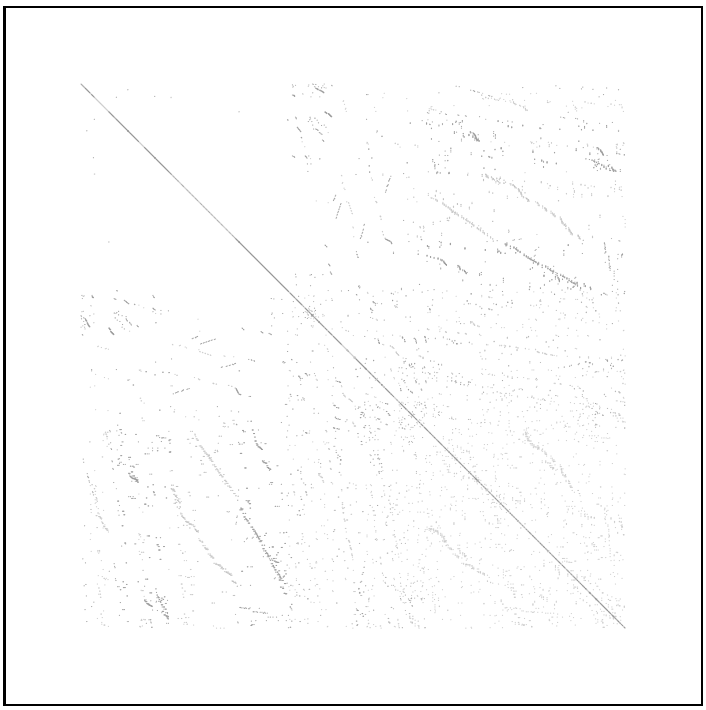

Figura 10.50: *Estructura sparse de la matriz convdifhor para* $n = 1234$ *reordenada con Grado Mínimo*

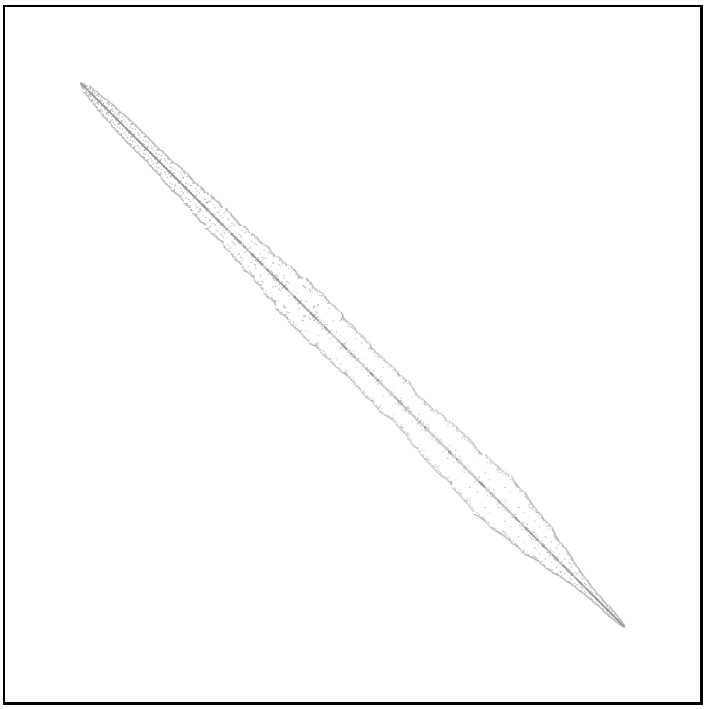

Figura 10.51: *Estructura sparse de la matriz convdifhor para $n = 1234$ reordenada con Cuthill-McKee Inverso*

En las Figuras 10.52-10.54, que representan las curvan de convergencia de los distintos métodos precondicionados con ILU(0) para el sistema original y para los reordenados con Grado Mínimo y Cuthill-McKee Inverso, respectivamente, puede verse cómo se suaviza la convergencia, además de acelerarse.

Tabla 10.5: *Ejemplo 9 (1234 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos con y sin precondicionar*

convdifhor(1234)		Jacobi	ILU	SSOR	Diagopt	MDG+ILU	RCM+ILU
VGMRES	k_{init}	no conv.	1	1	1	1	1
	k_{top}		100	100	500	100	100
	nº Iter.		79	96	229	62	40
	t(seg.)		0.1800	0.2700	2.0900	0.079	0.040
BiCG	nº Iter.	337	245	no conv.	271	no conv.	43
	t(seg.)	0.079	19.27		0.069		3.009
CGS	nº Iter.	204	60	82	177	47	29
	t(seg.)	0,040	0.019	0.040	0.029	0.029	0.019
BiCGSTAB	nº Iter.	275	53	67	201	43	25
	t(seg.)	0.059	0.029	0.040	0.040	0.019	0.010
TFQMR	nº Iter.	339	87	128	201	62	39
	t(seg.)	0.099	0.070	0.090	0.040	0.050	0.030
QMRCGSTAB	nº Iter.	293	54	70	212	45	27
	t(seg.)	0.079	0.039	0.040	0.069	0.029	0.019
LSQR	nº Iter.	>1234	>1234	>1234	>1234	>1234	722
	t(seg.)	-	-	-	-	-	50.70
MQMR	nº Iter.	295	>1234	>1234	322	>1234	44
	t(seg.)	1.240	-	-	1.509	-	3.140
MTFQMR	nº Iter.	206	59	76	196	47	27
	t(seg.)	1.180	0.109	0.189	1.120	0.060	0.029
MQMRCGSTAB	nº Iter.	318	57	68	219	46	25
	t(seg.)	2.840	0.109	0.159	1.350	0.069	0.029

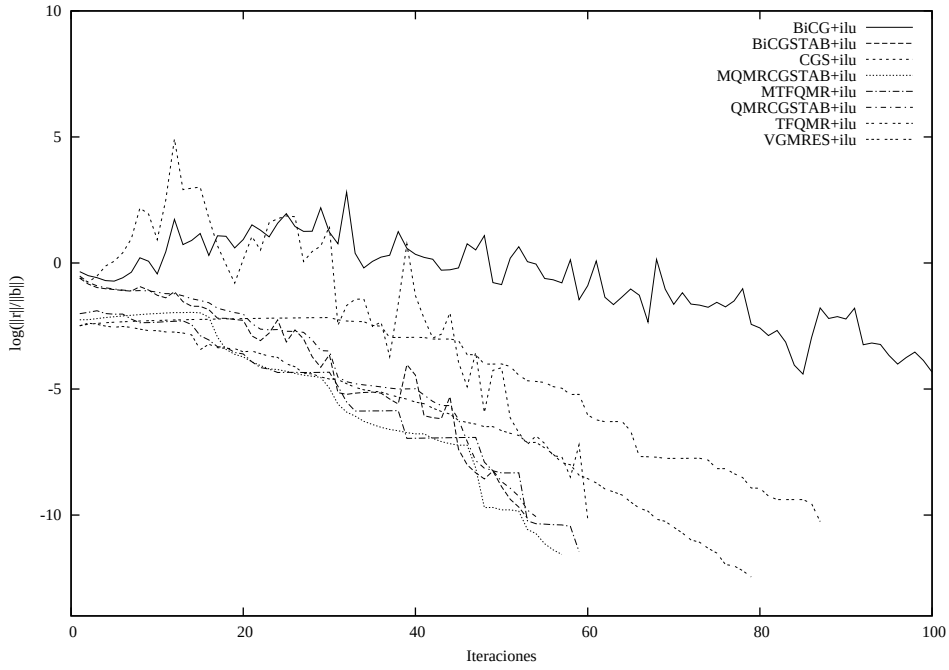

Figura 10.52: *Convergencia de distintos métodos de Krylov precondicionados con ILU(0) para convdifhor (1234 ecuaciones)*

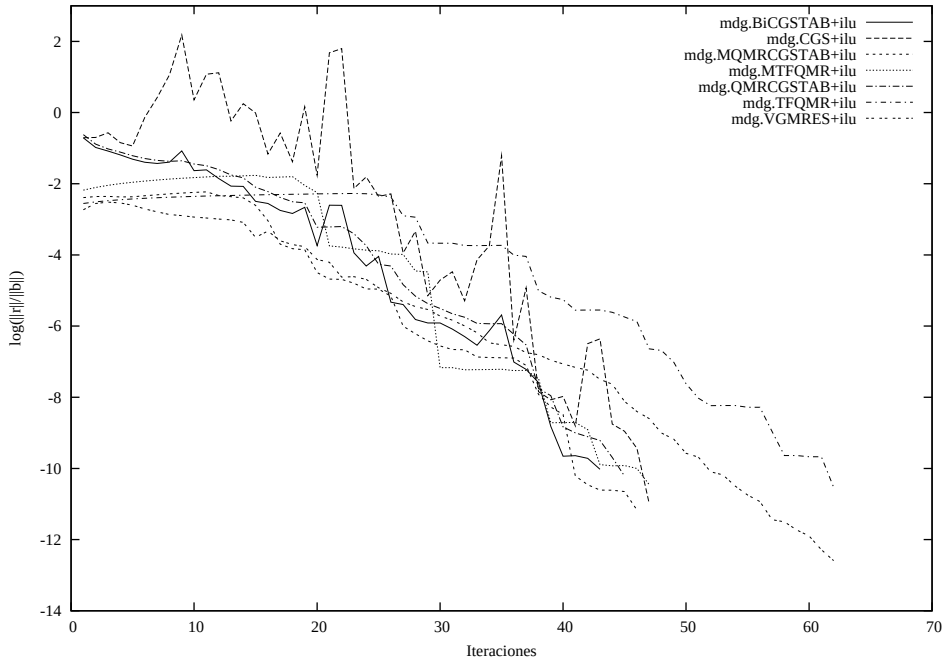

Figura 10.53: *Convergencia de distintos métodos de Krylov precondicionados con ILU(0) después de reordenar con Grado Mínimo para convdifhor (1234 ecuaciones)*

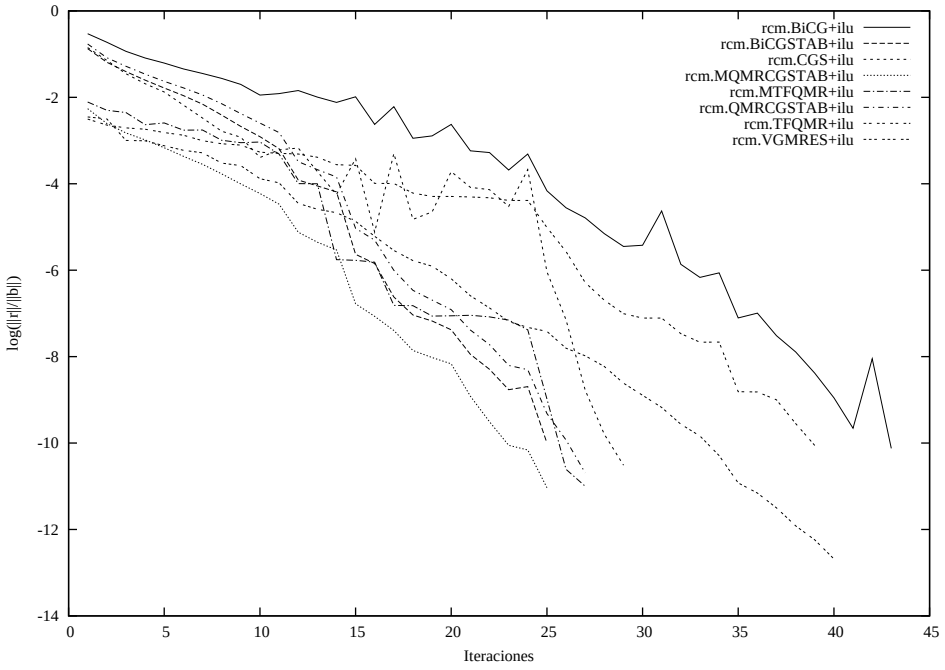

Figura 10.54: *Convergencia de distintos métodos de Krylov precondicionados con ILU(0) después de reordenar con Cutill-McKee Inverso para convdifhor (1234 ecuaciones)*

Igual que para la anterior discretización, el sistema de 3423 ecuaciones tampoco converge nunca sin el uso de técnicas de precondicionamiento. Las estrategias para el método VGMRES se han realizado en este caso utilizando la técnica *restart* que logra la convergencia en todos los casos en dos o tres iteraciones, aunque el coste, por lo que se refiere a tiempo de computación es bastante alto respecto a los otros métodos, con excepción de los Modificados TFQMR y QMRCGSTAB. Para los métodos QMR, MQMR, CGN y LSQR, la convergencia no se consigue en ningún caso y para el algoritmo BiCG, los cálculos con la traspuesta de los precondicionadores ILU(0) y SSOR son muy lentos. Los métodos CGS, TFQMR y MTFQMR, y MQMRCGSTAB sólo convergen con los precondicionadores ILU(0) y SSOR en un número menor de iteraciones que la dimensión del sistema. En todos los casos en que se consigue la convergencia, el mejor precondicionador vuelve a ser el ILU(0) (excepto para el BiCG). Para esta discretización, se ha estudiado la convergencia de los métodos de Krylov precondicionados con ILU(0) cuando el sistema es reordenado con Grado Mínimo, Mínimo Vecino y Cuthill-McKee Inverso y el comportamiento es muy parecido al del caso anterior. La reordenación con Mínimo Vecino presenta una convergencia muy similar a la de Cuthill-McKee Inverso, y mejora la conseguida con Grado Mínimo.

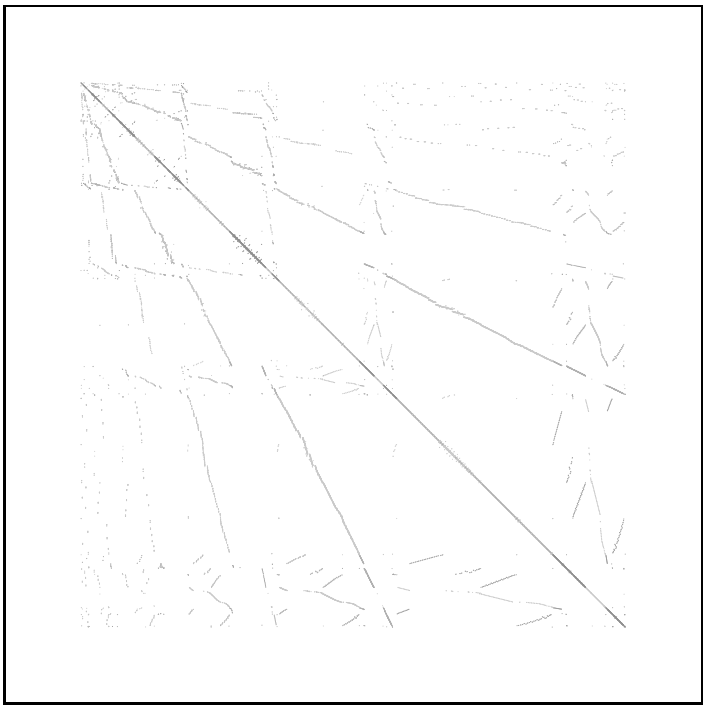

Figura 10.55: *Estructura sparse de la matriz convdifhor para* $n = 3423$

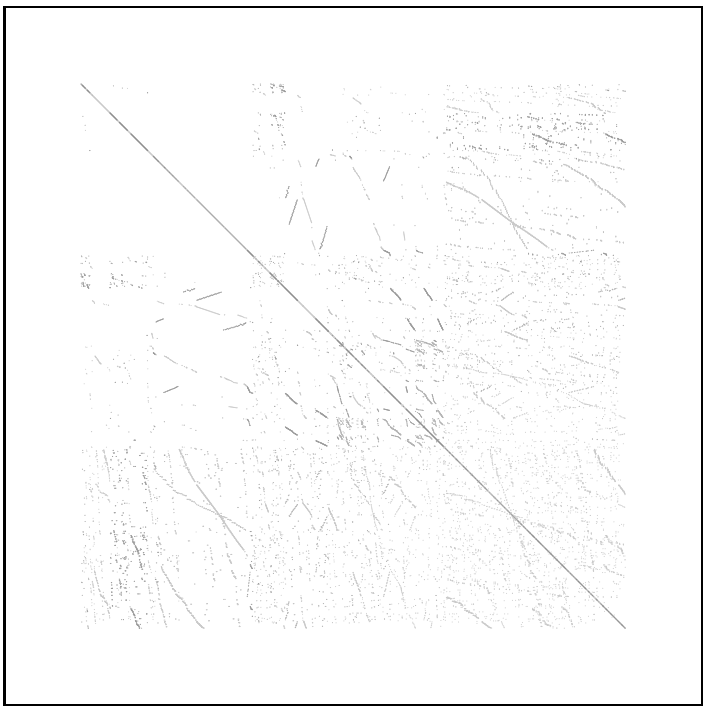

Figura 10.56: *Estructura sparse de la matriz convdifhor para* $n = 3423$ *reordenada con Grado Mínimo*

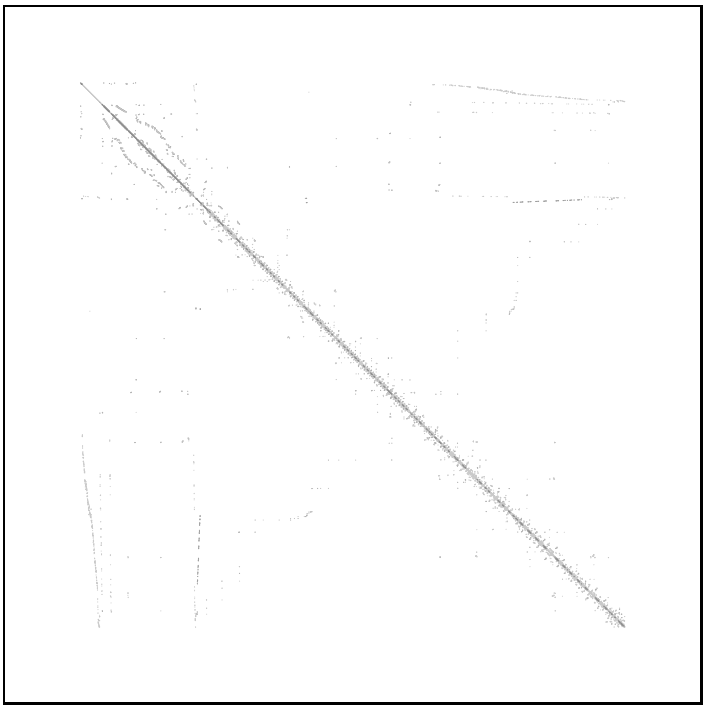

Figura 10.57: *Estructura sparse de la matriz convdifhor para* $n = 3423$ *reordenada con Mínimo Vecino*

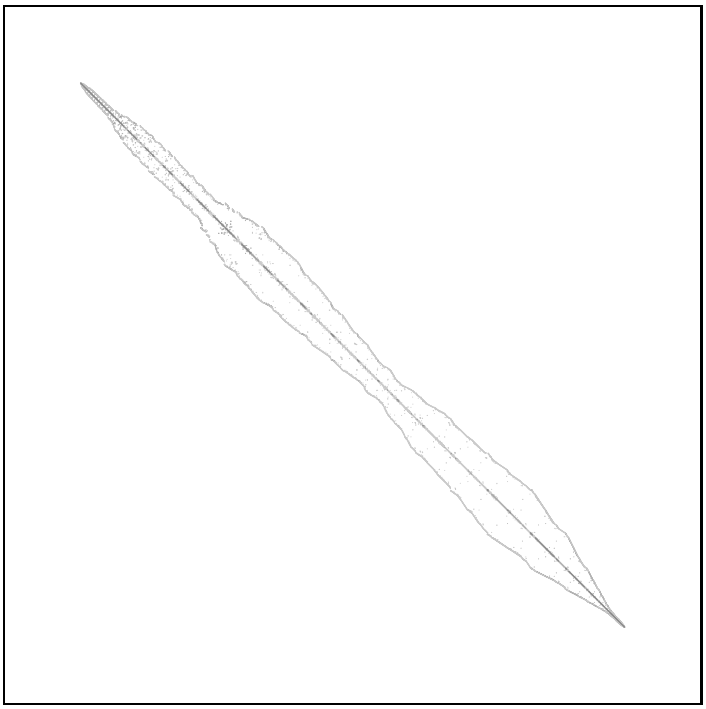

Figura 10.58: *Estructura sparse de la matriz convdifhor para* $n = 3423$ *reordenada con Cuthill-McKee Inverso*

Tabla 10.6: *Ejemplo 9 (3423 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos sin precondicionar y con distintos precondicionadores*

convdifhor(3423)		Jacobi	ILU	SSOR	Diagopt	MDG+ILU	MN+ILU	RCM+ILU
VGMRES	k_{init}	500	100	200	400	1	1	1
	k_{top}	500	100	200	400	200	200	200
	n^{o}Iter.	2	3	2	2	112	108	96
	t(seg.)	21.88	1.389	3.690	14.04	1.189	1.100	0.870
BiCG	n^{o}Iter.	1003	-	-	808	-	-	-
	t(seg.)	0.889	>500	>500	0.720	>500	>500	>500
CGS	n^{o}Iter.	>3423	146	211	>3423	no conv.	84	76
	t(seg.)	-	0.284	0.379	-		0.189	0.180
BiCGSTAB	n^{o}Iter.	687	126	181	534	91	75	65
	t(seg.)	0.610	0.269	0.319	0.459	0.179	0.170	0.160
TFQMR	n^{o}Iter.	>3423	253	358	>3423	>3423	139	119
	t(seg.)	-	0.680	0.889	-	-	0.400	0.329
QMRCGSTAB	n^{o}Iter.	842	137	183	540	99	75	66
	t(seg.)	1.12	0.359	0.420	0.730	0.239	0.219	0.180
MQMR	n^{o}Iter.	-	-	-	-	-	-	315
	t(seg.)	>500	>500	>500	>500	>500	>500	69.52
MTFQMR	n^{o}Iter.	712	158	256	no conv.	137	91	77
	t(seg.)	42.590	2.130	5.410		1.610	0.750	0.569
MQMRCGSTAB	n^{o}Iter.	872	151	191	550	104	79	72
	t(seg.)	93.209	1.959	3.040	28.710	0.949	0.600	0.519

En particular, las figuras 10.59 y 10.60 muestran el efecto de la reordenación en la convergencia de los métodos MTFQMR y MQMRCGSTAB precondicionados con ILU(0), respectivamente. Como puede comprobarse, la reordenación reduce casi a la mitad el número de iteraciones en ambos casos.

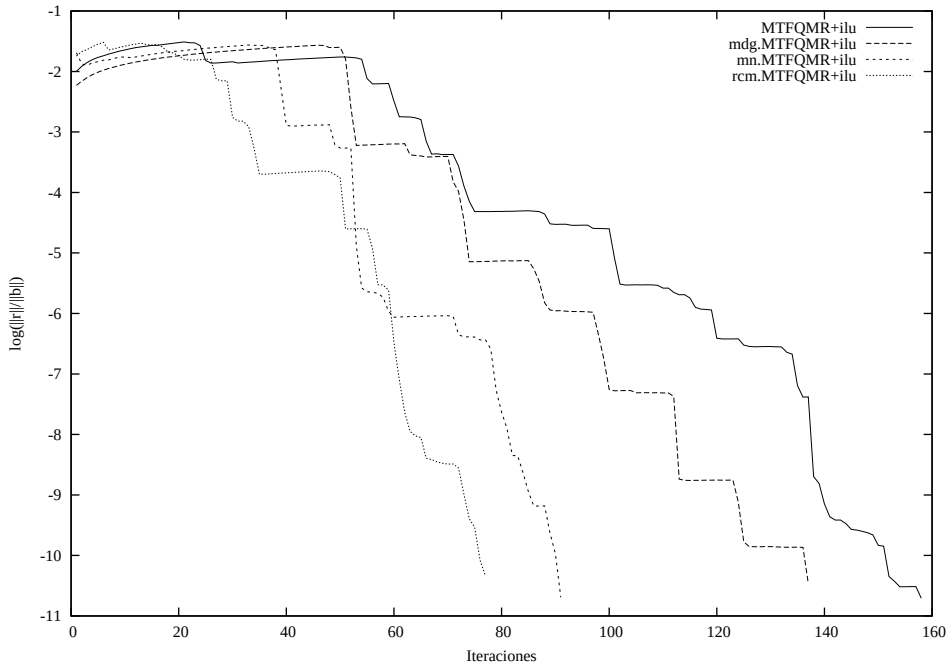

Figura 10.59: *Efecto de la reordenación en la convergencia del MTFQMR precondicionado con ILU(0) para convdifhor (3423 ecuaciones).*

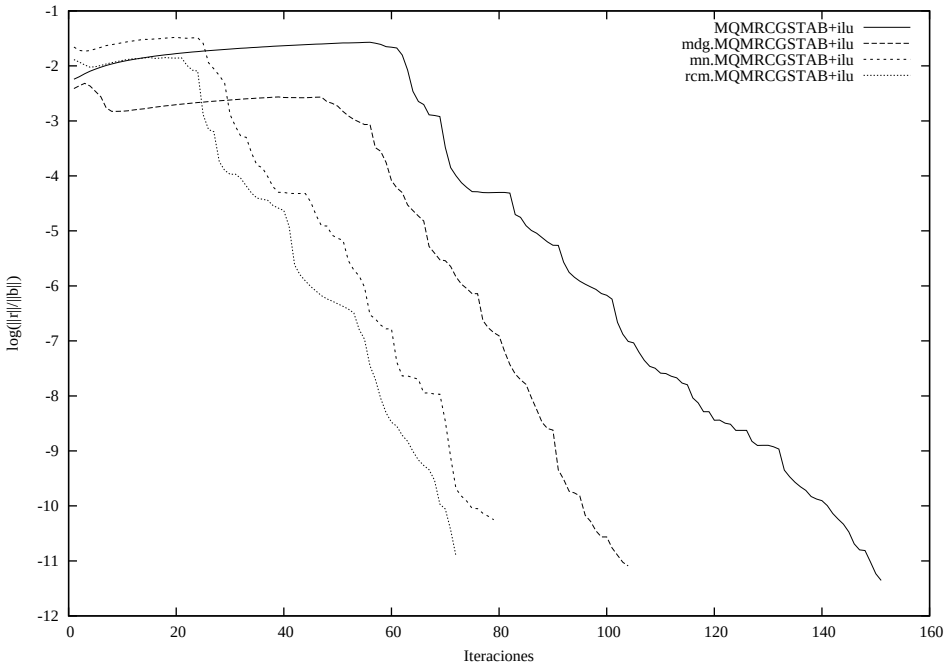

Figura 10.60: *Efecto de la reordenación en la convergencia del MQMRCGSTAB precondicionado con ILU(0) para convdifhor (3423 ecuaciones)*

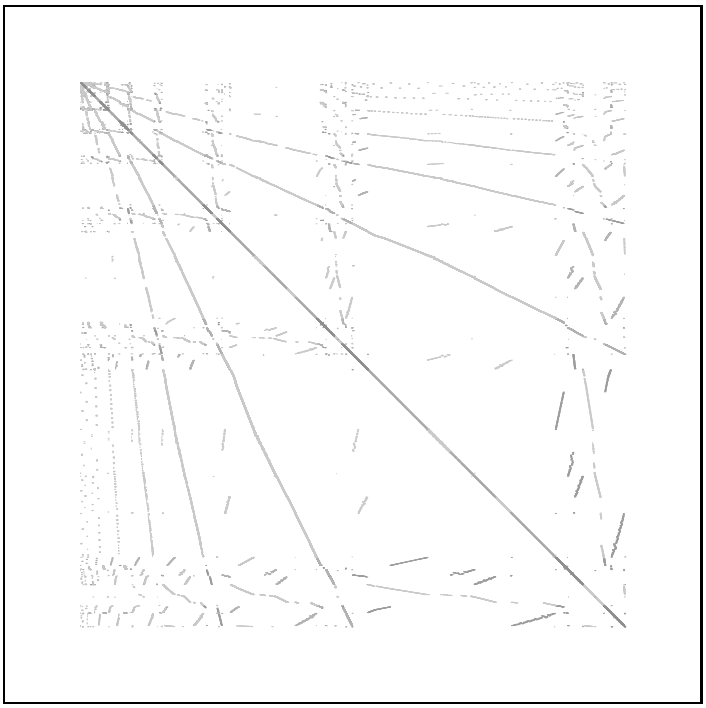

Figura 10.61: *Estructura sparse de la matriz convdifhor para* $n = 13190$

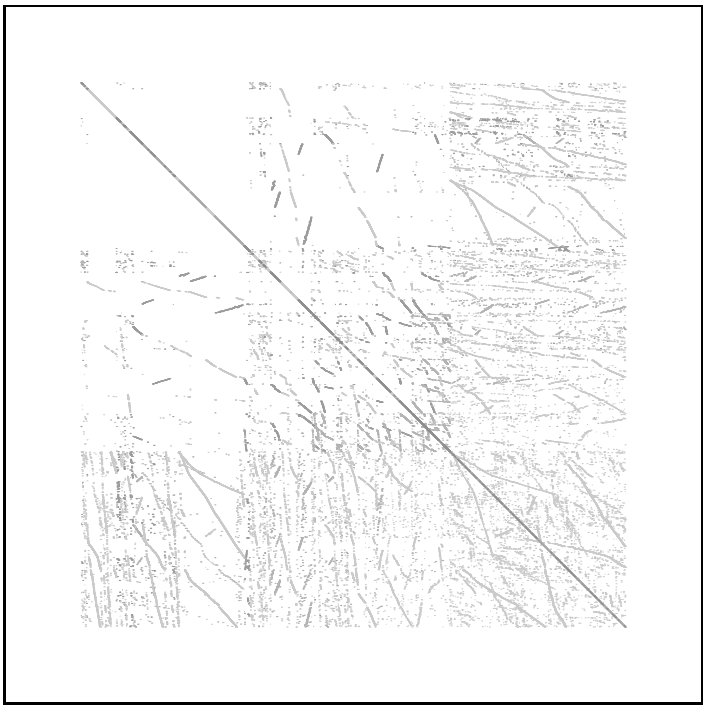

Figura 10.62: *Estructura sparse de la matriz convdifhor para* $n = 13190$ *reordenada con Grado Mínimo*

Tabla 10.7: *Ejemplo 9 (13190 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos con y sin precondicionamiento*

convdifhor(13190)		Jacobi	ILU	SSOR	Diagopt	MDG+ILU	RCM+ILU
VGMRES	k_{init}	1	1	1	1	1	1
	k_{top}	500	500	500	500	500	500
	n^{o}Iter.	-	501	-	-	257	83
	t(seg.)	>500	161.380	>500	>500	29.239	2.949
BiCGSTAB	n^{o}Iter.	4991	518	747	2981	248	62
	t(seg.)	25.760	5.819	7.839	17.450	2.639	1.220
QMRCGSTAB	n^{o}Iter.	5296	548	851	2929	258	63
	t(seg.)	35.38	6.989	12.24	17.969	3.149	1.330
MQMRCGSTAB	n^{o}Iter.	-	577	866	-	264	64
	t(seg.)	>500	99.110	341.710	>500	22.659	2.199

Para el último caso, la convergencia sólo se consigue para cuatro de los métodos de Krylov, VGMRES, BiCGSTAB, QMRCGSTAB, y para el MQMRCGSTAB, este último precondicionado con ILU(0) y SSOR, ya que con los precondicionadores de tipo diagonal la convergencia es lentísima, como se muestra en la tabla 10.7. Igual que en los casos anteriores cuando el sistema es reordenado, mejora la convergencia en número de iteraciones y tiempo de computación, siendo la reordenación más eficiente, nuevamente la de Cuthill-McKee Inverso.

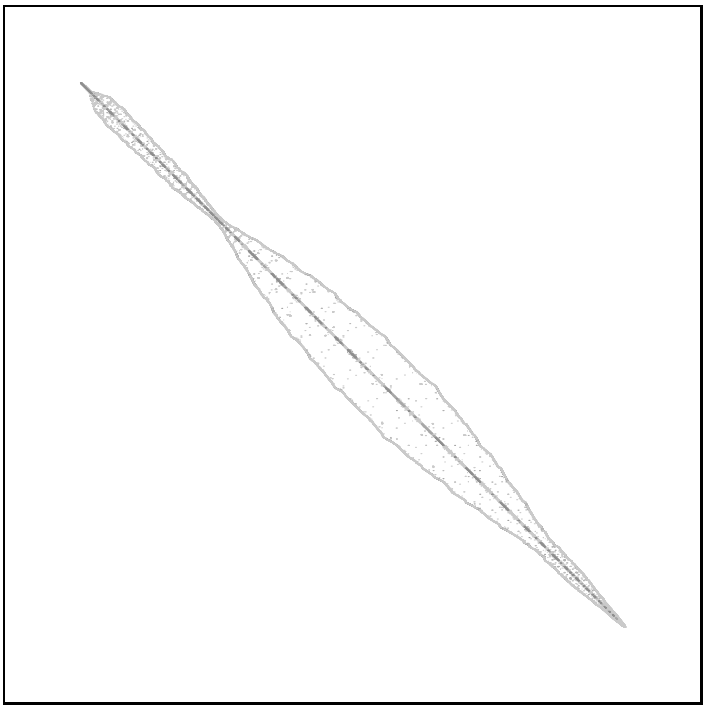

Figura 10.63: *Estructura sparse de la matriz convdifhor para* $n = 13190$ *reordenada con Cuthill-McKee Inverso*

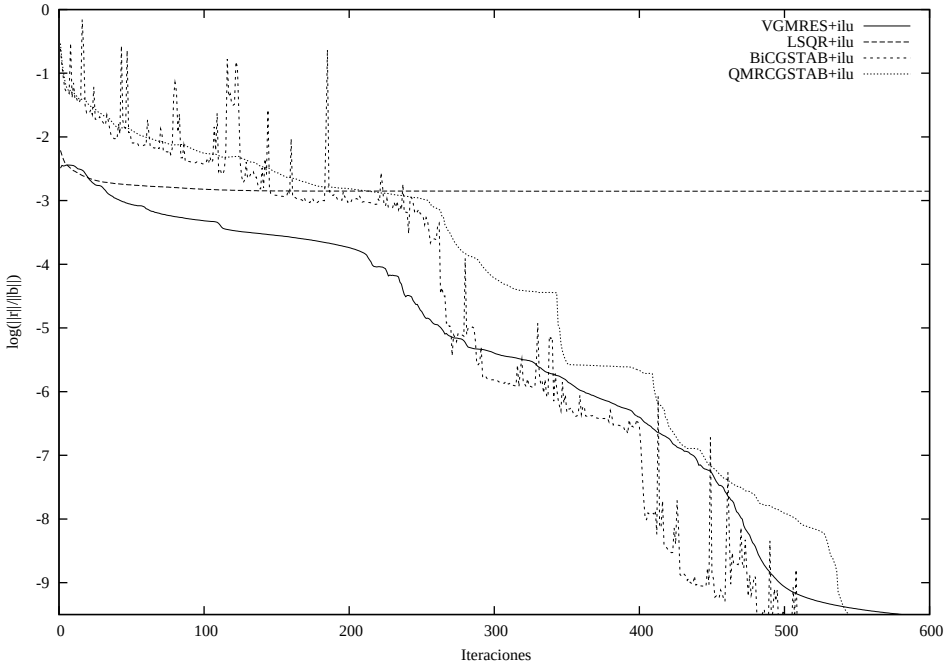

Figura 10.64: *Convergencia de distintos métodos de Krylov precondicionados con ILU(0) para convdifhor (13190 ecuaciones)*

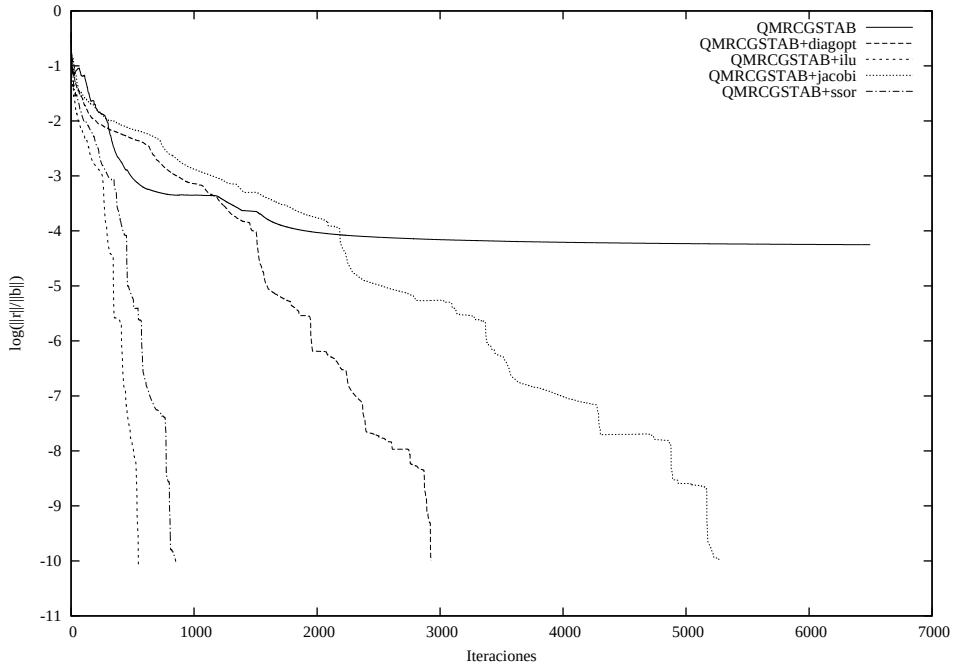

Figura 10.65: *Convergencia del QMRCGSTAB con diferentes precondicionadores para convdifhor (13190 ecuaciones)*

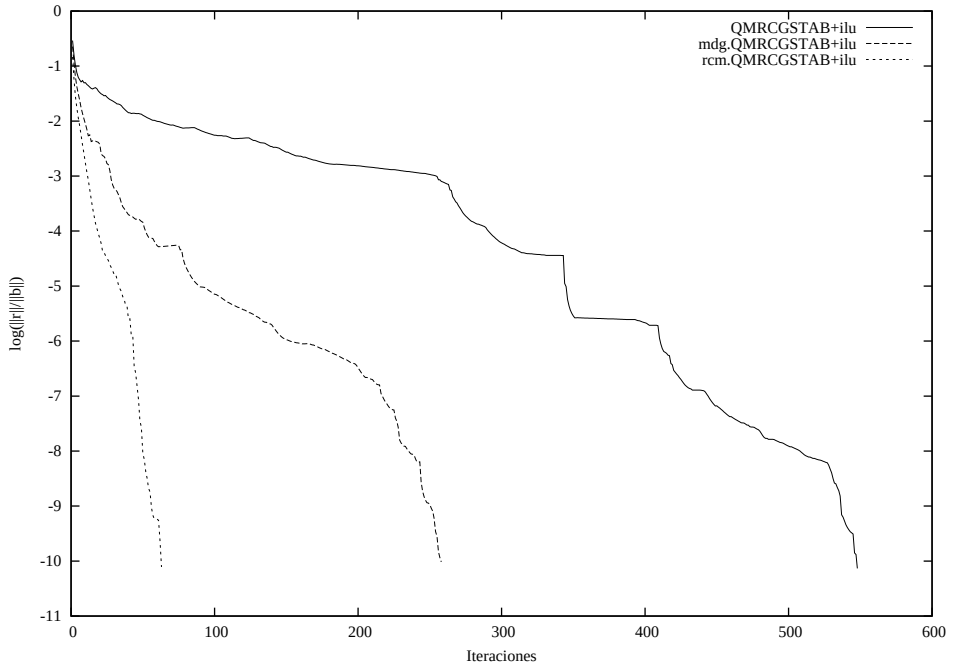

Figura 10.66: *Efecto de la reordenación en la convergencia del QMRCGSTAB precondicionado con ILU(0) para convdifhor (13190 ecuaciones)*

Las curvas representadas en la figura 10.64 muestran el comportamiento de los métodos de Krylov en este sistema precondicionado con ILU(0). El método LSQR precondicionado con ILU(0) se ve claramente como no llegua a converger en un número razonable de iteraciones. Por el contrario la convergencia de los otros métodos.Las Figuras 10.65 y 10.66 representan las curvas de convergencia del método QMRCGSTAB sin precondicionar y con dististos precondicionadores. Para el precondicionador ILU(0) la convergencia es mucho más rápida cuando el sistema es previamente reordenado. Este efecto es aquí más espectacular que los ejemplos anteriores. Por ejemplo, la reodenación con Cuthill-McKee Inverso reduce el número de iteraciones al 10 %.

Parece desprenderse de como conclusión preliminar, que la reordenación es más útil a medida que el condicionamiento del sistema es peor y el orden aumenta.

10.10. Ejemplo 10 (cuaref)

Se trata de un nuevo problema en $2-D$ de convección-difusión estacionario de un fluido sometido a un campo circular de velocidades, de ecuación $v_1\frac{\partial u}{\partial x} + v_2\frac{\partial u}{\partial y} - K\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$ sobre un dominio cuadrado Ω, con condiciones de tipo Dirichlet $u = 0$ en $x = 1$ y $u = 1$ en $x = 0$, y de Neumann, $\frac{\partial u}{\partial n} = 0$ en el resto de la frontera, siendo $K = \frac{\lambda}{c\rho}$, el coeficiente de difusión dado en $\left[\frac{m^2}{h}\right]$, y v_1, v_2, la componentes del campo de velocidades,

$$v_1 = C\,(y - 1/2)\left(x - x^2\right), \quad v_2 = C\,(1/2 - x)\left(y - y^2\right)$$

Se ha tomado $K = 1$ y para el coeficiente C de la velocidad un valor de 10^5.

Con las mallas correspondientes a distintas etapas de refinamiento, se obtienen, respectivamente, sistemas no simétricos de 1023, 4095 y 7520 ecuaciones. Las figuras 10.67-10.70, 10.72-10.75 y 10.77-10.80 muestran las estructuras *sparse* de las matrices con distintas reordenaciones, correspondientes a cada uno de estos sistemas de ecuaciones, respectivamente.

En este ejemplo, cuyos resultados se exponen en las tablas 10.8-10.10, para los tres sistemas de 1023, 4095 y 7520 ecuaciones, respectivamente, lo más significativo es que el precondicionador SSOR es el más efectivo en todos los casos en que se consigue la convergencia.

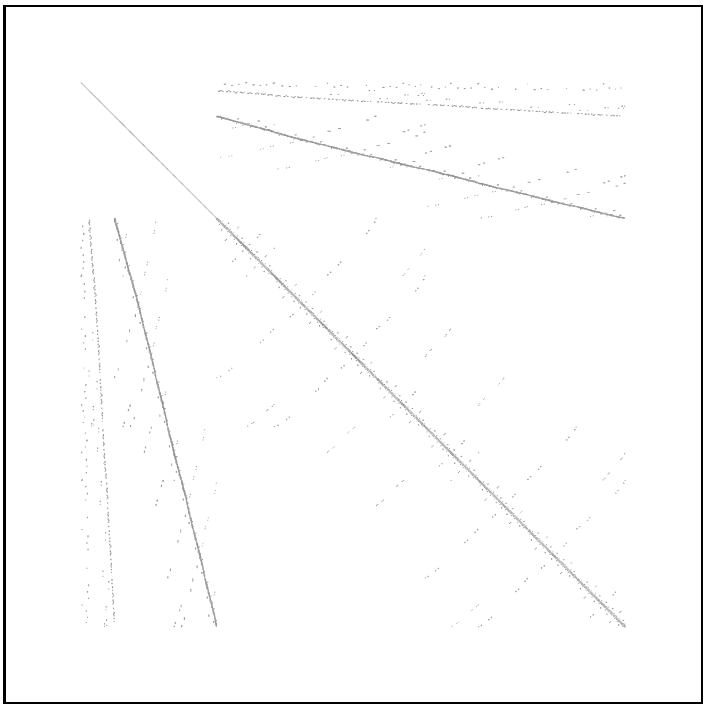

Figura 10.67: *Estructura sparse de la matriz cuaref para* $n = 1023$

Figura 10.68: *Estructura sparse de la matriz cuaref para* $n = 1023$ *reordenada con Grado Mínimo*

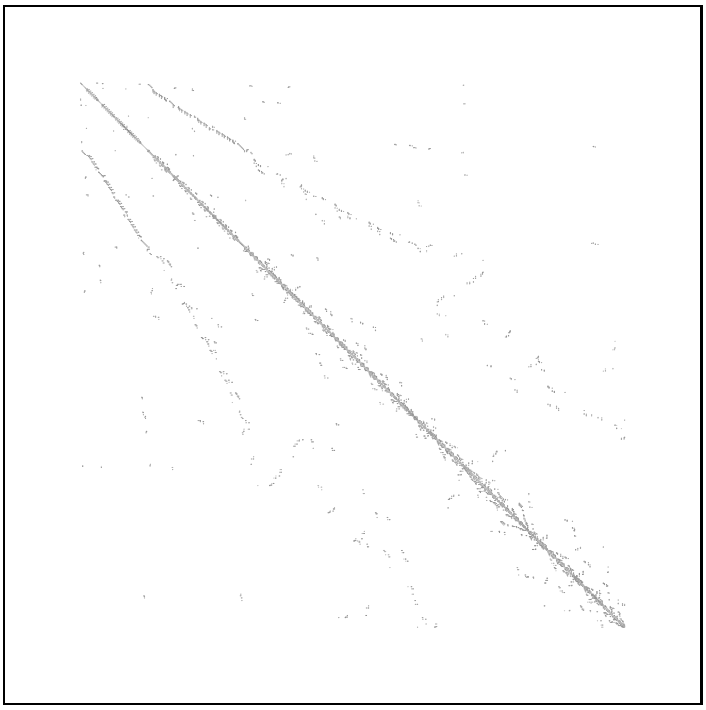

Figura 10.69: *Estructura sparse de la matriz cuaref para* $n = 1023$ *reordenada con Mínimo Vecino*

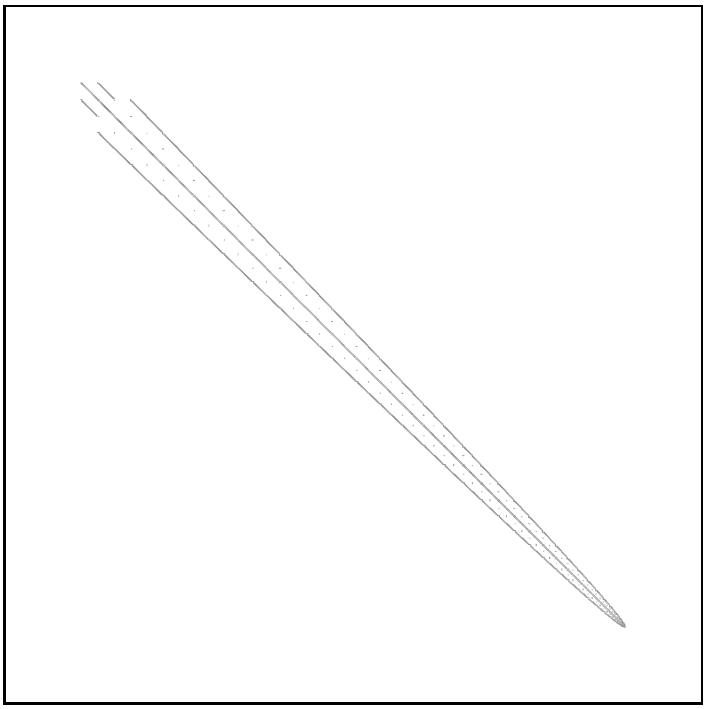

Figura 10.70: *Estructura sparse de la matriz cuaref para $n = 1023$ reordenada con Cuthill-McKee Inverso*

Tabla 10.8: *Ejemplo 10 (1023 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos con y sin precondicionamiento*

cuaref(1023)		SP	Jacobi	ILU	SSOR	Diagopt	MN+SSOR	RCM+SSOR
VGMRES	k_{init}	1	1	1	1	1	1	no conv.
	k_{top}	100	200	200	200	500	200	
	nºIter.	169	206	171	141	428	116	
	t(seg.)	7.019	4.569	0.909	0.560	7.880	0.359	
BiCG	nºIter.	>1023	745	189	146	729	no conv.	603
	t(seg.)	-	0.140	9.859	7.629	0.130		28.529
CGS	nºIter.	>1023	>1023	163	114	>1023	>1023	>1023
	t(seg.)	-	-	0.059	0.039	-	-	-
BiCGSTAB	nºIter.	655	527	133	107	497	97	202
	t(seg.)	0.070	0.079	0.050	0.050	0.079	0.029	0.060
TFQMR	nºIter.	>1023	>1023	317	198	>1023	>1023	>1023
	t(seg.)	-	-	0.159	0.100	-	-	-
QMRCGSTAB	nºIter.	690	551	138	108	500	97	219
	t(seg.)	0.140	0.140	0.069	0.050	0.120	0.039	0.079
MTFQMR	nºIter.	>1023	>1023	500	124	>1023	96	500
	t(seg.)	-	-	7.119	0.369	-	0.240	7.099
MQMRCGSTAB	nºIter.	690	534	133	108	493	100	387
	t(seg.)	11.29	6.240	0.450	0.280	5.329	0.25	3.549

Para el sistema de 1023 ecuaciones, los métodos QMR, CGN y LSQR y MQMR no convergen en ningún caso para un número de iteraciones menor que la dimensión, mientras que el algoritmo BiCG no converge si no se aplica alguna técnica de precondicionamiento. Los algoritmos CGS, TFQMR y MTFQMR sólo convergen precondicionados con ILU(0) y SSOR. Las diferentes técnicas de reordenación del sistema cuando se utiliza algún método de Krylov precondicionado con SSOR, si bien mejoran la convergencia en algunos casos, ocurre

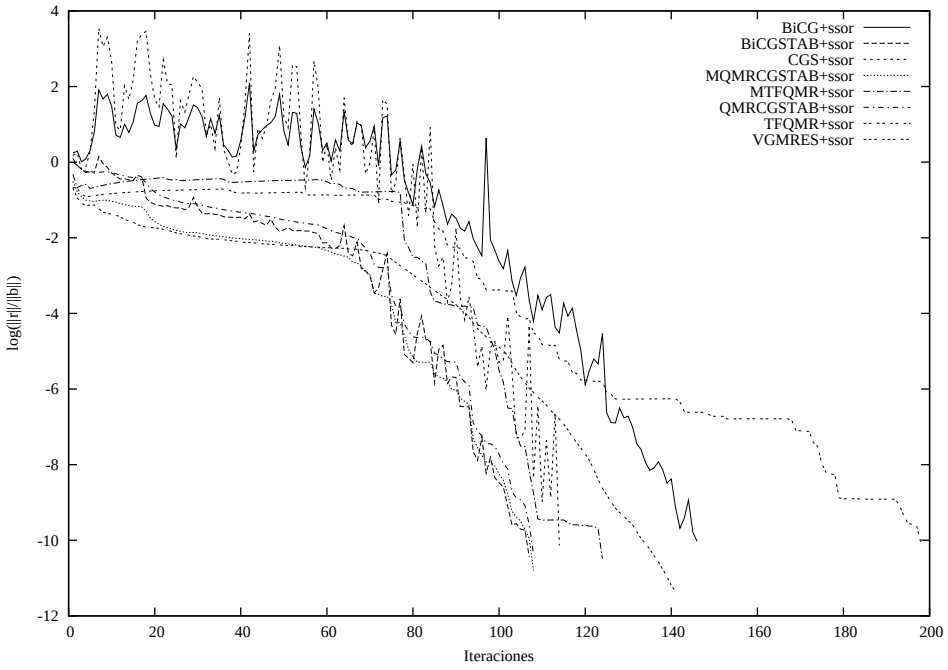

Figura 10.71: *Convergencia de diferentes métodos de Krylov precondicionados con SSOR para cuaref (1023 ecuaciones)*

que en otros no funcionan, por ejemplo, la renumeración con el algoritmo de Grado Mínimo sólo converge para al método VGMRES y además resulta más caro que cuando no está reordenado. Para la renumeración con Mínimo Vecino, los métodos BiCG, CGS y TFQMR no convergen, sin embargo en el resto de los métodos acelera la convergencia, tanto en iteraciones como en tiempo de computación. La reordenación con el algoritmo de Cuthill-McKee Inverso es en este caso peor que la anterior, ya que si bien logra la convergencia para el BiCG y TFQMR, ésta es mas costosa que sin la reordenación. Además no converge el VGMRES y el resto de métodos aumentan el número de iteraciones y tiempo.

La figura 10.71 representa las curvas correspondientes a la convergencia de los distintos métodos preconcidionados con SSOR. Los métodos más eficaces vuelven a ser BiCGSTAB, QMRCGSTAB y MQMRCGSTAB, estos dos últimos con curvas algo más suaves que el primero.

Los sistemas correspondientes a los otras dos mallas, de 4095 y 7520 ecuaciones respectivamente, sólo convergen para cuatro de los métodos de Krylov estudiados (ver tablas 10.9 y 10.10). El comportamiento es similar al caso anterior en cuanto a que el precondicionador SSOR es ahora el mejor. En lo que se refiere a la reordenación, se presenta el mismo comportamiento para la discretización correspondiente a 4095 ecuaciones. En cambio, para el sistema de 7520 ecuaciones es ahora, con excepción del método VGMRES, la reordenación con Cuthill-McKee Inverso la que produce menos iteraciones y consume menos tiempo.

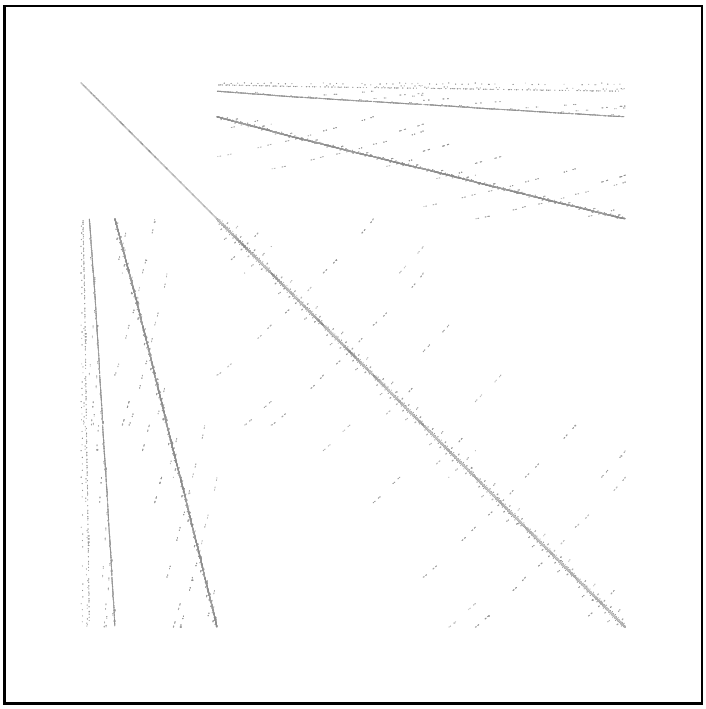

Figura 10.72: *Estructura sparse de la matriz cuaref para* $n = 4095$

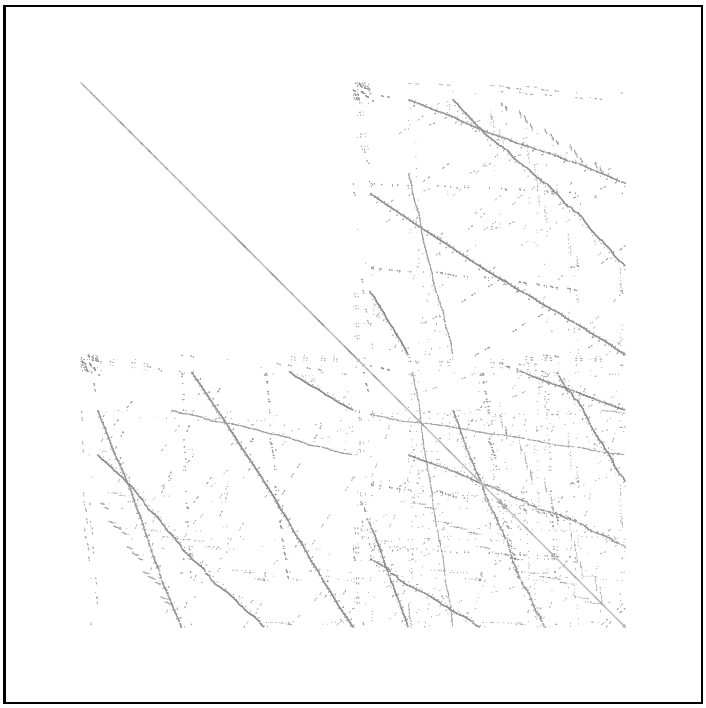

Figura 10.73: *Estructura sparse de la matriz cuaref para* $n = 4095$ *reordenada con Grado Mínimo*

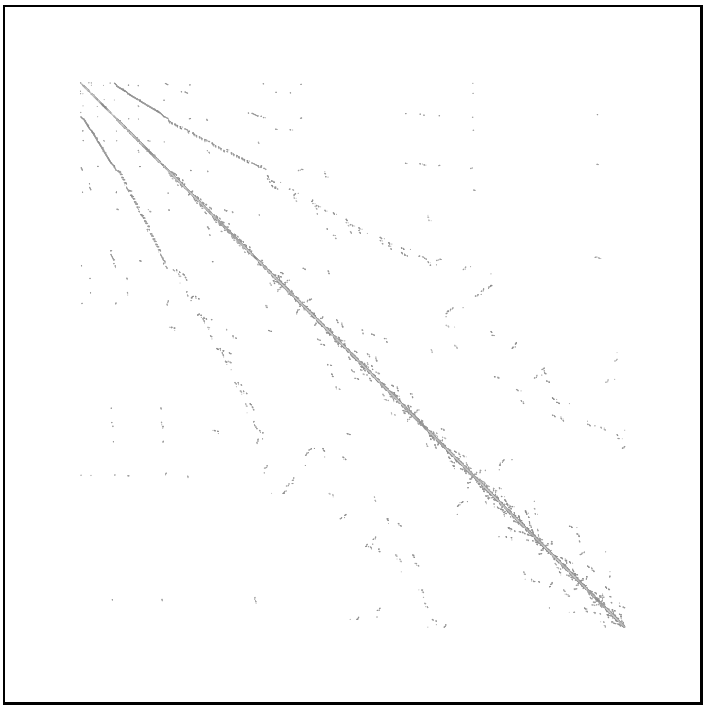

Figura 10.74: *Estructura sparse de la matriz cuaref para* $n = 4095$ *reordenada con Mínimo Vecino*

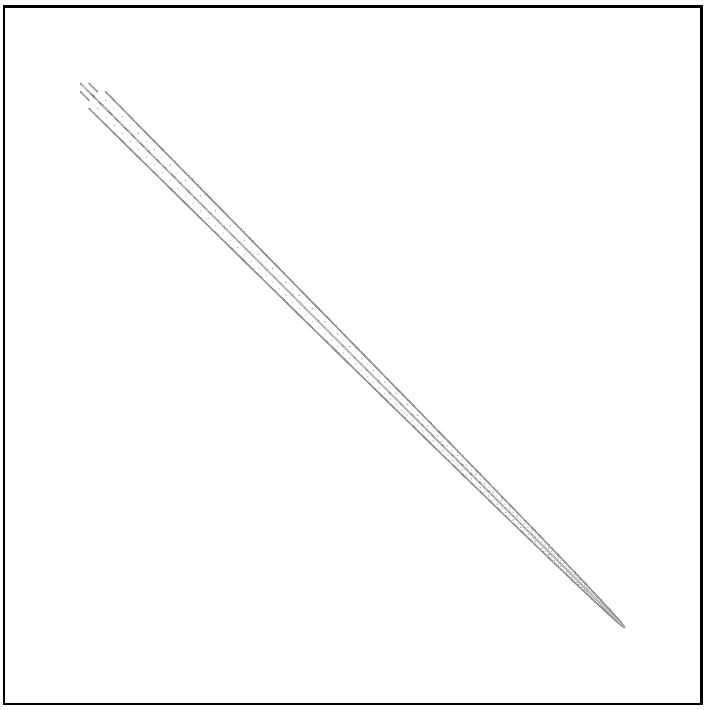

Figura 10.75: *Estructura sparse de la matriz cuaref para* $n = 4095$ *reordenada con Cuthill-McKee Inverso*

Las figuras 10.75-10.81, muestran respectivamente, las curvas de convergencia para los cuatro métodos precondicionados con SSOR y observamos el mismo comportamiento que en el anterior sistema.

Tabla 10.9: *Ejemplo 10 (4095 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos con y sin precondicionamiento*

cuaref (4095)		SP	Jacobi	ILU	SSOR	Diagopt	MN+SSOR	RCM+SSOR
VGMRES	k_{init}	1	no conv.	1	1	no conv.	1	no conv.
	k_{top}	500		500	500		500	
	nº Iter.	513		326	271		197	
	t(seg.)	202.029		14.529	10.050		4.949	
BiCGSTAB	nº Iter.	2275	1415	266	218	1123	157	425
	t(seg.)	2.049	2.049	0.75	0.519	1.420	0.319	0.819
QMRCGSTAB	nº Iter.	2412	1558	270	225	1177	163	470
	t(seg.)	3.910	2.779	0.830	0.590	2.179	0.639	1.090
MQMRCGSTAB	nº Iter.	-	1418	278	222	1137	182	-
	t(seg.)	>500	354.190	7.069	4.719	184,430	3.189	>500

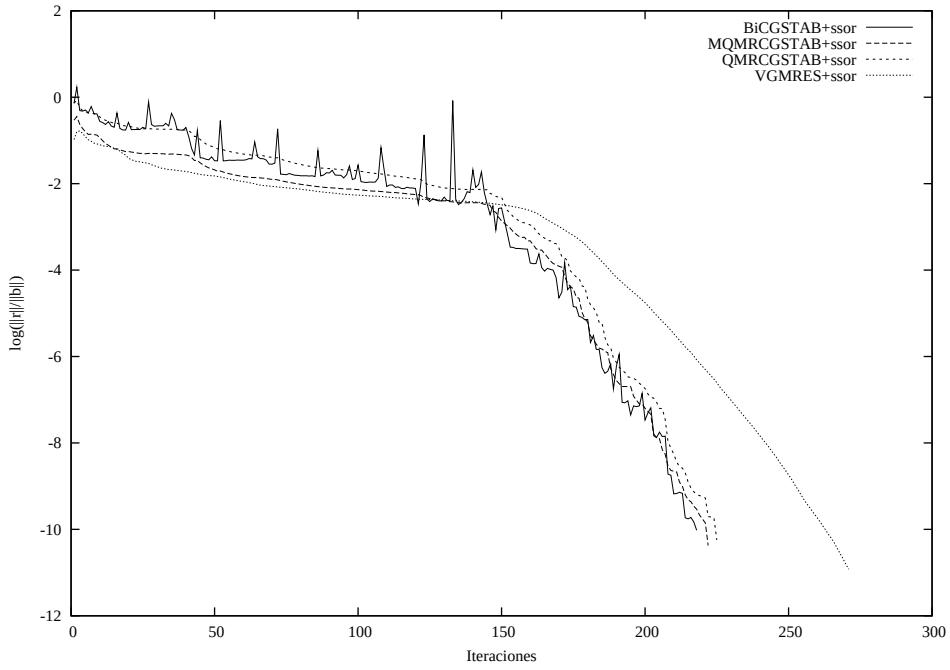

Figura 10.76: *Convergencia de diferentes métodos de Krylov precondicionados con SSOR para cuaref (4095 ecuaciones)*

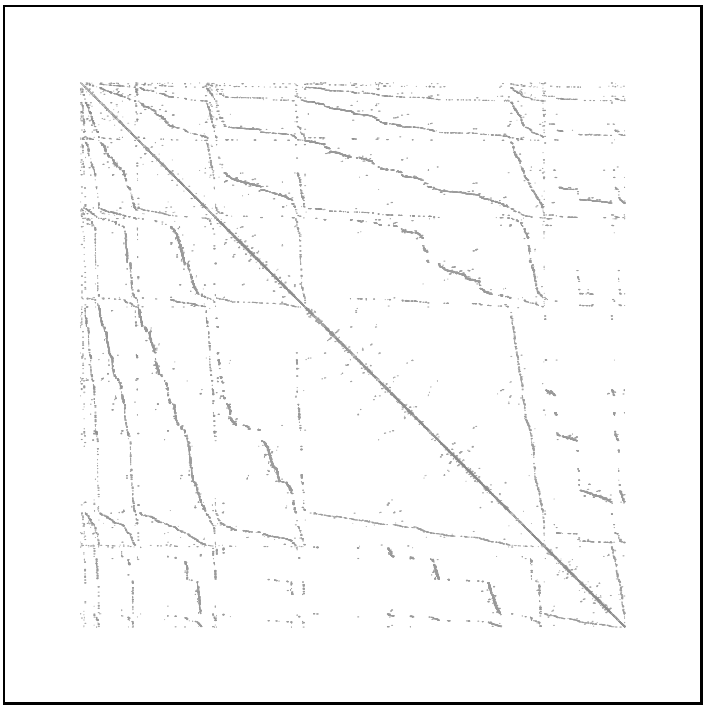

Figura 10.77: *Estructura sparse de la matriz cuaref para* $n = 7520$

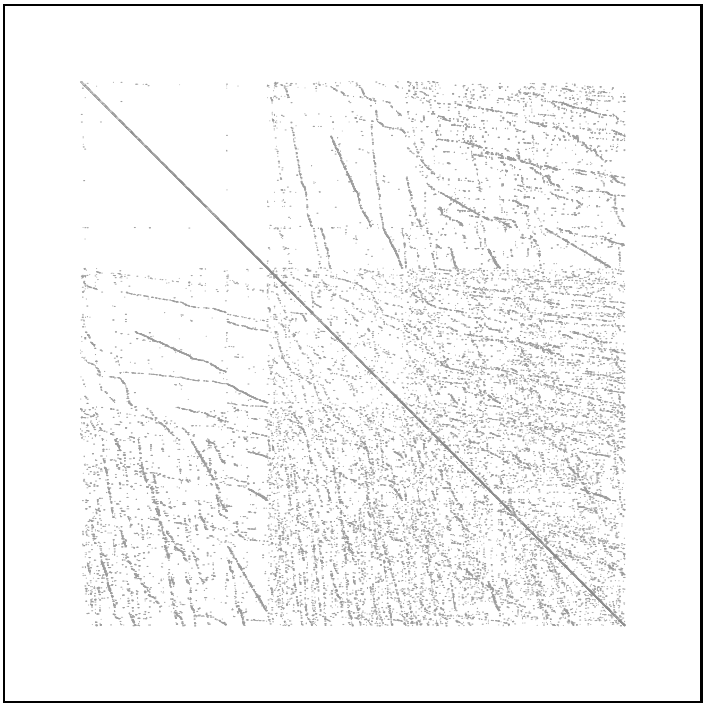

Figura 10.78: *Estructura sparse de la matriz cuaref para* $n = 7520$ *reordenada con Grado Mínimo*

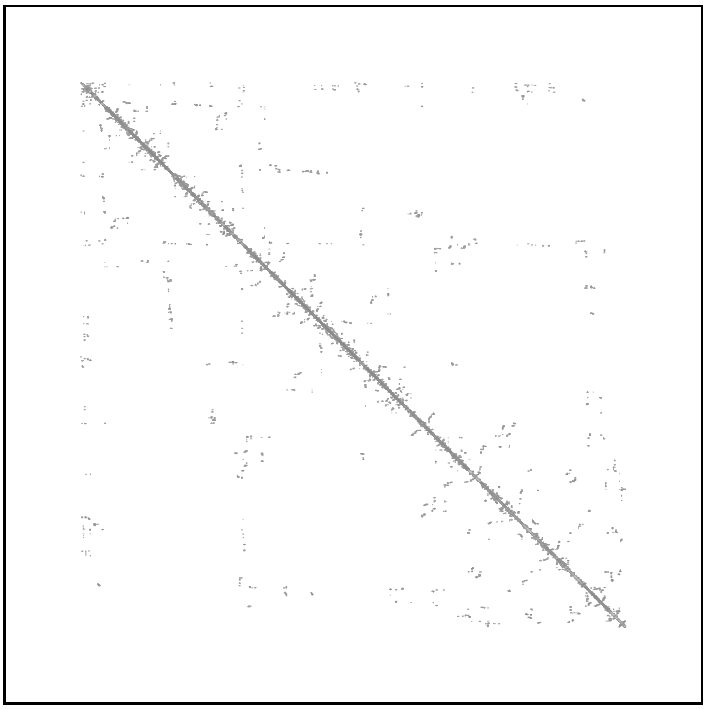

Figura 10.79: *Estructura sparse de la matriz cuaref para* $n = 7520$ *reordenada con Mínimo Vecino*

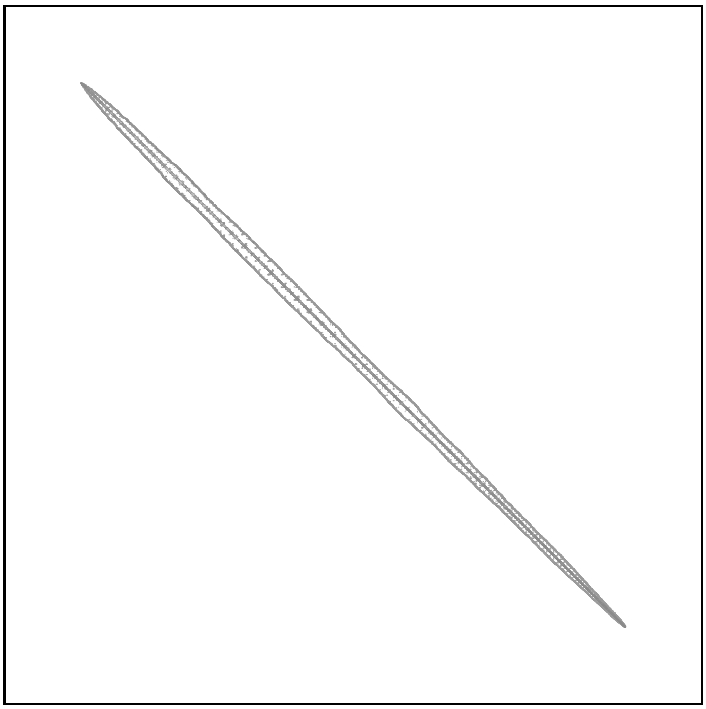

Figura 10.80: *Estructura sparse de la matriz cuaref para* $n = 7520$ *reordenada con Cuthill-McKee Inverso*

Tabla 10.10: *Ejemplo 10 (7520ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos con y sin precondicionamiento*

cuaref (7520)		SP	ILU	SSOR	MN+SSOR	RCM+SSOR
VGMRES	k_{init}	1	1	1	1	1
	k_{top}	500	500	500	500	1000
	n^{o}Iter.	516	354	351	160	532
	t(seg.)	423.889	30.309	31.780	5.869	73.349
BiCGSTAB	n^{o}Iter.	1726	371	335	190	132
	t(seg.)	4.250	2.210	1.860	1.080	0.720
QMRCGSTAB	n^{o}Iter.	1847	375	374	169	121
	t(seg.)	6.209	2.589	2.319	1.040	0.720
MQMRCGSTAB	n^{o}Iter.	-	372	290	128	117
	t(seg.)	$>$500	23.699	15.709	3.190	2.740

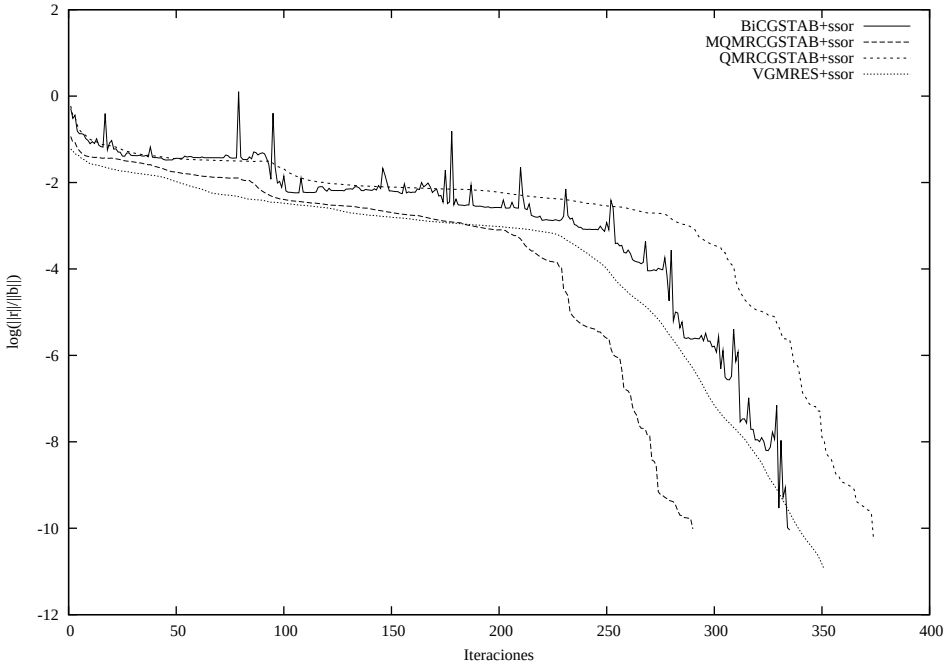

Figura 10.81: *Convergencia de diferentes métodos de Krylov precondicionados con SSOR para cuaref (7520 ecuaciones)*

10.11. Ejemplo 11 (isla)

Este problema se trata en dos etapas diferenciadas. La primera consiste en la construcción de un campo de velocidades a partir de datos observados y el segundo es un problema de convección-difusión (en el que nos vamos a centrar) definido en un dominio Ω bidimensional de frontera Γ, definido por la ecuación,

$$\frac{\partial c}{\partial t} + \mathbf{v} \cdot \nabla c - \nabla \left(K \nabla c \right) = f \qquad \text{en } \Omega$$

siendo $c = c(\mathbf{x}, t)$ la concentración de un elemento de fluido en movimiento; ésta depende de su vector de posición $\mathbf{x} = (x_1, x_2)$, y del tiempo t. El fluido que transporta la magnitud u, posee un campo de velocidades en régimen estacionario, $\mathbf{v} = (v_1, v_2)$. Se considera el estudio del modelo lineal, donde $K = K(\mathbf{x})$ es el coeficiente de difusión y $f = f(\mathbf{x}, t)$ respresenta las fuentes externas. Suponemos, además conocido el valor inicial $c(\mathbf{x}, 0) = c_0(\mathbf{x})$ en Ω, y unas condiciones de contorno en la frontera de tipo Dirichlet y de tipo mixta,

$$c = C(\mathbf{x}) \qquad \text{en } \Gamma_1$$

$$-K(\mathbf{x}) \frac{\partial c}{\partial \mathbf{n}} = h(\mathbf{x})\ c - G(\mathbf{x}) \qquad \text{en } \Gamma_2$$

El problema se ha resuelto utilizando el método de las características para la discretización temporal y la técnica estándar de elementos finitos. Se han considerado dos etapas del proceso de refinamientos que han dado lugar a dos sistemas no simétricos de 1220 y 12666 ecuaciones, respectivamente. Las figuras 10.82-10.84 y 10.88-10.90 muestran las estructuras *sparse* de las matrices con distintas reordenaciones, correspondientes a ambos sistemas de ecuaciones, respectivamente.

Los dos sistemas no simétricos, de 1220 y 12666 ecuaciones a que han dado lugar este ejemplo, se han analizado para los distintos métodos de Krylov estudiados y se han aplicado técnicas de precondicionamiento y reordenación de los sistemas como resulta reflejado en las tablas 10.11 y 10.12. Observamos que para el primer sistema de 1220 ecuaciones, la convergencia sin precondicionamiento sólo se consigue con los métodos VGMRES y BiCG. Para los métodos QMR, CGN, LSQR y MQMR, la convergencia no se consigue ni siquiera precondicionando o reordenando previamente el sistema. Por otra parte, el algoritmo BiCG sigue funcionando lentamente con los precondicionadores ILU(0) y SSOR y los métodos CGS, TFQMR y MTFQMR sólo convergen precondicionados con ILU(0).

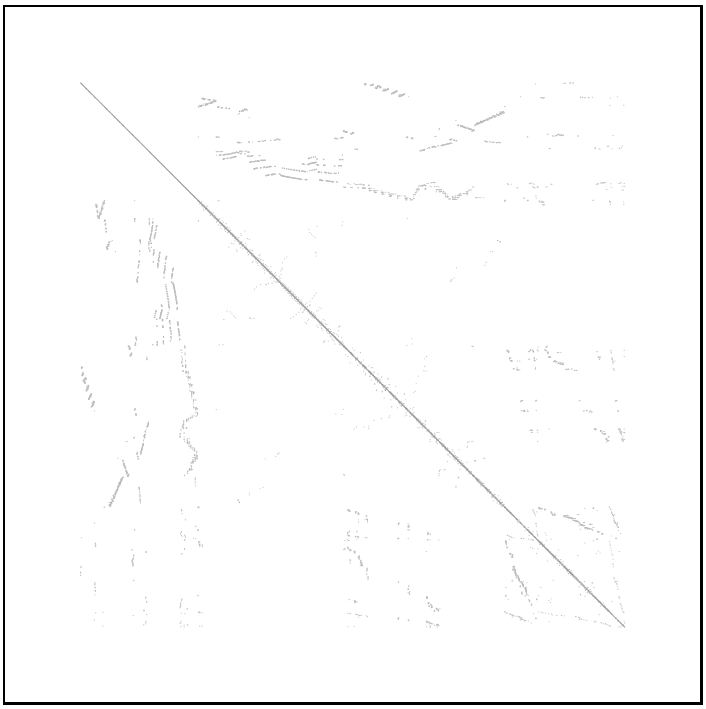

Figura 10.82: *Estructura sparse de la matriz isla para* $n = 1220$

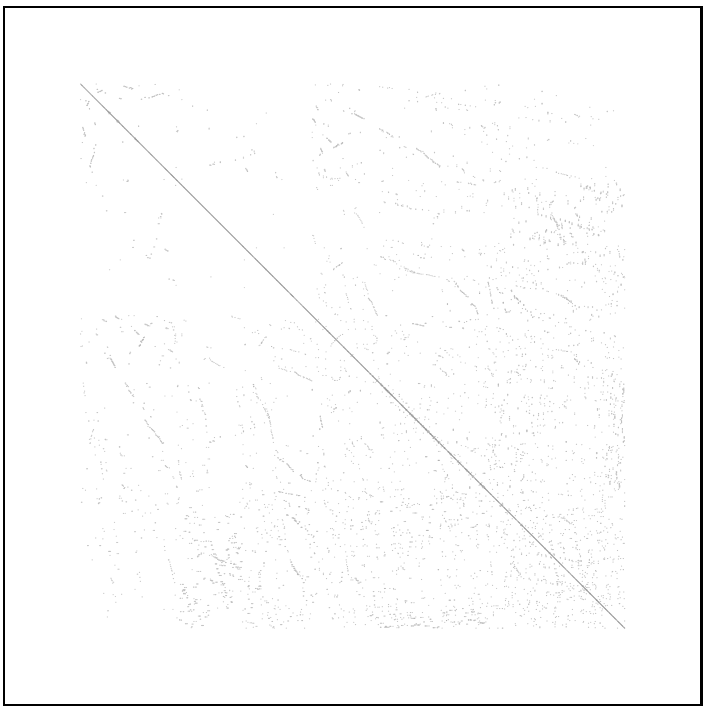

Figura 10.83: *Estructura sparse de la matriz isla para* $n = 1220$ *reordenada con Grado Mínimo*

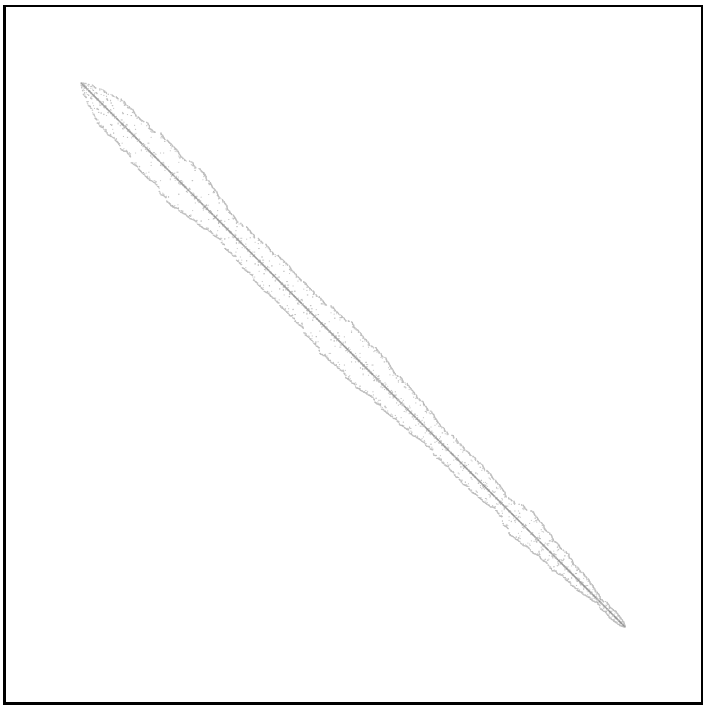

Figura 10.84: *Estructura sparse de la matriz isla para $n = 1220$ reordenada con Cuthill-McKee Inverso*

Tabla 10.11: *Ejemplo 11 (1220 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos con y sin precondicionamiento*

isla1220		SP	Jacobi	ILU	SSOR	Diagopt	MDG+ILU	RCM+ILU
VGMRES	k_{init}	1	1	1	1	1	1	1
	k_{top}	100	500	100	100	500	100	100
	n^{o}Iter.	110	207	69	91	211	54	37
	t(seg.)	1.37	1.55	0.109	0.360	1.640	0.060	0.039
BiCG	n^{o}Iter.	547	296	no conv.	no conv.	282	68	42
	t(seg.)	0.100	0.119			0.119	4.459	2.910
CGS	n^{o}Iter.	no conv.	>1220	60	>1220	>1220	60	31
	t(seg.)		-	0.030	-	-	0.030	0.019
BiCGSTAB	n^{o}Iter.	no conv.	166	51	65	159	37	26
	t(seg.)		0.060	0.029	0.039	0.060	0.020	0.010
TFQMR	n^{o}Iter.	>1220	>1220	92	>1220	>1220	61	37
	t(seg.)	-	-	0.059	-	-	0.039	0.019
QMRCGSTAB	n^{o}Iter.	no conv.	178	51	67	174	38	27
	t(seg.)		0.060	0.089	0.050	0.060	0.019	0.019
MQMR	n^{o}Iter.	>1220	>1220	>1220	>1220	>1220	67	41
	t(seg.)	-	-	-	-	-	4.470	2.910
MTFQMR	n^{o}Iter.	>1220	>1220	61	no conv.	>1220	42	30
	t(seg.)	-	-	0.129		-	0.060	0.030
MQMRCGSTAB	n^{o}Iter.	no conv.	162	51	50	174	37	27
	t(seg.)		0.899	0.089	0.119	1.040	0.050	0.030

En los casos en que se consigue la convergencia, el mejor precondicionador vuelve a ser el ILU(0). Asimismo, la utilización de alguna técnica de reordenación no sólo la mejora, sino que en casos como los métodos BiCG y MQMR consiguen converger. La renumeración con Cuthill-McKee Inverso es la más efectiva. Sin embargo, aún reordenando, la convergencia no se consigue con los métodos CGN, LSQR y QMR.

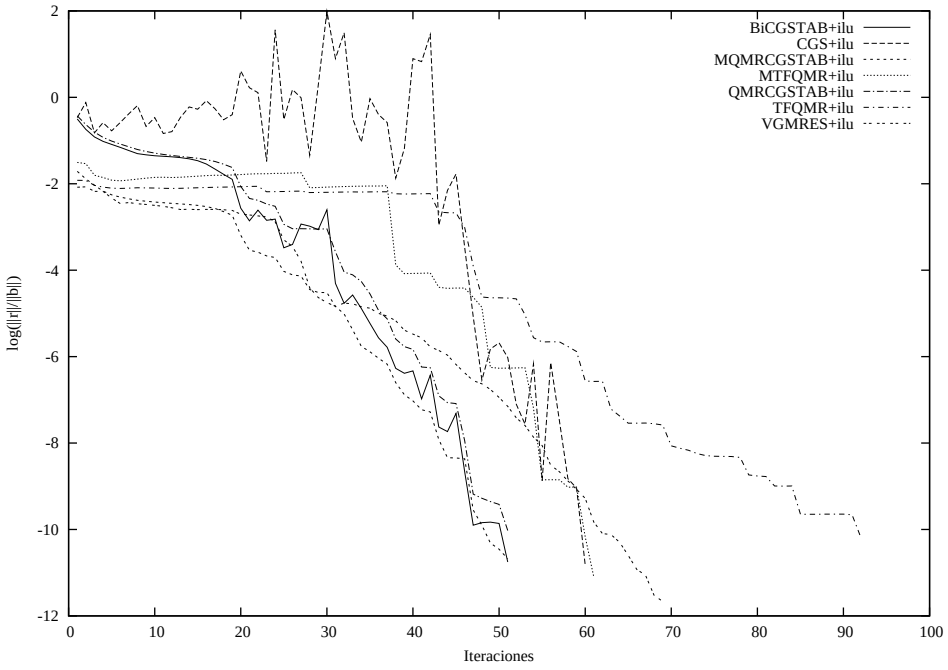

Figura 10.85: *Convergencia de diferentes métodos de Krylov precondicionados con ILU(0) para isla (1220 ecuaciones)*

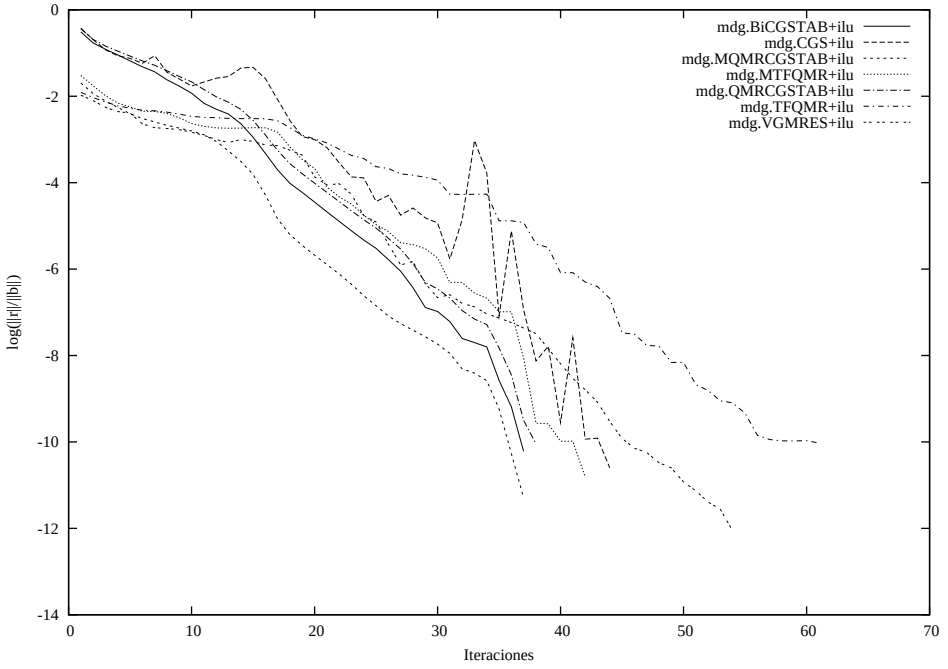

Figura 10.86: *Convergencia de diferentes métodos de Krylov precondicionados con ILU(0) después de reordenar con Grado Mínimo para isla (1220 ecuaciones)*

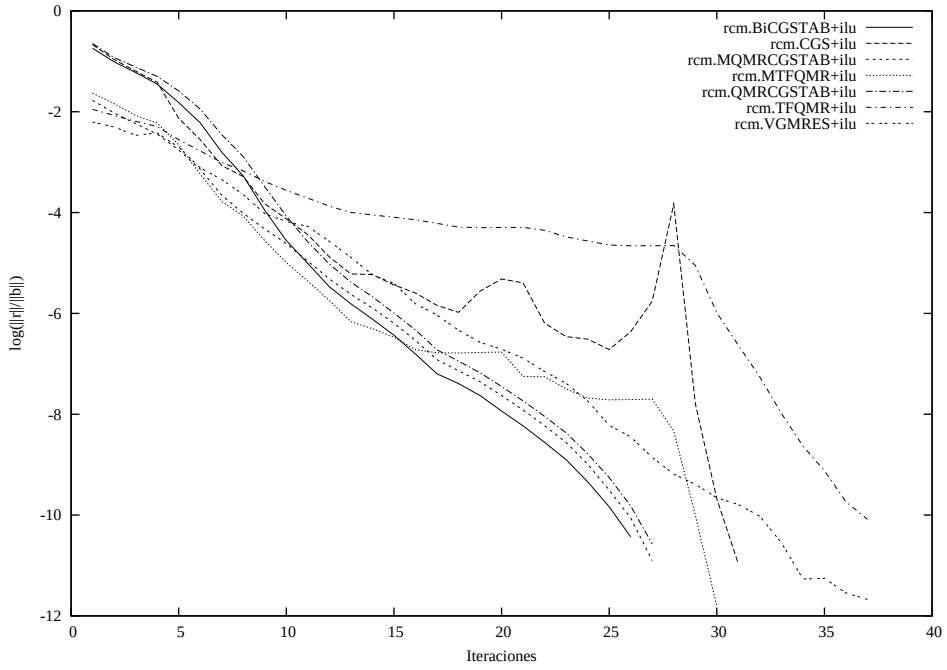

Figura 10.87: *Convergencia de diferentes métodos de Krylov precondicionados con ILU(0) después de reordenar con Cuthill McKee Inverso para isla (1220 ecuaciones)*

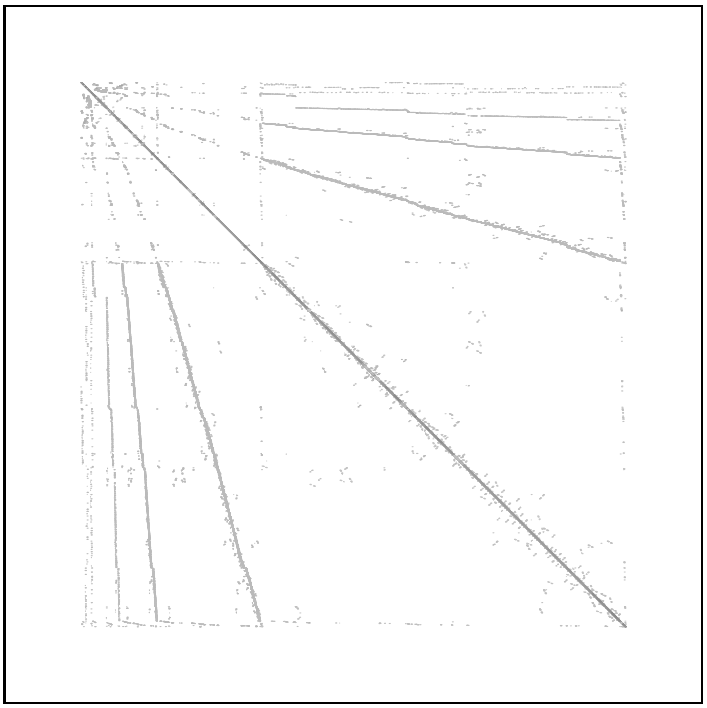

Figura 10.88: *Estructura sparse de la matriz isla para* $n = 12666$

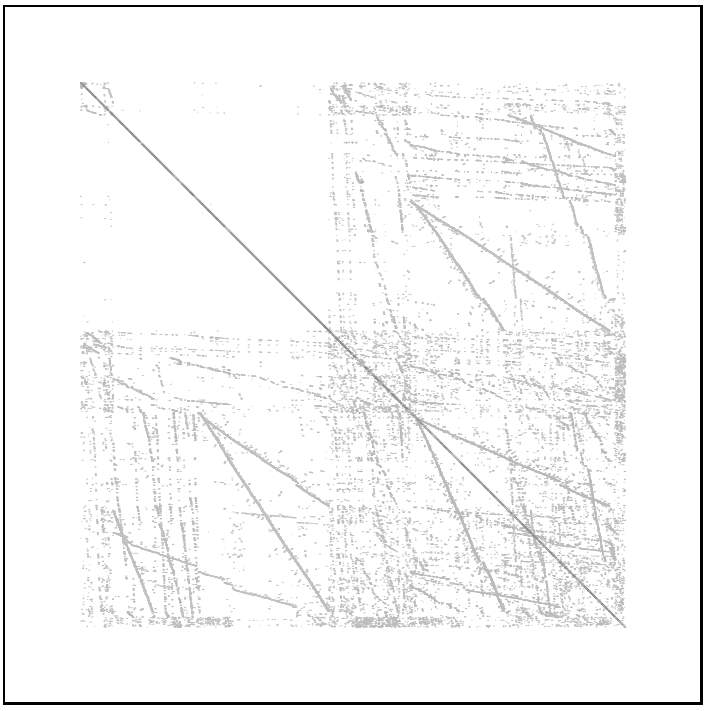

Figura 10.89: *Estructura sparse de la matriz isla para* $n = 12666$ *reordenada con Grado Mínimo*

Tabla 10.12: *Ejemplo 11 (12666 ecuaciones): número de iteraciones y tiempo de CPU (en segundos) para los distintos métodos sin precondicionar y con distintos precondicionadores*

isla(12666)		SP	ILU	SSOR	MDG+ILU	RCM+ILU
VGMRES	k_{init}	1	1	no conv.	1	1
	k_{top}	100	500		500	500
	n^{o}Iter.	140	296		119	58
	t(seg.)	64.589	37.290		5.779	1.639
CGS	n^{o}Iter.	no conv.	no conv.	no conv.	138	43
	t(seg.)				2.309	1.029
BiCGSTAB	n^{o}Iter.	no conv.	216	284	68	42
	t(seg.)		2.909	3.289	1.460	1.009
TFQMR	n^{o}Iter.	>12666	>12666	no conv.	197	124
	t(seg.)	-	-		3.769	2.420
QMRCGSTAB	n^{o}Iter.	>12666	249	303	76	43
	t(seg.)	-	3.500	3.850	1.600	1.080
MTFQMR	n^{o}Iter.	-	-	no conv.	149	41
	t(seg.)	>500	>500		8.019	1.310
MQMRCGSTAB	n^{o}Iter.	no conv.	242	336	77	39
	t(seg.)		18.509	32.099	2.860	1.230

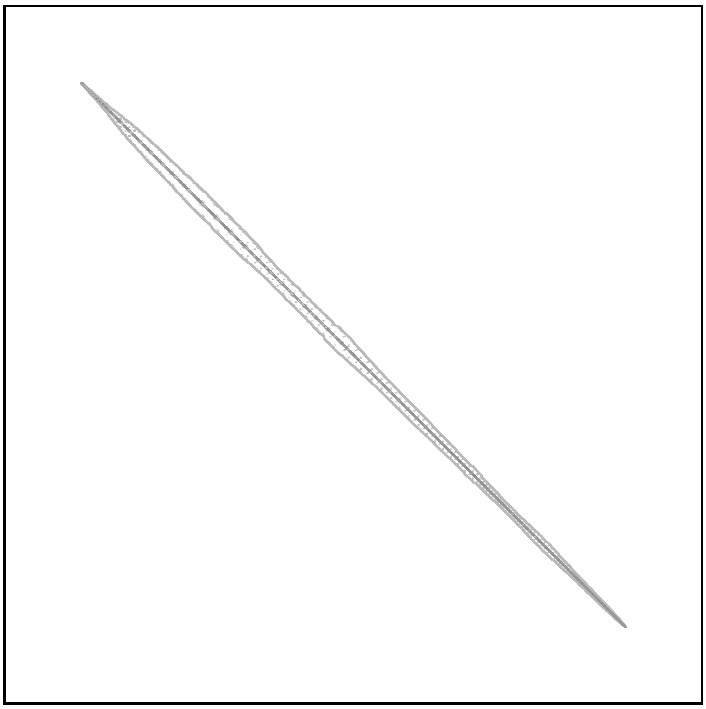

Figura 10.90: *Estructura sparse de la matriz isla para* $n = 12666$ *reordenada con Cuthill-McKee Inverso*

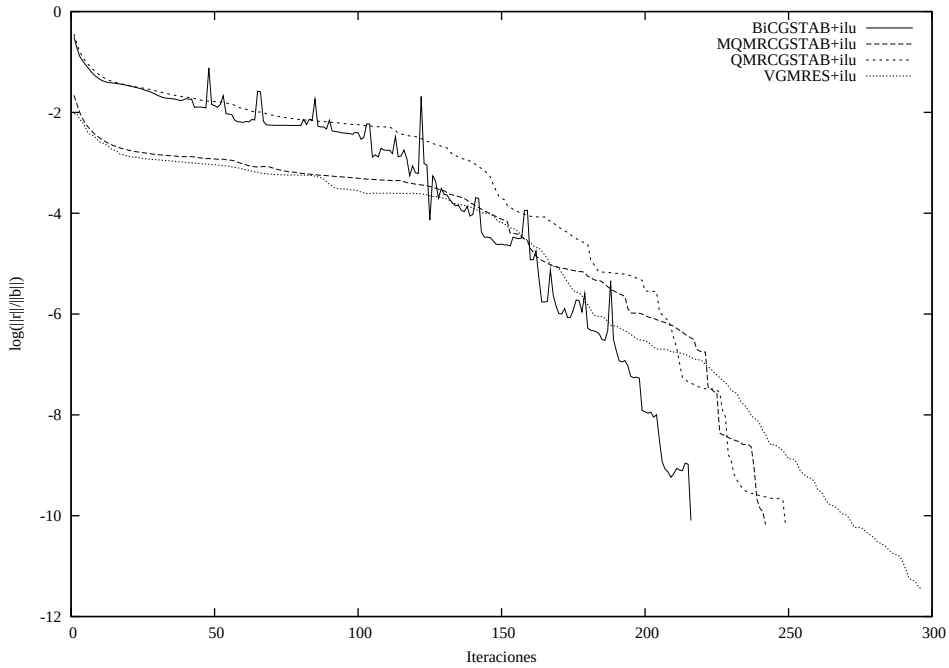

Figura 10.91: *Convergencia de diferentes métodos de Krylov precondicionados con ILU(0) para isla (12666 ecuaciones)*

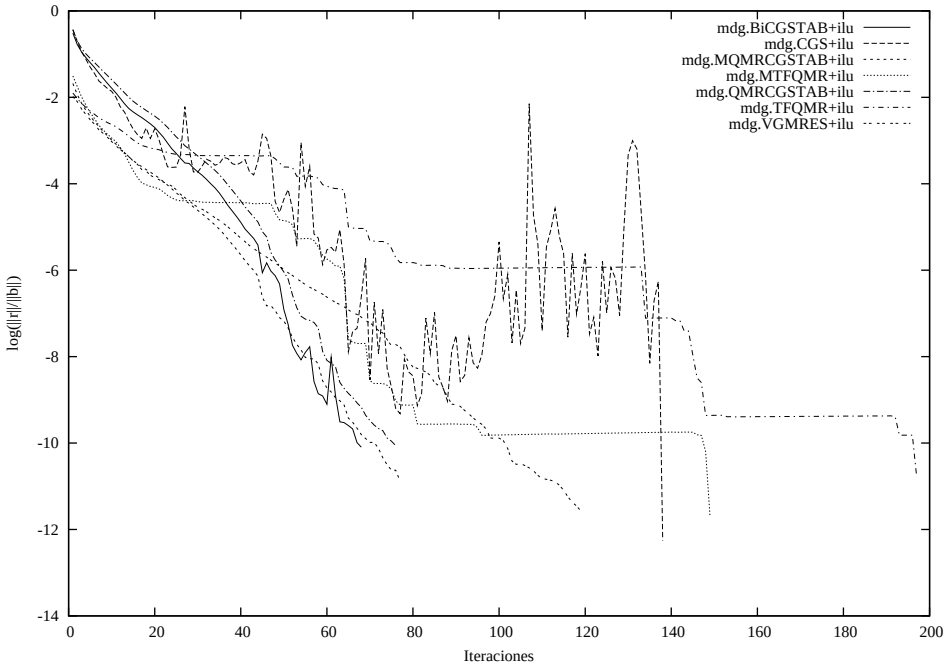

Figura 10.92: *Convergencia de diferentes métodos de Krylov precondicionados con ILU(0) después de reordenar con Grado Mínimo para isla (12666 ecuaciones)*

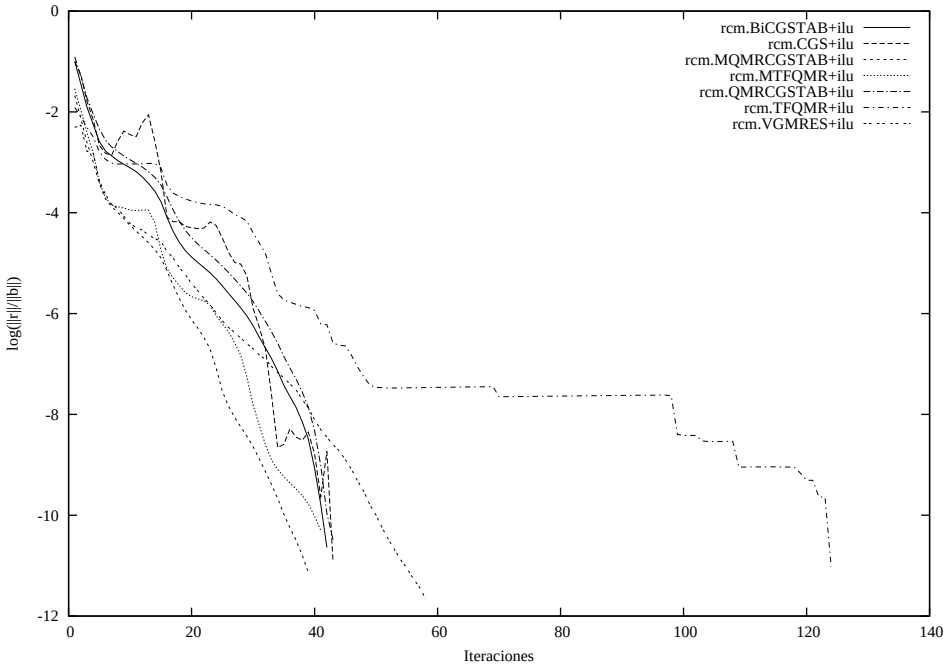

Figura 10.93: *Convergencia de diferentes métodos de Krylov precondicionados con ILU(0) después de reordenar con Cuthill McKee Inverso para isla (12666 ecuaciones)*

En las figuras 10.85-10.87 se presentan las curvas correspondientes a la convergencia de los distintos métodos precondicionados con ILU(0) con las diferentes renumeraciones y sin renumerar. La suavidad de las curvas para los sistemas reordenados se constata en las figuras 10.86 y 10.87.

Para el sistema de 12666 ecuaciones, el número de métodos que convergen se reduce a seis, debido al mayor número de condición de la matriz. Este caso sólo converge cuando no está renumerado, para los métodos, VGMRES (que es el único que converge sin utilizar alguna técnica de precondicionamiento), BiCGSTAB, QMRCGSTAB y MQMRCGSTAB, que lo hacen sólo cuando estan precondicionados con ILU(0) o SSOR.

Las figuras 10.91-10.93 muestran las curvas para los distintos algoritmos precondicionados con ILU(0) y las correspondientes renumeraciones con Grado Mínimo y Cuthill-McKee Inverso, en donde queda reflejado el beneficioso efecto de este último algoritmo.

Capítulo 11

Conclusiones y líneas futuras

En esta tesis se ha estudiado, en primer lugar, la resolución de sistemas de ecuaciones lineales de matriz *sparse* mediante métodos iterativos basados en subespacios de Krylov, dotados de distintas técnicas de precondicionamiento, así como la aplicación de algoritmos de reordenación para mejorar el efecto del precondicionamiento. Por otro lado, se han aportado unas variantes para los métodos de cuasi-mínimo residuo que hemos denominado métodos Modificados. Finalmente, se ha presentado un número considerable de experimentos numéricos que nos ha permitido extraer las siguientes conclusiones:

Para los sistemas simétricos, en general, la estrategia iterativa de resolución está clara: el métodos del Gradiente Conjugado es el más eficaz tanto en número de iteraciones y suavidad de la convergencia, como en coste computacional. Sin embargo, puede ocurrir que para matrices no definidas positivas este método no llegue a converger (ver por ejemplo la figura 10.8).

Para los sistemas no simétricos, la estrategia más adecuada no está igual de clara y depende, en general, de varios factores.

De los métodos de ortogonalización, el algoritmo GMRES y sus variantes, que hemos aplicado en todos los ejemplos, podemos destacar que aunque se trata de un método muy robusto y seguro, es generalmente más costoso con respecto a otros métodos estudiados, especialmente en lo que se refiere a memoria y a tiempo de CPU. En cambio presenta una convergencia monotónica en un número de iteraciones relativamente corto. El VGMRES ha permitido utilizar de forma combinada el *full*-GMRES y el *restarted*-GMRES, lo que redunda en un ahorro en el coste computacional.

Los métodos de biortogonalización han resultado los más adecuados, sobre todo para los sistemas de menor dimensión. Generalmente, los métodos BiCGSTAB, QMRCGSTAB y su versión modificada MQMRCGSTAB, son los que presentan una convergencia más rápida y barata en todos los casos, siendo los dos últimos los que presentan curvas de convergencia más suaves, con menos fluctuaciones, aunque el tiempo de computación puede ser algo mayor. Además, junto con el VGMRES, son los que han funcionado eficientemente en casi todos los problemas no simétricos presentados.

Respecto a las versiones modificadas de los métodos de cuasi-minimo residuo, propuestas en esta tesis, los resultados obtenidos indican que, en general proporcionan curvas de convergencias más suaves que las versiones originales, aunque el coste computacional suele estar por encima. En los ejemplos analizados se ha comprobado que los algoritmos Modificados poseen características similares a las del GMRES, pero con un menor coste computacional que éste. Este comportamiento más robusto de los métodos de tipo QMR Modificados ha permitido alcanzar la convergencia en algunos casos en que los métodos originales no lo consiguieron.

La familia de métodos basados en la Ecuación Normal, representada aquí por los algoritmos CGN y LSQR, no parece competitiva para el tipo de ejemplos que proponemos, en su mayoría problemas de Convección-Difusión, porque su coste resulta elevadísimo. En efecto, no se ha logrado la convergencia en prácticamente ninguno de los casos, ya que, o bien el número de iteraciones necesario era mayor que la dimensión del sistema, o bien el tiempo de computación era mucho más elevado que el de la mayoría de los otros métodos, incluido el VGMRES.

Las aplicaciones numéricas indican que no sólo es fundamental utilizar técnicas de precondicionamiento para la resolución de los sistemas de forma eficiente, ya que aceleran la convergencia y la suavizan, sino también la elección del precondicionador adecuado, dependiendo de cada problema. En general, de los precondicionadores utilizados, el ILU(0) resultó el más competitivo, aunque el precondicionador SSOR pueda ser en ocasiones menos costoso en tiempo de CPU. En cualquier caso, debemos señalar que en determinados problemas y para determinados métodos no es posible la convergencia cuando se precondiciona el sistema, debido a que pueden existir algunas entradas nulas en la diagonal de la matriz.

Se ha comprobado, asimismo, que el efecto de la renumeración mejora sensiblemente la velocidad de convergencia y, en particular, reduce el coste computacional cuando se utilizan los precondicionadores ILU(0) y SSOR, ya que se mejora las características de los mismos. Por otro lado, el tiempo requerido para la reordenación es siempre muy inferior al total del proceso.

Queda pues patente, a modo de resumen, que lo que hemos definido como estrategia RPK (*Reordenación, Precondicionamiento y Método de Krylov*) para la resolución de sistemas de ecuaciones lineales debe considerarse de manera que cada uno de los tres aspectos sean igualmente importantes e irrenunciables. Asimismo, los métodos QMR Modificados suponen una elección a caballo entre los métodos clásicos de QMR y el GMRES.

Este trabajo abre varias líneas futuras que deben ser estudiadas con profundidad.

En primer lugar, estudiar la posibilidad de aplicar a los métodos de cuasimínimo residuo modificados propuestos en esta tesis, una técnica de *restart* que permita la implementación de estos algoritmos de forma similar a como se hace en el VFGMRES (ver Galán y otros [24]), lo que supondría una mejora conside-

rable del comportamiento, en cuanto a la convergencia y coste computacional.

Por otro lado, sería muy interesante realizar un estudio para comprobar cómo afecta a la convergencia de los métodos de cuasi-mínimo residuo modificados, MTFQMR y MQMRCGSTAB, una determinada elección del vector inicialización $\mathbf{r}_0^*$ y estudiar la posibilidad de encontrar nuevas formas de precondicionamiento dependiendo de ésta, tal como proponen Suárez y otros [82] para los algoritmos CGS y BiCGSTAB.

Habría que extender este estudio a otros precondicionadores como los basados en la inversa aproximada.

Finalmente, y de forma similar, sería también de interés ampliar el estudio con otras técnicas de reordenación que permitan tener en cuenta los valores numéricos de los elementos de las matrices.

Bibliografía

[1] W.E. ARNOLDI, The Principle of Minimized Iteration in the Solution of the Matrix Eingenvalue Problem, *Quart. Appl. Math.*, 9, pp.17-29 (1951).

[2] O. AXELSSON, A Restarted Version of a Generalized Preconditioned Conjugate Gradient Method, *Comunications in Applied Numerical Methods*, 4, pp. 521-530 (1988).

[3] O. AXELSSON, *Iterative Solution Methods*, Cambridge University Press (1996).

[4] R. BARRET, M.BERRY, T.F. CHAN, J. DEMMEL, J. DONATO, J. DONGARRA, V. EIJKHOUT, R. POZO, C. ROMINE AND H. VAN DER VORST, *Templates for th Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, (1994).

[5] M. BENZI, C.D. MEYER Y M. TUMA, A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method, *SIAM J. Sci. Comput.*, 17, 5, pp. 1135-1149 (1996).

[6] T.F. CHAN, E. GALLOPOULOS, V. SIMONCINI, T. SZETO Y C.H. TONG, A Quasi-Minimal Residual Variant of the Bi-CGSTAB Algorithm for Nonsymmetric Systems, *SIAM J. Sci. Statist. Comput.*, 15, pp. 338-247 (1994).

[7] T.F. CHAN Y T. SZETO, Composite Step Product Methods for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Comput.*, 17, 6, pp. 1491-1508 (1996).

[8] E.H. CUTHILL Y J.M. MCKEE, Reducing the Bandwidth of Sparse Symmetric Matrices, *Proc. 24th National Conference of the Association for Computing Machinery*, Brondon Press, New Jersey, pp. 157-172 (1969)

[9] A.C. DAMHAUG Y K.M. MATHISEN, On Sparse Matrix Methods in Computational Structural Mechanics, en *Numerical Methods in Engineering'92*, Ch. Hirsch et al. (Editors), pp. 527-532 (1992).

[10] JKM. DICKINSON Y P.A. FORSYTH, Preconditioned Conjugate Gradient Methods for Three-Dimensional Linear Elasticity, *Int. J. Num. Meth. Eng.*, 37, pp. 2211-2234 (1994).

[11] Q.V. DINH, B. MANTEL, J. PERIAUX Y B. STOUFFLET, Contribution to Problems T4 and T6 Finite Element GMRES and Conjugate Gradient Solvers, *Tech. Rep.* (1993).

[12] I.S. DUFF, R.G. GRIMES Y J.G. LEWIS, Sparse Matrix Test Problems, *ACM Trans. Math. Sof.* 15, 1, pp. 1-14 (1989).

[13] L.C. DUTTO, The Effect of Ordering on Preconditioned GMRES Algorithm, *Int. Jour. Num, Meth. Eng*. 36, pp. 457-497 (1993).

[14] J.M. ESCOBAR, G. MONTERO Y R. MONTENEGRO, Efficient Tools for 3-D Electromagnetic Simulation Using FEM, en *7th International Colloquium on Differential Equations*, Plovdiv, Bulgaria (1996).

[15] S. FERNÁNDEZ MÉNDEZ, *Mesh-Free Methods and Finite Elements: Friend or Foe?*, Doctoral Thesis, Universitat Politècnica de Catalunya (2001).

[16] L. FERRAGUT, G. WINTER, G. MONTERO Y R. MONTENEGRO, *NEPTUNO, un Sistema Adaptativo de Elementos Finitos en 2-D*, Dpto. de Matemáticas, Univ. Las Palmas Gran Canaria (1989).

[17] R. FLETCHER, Conjugate Gradient Methods for Indefinite Systems, *Lectures Notes in Math.,* 506, pp. 73-89 (1976).

[18] E. FLOREZ, D. GARCIA, L. GONZALEZ Y G. MONTERO, The Effect of Ordering on Sparse Approximate Inverse Preconditioners for Nonsymmetric Problems, *Advances in Engineering Software*, 33, pp. 611-619, (2002).

[19] E. FLOREZ, D. GARCIA, L. GONZALEZ Y G. MONTERO, Does the Ordering Affect to Sparse Inverse Preconditioners?, *en MS'2000*, pp. 413-422, Las Palmas G.C. (2000).

[20] R.W. FREUND, A Transpose-Free Quasi-Minimal Residual Algorithm for non-Hermitian Linear Systems, *SIAM J. Sci. Comput.,* 14, pp. 470-482 (1993).

[21] R.W. FREUND Y N.M. NACHTIGAL, QMR: a Quasi-Minimal Residual Method for non-Hermitian Linear Systems, *Numerische Math.* 60, pp. 315-339 (1991).

[22] R.W. FREUND Y N.M. NACHTIGAL, An Implementation of the QMR Method Based on Coupled Two-Term Recurrences, *SIAM J. Sci. Comput.*, 15, 2, pp. 313-337 (1994).

[23] R.W. FREUND, M.H. GUTKNECHT Y N.M. NACHTIGAL, An Implementation of the Look-ahead Lanczos Algorithm for non-Hermitian Matrices, *SIAM J. Sci. Comput.*, 14, 1, pp. 137-158 (1993).

[24] M. GALÁN, G. MONTERO Y G. WINTER, Variable GMRES: an Optimizing Self-Configuring Implementation of GMRES(k) with Dynamic Memory Allocation, *Tech. Rep. of CEANI*, (1994).

[25] M. GALÁN, G. MONTERO Y G. WINTER, A Direct Solver for the Least Square Problem Arising From GMRES(k), *Com. Num. Meth. Eng.*, 10, pp. 743-749 (1994).

[26] M. GALÁN, *Avances en el Método de Residuo Mínimo Generalizado (algoritmo GMRES), su Desarrollo en ANSI-C con Algoritmos de Paralelización y Vectorización, y sus Aplicaciones al Método de los Elementos Finitos,* Tesis Doctoral, Univ. de Las Palmas de Gran Canaria (1994).

[27] D. GARCÍA, G. MONTERO A. SUÁREZ E. FLÓREZ Y L. GONZÁLEZ, Aplicaciones de Métodos Basados en Subespacios de Krylov para Sistemas Lineales Sparse No Simétricos. *XVI C.E.D.Y.A./ V C.M.A.*, pp. 1123-1132, Las Palmas G.C. (1999).

[28] A. GEORGE, Computer Implementation of the Finite Element Method, *Report Stan CS-71-208,* (1971).

[29] A. GEORGE Y J.W. LIU, The Evolution of the Minimum Degree Ordering Algorithms, *SIAM Rev.* 31, pp. 1-19 (1989).

[30] G.H. GOLUB Y KAHAN W., Calculating the Singular Values and Pseudoinverse of a Matrix. *SIAM J. Numer. Anal. 2,* pp. 205-224 (1965).

[31] L. GONZÁLEZ, G. MONTERO, Y E. FLÓREZ, Aproximate Inverse Preconditioners Using Frobenius Inner Product I: Theorical Results, 8^{th} *ILAS CONFERENCE,* Barcelona, (1999).

[32] L. GONZÁLEZ, G. MONTERO, E. FLÓREZ Y D. GARCÍA, Aproximada inversa y producto escalar de Frobenius, en *XVI C.E.D.Y.A., V/ C.M.A.*, pp. 1141-1148, Las Palmas G.C. (1999).

[33] A. GREENBAUM Y L.N. TREFETHEN, GMRES/CR and Arnoldi/Lanczos as Matrix Approximation Problems, *SIAM J. Sci. Comput.*, 15, 2, pp. 359-368 (1994).

[34] A. GREENBAUM, *On Role of the Left Starting Vector in the Two-Ssided Lanczos Algorithm and Nonsymmetric Linear System Solvers*, University of Washington, Mathematics Department, (1997).

[35] A. GREENBAUM, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, (1997).

[36] M. GROTE Y H.D. SIMON, Parallel Preconditioning and Approximate Inverses on the Connection Machine, en *Sixth SIAM Conference on Parallel Processing for Scientific Computing*, Vol. II, pp. 519-523, Philadelphia, (1992).

[37] M.H. GUTKNECHT, Variants of BICGSTAB for Matrices with Complex Spectrum, *SIAM J. Sci. Comput.*, 14, 5, pp. 1020-1033 (1993).

[38] W. HACKBUSCH, *Iterative Solution of Large Sparse Systems of Equations,* Applied Mathematical Sciences 95, Springer-Verlag, New York, (1994).

[39] M.R. HESTENES Y E. STIEFEL, Methods of Conjugate Gradients for Solving Linear Systems, *Jour. Res. Nat. Bur. Sta.* 49, 6, pp. 409-436, (1952).

[40] A. HUERTA, A. RODRÍGUEZ-FERRÁN, J. SARRATE, P. DÍEZ. Y S. FERNÁNDEZ-MÉNDEZ, Numerical Modelling, a Tool to Improve the Design of Active Carbon Canisters, en *Abstracts of the Sixth U.S. National Congress on Computational Mechanics,* Dearborn, Michigan, (2001).

[41] A. JENNINGS Y G.M. MALIK, The Solution of Sparse Linear Equations by Conjugate Gradient Method, *Int. Jour. Num. Meth. Eng.* 12, pp. 141-158, (1978).

[42] W. JOUBERT, A Robust GMRES-based Adaptive Polynomial Preconditioning Algorithm for Nonsymmetric Linear Systems, *SIAM J. Sci. Comput.*, 15, 2, pp. 427-439 (1994).

[43] E.M. KASENALLY, GMBACK: A Generalised Minimum Backward Error Algorithm for Nonsymmetric Linear Systems, *SIAM J. Sci. Comput.,* 16, 3, pp. 698-719 (1995).

[44] C. LANCZOS, Solution of Systems of Linear Equations by Minimized Iterations, *Jour. Res. Nat. Bur. Sta.* 49, 1, pp. 33-53, (1952).

[45] R.R. LEWIS, Simulated Annealing for Profile and Fill Reduction of Sparce Matrices, *Int. Jour. Num, Meth. Eng*. 37, pp. 905-925 (1994).

[46] I.L. LIM, I.W. JOHNSTON Y S.K. CHOI, A Comparison of Algorithms for Profile Reduction of Sparse Matrices, *Computers & Structures,* 57, 2, pp. 297-302 (1995).

[47] T-Z. MAI, Modified Lanczos Method for Solving Large Sparse Linear Systems, *Com. Num. Meth. Eng.,* 9, pp. 67.79 (1993).

[48] G. MARTIN, Methodes de Preconditionnemen par Factorisation Incomplete, *Memoire de Maitrise*, Universite Laval, Quebec, Canada, (1991).

[49] M. MONGA-MADE, Ordering Strategies for Modified Block Incomplete Factorizations, *SIAM J. Sci. Comput.,* 16, 2, pp. 378-399 (1995).

[50] R. MONTENEGRO, G. MONTERO, G. WINTER Y L. FERRAGUT, Aplicación de Métodos de Elementos Finitos Adaptativos a Problemas de Convección-Difusión en 2-D, *Rev. Int. Met. Num. Cal. y Dis. en Ing.*, 5, 4, pp. 535-560 (1989).

[51] G. MONTERO Y A. SUÁREZ, Left-Right Preconditioning Versions of BCG-Like Methods, *Int. J. Neur., Par. & Sci. Comput.*, 3, pp. 487-501 (1995).

[52] G. MONTERO, G. WINTER, A. SUÁREZ, M. GALÁN Y D. GARCÍA, Contribution to Iterative Methods for Nonsymmetric Linear Systems: GMRES, BCG and QMR Type Methods, en *Computational Methods and Neural Networks. Part Two: Computational Methods,* M. Sambandhamy M.P. Bekakos, Eds., Dynamic Publishers, Inc., pp. 97-128 (1999).

[53] G. MONTERO, R. MONTENEGRO, G. WINTER Y L. FERRAGUT, Aplicacion de Esquemas EBE en Procesos Adaptativos, *Rev. Int. Met. Num. Cal. Dis. Ing.* 6, pp.311-332 (1990).

[54] G. MONTERO Y A. SUÁREZ, Efecto del Precondicionamiento y Reordenación en los Métodos CGS y BI-CGSTAB, en *III Congreso de Métodos Numéricos en Ingeniería*, pp. 1306-1315, Zaragoza (1996).

[55] G. MONTERO, R. MONTENEGRO Y J.M. ESCOBAR, *MEM3D, un Código de Elementos Finitos Tridimensional*, Dpto. de Matemáticas, Univ. Las Palmas de G. C. (1996).

[56] G. MONTERO, G. WINTER, P. ALMEIDA Y P. CUESTA, Resolución de Sistemas no Simétricos con Doble Gradiente Conjugado Precondicionado en Procesos Adaptativos, en *XII C.E.D.Y.A., II Congr. de Mat. Apl.*, pp. 625-630, Oviedo (1991).

[57] G. MONTERO, M. GALÁN, P. CUESTA Y G. WINTER, Effects of Stochastic Ordering on Preconditioned GMRES Algorithm, *Advances in Structural Optimization*, pp. 241-246 (1994).

[58] G. MONTERO, A. SUÁREZ Y D. GARCÍA, Implementation of QMR-Like Methods Using a New Solver for the Quasi-Minimization Problem, en *Encuentro de Análisis Matricial y Aplicaciones (EAMA 97)*, pp. 229-236, Sevilla (1997).

[59] G. MONTERO, L. GONZÁLEZ, D. GARCÍA Y A. SUÁREZ, Aproximate Inverse Using Frobenius Inner Product II: Computational Aspects, *8^{th} ILAS CONFERENCE,* Barcelona, (1999).

[60] G. MONTERO,L. GONZÁLEZ, E. FLÓREZ, M.D. GARCÍA Y A. SUÁREZ., Approximate Inverse Computation Using Frobenius Inner Product, *Num. Lin. Alg. Appl., 9,* pp. 239-247 (2002).

[61] N.M. NACHTIGAL, S.C. REDDY Y L.N. TREFETHEN, How Fast Are Nonsymmetric Matrix Iterations?, *SIAM J. Matr. Anal. Appl.,* 13, 3, pp. 796-825 (1992).

[62] N.M. NACHTIGAL, L. REICHEL Y L.N. TREFETHEN, A Hybrid GMRES Algorithm for Nonsymmetric Linear Systems, *SIAM J. Matr. Anal. Appl.,* 13, 3, pp. 778-795 (1992).

[63] C.C. PAIGE Y M.A. SAUNDERS, LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares, *ACM Trans. Math. Sof.* 8, 1, pp. 43-71 (1982).

[64] B.N. PARLETT, D.R. TAYLOR Y Z.A. LIU, A Look-Ahead Lanczos Algorithm for Unsymetric Matrices, *Mathematics of Computation,* 44, 169, pp. 105-124 (1985).

[65] G.H. PAULINO, I.F.M. MENEZES, M. GATTAS Y S. MUKHERJEE, A New Algorithm for Finding a Pseudoperipheral Vertex or the Endpoints of a Pseudodiameter in a Grapf, *Num. Meth. Eng.,* 10, pp. 913-926 (1994).

[66] A. PETERS, Non-Symmetric CG-Like Schemes and the Finite Element Solution of the Advection-Dispersion Equation, *Int. Jour. Num, Meth. Fluids*, 17, pp. 955-974 (1993).

[67] C. POMMERELL Y W. FICHTNER, Memory Aspects and Performance of Iterative Solvers, *SIAM J. Sci. Comput.*, 15, 2, pp. 460-473 (1994).

[68] G. RADICATI Y M. VITALETTI, Sparse Matrix-Vector Product and Storage Representations on the IBM 3090 with Vector Facility, *Report G513-4098, IBM-ECSEC*, Rome, (1986).

[69] J.J. RUSCH, Using a Complex Version of GMRES for Solving Optical Sattering Problems, en *Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen Eds. Elsevier Science Publishers B.V. (North-Holland)*, pp. 459-468 (1992).

[70] Y. SAAD Y M. SCHULTZ, GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Statist. Comput.*, 7, pp. 856-869 (1986).

[71] Y. SAAD, A Flexible Inner-Outer Preconditioned GMRES Algorithm, *SIAM J. Sci. Comput.,* 14, pp. 461-469 (1993).

[72] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, PWS PUBLISHING COMPANY, Boston, (1996).

[73] P. SAINT-GEORGES Y G. WARZEE, High-Performance PCG Solvers for FEM Structural Analysis, *Int. Jour. Num, Meth. Eng*. 39, pp. 1313-1340 (1996).

[74] P.E. SAYLOR, Richardson's Iteration with Dynamic Parameters and the SIP Incomplete Factorization for the Solution of Linear Systems of Equations, *Society of Petroleum Engineers J.,* pp. 691-708 (1981).

[75] J.N. SHADID Y R.S. TUMINARO, A Comparison of Preconditioned Nonsymmetric Krylov Methods on a Large-Scale Mimd Machine, *SIAM J. Sci. Comput.,* 15, 2, pp. 440-459 (1994).

[76] P. SONNEVELD, CGS: a Fast Lanczos-Type Solver for Nonsymmetric Linear Systems, *SIAM J.Sci. Statist. Comput.* 10, pp.36-52 (1989).

[77] G. STARKE, Alterating Direction Preconditioning for Nonsymmetric Systems of Linear Equations, *SIAM J. Sci. Comput.,* 15, 2, pp. 369-384 (1994).

[78] E. STEINTHORSSON Y T. I-P. SHIH, Methods for Reducing Approximate-Factorization Errors in Two- and Three-Factored Schemes, *SIAM J. Sci. Comput.,* 14, 5, pp. 1214-1236 (1993).

[79] A. SUÁREZ, *Contribución a Algoritmos de Biortogonalización para la Resolución de Sistemas de Ecuaciones Lineales*, Tesis Doctoral, Univ. de Las Palmas de G. C. (1995).

[80] A. SUÁREZ, G. MONTERO Y M. GALÁN, Estudio del Comportamiento de Diferentes Precondicionadores para Doble Gradiente Conjugado en Problemas de Convección-Difusión, en *XIII C.E.D.Y.A. Congreso de Matemática Aplicada*, pp. 822-827, Madrid (1993).

[81] A. SUÁREZ, D. GARCÍA, E. FLÓREZ Y G. MONTERO, Preconditioning Krylov Methods, en *Nato Book Series, Algorithms for Sparse Large Scale Linear Algebraic Systems: State of the Art and Applications in Science and Engineering* (1997).

[82] A. SUÁREZ, G. MONTERO, D. GARCÍA Y E. FLÓREZ, Efecto de la Elección del Vector Inicial en la Convergencia de los Algoritmos CGS y BICGSTAB, en *Encuentro de Análisis Matricial y Aplicaciones (EAMA 97)*, Sevilla (1997).

[83] M. SUARJANA Y K.H. LAW, A Robust Incomplete Factorization Based on Value and Space Constrainsts, *Int. Jour. Num, Meth. Eng*. 38, pp. 1703-1719 (1995).

[84] L.N. TREFETHEN, D. BAU, *Numerical Linear Algebra,* SIAM, Philadelphia, (1997)

[85] H.A. VAN DER VORST, Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems, *SIAM J. Sci. Comput.,* 13, pp. 631-644 (1992).

[86] H.A. VAN DER VORST, Conjugate Gradient Type Methods for Nonsymmetric Linear Systems, en *Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen Eds. Elsevier Science Publishers B.V. (North-Holland),* pp. 67-76 (1992).

[87] H.A. VAN DER VORST Y C. VUIK, GMRESR: A Family of Nested GMRES Methods, *Tech, Rep. 91-80, Delft University of Technology, Mathematics and Informatics*, Delft, The Netherlands, (1991).

[88] M. VIDRASCU, An Evaluation of the Linear Systems Arising from 3D Elasticity Problems, *Comp. Meth. Appl. Mech. Eng.*, 101, pp. 463-477 (1992).

[89] C. VUIK, A Comparison of some GMRES-Like Methods, *en Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen Eds. Elsevier Science Publishers B.V. (North-Holland)*, pp. 155-161 (1992).

[90] H.F. WALKER, Implementation of the GMRES Method Using Householder Transformations, *SIAM J. Sci. Comput.,* 9, pp. 152-163 (1988).

[91] G. WINTER, G. MONTERO, L. FERRAGUT Y R. MONTENEGRO, Adaptive Strategies Using Standard and Mixed Finite Elements for Wind Field Adjustment, *Solar Energy*, 54, 1, pp. 49-56 (1995).

[92] G. WINTER, G. MONTERO Y J. BETANCOR, *CONDIF3D, un Sistema de Elementos Finitos en 3-D para Simulación del Transporte y Difusión de Contaminantes en la Atmósfera*, C.E.A.N.I., Univ. Las Palmas de Gran Canaria (1997).

[93] G. WINTER, G. MONTERO, L. FERRAGUT Y R. MONTENEGRO, Aplicación de Esquemas EBE de Gradiente Conjugado en Suavizados del Método Multimalla para Procesos Adaptativos, en *XI C.E.D.Y.A. I Congreso de Matemática Aplicada*, pp. 369-375, Fuengirola, Málaga (1989).

[94] G. WINTER, G. MONTERO P. CUESTA, M. GALÁN Y E. FLÓREZ, Resolución Numérica del Problema Convección-Difusión en un Intercambiador de Calor de Flujo Cruzado con Lecho Granular Móvil, *en XII C.E.D.Y.A III Congreso de Matemática Aplicada*, pp. 258-265, Madrid (1993).

[95] P.M. DE ZEEUW, Incomplete Line LU as Smoother and as Preconditioner, *en Eighth GAMM-Seminar*, pp. 215-224, Kiel (1992).

[96] L. ZHOU Y H.F. WALKER, Residual Smoothing Techniques for Iterative Methods, *SIAM J. Sci. Comput.*, 15, 2, pp. 297-312 (1994).