UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

DEPARTAMENTO DE MATEMÁTICAS

TESIS DOCTORAL

CONTRIBUCIÓN A ALGORITMOS DE BIORTOGONALIZACIÓN PARA LA RESOLUCIÓN DED SISTEMAS DE ECUACIONES LINEALES

ANTONIO F. SUÁREZ SARMIENTO

Las Palmas de Gran Canaria, Noviembre de 1995

15/1995-96

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
UNIDAD DE TERCER CICLO Y POSTGRADO

Reunido el día de la fecha, el Tribunal nombrado por el Excmo. Sr. Rector Magfco. de esta Universidad, el aspirante expuso esta TESIS DOCTORAL.

Terminada la lectura y contestadas por el Doctorando las objeciones formuladas por los señores jueces del Tribunal, éste calificó dicho trabajo con la nota de Apto cum laude

Las Palmas de G. C., a 20 de diciembre de 1995.

El Presidente: Dr. D. Francisco Rubio Royo,

El Secretario: Dr. D. Pedro Almeida Benítez,

El Vocal: Dr. D. Gabriel Winter Althaus,

El Vocal: Dr. D. Felipe Petriz Calvo,

El Vocal: Dr, D. Iñigo Arregui Alvarez,

El Doctorando: D. Antonio Suárez Sarmiento,

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

DEPARTAMENTO DE MATEMÁTICAS

TESIS DOCTORAL

CONTRIBUCIÓN A ALGORITMOS DE BIORTOGONALIZACIÓN PARA LA RESOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES

Antonio F. Suárez Sarmiento

Noviembre de 1995

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
DEPARTAMENTO DE MATEMÁTICAS

TESIS DOCTORAL

CONTRIBUCIÓN A ALGORITMOS DE BIORTOGONALIZACIÓN PARA LA RESOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES

PROGRAMA DE ELEMENTOS FINITOS EN LA INGENIERÍA

Presentada por Antonio F. Suárez Sarmiento
Dirigida por Dr. D. Gustavo Montero García

El Director

El Doctorando

Las Palmas de G.C. a 2 de Noviembre de 1995

AGRADECIMIENTOS

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

A Gabriel Winter Althaus, por sugerir, junto con Gustavo, el atractivo tema que constituyó el núcleo de esta tesis y por sus consejos siempre oportunos.

A D. Luis Álvarez Amador, por preocuparse en su momento de mi reingreso a las tareas docentes en la Universidad.

Las Palmas de G.C. a 2 de Noviembre de 1995

A mi familia

RESUMEN

En este trabajo se presentan alternativas para la resolución de los sistemas de ecuaciones lineales a que da lugar la aplicación de métodos como diferencias finitas ó elementos finitos en la obtención de soluciones aproximadas de problemas de contorno en derivadas parciales.

En principio, se hace una síntesis comparativa de distintos métodos iterativos basados en los subespacios de Krylov, dedicando especial atención a los métodos CGS y Bi-CGSTAB.

Después de exponer las distintas formas de precondicionar y definir los precondicionadores Diagonal, SSOR e ILU(0), establecemos los algoritmos apropiados de estos métodos para aplicarlos al sistema precondicionado. Se plantea la equivalencia, para métodos como el CGS y el Bi-CGSTAB, entre estas diferentes formas de precondicionar, (por la derecha, por la izquierda y por ambos lados), con una adecuada elección del vector de inicialización del algoritmo.

Asimismo, se contrastan la bondad de los distintos algoritmos utilizando estos precondicionadores en la resolución de algunos problemas clásicos en MEF. En la aplicación del precondicionador ILU(0) se han aplicado técnicas de renumeración que han supuesto mejoras en la convergencia.

Se presentan variantes para Bi-CG, CGS y Bi-CGSTAB, que hemos llamado, respectivamente, Bi-CG*, CGS* y Bi-CGSTAB*, que ofrecen, en muchas ocasiones, curvas de convergencia más suaves y uniformes.

Por último, se aplican estas técnicas y algoritmos a la resolución de un problema de control en la frontera.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1: SÍNTESIS COMPARATIVA DE MÉTODOS PARA LA RESOLUCIÓN DE ECUACIONES LINEALES BASADOS EN LOS SUBESPACIOS DE KRYLOV. 4

1.1 - Subespacios de Krylov. 5

1.2 - Métodos iterativos para sistemas no simétricos. 7

1.3 - Métodos basados en la ecuación normal. 8

1.4 - Métodos de ortogonalización. GMRES. 9

1.5 - Métodos de biortogonalización tipo Bi-CG. 11

1.5.1 - Bi-CG. 12

1.5.2 - CGS. 14

1.5.3 - Bi-CGSTAB. 17

1.5.4 - QMR. 21

CAPÍTULO 2: PRECONDICIONAMIENTO. 24

2.1 - Condicionamiento de un sistema. 24

2.2 - Técnica de precondicionamiento. 25

2.3 - Convergencia y precondicionamiento. 26

2.3.1 - Sistemas simétricos. 27

2.3.2 - Sistemas no simétricos. 27

2.4 - Sistemas precondicionados. 28

2.5 - Precondicionadores en sistemas no simétricos. 29

2.5.1 - Precondicionadores Diagonal, SOR y SSOR. 29

2.5.2 - Factorizaciones incompletas. 32

CAPÍTULO 3: ALMACENAMIENTO COMPACTO. 35

3.1 - Almacenamiento de la matriz del sistema. 36

3.2 - Almacenamiento de la matriz de precondicionamiento factorizada ILU. 37

3.3 - Almacenamiento de la matriz de precondicionamiento factorizada SSOR. 38

CAPÍTULO 4: EFECTO DE LA RENUMERACIÓN EN EL PRECONDICIONAMIENTO. 40

CAPÍTULO 5: ALGORITMOS PRECONDICIONADOS. 44

5.1 - Doble Gradiente Conjugado. 45

5.2 - CGS. 54

5.3 - Algoritmo CGS*. 61

5.4 - Bi-CGSTAB. 64

5.5 - Algoritmo Bi-CGSTAB*. 70

CAPÍTULO 6: APLICACIONES TEST. 73

6.1 - Problema 1. 76

6.2 - Problema 2. 84

6.3 - Problema 3. 91

6.4 - Problema 4. 106

6.5 - Observaciones generales. 116

CAPÍTULO 7: APLICACIONES EN LA RESOLUCIÓN DE PROBLEMAS DE CONTROL EN LA FRONTERA CON OPERADOR DE LAPLACE. 118

7.1 - Definición del problema. 118

7.2 - Discretización. 119

7.3 - Optimización. 119

7.4 - Métodos de resolución. 121

7.5 - Resultados numéricos. 124

CONCLUSIONES Y LÍNEAS FUTURAS. 131

REFERENCIAS. 133

INTRODUCCIÓN

La formulación matemática de las leyes que rigen muchos problemas físicos es complicada. En las expresiones correspondientes figuran, con frecuencia, derivadas parciales de orden superior que hacen muy difícil ó imposible alcanzar la solución de forma analítica.

La imposibilidad de obtener la solución exacta traducida en una expresión matemática a la que se ajusten las diversas variantes a que puede dar lugar un fenómeno físico variando los parámetros que intervengan en su definición, no debe empañar, en absoluto, el carácter útil y determinante que tiene esta formulación para dar soluciones numéricas por procedimientos aproximados a cada problema concreto, caracterizado por determinadas constantes físicas.

Este carácter numérico de las matemáticas data de la antigüedad, en la que esta ciencia debía hacer frente a problemas prácticos con la finalidad de obtener una solución en forma de números y su desarrollo en forma de planteamientos nuevos para métodos numéricos ha conllevado que los modelos matemáticos para representar los fenómenos físicos se hayan ido perfeccionando de tal forma, que se encuentra con frecuencia 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 (con diferencias finitas, elementos finitos, volúmenes finitos), que nos 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 incógnitas son precisamente estos valores numéricos puntuales de la solución aproximada al problema físico que ha conducido al sistema.

La resolución de estos sistemas de ecuaciones, elemento fundamental en este proceso de dar solución a un problema, con un planteamiento de técnicas y métodos apropiados, constituye el objeto primordial de esta tesis.

Como la eficiencia de un método depende de su facilidad de implementación, la "pléyade" de métodos apropiados para aproximar la solución ha ido desarrollándose paralelamente a la evolución de los ordenadores. Así, el conocido Gradiente Conjugado publicado por Hestenes y Stiefel en 1952 ha adquirido relevancia prácticamente en los últimos veinticinco años, que es cuando ha encontrado el soporte informático adecuado.

Los errores de redondeo y el efecto de relleno que se producen en la aplicación de los métodos directos, han hecho más adecuados los métodos iterativos para la resolución estos sistemas. En el conjunto de estos métodos, tienen especial relevancia los algoritmos basados en los subespacios de Krylov con un desarrollo espectacular en los últimos tiempos.

Con estos antecedentes, los objetivos específicos de esta tesis, con la finalidad común de proporcionar herramientas adecuadas para la obtención de soluciones aproximadas de sistemas de ecuaciones lineales son:

- Hacer un análisis general de los métodos tipo Bi-CG.

- Comprobar el efecto de un precondicionamiento previo del sistema en la convergencia de los algoritmos.

- Implementar variantes en los algoritmos ya establecidos que supongan una mejora en su comportamiento.

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

- Realizar un estudio comparativo del comportamiento de estos métodos de bio-ortogonalización entre sí, y con otros métodos como el GMRES ó los de tipo QMR, en algunos problemas de aplicación.

Estos objetivos nos han hecho estructurar la tesis en los capítulos que brevemente reseñamos ahora. En el capítulo 1 se presentan la familia de métodos basados en los subespacios de Krylov, con especial atención al CGS y Bi-CGSTAB, métodos objeto principal de esta tesis. En el capítulo 2 se describen las distintas formas de precondicionamiento y los precondicionadores que se utilizarán en las aplicaciones posteriormente. En el capítulo 3 se describe la forma de almacenamiento compacta que utilizaremos para trabajar con las matrices del sistema y con las matrices de precondicionamiento. En el capítulo 4 se exponen las técnicas de renumeración que se aplican en la utilización del precondicionador ILU(0). En el capítulo 5 se deducen los algoritmos precondicionados para los distintos métodos, estableciéndose la relación directa entre las distintas formas de precondicionamiento y la adecuada elección del vector inicialización del algoritmo en los métodos CGS y Bi-CGSTAB. Se presentan asimismo otras versiones para estos métodos que hemos denotado por CGS* y Bi-CGSTAB*. Por último, se aplican las técnicas y métodos expuestos en los capítulos anteriores a diversos problemas discretizados por MEF en el capítulo 6 y a un problema de optimización de control óptimo en la frontera en el capítulo 7.

CAPÍTULO 1.

SÍNTESIS COMPARATIVA DE MÉTODOS PARA LA RESOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES BASADOS EN LOS SUBESPACIOS DE KRYLOV.

La aplicación de diferencias finitas ó de elementos finitos, entre otros métodos, para la obtención de soluciones aproximadas de problemas de contorno en derivadas parciales, conduce a grandes sistemas de ecuaciones lineales $Ax = b$, donde la matriz A es tipo "sparse".

Los métodos directos que se utilizan para la resolución de sistemas de ecuaciones lineales están basados, generalmente, en variantes de la eliminación Gaussiana. Construyen una matriz triangular inferior L y otra triangular superior U, tal que $LU=A$. Las matrices triangulares L y U, son, asimismo, "sparse", pero más densas ó "llenas" que la matriz original A, produciéndose el llamado efecto de "relleno" ó "fill-in", que aumenta con el número de nodos y con la dimensión del problema, de tal forma, que, a igualdad de nodos, para una matriz obtenida de una discretización de un problema en 3-D es mayor que para uno de 2-D.

Este efecto, que incrementa los requerimientos de almacenaje y de coste computacional, unido a la presencia de los errores de redondeo que influyen en los valores de la solución, teóricamente exacta, hacen más adecuados los métodos iterativos para la resolución de estos sistemas. Con todo, las técnicas de reordenación, basadas en teoría de grafos, como el algoritmo inverso de Cuthill-Mckee [8], [2] ó el algoritmo del grado mínimo y sus variantes, [22], conducen a formas más apropiadas de almacenamiento y a la determinación de matrices de permutación, tales que en la aplicación de métodos de eliminación al sistema equivalente, las matrices intermedias

generadas, permanezcan tan "sparse" como sea posible, disminuyendo este "efecto fill-in".

En los métodos iterativos, a partir de un vector inicial x_0, se genera una secuencia de vectores $\{x_i\}$, que converge a la solución buscada. A pesar de la convergencia, relativamente lenta, el carácter "sparse" de A hace posible efectuar un elevado número de iteraciones sin un trabajo excesivo. Por otra parte, los errores de redondeo, importantes en los métodos directos, influyen, en general, en la velocidad de convergencia, pero no en la aproximación final.

Además de los métodos clásicos de este tipo, (Jacobi, Gauss-Seidel, Relajación, ...) que se ajustan a estas precisiones, se han desarrollado otros que presentan ventaja respecto a los mismos, sobre todo en estos sistemas "sparse". Entre ellos destacan los basados en los llamados subespacios de Krylov.

1.1 - SUBESPACIOS DE KRYLOV

En los métodos iterativos, los sucesivos valores de la solución aproximada en la resolución de $Ax = b$, vienen dados por la relación de recurrencia

$$x_{i+1} = x_i + B^{-1}(b - Ax_i),$$

ó bien,

$$Bx_{i+1} = Cx_i + b, \text{ siendo } C = B - A.$$

Distintas elecciones para las matrices B y C en función de la matriz A, conducen a los métodos clásicos de Jacobi, Gauss-Seidel ó de relajación.

Pero estas expresiones también se pueden escribir en función del vector residuo $r_i = b - Ax_i$, como $x_{i+1} = x_i + B^{-1}r_i$. De esta forma, eligiendo una aproximación inicial x_0, los valores de las sucesivas iteraciones se podrían calcular por las respectivas expresiones,

$$x_1 = x_0 + B^{-1}r_0$$

$$x_2 = x_1 + B^{-1}r_1 = x_0 + B^{-1}r_0 + B^{-1}(b - Ax_1) = x_0 + B^{-1}r_0 + B^{-1}(b - Ax_0 - AB^{-1}r_0) =$$
$$= x_0 + 2B^{-1}r_0 - B^{-1}A(B^{-1}r_0)$$

$$x_3 = x_2 + B^{-1}r_2$$

$$x_4 = x_3 + B^{-1}r_3$$

.....

$$x_i = x_{i-1} + B^{-1}r_{i-1}$$

En el caso que $B = I$ quedaría,

$$x_1 = x_0 + r_0$$

$$x_2 = x_1 + r_1 = x_0 + r_0 + (b - Ax_1) = x_0 + r_0 + (b - Ax_0 - Ar_0) =$$
$$= x_0 + 2r_0 - Ar_0$$

$$x_3 = x_2 + r_2 = x_0 + 2r_0 - Ar_0 + (b - Ax_2) =$$
$$= x_0 + 2r_0 - Ar_0 + (b - Ax_0 + 2Ar_0 - A^2r_0) = x_0 + 3r_0 + Ar_0 - A^2r_0$$

$$x_4 = x_3 + r_3$$

.....

$$x_i = x_{i-1} + r_{i-1}$$

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

$$x_i = x_0 + SPAN\{r_0, Ar_0, A^2r_0, ..., A^{i-1}r_0\}$$

$$x_i = x_0 + [r_0, Ar_0, A^2r_0, ..., A^{i-1}r_0]$$

El subespacio $K^i(A;r_0)$, de base $\left[r_0, Ar_0, A^2r_0, ..., A^{i-1}r_0\right]$, es llamado espacio de Krylov de dimensión i correspondiente a la matriz A y residuo inicial r_0.

1.2 - MÉTODOS ITERATIVOS PARA SISTEMAS NO SIMÉTRICOS.

En los sistemas lineales de ecuaciones cuya matriz de coeficientes es simétrica y definida positiva, el algoritmo del Gradiente Conjugado [26], constituye una robusta herramienta para su resolución. Con este método, prácticamente desarrollado a partir de 1970 y que en ausencia de errores de redondeo, alcanza teóricamente la solución exacta en n iteraciones como máximo, se obtienen soluciones aproximadas que cumplen los requisitos esenciales de:

(a) minimizar una cierta norma del vector residuo sobre un subespacio de Krylov generado por A, 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, el CG no es aplicable cuando la matriz del sistema no cumple las condiciones de simetría y de ser definida positiva. Para matrices no simétricas, no existen, en general, métodos que cumplan ambos requerimientos (a) y (b), sin añadir algún inconveniente ó desventaja. Antes bien, los métodos utilizados para estos sistemas, se obtienen generalmente por adaptarse preferentemente a uno de los dos requisitos enunciados anteriormente, pero no a los dos simultáneamente.

Distinguiremos así tres familias que acogen a la mayoría de los métodos que resuelven sistemas con matrices generales no singulares basados en estos subespacios:

- Métodos basados en la ecuación normal.
- Métodos de ortogonalización.
- Métodos de biortogonalización.

Una clasificación detallada se puede observar en [3] y [36].

1.3 - MÉTODOS BASADOS EN LA ECUACIÓN NORMAL

Sea el sistema $Ax = b$, siendo A una matriz real, no singular. Multiplicando por A^T, resultaría un sistema equivalente al primero, $A^T Ax = A^T b$, pero con matriz simétrica definida positiva, que recibe el nombre de sistema de ecuaciones normal, al que podemos aplicar el algoritmo del CG.

El método conocido como **CGN** [36], [7], construye una sucesión de vectores

$$x_i \in x_0 + \left[A^T r_0, (A^T A)A^T r_0, \ldots, (A^T A)^{n-1} A^T r_0\right]$$

con residuo mínimo en cada paso, sin efectuar el cálculo explícito del producto $A^T A$. El esquema de cálculo viene dado por el **algoritmo del CGN** siguiente:

Valor inicial x_0 ; $\mathbf{r_0} = b - A\mathbf{x_0}$;

$\beta_0 = 0, \mathbf{p_0} = 0;$

Si $\|\mathbf{r_i}\| / \|\mathbf{r_0}\| \geq \varepsilon, (i = 1,2,3,\ldots)$;

$$\mathbf{p_i} = \mathbf{A^T r_{i-1}} + \beta_{i-1}\mathbf{p_{i-1}} \ ;$$

$$\alpha_i = \frac{\left(\mathbf{A^T r_{i-1}}\right)^{\mathbf{T}}\left(\mathbf{A^T r_{i-1}}\right)}{\left(\mathbf{A p_i}\right)^{\mathbf{T}}\left(\mathbf{A p_i}\right)};$$

$$\mathbf{x_i} = \mathbf{x_{i-1}} + \alpha_i \mathbf{p_i} \ ;$$

$$\mathbf{r_i} = \mathbf{r_{i-1}} - \alpha_i \mathbf{A p_i} \ ;$$

$$\beta_i = \frac{\left(\mathbf{A^T r_i}\right)^{\mathbf{T}}\left(\mathbf{A^T r_i}\right)}{\left(\mathbf{A^T r_{i-1}}\right)^{\mathbf{T}}\left(\mathbf{A^T r_{i-1}}\right)};$$

fin

Al igual que el CG, los sucesivos valores de la solución aproximada se obtienen por una relación de recurrencia de tres términos, con lo que en principio se cumplen, asimismo, las condiciones (a) y (b), de minimizar una norma del vector residuo manteniendo un bajo coste computacional, a las que se ajustaba este método.

Sin embargo, el CGN presenta algunos inconvenientes:

- El número de condicionamiento del nuevo sistema resulta ser $K_2(A^TA) = K_2(A)^2$, y para sistemas mal condicionados, (valores de $K_2(A)$ relativamente grandes), disminuye la efectividad del algoritmo y la rapidez de convergencia.

- En cada iteración figuran dos productos matriz por vector, los correspondientes a las matrices A y A^T, aumentando el coste computacional del algoritmo.

Una mejora del algoritmo, que intenta corregir el posible empeoramiento del número de condición, la proponen Paige y Saunders aplicando el proceso de Lanczos al sistema simétrico $\begin{pmatrix} I & A \\ A^T & -\lambda^2 I \end{pmatrix}\begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$, dando origen al llamado LSQR [38].

1.4 - MÉTODOS DE ORTOGONALIZACIÓN. GMRES.

Estos métodos, basados en recurrencias largas, conllevan que el trabajo computacional y la cantidad de memoria requerida aumenten con el número de iteraciones. Consecuentemente, se hacen necesarios criterios de truncamiento, utilizándose un espacio de Krylov de dimensión k considerablemente menor que la dimensión n del sistema.

Con el GMRES [47], se construye una secuencia de vectores $\{x_i\}$, siendo

$$x_i \in x_0 + \left[r_0, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\right]$$

con la condición de mínimo para $\|\mathrm{r}_i\|_2 = \|b - Ax_i\|_2$, que equivale a la condición de ortogonalidad

$$r_i \perp \left[Ar_0, A^2 r_0, \ldots, A^i r_0\right]$$

que se implementa en sentido estricto, de tal forma que en el paso i-ésimo, se trabaja con combinaciones lineales de i vectores.

Esta condición hace que se verifique la propiedad (a) de minimización del vector residuo, obteniéndose un método robusto, de convergencia rápida y con curvas de convergencia monótonas sin "picos", pero a costa de exigir un elevado coste computacional y una alta disponibilidad de almacenaje que van incrementándose con el orden de las iteraciones. Estos aspectos hacen, consiguientemente, que el método no se ajuste a las dos características que habíamos enunciado anteriormente como requisitos ideales para estos métodos iterativos. El **algoritmo GMRES** correspondiente queda como sigue:

1- Inicialización

Valor inicial x_0

Dimensión subespacio Krylov k

$$\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$$

$$\mathbf{v_1} = \mathbf{r_0} / \|\mathbf{r_0}\|$$

2- Para $j = 1,2,\ldots,k$

$$h_{ij} = \left(\mathbf{A}\mathbf{v_j}, \mathbf{v_i}\right), i = 1,2,\ldots,j$$

$$\hat{\mathbf{v}}_{j+1} = \mathbf{A}\mathbf{v_j} - \sum_{i=1}^{j} h_{ij}\mathbf{v_i}$$

$$h_{j+1,j} = \left\| \hat{\mathbf{v}}_{j+1} \right\|$$

$$\mathbf{v}_{j+1} = \hat{\mathbf{v}}_{j+1} / h_{j+1,j}$$

3- Hallar $\mathbf{y}_k$ que minimiza $J(\mathbf{y}) = \left\| \beta \mathbf{e}_1 - \overline{\mathbf{H}}_k \mathbf{y} \right\|$, donde $\mathbf{e}_1$ es el primer vector de la base canónica en R^{k+1} y $\overline{\mathbf{H}}_k$ es la matriz de elementos h_{ij}. 4- Determinar $\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{V}_k \mathbf{y}_k$ donde $\mathbf{V}_k$ es la matriz de vectores columna $\left[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\right]$.

5- Calcular $\mathbf{r}_{n+1} = \mathbf{b} - \mathbf{A}\mathbf{x}_{n+1}$

Si $\left\| \mathbf{r}_{n+1} \right\| / \left\| \mathbf{r}_n \right\| \geq \varepsilon$, $\mathbf{v}_1 = \mathbf{r}_{n+1} / \left\| \mathbf{r}_{n+1} \right\|$. Ir a 2.

fin

Dado que se requieren *i* vectores en memoria en la *i*-ésima iteración, para restringir trabajo y almacenamiento se pueden ejecutar *m* iteraciones, obtener la solución aproximada y usar esta como inicialización en una siguiente aplicación del algoritmo. Este truncamiento y reinicio afecta a la convergencia del método. Recientemente se han desarrollado nuevas versiones que parecen ser más eficaces como el FGMRES [46], el GMRESR [56] ó el VGMRES [20].

1.5 - MÉTODOS DE BIORTOGONALIZACIÓN TIPO Bi-CG

El principal problema computacional del GMRES es que cada vector residuo generado ha de ser ortogonal con respecto a todos los vectores residuos anteriores.

En los métodos de bi-ortogonalización, las soluciones aproximadas se obtienen aplicando una condición de Galerkin y no por una propiedad de minimización estricta como en el GMRES. Así, las fórmulas de recurrencia son reducidas y el almacenamiento no aumenta con el número de iteraciones, pero presentando, en su defecto, un comportamiento irregular en la convergencia, con oscilaciones y abundantes "picos" que pueden conducir a criterios de parada con soluciones erróneas. Es por ello conveniente disponer en los algoritmos de una

"reinicialización" de los mismos, volviendo a calcular el parámetro que nos defina este criterio en base al vector residuo real $r_i = b - Ax_i$ correspondiente a la última iteración obtenida, en lugar de determinarlo con el vector obtenido con la fórmula de recurrencia respectiva.

1.5.1.- DOBLE GRADIENTE CONJUGADO (Bi-CG)

El Bi-CG [14], construye al igual que el GMRES, una secuencia de vectores

$$x_i \in x_0 + \left[r_0, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\right]$$

que conduce a una expresión

$$x_i = x_0 + q_{i-1}(A) r_0$$

donde $q_{i-1}(A)$ es un polinomio de grado (i-1).

Para el vector residuo quedaría,

$$r_i = b - Ax_i = b - Ax_0 - Aq_{i-1}(A) r_0 = r_0\left[1 - Aq_{i-1}(A)\right] = p_i(A) r_0$$

En este método, $p_i(A)$ no es determinado por la condición "estricta" que se impone en el GMRES, sino por la condición de ortogonalidad

$$r_i \perp \left[\bar{r}_0, A^T \bar{r}_0, \left(A^T\right)^2 \bar{r}_0, \ldots, \left(A^T\right)^{i-1} \bar{r}_0\right]$$

siendo $\bar{r}_0$, el vector residuo inicial del sistema auxiliar $A^T y = b$, que resolveremos paralelamente al sistema original $Ax = b$. determinando los sucesivos vectores residuos y direcciones conjugadas que figuran en el algoritmo, por las respectivas relaciones de recurrencia, pero sin analizar ni considerar su convergencia porque no influye en la solución del mismo.

Asimismo, se verifica

$$\bar{r}_i \perp \left[r_0, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\right]$$

cumpliéndose la relación de biortogonalidad $\left(r_j, \bar{r}_k\right) = 0$, para $j \neq k$.

De esta forma, el **algoritmo Bi-CG** contempla el cálculo de dobles vectores residuos y dobles direcciones conjugadas, además de efectuar dobles productos matriz por vector, debidos a la matriz del sistema $Ax = b$ y a la matriz del sistema auxiliar $A^T y = b$:

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

Elegimos $\underline{\mathbf{r}}_0$, tal que $\underline{\mathbf{r}}_0^{\mathrm{T}} \mathbf{r}_0 \neq 0$;

$\rho_0 = 1$;

$\underline{\mathbf{p}}_0 = \mathbf{p}_0 = \mathbf{0}$;

Si $\|\mathbf{r}_{i-1}\| / \|\mathbf{r}_0\| \geq \varepsilon$, (i=1,2,3,....),

$\rho_i = \underline{\mathbf{r}}_{i-1}^{\mathrm{T}} \mathbf{r}_{i-1}$;

$\beta_i = \rho_i / \rho_{i-1}$;

$\mathbf{p}_i = \mathbf{r}_{i-1} + \beta_i \mathbf{p}_{i-1}$;

$\underline{\mathbf{p}}_i = \underline{\mathbf{r}}_{i-1} + \beta_i \underline{\mathbf{p}}_{i-1}$;

$\mathbf{v} = \mathbf{A}\ \mathbf{p}_i$;

$\alpha_i = \rho_i / \left(\underline{\mathbf{p}}_i^{\mathrm{T}} \mathbf{v}\right)$;

$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i \mathbf{p}_i$;

$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i \mathbf{v}$;

$\underline{\mathbf{v}} = \mathbf{A}^{\mathrm{T}} \underline{\mathbf{p}}_i$;

$\underline{\mathbf{r}}_i = \underline{\mathbf{r}}_{i-1} - \alpha_i \underline{\mathbf{v}}$;

fin

El cálculo por iteración es el doble que el GC, pero presenta la ventaja frente a otras generalizaciones del mismo como el CGN, ó el método del Residuo mínimo [7], de conservar el condicionamiento de la matriz del sistema original por lo que su aplicación no empeora la convergencia de sistemas inicialmente mal condicionados.

1.5.2.- CONJUGATE GRADIENT SQUARED (CGS)

Sonnenveld [49], observa que en caso de convergencia del Doble Gradiente, los vectores $\{r_i\}$ y $\{\bar{r}_j\}$ tienden a cero, pero la convergencia de los "residuos auxiliares" no es explotada y solo 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 $\{r_i\}$.

Usando las expresiones polinómicas para los vectores residuos, $r_i = P_i(A)r_0$ y $\bar{r}_j = P_j(A^T)\bar{r}_0$, queda para la relación de biortogonalidad

$$(r_i, \bar{r}_j) = \left[P_i(A)r_0, P_j(A^T)\bar{r}_0\right] = \left[P_i(A)P_j(A)r_0, \bar{r}_0\right]$$

lo que sugiere trabajar con aproximaciones $\hat{x}_i$, para las cuales

$$\hat{r}_i = b - A\hat{x}_i = P_i^2(A)r_0.$$

Obtendremos el algoritmo en base a estos nuevos vectores residuos, efectuando las transformaciones pertinentes en el algoritmo del Bi-CG que permitan generar $P_i^2(A)r_0$. Para ello se usarán, además de las expresiones polinómicas de los vectores residuos, las correspondientes a las direcciones conjugadas de los sistemas original y auxiliar, $p_i = T_{i-1}(A)r_0$ y $\bar{p}_i = T_{i-1}(A^T)\bar{r}_0$.

En el Doble Gradiente Conjugado, $\rho_i = \bar{r}_{i-1}^T r_{i-1}$. Sustituyendo los vectores residuos de los sistemas original y auxiliar, calculando la traspuesta y utilizando el vector residuo que propone Sonnenveld,

$$\rho_i = \left[P_{i-1}(A^T)\bar{r}_0\right]^T\left[P_{i-1}(A)r_0\right] = \bar{r}_0^T P_{i-1}^2(A)r_0 = \bar{r}_0^T \hat{r}_{i-1}.$$

En el Doble Gradiente Conjugado, $\alpha_i = \rho_i / (\bar{p}_i^T A p_i)$. Sustituyendo las direcciones conjugadas por sus expresiones polinómicas y llamando $\hat{p}_i = T_{i-1}^2(A)r_0$, queda,

$$\alpha_i = \rho_i / \left(\bar{r}_0^T T_{i-1}(A) A T_{i-1}(A) r_0\right) = \rho_i / \left(\bar{r}_0^T A T_{i-1}^2(A) r_0\right) = \rho_i / \left(\bar{r}_0^T A \hat{p}_i\right).$$

La expresión del vector residuo del sistema original en el Doble Gradiente Conjugado es $r_i = r_{i-1} - \alpha_i A p_i$. Sustituyendo,

$$P_i(A) r_0 = P_{i-1}(A) r_0 - \alpha_i A T_{i-1}(A) r_0$$

y elevando al cuadrado,

$$P_i^2(A) = P_{i-1}^2(A) - \alpha_i A\left[2 P_{i-1}(A) T_{i-1}(A) - \alpha_i A T_{i-1}^2(A)\right].$$

Para hallar una expresión para el producto $P_{i-1}(A) T_{i-1}(A)$, partiremos de la igualdad, $T_i(A) = P_i(A) + \beta_{i+1} T_{i-1}(A)$, que permite calcular por recurrencia las direcciones conjugadas en el Doble Gradiente. Elevando al cuadrado y operando,

$$T_i(A) P_i(A) = P_i^2(A) + \beta_{i+1} P_i(A) T_{i-1}(A)$$

llamando $P_i(A) T_{i-1}(A) = q_i$, sustituyendo y efectuando operaciones,

$$\hat{r}_i = \hat{r}_{i-1} - \alpha_i A\left[2\hat{r}_{i-1} + 2\beta_i q_{i-1} - \alpha_i A \hat{p}_i\right]$$

que es la expresión del vector residuo en el CGS.

La fórmula de recurrencia para los nuevos vectores $\hat{p}_i$ se obtendrá partiendo de la relación, $T_{i-1}(A) = P_{i-1}(A) + \beta_i T_{i-2}(A)$. Elevando al cuadrado y sustituyendo queda,

$$\hat{p}_i = \hat{r}_{i-1} + 2\beta_i q_{i-1} + \beta_i^2 p_{i-1} = \hat{r}_{i-1} + \beta_i q_{i-1} + \beta_i\left(q_{i-1} + \beta_i p_{i-1}\right)$$

La relación de recurrencia para los vectores q_i . resultará de multiplicar miembro a miembro:

$$\left\{\begin{array}{l} T_{i-1}(A) = P_{i-1}(A) + \beta_i T_{i-2}(A) \\ P_i(A) = P_{i-1}(A)\alpha_i A T_{i-1}(A) \end{array}\right\}$$

quedando,

$$P_i(A) T_{i-1}(A) = P_{i-1}^2(A) + \beta_i P_{i-1}(A) T_{i-2}(A) - \alpha_i(A) T_{i-1}(A) P_{i-1}(A) - \alpha_i \beta_i A T_{i-1}(A) T_{i-2}(A)\,,$$

Operando resulta una expresión para $T_{i-1}(A)T_{i-2}(A)$, en función de vectores conocidos,

$$T_{i-1}(A)T_{i-2}(A) = \beta_i T_{i-2}^2(A) + \beta_i P_{i-1}(A)T_{i-2}(A)$$

y sustituyendo,

$$q_i = \hat{r}_i + \beta_i q_{i-1} - \alpha_i A\left[\hat{r}_{i-1} + \beta_i q_{i-1} + \beta_i(\beta_i p_{i-1} + q_{i-1})\right]$$

$$q_i = \hat{r}_{i-1} + \beta_i q_{i-1} - \alpha_i A\hat{p}_i$$

Igualmente, operando de forma similar, quedaría para las sucesivas soluciones aproximadas la expresión,

$$x_i = x_{i-1} + \alpha_i\left[\hat{r}_{i-1} + \beta_i q_{i-1} + (\hat{r}_{i-1} + \beta_i q_{i-1} - \alpha_i A\hat{p}_i)\right]$$

Con estas relaciones, resulta el **algoritmo CGS** [49], en el que no figuran los vectores residuos ni las direcciones conjugadas del sistema auxiliar, eliminando así los productos A^T por vector:

Valor inicial $\mathbf{x_0}$, $\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$;

Elegimos $\bar{\mathbf{r}}_0$, tal que $\bar{\mathbf{r}}_0^{\mathbf{T}}\mathbf{r_0} \neq 0$;

$\rho_0 = 1$;

$\mathbf{p_0} = \mathbf{q_0} = \mathbf{0}$;

Si $\|\mathbf{r_i}\|/\|\mathbf{r_0}\| \geq \varepsilon, (i = 1,2,3,...)$;

$\rho_i = \bar{\mathbf{r}}_0^{\mathbf{T}}\mathbf{r}_{\mathbf{i-1}}$;

$\beta_i = \rho_i/\rho_{i-1}$;

$\mathbf{u} = \mathbf{r}_{\mathbf{i-1}} + \beta_i\mathbf{q}_{\mathbf{i-1}}$;

$\mathbf{p}_i = \mathbf{u} + \beta_i(\mathbf{q}_{\mathbf{i-1}} + \beta_i\mathbf{p}_{\mathbf{i-1}})$;

$\mathbf{v} = \mathbf{A}\mathbf{p}_i$;

$\alpha_i = \rho_i/(\bar{\mathbf{r}}_0^{\mathbf{T}}\mathbf{v})$;

$\mathbf{q}_i = \mathbf{u} - \alpha_i\mathbf{v}$;

$\mathbf{w} = \mathbf{u} + \mathbf{q}_i$;

$\mathbf{x_i} = \mathbf{x_{i-1}} + \alpha_i \mathbf{w}$;

$\mathbf{r_i} = \mathbf{r_{i-1}} - \alpha_i \mathbf{A w}$;

fin

Cuando el Bi-CG converge, como la expresión polinómica del vector residuo es $r_i = P_i(A)r_0$, el operador $P_i(A)$ reduce el vector inicial r_0, a otro menor r_i, y, dado que en el CGS se utilizan vectores $\hat{r}_i = P_i^2(A)r_0$, cabe esperar que al aplicarlo dos veces resultará mejorada la convergencia. Eso ocurre con mucha frecuencia, y, en general, el CGS converge más rápidamente que el Bi-CG, pero al igual que en este, las curvas de convergencia presentan bruscas oscilaciones que pueden conducir a la cancelación del algoritmo con soluciones erróneas carentes de significado.

Por ello, asimismo, una vez alcanzado el criterio de parada adoptado, se debe calcular el vector residuo real, $r_i = b - Ax_i$, para proceder a una realimentación del algoritmo en caso necesario.

1.5.3.- BI-CGSTAB

De la misma forma que en el CGS los vectores residuos verifican $\hat{r}_i = P_i^2(A)r_0$, pueden construirse otros métodos iterativos en los cuales los valores aproximados de la solución x_i, sean obtenidos de modo que los residuos vengan dados por $r_i = Q_i(A)P_i(A)r_0$, con otro "polinomio reductor" $Q_i(A)$. Las relaciones de recurrencia del algoritmo donde intervenga este polinomio no han de ser excesivamente complicadas y los parámetros que figuren en su definición fácilmente optimizables.

Van der Vorst, en el BI-CGSTAB [55], propone obtener un vector residuo por aplicación sucesiva de dos "reductores" distintos,

$$\hat{r}_i = Q_i(A)P_i(A)r_0$$

en lugar de $\hat{r}_i = P_i^2(A)r_0$, como en el CGS.

Sugiere para $Q_i(A)$, la expresión

$$Q_i(A)=(1-w_1A)(1-w_2A)...(1-w_iA)$$

determinando w_i, por la condición de mínimo para r_i en la iteración i-ésima.

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

Las fórmulas de recurrencia en el Bi-CG para los sucesivos vectores residuos y direcciones conjugadas son,

$$r_i = P_i(A)r_0$$
$$p_{i+1} = T_i(A)r_0$$
$$T_i(A)r_0 = \left[P_i(A)+\beta_{i+1}T_{i-1}(A)\right]r_0$$
$$P_i(A)r_0 = \left[P_{i-1}(A)-\alpha_i AT_{i-1}(A)\right]r_0$$

y para el Bi-CGSTAB se obtiene la relación,

$$\hat{r}_i = Q_i(A)P_i(A) = (1-w_iA)Q_{i-1}(A)-\left[P_{i-1}(A)-\alpha_i AT_{i-1}(A)\right]r_0 =$$
$$=\left[Q_{i-1}(A)P_{i-1}(A)-\alpha_i AQ_{i-1}(A)T_{i-1}(A)\right]r_0 - w_iA\left[Q_{i-1}(A)P_{i-1}(A)-\alpha_i AQ_{i-1}(A)T_{i-1}(A)\right]r_0$$

Igualando $Q_{i-1}(A)T_{i-1}(A)r_0$ a un nuevo vector $\hat{p}_i$, queda,

$$\hat{r}_i = \hat{r}_{i-1} - \alpha_i A\hat{p}_i - w_iA(r_{i-1}-\alpha_i A\hat{p}_i).$$

Necesitamos una relación de recurrencia para este vector $\hat{p}_i$ que obtendremos, asimismo, a partir del Bi-CG:

$$\hat{P}_{i+1} = Q_i(A)T_i(A) = Q_i(A)\left[P_i(A)+\beta_{i+1}T_{i-1}(A)\right]r_0 -$$
$$= Q_i(A)P_i(A)r_0 + \beta_{i+1}(1-w_iA)Q_{i-1}(A)T_{i-1}(A)r_0 =$$
$$= Q_i(A)P_i(A)r_0 + \beta_{i+1}Q_{i-1}(A)T_{i-1}(A)r_0 - \beta_{i+1}w_iAQ_{i-1}(A)r_0$$

Sustituyendo, resulta

$$\hat{p}_{i+1} = \hat{r}_i + \beta_{i+1}\hat{p}_i - \beta_{i+1}w_iA\hat{p}_i\ .$$

Incorporando la ortogonalidad de $P_i(A)r_0$, respecto a los vectores $R_j(A^T)\bar{r}_0$, con $j<i$, determinaremos la expresión de $\beta_{i+1}=\rho_{i+1}/\rho_i$, en función de los nuevos vectores. Para el numerador tenemos,

$$\rho_{i+1}=\left[P_i(A^T)\bar{r}_0,P_i(A)r_0\right]=\left[(-1)^i\alpha_1\alpha_2\alpha_3\ldots\alpha_i(A^T)^i\bar{r}_0,P_i(A)r_0\right]=$$
$$=\frac{\alpha_1\alpha_2\alpha_3\ldots\alpha_i}{w_1w_2w_3\ldots w_i}\left[Q_i(A^T)\bar{r}_0,P_i(A)r_0\right]=\frac{\alpha_1\alpha_2\alpha_3\ldots\alpha_i}{w_1w_2w_3\ldots w_i}\left[\bar{r}_0^TQ_i(A)P_i(A)r_0\right]=$$
$$=\frac{\alpha_1\alpha_2\alpha_3\ldots\alpha_i}{w_1w_2w_3\ldots w_i}\bar{r}_0^T\hat{r}_i=\frac{\alpha_1\alpha_2\alpha_3\ldots\alpha_i}{w_1w_2w_3\ldots w_i}\hat{\rho}_{i+1}$$

quedando entonces,

$$\beta_{i+1}=\frac{\rho_{i+1}}{\rho_i}=\frac{\alpha_i}{w_i}\frac{\hat{\rho}_{i+1}}{\hat{\rho}_i}$$

Operando de forma similar para obtener la nueva relación que defina al parámetro α_i,

$$\alpha_i=\frac{\rho_i}{\bar{p}_i^TAp_i}=\frac{\rho_i}{\left[T_{i-1}(A^T)\bar{r}_0\right]^TAT_{i-1}(A)r_0}=\frac{\rho_i}{\left[T_{i-1}(A^T)\bar{r}_0,AT_{i-1}(A)r_0\right]}=$$

$$=\frac{\rho_i}{\left[(-1)^{i-1}\alpha_1\alpha_2\alpha_3\ldots\alpha_{i-1}(A^T)^{i-1}\bar{r}_0,AT_{i-1}(A)r_0\right]}=$$

$$=\frac{w_1w_2w_3\ldots w_{i-1}}{\alpha_1\alpha_2\alpha_3\ldots\alpha_{i-1}}\frac{\rho_i}{\left[(-1)^{i-1}w_1w_2w_3\ldots w_{i-1}(A^T)^{i-1}\bar{r}_0,AT_{i-1}(A)r_0\right]}=$$

$$=\frac{w_1w_2w_3\ldots w_{i-1}}{\alpha_1\alpha_2\alpha_3\ldots\alpha_{i-1}}\frac{\rho_i}{\left[Q_{i-1}(A^T)\bar{r}_0,AT_{i-1}(A)r_0\right]}=\frac{\hat{\rho}_i}{\bar{r}_0^TA\hat{p}_i(A)}.$$

La relación de recurrencia para el vector solución incógnita se obtendría de forma análoga,

$$x_i=x_{i-1}+\alpha_i\hat{p}_i+w_i(\hat{r}_{i-1}-\alpha_iA\hat{p}_i)$$

El parámetro w_i, que interviene en la construcción del "polinomio reductor" $Q_i(A)$ y que figura en las relaciones de recurrencia del algoritmo, lo obtendremos con la condición de minimizar la norma del vector residuo $\|\hat{r}_i\|_2$. Introduciendo, a efectos operativos, los vectores v_i, s_i, t_i, definidos por las relaciones $A\hat{p}_i = v_i$, $s_i = \hat{r}_{i-1} - \alpha_i v_i$, $t_i = As_i$, podremos escribir este vector como

$$\hat{r}_i = s_i - w_i t_i$$

y la norma $\|.\|_2$, sería

$$F = \|\hat{r}_i\|_2 = (s_i - w_i t_i)^T (s_i - w_i t_i).$$

Igualando la derivada a cero, $\frac{dF}{dw_i} = 0$ y operando, determinaremos la expresión de w_i,

$$w_i = \frac{s_i^T t_i}{t_i^T t_i}.$$

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, podemos escribir el **algoritmo Bi-CGSTAB:**

Valor inicial $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;

Elegimos $\bar{\mathbf{r}}_0$, tal que $\bar{\mathbf{r}}_0^{\mathrm{T}} \mathbf{r}_0 \neq 0$;

$\rho_0 = \alpha_0 = w_0 = 1$;

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

Si $\|\mathbf{r}_{i-1}\| / \|\mathbf{r}_0\| \geq \varepsilon, (i = 1,2,3,...)$;

$\rho_i = \bar{\mathbf{r}}_0^{\mathrm{T}} \mathbf{r}_{i-1}$;

$\beta_i = (\rho_i / \rho_{i-1})(\alpha_{i-1} / w_{i-1})$;

$\mathbf{p}_i = \mathbf{r}_{i-1} + \beta_i (\mathbf{p}_{i-1} + w_{i-1} \mathbf{v}_{i-1})$;

$\mathbf{v}_i = \mathbf{A}\mathbf{p}_i$;

$$\alpha_i = \rho_i / (\bar{r}_0^T v_i);$$

$$\mathbf{s} = \mathbf{r}_{i-1} - \alpha_i \mathbf{v}_i \ ;$$

$$\mathbf{t} = \mathbf{A}\mathbf{s};$$

$$w_i = (\mathbf{t}^T \mathbf{s}) / (\mathbf{t}^T \mathbf{t});$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i \mathbf{p}_i + w_i \mathbf{s};$$

$$\mathbf{r}_i = \mathbf{s} - \mathbf{w}_i \mathbf{t};$$

fin

En el algoritmo figuran dos productos matriz por vector y cuatro productos internos, mientras el CGS exige los mismos productos matriz-vector pero solo dos productos internos.

En la mayoría de los casos la convergencia del BI-CGSTAB es más rápida y uniforme que en el CGS, e incluso con menor carga computacional para alcanzar una determinada tolerancia, pues la reducción del número de iteraciones compensa el pequeño incremento computacional por iteración que suponen estos dos productos adicionales.

Sin embargo, la presencia en ocasiones, de forma similar a lo que ocurre en el Bi-CG y CGS, de bruscas variaciones en la norma del vector residuo hacen aconsejable proceder al final del algoritmo al cálculo del vector residuo real correspondiente a la solución aproximada obtenida, para efectuar la corrección oportuna en caso que fuese necesaria.

1.5.4.- QMR

El método propuesto por R. Freund y N. Nachtigal [17], [18], usa una variante del proceso de Lanczos para generar una base en el subespacio de Krylov inducido por la matriz del sistema. Conservando la propiedad de los métodos tipo-

gradiente de utilizar relaciones de recurrencia que impliquen bajo coste computacional, intenta suavizar las fluctuaciones que tienen lugar en la convergencia debidas al hecho de no usar una condición de minimización para la generación de los vectores residuos. Plantea para ello, sin llegar a la condición estricta que impone el GMRES, una técnica para "cuasi-minimizar" estos vectores, condición que da nombre al método.

Aplicando esta técnica de "suavizado de residuos" al CGS y al BI-CGSTAB [59], [15], se obtienen los **algoritmos** llamados QMRCGS y **QMRCGSTAB** [6]. Exponemos aquí el **algoritmo QMRCGSTAB**:

Aproximación inicial $\mathbf{v}_0$;

Elegimos $\underline{\mathbf{r}}_0$ arbitrario, tal que $\underline{\mathbf{r}}_0^{\mathrm{T}}\,\mathbf{r}_0 \neq 0$.

$\rho_0 = \alpha_0 = \omega_0 = 1;\quad \theta_0 = \eta_0 = 0;\quad \gamma = \|\mathbf{r}_0\|;\quad \mathbf{p}_0 = \mathbf{y}_0 = \mathbf{d}_0 = \mathbf{0};$

Si $\|\mathbf{r}_{i-1}\|/\|\mathbf{r}_0\| \geq \varepsilon$ (i=1,2,3,....),

$\rho_i = \underline{\mathbf{r}}_0^{\mathrm{T}}\,\mathbf{r}_{i-1};\quad \beta_i = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1});\quad \mathbf{p}_i = \mathbf{r}_{i-1} + \beta_i(\mathbf{p}_{i-1} - \omega_{i-1}\,\mathbf{y}_{i-1});$

$\alpha_i = \rho_i/(\underline{\mathbf{r}}_0^{\mathrm{T}}\,\mathbf{y}_i);\quad \mathbf{s}_i = \mathbf{r}_{i-1} - \alpha_i\,\mathbf{y}_i;$

Primera cuasi-minimización y solución intermedia

$\underline{\theta}_i = \|\mathbf{s}_i\|/\gamma;\quad c = 1/\sqrt{1+\underline{\theta}_i^2};\quad \underline{\gamma} = \gamma\,\underline{\theta}_i\,c;\quad \underline{\eta}_i = c^2\alpha_i;$

$\underline{\mathbf{d}}_i = \mathbf{p}_i + \frac{\theta_{i-1}^2\eta_{i-1}}{\alpha_i}\mathbf{d}_{i-1};\quad \underline{\mathbf{v}}_i = \mathbf{v}_{i-1} + \underline{\eta}_i\,\underline{\mathbf{d}}_i$

Cálculo de $\mathbf{t}_i, \omega_i, \mathbf{r}_i$

$\omega_i = (\mathbf{t}_i^{\mathrm{T}}\mathbf{s}_i)/(\mathbf{t}_i^{\mathrm{T}}\mathbf{t}_i);\quad \mathbf{r}_i = \mathbf{s}_i - \omega_i\,\mathbf{t}_i;$

Segunda cuasi-minimización y solución

$\theta_i = \|\mathbf{r}_i\|/\underline{\gamma};\quad c = 1/\sqrt{1+\theta_i^2};\quad \gamma = \underline{\gamma}\,\theta_i\,c;\quad \eta_i = c^2\omega_i;$

$\mathbf{d}_i = \mathbf{s}_i + \frac{\theta_i^2\eta_i}{\omega_i}\underline{\mathbf{d}}_i;\quad \mathbf{v}_i = \underline{\mathbf{v}}_i + \eta_i\,\mathbf{d}_i;$

fin

Hubiese sido muy extenso el relacionar todos los métodos iterativos, con sus variantes respectivas, que se han desarrollado en los últimos años

para la resolución de sistemas de ecuaciones lineales. Con todo, hemos expuesto los más representativos de las distintas familias en que se pueden encuadrar el conjunto de los mismos, que son, con mucho, los más utilizados.

Para matrices no simétricas, es difícil decidir y elegir "a priori" el método que proporcione mejores resultados. Las variantes del Bi-CG, CGS y Bi-CGSTAB, por su facilidad de implementación y su rapidez de convergencia, son efectivos en gran número de problemas. En caso de convergencia excesivamente lenta de estos, son los métodos tipo GMRES los más indicados. El LSQR siempre converge teóricamente, pero puede ser a costa de un gran número de iteraciones, especialmente en los sistemas mal condicionados. En [36] y [48], se comparan resultados obtenidos en diversos problemas utilizando estos algoritmos.

CAPÍTULO 2

PRECONDICIONAMIENTO

El comportamiento de los métodos de Krylov en la resolución de sistemas de ecuaciones lineales mejora sustancialmente con las técnicas de precondicionamiento. Es más, en ciertos problemas, como justificaremos en las aplicaciones prácticas, es indispensable la implementación de un adecuado precondicionador para asegurar la convergencia con los métodos propuestos.

Introducimos en este capítulo ideas generales sobre precondicionamiento de sistemas y relacionamos algunos de los precondicionadores más utilizados.

2.1 - CONDICIONAMIENTO DE UN SISTEMA

Cuando en un sistema, variaciones relativas mínimas en los coeficientes de la matriz ó en el vector segundo miembro, producen variaciones relativas considerablemente más significativas en la solución del mismo, se dice que está mal condicionado.

Si representamos los nuevos sistemas con estos incrementos, por

$$A(x+\Delta x)=b+\Delta b$$
$$(A+\Delta A)(x+\Delta x)=b$$

se comprueba, [6],[17], que

$$\frac{\|\Delta x\|}{\|x\|}=\left[\|A\|\|A^{-1}\|\right]\frac{\|\Delta b\|}{\|b\|}$$

$$\frac{\|\Delta x\|}{\|x+\Delta x\|} = \left[\|A\|\|A^{-1}\|\right]\frac{\|\Delta A\|}{\|A\|}.$$

Al factor $K(A)=\left[\|A\|\|A^{-1}\|\right]$, donde $\|\ .\ \|$ es una norma matricial definida en el conjunto de las matrices cuadradas, se le denomina número de condicionamiento del sistema. Evidentemente, cuanto menor sea su valor, menor será la variación de la solución ante las posibles fluctuaciones en los valores de los elementos de A y b.

Usando la norma espectral $\|\ .\ \|_2$, $K(A)=\frac{\mu_M}{\mu_m}\geq 1$, donde μ_M y μ_m son, respectivamente, los valores singulares máximo y mínimo de la matriz del sistema, valores que se pueden calcular por $\mu_i=\sqrt{\lambda_i\left(AA^T\right)}$, siendo $\lambda_i\left(AA^T\right)$ los correspondientes valores propios de AA^T.

Cuando A es simétrica, $\mu_i(A)\equiv\lambda_i(A)$, y el número de condicionamiento sería $K(A)=\frac{\lambda_M}{\lambda_m}\geq 1$.

Con esta definición, la "bondad" del condicionamiento de una matriz viene dada, en principio, por la proximidad de $K(A)$ a la unidad, de tal forma, que en el caso que los valores singulares extremos coincidieran, el número de condicionamiento adoptaría este valor óptimo.

2.2 - TÉCNICA DE PRECONDICIONAMIENTO

En numerosas ocasiones, la matriz y el vector segundo miembro del sistema se calculan de forma aproximada, pudiendo existir ciertas diferencias con los valores numéricos que reflejen exactamente el problema. En estos casos, un mal condicionamiento del sistema afectaría negativamente a la convergencia.

Se hace necesario así, mejorar este condicionamiento utilizando adecuadas técnicas de precondicionamiento. Técnicas que, en general, consisten en transformar el sistema en otro de idéntica solución pero con menor $K(A)$.

Para ello multiplicaremos en $Ax = b$ por una matriz M, llamada matriz de precondicionamiento,

$$MAx = Mb$$

tal que $K(MA) < K(A)$.

El menor valor de $K(A)$ corresponde a $M = A^{-1}$, quedando $K(A^{-1}A) = 1$, que es, obviamente, el caso ideal y el sistema convergería en una sola iteración, pero el coste computacional del cálculo de A^{-1} equivaldría a resolverlo por un método directo.

Esta circunstancia sugiere para M una matriz lo más próxima posible a A^{-1}, sin que su determinación suponga un elevado coste. Generalmente, se opta por considerar como matriz de precondicionamiento a M^{-1}, y obtener M como aproximación de A, escribiendo,

$$M^{-1}Ax = M^{-1}b.$$

Por otro lado, en los algoritmos precondicionados de los distintos métodos figuran productos de la matriz inversa por un vector que no deben exigir excesivo trabajo adicional. Para ello, la matriz M debe ser fácilmente invertible, por ejemplo una matriz diagonal, ó una matriz factorizada adecuadamente para efectuar esos productos por procesos de remonte, sin necesidad de calcular M^{-1}.

2.3 - CONVERGENCIA Y PRECONDICIONAMIENTO

El precondicionamiento de un sistema tiene por finalidad mejorar la convergencia del método aplicado respecto a la convergencia del sistema sin precondicionar. Consideremos esta influencia en sistemas simétricos y no simétricos.

2.3.1- SISTEMAS SIMÉTRICOS.

En la resolución de sistemas simétricos, la razón de convergencia del Gradiente Conjugado

$$\|x - x_i\|_A \leq 2\left(\frac{\sqrt{K(A)-1}}{\sqrt{K(A)+1}}\right)^i \|x - x_0\|_A \quad [29]$$

depende del número de condicionamiento $K(A)$, función de los autovalores mayor y menor de la matriz del sistema.

Sin embargo, en la práctica, después de un cierto número de iteraciones la convergencia se hace "superlineal", como si el número de condicionamiento inicial fuese sustituído por otro menor, de tal forma que la razón de convergencia depende, además, de la distribución total de los valores propios de A.

Las técnicas de precondicionamiento, tienen por objeto transformar el sistema $Ax = b$, en otro $M^{-1}Ax = M^{-1}b$, con unos nuevos autovalores que conduzcan a una convergencia más rápida, bien disminuyendo el número de condicionamiento $K(A)$, bien mejorando la distribución de los autovalores más pequeños del espectro [25], [53] .

Las distintas iteraciones x_i, del sistema precondicionado verificarán,

$$\|x - x_i\|_{M^{-1}A} \leq 2\left(\frac{\sqrt{K(M^{-1}A)-1}}{\sqrt{K(M^{-1}A)+1}}\right)^i \|x - x_0\|_{M^{-1}A}$$

con $x_i \in x_0 + K^i(M^{-1}A; M^{-1}r_0)$.

2.3.2- SISTEMAS NO SIMÉTRICOS.

En sistemas no simétricos, es complicado probar que el sistema precondicionado resultante posee un espectro de autovalores que mejore la convergencia

respecto al sistema original. Pero, por analogía, y, aún sin demostraciones matemáticas que lo confirmen, se podría esperar un comportamiento similar. Esta conclusión, corroborada numéricamente en todas las aplicaciones, nos va a permitir tratar técnicas de precondicionamiento en los métodos tipo doble-gradiente, al igual que se utilizan en el Gradiente Conjugado, con las salvedades correspondientes que contemplen la no simetría del sistema.

2.4 - SISTEMAS PRECONDICIONADOS

Dependiendo de la forma de plantear el producto de la inversa de la matriz de precondicionamiento por la matriz del sistema, y aprovechando la descomposición en factores de aquella, se distinguen los siguientes casos:

- Precondicionamiento por la izquierda

$$M^{-1}Ax = M^{-1}b \begin{Bmatrix} M^{-1}A = \tilde{A} \\ M^{-1}b = \tilde{b} \end{Bmatrix} \tilde{A}x = \tilde{b}$$

- Precondicionamiento por la derecha

$$AM^{-1}Mx = b \begin{Bmatrix} AM^{-1} = \tilde{A} \\ Mx = \tilde{x} \end{Bmatrix} \tilde{A}\tilde{x} = b$$

- Precondicionamiento por ambos lados

Expresando M factorizada como $M = M_1 M_2$,

$$M_1^{-1}AM_2^{-1}M_2x = M_1^{-1}b \begin{Bmatrix} M_1^{-1}AM_2^{-1} = \tilde{A} \\ M_2x - \tilde{x} \\ M_1^{-1}b = \tilde{b} \end{Bmatrix} \tilde{A}\tilde{x} = \tilde{b}$$

Las características de cada problema y, en definitiva, de la matriz A, del precondicionador utilizado e, incluso, de la tolerancia exigida para el criterio de

parada adoptado, hacen más eficiente una forma u otra de precondicionamiento, sin que se puedan establecer "a priori" bases de elección que nos inclinen por una de ellas.

2.5- PRECONDICIONADORES EN SISTEMAS NO SIMÉTRICOS.

En principio, los precondicionadores han de cumplir los requisitos generales:

a) Facilidad de implementación, evitando un coste computacional excesivo del producto de M^{-1} por cualquier vector.

b) Mejorar la convergencia.

Una matriz que sea una aproximación más ó menos cercana de A, obtenida con estos criterios, puede ser un buen precondicionador.

Los ordenadores con múltiples procesadores en paralelo ofrecen una gran versatilidad. Grote y Simon [24] proponen obtener M como aproximación de A^{-1} minimizando una norma que reduce el cálculo a n "pequeños" problemas independientes de mínimos cuadrados. Asimismo, en [48] y [37] se plantea efectuar esta aproximación por una expresión polinómica $P(A)$.

El campo de posibles precondicionadores aplicables en ordenadores que no tengan esa característica es también muy amplio. Expondremos, en líneas generales, algunos de los más utilizados.

2.5.1 - PRECONDICIONADORES DIAGONAL, SOR Y SSOR.

El método de Richardson, método iterativo muy simple, de relación de recurrencia $x_{i+1} = x_i + \alpha(b - Ax_i)$, con $\alpha > 0$, permite, por comparación con otros métodos iterativos, definir ciertas matrices utilizables como precondicionadores.

Para ello, contrastaremos la fórmula de recurrencia para la solución que resulta de aplicar el método de Richardson al sistema precondicionado por

una matriz genérica M, con la fórmula correspondiente que se obtiene aplicando los métodos, teóricamente superiores, de Jacobi, SOR y SSOR al sistema sin precondicionar.

- Precondicionador Diagonal

Aplicando el método de Richardson al sistema precondicionado $M^{-1}Ax = M^{-1}b$, queda, para el cálculo de los sucesivos valores de la solución,

$$x_{i+1} = x_i + \alpha\left(M^{-1}b - M^{-1}Ax_i\right)$$

y, multiplicando por la matriz de precondicionamiento,

$$Mx_{i+1} = Mx_i + \alpha\left(b - Ax_i\right).$$

Por otro lado, considerando la descomposición de la matriz A de la forma $A = D - E - F$, (con D matriz diagonal formada por los elementos de la diagonal principal de la misma y E y F matrices triangulares), y utilizando el método de Jacobi para la resolución del sistema $Ax = b$, resulta,

$$x_{i+1} = D^{-1}\left(E + F\right)x_i + D^{-1}b$$

multiplicando por la matriz diagonal, y operando,

$$Dx_{i+1} = Dx_i + \left(b - Ax_i\right).$$

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, menos robusto y más simple, cuando este se aplica al sistema precondicionado con la matriz diagonal D.

Resulta así un precondicionador elemental, que se conoce como precondicionador Diagonal, fácil de implementar y con matriz inversa que se determina con muy bajo coste computacional.

- Precondicionador SOR

Aplicando el método SOR al sistema $Ax=b$, y con la misma descomposición anterior $D=A+E+F$, donde w es el llamado parámetro de relajación, queda para la fórmula de recurrencia de la solución,

$$x_{i+1}=(D-wE)^{-1}\left[(1-w)D+wF\right]x_i+w(D-wE)^{-1}b$$
$$(D-wE)x_{i+1}=(D-wE)x_i+w(b-Ax_i)$$

Comparando de nuevo con el método de Richardson aplicado al sistema precondicionado $Mx_{i+1}=Mx_i+(b-Ax_i)$, definamos, en esta ocasión, la matriz de precondicionamiento $M=(D-wE)$.

- Precondicionador SSOR

Aplicando ahora el método SSOR al sistema sin precondicionar, se obtiene para la solución,

$$x_{i+1}=\left(\frac{D}{w}-F\right)^{-1}\left(\frac{1-w}{w}D+E\right)\left(\frac{D}{w}-E\right)^{-1}\left(\frac{1-w}{w}D+F\right)x_i+\left(\frac{D}{w}-F\right)^{-1}\frac{2-w}{w}D\left(\frac{D}{w}-E\right)^{1}b$$

operando, para expresar la relación de forma que se pueda comparar con $Mx_{i+1}=Mx_i+\alpha(b-Ax_i)$, queda,

$$\frac{1}{w(2-w)}(D-wE)D^{-1}(D-wF)x_{i+1}=\frac{1}{(2-w)}(D-wE)D^{-1}(D-wF)x_i+(b-Ax_i)$$

con lo que resulta como matriz de precondicionamiento,

$$M=\frac{1}{w(2-w)}(D-wE)D^{-1}(D-wF).$$

En el caso de aplicar este precondicionador a sistemas simétricos, la expresaremos como producto de dos matrices triangulares traspuestas,

$$M = \left[\frac{(D-wE)D^{-1/2}}{\sqrt{w(2-w)}}\right]\left[\frac{(D-wE)D^{-1/2}}{\sqrt{w(2-w)}}\right]^T$$

ya que, en estos casos, $(D-wF)=(D-wE)^T$.

Para sistemas no simétricos, lo escribiremos como producto de dos matrices triangulares, inferior y superior, respectivamente,

$$M = (I-wED^{-1})\left[\frac{D-wF}{w(2-w)}\right].$$

2.5.2 - FACTORIZACIONES INCOMPLETAS

La propiedad que debe verificar, en principio, una matriz de precondicionamiento M de ser una aproximación, más ó menos cercana, de la matriz del sistema, sugiere como un procedimiento para obtener aquella, descomponer A de la forma, $A \approx A_1 A_2$, de manera que no suponga un excesivo esfuerzo computacional, y adoptar para la misma,

$$M = A_1 A_2 .$$

Además de otras factorizaciones,como la ILLU [57], [58], por bloques, destacaremos la basada en la factorización en dos matrices triangulares LU, usualmente utilizada en los métodos directos.

- Precondicionador ILU(0)

Resulta de descomponer A en dos matrices triangulares, inferior y superior, respectivamente, L y U,

$$A \approx LU = M$$

cuyos elementos t_{ij} son tales que

$$t_{ij} = 0 \text{ si } a_{ij} = 0$$
$$(A - LU)_{ij} = 0 \text{ si } a_{ij} \neq 0$$

es decir, que los elementos nulos de la matriz del sistema, continúan siendo ceros en las posiciones respectivas de las matrices triangulares.

Si no incurriésemos en esta simplificación, el coste computacional se incrementaría y equivaldría a resolver el sistema por un método directo.

- Precondicionadores ILU(n)

Para las matrices tridiagonales ó pentadiagonales que resultan de la discretización de problemas con ecuaciones en derivadas parciales elípticas, se pueden aplicar otros niveles de factorización, consistentes en "rellenar" alguna diagonal de las matrices factores de la descomposición, que en ILU(0) serían nulas. La aproximación sería mayor a costa de incrementar el trabajo computacional de la obtención de la descomposición.

Son los precondicionadores DIAGONAL, SSOR e ILU(0), los que utilizaremos posteriormente en las aplicaciones, y nos servirán para contrastar la mejora que aportan a la eficiencia de los diferentes algoritmos.

El DIAGONAL, es con mucho, el que menos esfuerzo computacional exige. Se calcula directamente tomando los elementos de la diagonal principal de A y los productos matriz inversa por vector se efectúan, asimismo, de forma inmediata. Sin embargo, su aplicación se ve reducida. En muchos sistemas mal condicionados, representativos de problemas físicos con capas límites, singularidades ó condiciones de contorno especiales, no mejora sustancialmente la convergencia.

Los precondicionadores SSOR e ILU(0), exigen más esfuerzo computacional inicial para su construcción, sobre todo este último, y los productos

matriz inversa por vector se realizan por procesos de remonte, pero su aplicación da lugar a buenos resultados en sistemas que con el precondicionador DIAGONAL no convergen.

CAPÍTULO 3

ALMACENAMIENTO COMPACTO

El carácter "sparse" de las matrices de los sistemas a que hacemos referencia en este trabajo, obtenidos, en general, al discretizar ecuaciones en derivadas parciales, hace que el número de elementos distintos de cero en una matriz de orden n sea sustancialmente inferior a n^2.

Existen distintas formas de almacenamiento para estas matrices, que datan de las primeras experiencias numéricas en el campo de la Ingeniería. El primer intento de reducir almacenamiento innecesario a destacar apareció en Zienkiewicz [60], donde se introdujo la técnica de almacenamiento en perfil o línea de cielo. Se trataba de almacenar (en el caso general de una matriz no simétrica) la parte triangular superior e inferior por columnas y filas respectivamente, empezando en cada una por el primer elemento no nulo. Luego, estos bloques se disponían en un vector y se accedían a ellos mediante otro vector de punteros. No cabe duda de que el principal inconveniente de este método estribaba en que existían aún muchos elementos nulos dentro del perfil que eran almacenados. Otra idea consistió en almacenar una banda de la matriz que contuviera a todos los elementos no nulos. A pesar de ser un método óptimo para matrices con estructura en banda, adolecía en general del mismo defecto que el anterior, ya que, en general, en la banda solían aparecer diversos elementos nulos que eran almacenados.

Una técnica más reciente, que tuvo su auge con la aparición de ordenadores paralelos, es la de almacenamiento elemento a elemento. En este caso, no se construía la matriz de rigidez, sino que todas las operaciones se realizaban a partir de las matrices elementales que eran las almacenadas realmente (ver por ejemplo Montero y otros [33]. Cada matriz elemental se introducía por columnas en un macrovector, al

cual se accedía mediante punteros. La efectividad de estos métodos, sin embargo, está directamente relacionada con la utilización del paralelismo masivo en la computación.

En la aplicación de los algoritmos que se exponen en capítulos sucesivos se opera con la matriz del sistema y con las matrices de los precondicionadores establecidos. Se ha adoptado un almacenamiento en forma compacta, [44], [50], que supone un ahorro computacional a ese efecto, sustituyendo cada matriz por dos matrices de perfil irregular, una con los elementos no nulos correspondientes y otra donde se reflejen las posiciones respectivas que ocupan estos elementos. Esto nos permite almacenar en una única matriz de elementos no nulos cada precondicionador, aunque esté factorizado en dos matrices triangulares, y mantener la misma matriz indicativa de posiciones para la matriz del sistema y para estas matrices de precondicionamiento.

3.1 - ALMACENAMIENTO DE LA MATRIZ DEL SISTEMA

En una de las matrices dispondremos en la primera columna los elementos de la diagonal, y en las restantes los elementos no nulos de cada fila. En la segunda matriz, de posiciones, figurarán en la primera columna el número de elementos no nulos de cada fila, y en las siguientes las posiciones de columnas respectivas que ocupan.

Por ejemplo, para una matriz

$$\begin{pmatrix} a_{11} & a_{12} & 0 & a_{14} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ a_{41} & 0 & a_{43} & a_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{pmatrix}$$

las matrices de elementos no nulos y la matriz de posiciones serían,

$$\begin{pmatrix} a_{11} & a_{12} & a_{14} \\ a_{22} & a_{21} & \\ a_{33} & a_{32} & a_{34} \\ a_{44} & a_{41} & a_{43} \\ a_{55} & a_{56} & \\ a_{66} & a_{65} & \end{pmatrix} \qquad \begin{pmatrix} 3 & 2 & 4 \\ 2 & 1 & \\ 3 & 2 & 4 \\ 3 & 1 & 3 \\ 2 & 6 & \\ 2 & 5 & \end{pmatrix}$$

3.2 - ALMACENAMIENTO DE LA MATRIZ DE PRECONDICIONAMIENTO ILU FACTORIZADA

Ya hemos comentado que en la factorización ILU(0), los elementos nulos de la matriz del sistema continuaban siendo ceros en las posiciones respectivas de las matrices triangulares. Además, convenimos en igualar a la unidad todos los elementos de la diagonal de la matriz triangular inferior L, y escribir los elementos que resulten de la diagonal principal de A después de factorizar, en la diagonal de la matriz triangular superior U.

Esto nos va a permitir utilizar una sola matriz de elementos para almacenar conjuntamente las dos matrices triangulares y conservar la misma matriz de posiciones de la matriz del sistema A.

Por ejemplo, la factorización para la matriz del sistema anterior quedaría de la forma,

$$\begin{pmatrix} a_{11} & a_{12} & 0 & a_{14} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ a_{41} & 0 & a_{43} & a_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{pmatrix} \approx \begin{pmatrix} 1 & & & & & \\ l_{21} & 1 & & & & \\ 0 & l_{32} & 1 & & & \\ l_{41} & 0 & l_{43} & 1 & & \\ 0 & 0 & 0 & 0 & 1 & \\ 0 & 0 & 0 & 0 & l_{65} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & 0 & u_{14} & 0 & 0 \\ & u_{22} & 0 & 0 & 0 & 0 \\ & & u_{33} & u_{34} & 0 & 0 \\ & & & u_{44} & 0 & 0 \\ & & & & u_{55} & u_{56} \\ & & & & & u_{66} \end{pmatrix}$$

y las dos matrices L y U, se almacenarían en la matriz única,

$$\begin{pmatrix} u_{11} & u_{12} & 0 & u_{14} & 0 & 0 \\ l_{21} & u_{22} & 0 & 0 & 0 & 0 \\ 0 & l_{32} & u_{33} & u_{34} & 0 & 0 \\ l_{41} & 0 & l_{43} & u_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & u_{55} & u_{56} \\ 0 & 0 & 0 & 0 & l_{65} & u_{66} \end{pmatrix}$$

cuyos elementos no nulos ocupan idénticos lugares que en la matriz del sistema, y, por consiguiente, definida por la misma matriz de posiciones.

3.3 - ALMACENAMIENTO DE LA MATRIZ DE PRECONDICIONAMIENTO FACTORIZADA SSOR.

La matriz de precondicionamiento, denominada Precondicionador SSOR, obtenida en el Capítulo 2, la escribíamos como,

$$M = \left(I - wED^{-1}\right)\left[\frac{D - wF}{w(2-w)}\right], \text{ siendo } A=D-E-F$$

y, donde

I: matriz unidad.

E: matriz triangular inferior cuyos elementos de la diagonal son todos nulos.

F: matriz triangular superior con los elementos de la diagonal, asimismo, ceros.

D: matriz diagonal cuyos elementos son los de la diagonal principal de *A*.

Analizando ahora la morfología de las matrices que figuran en la descomposición de *M*, observamos que la matriz producto wED^{-1} es una matriz triangular, con elementos nulos en la diagonal y que mantiene los mismos elementos nulos, en las mismas posiciones, que la matriz del sistema, lo que conlleva a que la estructura de la matriz $\left(I - wED^{-1}\right)$ sea similar, pero con los elementos de la diagonal iguales a la unidad.

Para el ejemplo anterior, esta matriz quedaría de la forma

$$\left(I - wED^{-1}\right) = \begin{pmatrix} 1 & & & & & \\ s_{21} & 1 & & & & \\ 0 & s_{32} & 1 & & & \\ s_{41} & 0 & s_{43} & 1 & & \\ 0 & 0 & 0 & 0 & 1 & \\ 0 & 0 & 0 & 0 & s_{65} & 1 \end{pmatrix}$$

La segunda matriz factor del precondicionador, $\left[\frac{D - wF}{w(2-w)}\right]$, será una matriz triangular superior que conserva también los elementos nulos en los mismos lugares que A, pero con valores distintos de cero en la diagonal. Para el caso concreto que hemos planteado en este capítulo quedaría como,

$$\left[\frac{D - wF}{w(2-w)}\right] = \begin{pmatrix} r_{11} & r_{12} & 0 & r_{14} & 0 & 0 \\ & r_{22} & 0 & 0 & 0 & 0 \\ & & r_{33} & r_{34} & 0 & 0 \\ & & & r_{44} & 0 & 0 \\ & & & & r_{55} & r_{56} \\ & & & & & r_{66} \end{pmatrix}$$

Es decir, que al igual que para el precondicionador ILU, bastaría una matriz de elementos no nulos para almacenar conjuntamente a las dos matrices factores:

$$\begin{pmatrix} r_{11} & r_{12} & 0 & r_{14} & 0 & 0 \\ s_{21} & r_{22} & 0 & 0 & 0 & 0 \\ 0 & s_{32} & r_{33} & r_{34} & 0 & 0 \\ s_{41} & 0 & s_{43} & r_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{55} & r_{56} \\ 0 & 0 & 0 & 0 & s_{65} & r_{66} \end{pmatrix}$$

Asimismo, no habría necesidad de crear una nueva matriz de posiciones porque es idéntica a la construída para el almacenamiento de la matriz del sistema.

CAPÍTULO 4

EFECTO DE LA RENUMERACIÓN EN EL PRECONDICIONAMIENTO.

En el capítulo 2 se observó que en el sistema precondicionado $M^{-1}Ax = M^{-1}b$, la matriz de precondicionamiento ideal sería $M = A$, y el sistema convergería en una sola iteración. Esto, aunque es inviable en la práctica, ya que el coste computacional del cálculo de M equivaldría a su resolución por un método directo, sugiere, para esta matriz de precondicionamiento, una matriz lo más aproximada posible a la del sistema valorando el esfuerzo que su determinación suponga.

Así, hemos utilizado el precondicionador ILU(0), aproximando A por el producto de dos matrices triangulares, una inferior y otra superior, que conservan los mismos elementos nulos en las mismas posiciones, sustituyendo solamente aquellos elementos distintos de cero.

Evidentemente, el coste de esta factorización es notablemente inferior al de la descomposición LU, porque en estas matrices dispersas el número de elementos no nulos es relativamente reducido, y evitamos el cálculo de los valores que figuran en las matrices triangulares en las posiciones para las que los elementos correspondientes de la matriz del sistema eran ceros.

Nos proponemos ahora utilizar técnicas de reordenación ya establecidas, que se aplican fundamentalmente en la resolución de sistemas de ecuaciones lineales por métodos directos, al objeto de conseguir que la factorización ILU(0) sea más cercana a la factorización completa LU, y obtener una mejora en el efecto del precondicionamiento en los métodos de bi-ortogonalización: Bi-CG, CGS y

Bi-CGSTAB. Mejora que se ve reflejada en los resultados obtenidos para las aplicaciones prácticas.

L.C. Dutto estudia en [11] el efecto de la reordenación en la resolución de la ecuación de Navier-Stokes con el método GMRES.

Estas técnicas, basadas en la teoría de grafos, disminuyen la envoltura ó el perfil de una matriz e intentan reducir el coste de su almacenamiento y evitar en lo posible el efecto fill-in en la factorización LU, conservando el número máximo de sus elementos nulos en las matrices triangulares de la descomposición.

La disminución del efecto de "relleno" y, consiguientemente, la existencia de más elementos iguales a cero en L y U, hace que la factorización incompleta ILU(0) cuando se aplica a la matriz de un sistema lineal A se asemeje más a su factorización completa, constituyendo entonces, en teoría, un mejor precondicionador.

La reordenación no va a afectar, sin embargo, al almacenamiento de la matriz, ya que el número de elementos no nulos, que son los que almacenamos en la forma compacta que utilizada, se sigue conservando aunque ocupen distintas posiciones en la misma.

Para efectuar la renumeración y construir la nueva matriz que defina al precondicionador ILU, hemos aplicado el Algoritmo Inverso de Cuthill-Mckee propuesto por George [21], para mejorar el algoritmo inicial de Cuthill-Mckee [8], y el Algoritmo del Mínimo Vecino [30] que junto con el algoritmo del Mínimo Fill-in [30], entre otros, son variantes del Algoritmo del Mínimo Grado propuesto por George y Liu [22]:

Algoritmo ICM (Cuthill-Mckee inverso)

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 (extremo y con pocas conexiones) y renumerarlo como x_1.

3 - Renumerar los nodos conectados a x_i en orden ascendente de grado.

4 - Efectuar el ordenamiento inverso.

Algoritmo del Mínimo vecino

1 - Construir grafo $g(x)=\langle V,E\rangle$.

2 - Mientras $V\neq\Phi$

2.1 - Elegir un nodo v de grado mínimo en $g(x)=\langle V,E\rangle$ y ordenar v como nodo siguiente.

2.2 - Definir

$$V_v=V-\{v\} \qquad E_v=\{(a,b)\in E / a,b\in V_v\}$$

y hacer $V=V_v$, $E=E_v$ y $g(x)=\langle V,E\rangle$.

La elección del nodo inicial que figura en el paso 2 de los algoritmos anteriores es importante para la eficiencia de los mismos. Para su determinación utilizaremos el algoritmo de George [2] basado en la construcción de unas particiones del conjunto de nodos, llamadas estructuras con niveles enraizadas.

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 $\varepsilon(x) = Max\{d(x,y) / y \in V\}$, el **Algoritmo de George** se escribe de la forma siguiente:

1 - Elegir un nodo arbitrario r de V.

2 - Generar una estructura con niveles enraizada en r:

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

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:

$$\{L_1(x), L_2(x), \ldots, L_{\varepsilon(x)}(x)\}$$

5 - Si $\varepsilon(x) > \varepsilon(r)$, establecer $x \rightarrow r$ y volver al paso 3.

6 - Caso contrario, tomamos x como nodo inicial.

CAPÍTULO 5

ALGORITMOS PRECONDICIONADOS

En capítulos anteriores, hemos precisado las ventajas de la técnica de Precondicionamiento en la resolución de sistemas de ecuaciones lineales por los métodos iterativos relacionados en el Capítulo 1, de tal forma, que en muchos casos es imprescindible su implementación para obtener una solución aceptable en un número razonable de iteraciones. Se han considerado, también, algunos precondicionadores que, por su sencillez de aplicación y su efectividad, son de los más utilizados.

Se hace necesario, pués, adecuar los algoritmos de estos métodos para su aplicación en los sistemas previamente precondicionados.

El procedimiento general, será transformar el sistema original $Ax = b$, mediante un precondicionamiento por la izquierda, por la derecha ó por ambos lados, en el sistema precondicionado $\tilde{A}\tilde{x} = \tilde{b}$, y aplicar a este último sistema el algoritmo del método elegido. Pero, conforme se desarrolla el algoritmo, efectuaremos transformaciones, introduciendo nuevos vectores y aprovechando las características que debe reunir la matriz de precondicionamiento, de modo que todos los algoritmos deben cumplir con carácter común, unos requerimientos generales que garanticen su efectividad:

- No se tendrá que efectuar explícitamente el producto de la matriz de precondicionamiento por la matriz del sistema, debido a la carga computacional que esto conllevaría.

- A pesar de estar resolviendo el sistema $\tilde{A}\tilde{x} = \tilde{b}$, el algoritmo debe proporcionar la solución del sistema original, estableciendo una relación de recurrencia para la misma.

- Asimismo, el criterio de parada se define a partir de los vectores residuos del sistema $Ax=b$, para lo cual, también, debe figurar en el algoritmo precondicionado la relación de recurrencia que nos permita su determinación.

En este capítulo, obtendremos los Algoritmos precondicionados de los métodos objeto de esta tesis: Bi-CG, CGS y Bi-CGSTAB, estableciendo comparaciones en cada uno para las distintas formas de precondicionamiento y extrayendo las conclusiones que se derivan de las mismas.

Asimismo, presentamos sendas variantes para los algoritmos clásicos precondicionados de estos métodos que hemos denotado, respectivamente por Bi-CG*, CGS* y Bi-CGSTAB*, con comportamiento en aritmética finita distinto al de los originales y cuya aplicación da lugar, en muchas ocasiones, a convergencia más rápida y suave.

5.1 - DOBLE GRADIENTE CONJUGADO

Estableceremos para las formas de precondicionamiento por la derecha, por la izquierda y por ambos lados, las relaciones entre las matrices, vectores incógnitas y vectores segundo miembro, respectivos, de los sistemas original y precondicionado. Estas expresiones junto con las de los vectores residuos se sustituirán en el algoritmo del Bi-CG expuesto en el Capítulo 1, para realizar las transformaciones comentadas anteriormente.

- Precondicionamiento por la derecha

Precondicionando por la derecha con la matriz M, tendríamos,

$$\tilde{A}\tilde{x} = \tilde{b} \begin{Bmatrix} \tilde{A} - AM^{-1} \\ \tilde{x} = Mx \\ \tilde{b} = b \end{Bmatrix}$$

Utilizando el Bi-CG para este nuevo sistema, tendríamos que construir la matriz

$$\begin{pmatrix} 0 & \tilde{A} \\ \tilde{A}^T & 0 \end{pmatrix}$$

y aplicar el CG al sistema simétrico

$$\begin{pmatrix} 0 & \tilde{A} \\ \tilde{A}^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{y} \\ \tilde{x} \end{pmatrix} = \begin{pmatrix} \tilde{b} \\ \tilde{b}_1 \end{pmatrix},$$

que conduce a resolver paralelamente al sistema precondicionado $\tilde{A}\tilde{x} = \tilde{b}$, el sistema auxiliar, asimismo precondicionado,

$$\tilde{A}^T\tilde{y} = \tilde{b}_1 \begin{Bmatrix} \tilde{A}^T = M^{-T}A^T \\ \tilde{y} = y \\ \tilde{b}_1 = M^{-T}b_1 \end{Bmatrix}$$

Las relaciones entre los vectores residuos de los sistemas con y sin precondicionamiento quedarían, respectivamente,

$\tilde{r}_i = \tilde{b} - \tilde{A}\tilde{x}_i = b - AM^{-1}Mx_i = r_i$, para el sistema original

$\tilde{\tilde{r}}_i - \tilde{b}_1 - \tilde{A}^T\tilde{y}_i = M^{-T}b_1 - M^{-T}A^Ty_i = M^{-T}\left(b_1 - A^Ty_i\right) = M^{-T}\bar{r}_i$, para el sistema auxiliar.

Aplicando ahora el algoritmo del Bi-CG a estos sistemas precondicionados y estableciendo las nuevas expresiones para los distintos parámetros y vectores:

Elegimos $\tilde{\tilde{r}}_0$

$\tilde{\rho}_0 = 1$

$$\tilde{p}_0 = \tilde{\bar{p}}_0 = 0$$

$$\tilde{\rho}_i = \tilde{\bar{r}}_{i-1}^T \tilde{r}_{i-1} = \tilde{\bar{r}}_{i-1}^T r_{i-1}$$

$$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$$

$$\tilde{p}_i = \tilde{r}_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1} = r_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1}$$

$$\tilde{\bar{p}}_i = \tilde{\bar{r}}_{i-1} + \tilde{\beta}_i \tilde{\bar{p}}_{i-1}$$

$\tilde{v}_i = \tilde{A}\tilde{p}_i = AM^{-1}\tilde{p}_i$; haciendo $My_i = \tilde{p}_i$ queda $\tilde{v}_i = Ay_i$

$$\tilde{\alpha}_i = \tilde{\rho}_i / \left(\tilde{\bar{p}}_i^T \tilde{v}_i\right)$$

$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{p}_i$; multiplicando por M^{-1}, $x_i = x_{i-1} + \tilde{\alpha}_i y_i$

$\tilde{r}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{v}_i$; $r_i = r_{i-1} - \tilde{\alpha}_i \tilde{v}_i$

$\tilde{\bar{v}}_i = \tilde{A}^T \tilde{\bar{p}}_i = M^{-T} A^T \tilde{\bar{p}}_i$; multiplicando por M^T, $M^T \tilde{\bar{v}}_i = A^T \tilde{\bar{p}}_i$

$$\tilde{\bar{r}}_i = \tilde{\bar{r}}_{i-1} - \tilde{\alpha}_i \tilde{\bar{v}}_i$$

Resulta así, el siguiente **Algoritmo Bi-CG con precondicionamiento por la derecha**:

Valor inicial $\mathbf{x}_0, \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;

Elegimos $\tilde{\bar{\mathbf{r}}}_0$, tal que $\tilde{\bar{\mathbf{r}}}_0^{\mathrm{T}} \mathbf{r}_0 \neq \mathbf{0}$;

$\tilde{\rho}_0 = 1$;

$\tilde{\mathbf{p}}_0 = \tilde{\bar{\mathbf{p}}}_0 = \mathbf{0}$;

Si $\|\mathbf{r}_i\| / \|\mathbf{r}_0\| \geq \varepsilon, (i = 1,2,3,\ldots)$;

$\tilde{\rho}_i = \tilde{\bar{\mathbf{r}}}_{\mathbf{i-1}}^{\mathbf{T}} \mathbf{r}_{\mathbf{i-1}}$;

$\tilde{\beta}_i = \tilde{\mathbf{r}}_{\mathbf{i}} / \tilde{\mathbf{r}}_{\mathbf{i-1}}$;

$\tilde{\mathbf{p}}_{\mathbf{i}} = \mathbf{r}_{\mathbf{i-1}} + \tilde{\beta}_i \tilde{\mathbf{p}}_{\mathbf{i-1}}$;

$\tilde{\bar{\mathbf{p}}}_{\mathbf{i}} = \tilde{\bar{\mathbf{r}}}_{\mathbf{i-1}} + \tilde{\beta}_i \tilde{\bar{\mathbf{p}}}_{\mathbf{i-1}}$;

$\mathbf{M}\mathbf{y}_{\mathbf{i}} = \tilde{\mathbf{p}}_{\mathbf{i}}$;

$\tilde{\mathbf{v}}_{\mathbf{i}} = \mathbf{A}\mathbf{y}_{\mathbf{i}}$;

$\tilde{\alpha}_i = \tilde{\rho}_i / \left(\tilde{\bar{\mathbf{p}}}_{\mathbf{i}}^{\mathrm{T}} \tilde{\mathbf{v}}_{\mathbf{i}}\right)$;

$\mathbf{x}_{\mathbf{i}} = \mathbf{x}_{\mathbf{i-1}} + \tilde{\alpha}_i \mathbf{y}_{\mathbf{i}}$;

$$\mathbf{r}_i = \mathbf{r}_{i-1} - \tilde{\alpha}_i \tilde{\mathbf{v}}_i;$$

$$\mathbf{M}^T \tilde{\tilde{\mathbf{v}}}_i = \mathbf{A}^T \tilde{\tilde{\mathbf{p}}}_i \ ;$$

$$\tilde{\tilde{\mathbf{r}}}_i = \tilde{\tilde{\mathbf{r}}}_{i-1} - \tilde{\alpha}_i \tilde{\tilde{\mathbf{v}}}_i;$$

fin

- **Precondicionamiento por la izquierda**

El sistema precondicionado sería,

$$\tilde{A}\tilde{x} = \tilde{b} \begin{cases} \tilde{A} = M^{-1}A \\ \tilde{x} = x \\ \tilde{b} = M^{-1}b \end{cases}$$

Construyendo la matriz simétrica propia del método, se obtendría para el sistema auxiliar precondicionado,

$$A^T M^{-T} M^T y = b_2$$

lo que daría lugar a las relaciones entre los vectores residuos de los sistemas antes y después del precondicionamiento,

$\tilde{r}_i = M^{-1} r_i$ para el sistema original,

$\tilde{\tilde{r}}_i = \bar{r}_i$, para el sistema auxiliar.

Utilizando, al igual que anteriormente, el algoritmo del Bi-CG, y efectuando, asimismo, las transformaciones oportunas para que las soluciones aproximadas que nos proporcione sean las del sistema original, queda,

Elegimos $\tilde{\tilde{r}}_0$

$\tilde{\rho}_0 = 1$

$\tilde{p}_0 = \tilde{\tilde{p}}_0 = 0$

$\tilde{\rho}_i = \tilde{\tilde{r}}_{i-1}^T \tilde{r}_{i-1} = \bar{r}_{i-1}^T M^{-1} r_{i-1}$, haciendo $Mz_{i-1} = r_{i-1}$ queda $\tilde{\rho}_i = \bar{r}_{i-1}^T z_{i-1}$

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$

$\tilde{p}_i = \tilde{r}_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1} = z_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1}$

$$\tilde{\tilde{p}}_i = \tilde{\tilde{r}}_{i-1} + \tilde{\beta}_i \tilde{\tilde{p}}_{i-1}$$

$$\tilde{v}_i = \tilde{A}\tilde{p}_i = M^{-1}A\tilde{p}_i;\ \text{multiplicando por } M,\quad M\tilde{v}_i = A\tilde{p}$$

$$\tilde{\alpha}_i = \tilde{\rho}_i / \left(\tilde{\tilde{p}}_i^T \tilde{v}_i\right)$$

$$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{p}_i;\ x_i = x_{i-1} + \tilde{\alpha}_i \tilde{p}_i$$

$$\tilde{r}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{v}_i;\ \text{multiplicando por } M,\ r_i = r_{i-1} - \tilde{\alpha}_i A\tilde{p}_i$$

$$\tilde{\tilde{v}}_i = \tilde{A}^T \tilde{\tilde{p}}_i = A^T M^{-T} \tilde{\tilde{p}}_i\ ,\ \text{haciendo } M^T y_i = \tilde{\tilde{p}}_i \text{ queda } \tilde{\tilde{v}}_i = A^T y_i$$

$$\tilde{\tilde{r}}_i = \tilde{\tilde{r}}_{i-1} - \tilde{\alpha}_i \tilde{\tilde{v}}_i$$

Queda, entonces, el **Algoritmo Bi-CG con precondicionamiento por la izquierda**:

Valor inicial $\mathbf{x_0}$, $\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$;

Elegimos $\tilde{\tilde{\mathbf{r}}}_0$, tal que $\tilde{\tilde{\mathbf{r}}}_0^{\mathbf{T}}\mathbf{r_0} \neq \mathbf{0}$;

$\tilde{\rho}_0 = 1$;

$\tilde{\mathbf{p}}_0 = \tilde{\tilde{\mathbf{p}}}_0 = \mathbf{0}$;

Si $\|\mathbf{r}_i\| / \|\mathbf{r_0}\| \geq \varepsilon, (i = 1,2,3,\ldots)$;

$\mathbf{M}\mathbf{z}_{i-1} = \mathbf{r}_{i-1}$;

$\tilde{\rho}_i = \tilde{\mathbf{r}}_{i-1}^{\mathbf{T}} \mathbf{z}_{i-1}$;

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$;

$\tilde{\mathbf{p}}_i = \mathbf{z}_{i-1} + \tilde{\beta}_i \tilde{\mathbf{p}}_{i-1}$;

$\tilde{\tilde{\mathbf{p}}}_i = \tilde{\tilde{\mathbf{r}}}_{i-1} + \tilde{\beta}_i \tilde{\tilde{\mathbf{p}}}_{i-1}$;

$\mathbf{M}\tilde{\mathbf{v}}_i = \mathbf{A}\tilde{\mathbf{p}}_i$;

$\tilde{\alpha}_i = \tilde{\rho}_i / \left(\tilde{\tilde{\mathbf{p}}}_i^{\mathbf{T}} \tilde{\mathbf{v}}_i\right)$;

$\mathbf{x}_i = \mathbf{x}_{i-1} + \tilde{\alpha}_i \tilde{\mathbf{p}}_i$;

$\mathbf{r}_i = \mathbf{r}_{i-1} - \tilde{\alpha}_i \mathbf{A}\tilde{\mathbf{p}}_i$;

$\mathbf{M}^{\mathbf{T}}\mathbf{y}_i = \tilde{\tilde{\mathbf{p}}}_i$;

$\tilde{\tilde{\mathbf{v}}}_i = \mathbf{A}^{\mathbf{T}}\mathbf{y}_i$;

$$\tilde{\bar{\mathbf{r}}}_i = \tilde{\bar{\mathbf{r}}}_{i-1} - \tilde{\alpha}_i \tilde{\bar{\mathbf{v}}}_i;$$

fin

- Precondicionamiento por ambos lados

En el caso que factorizáramos la matriz de precondicionamiento como $M = LU$, (se ha utilizado este tipo de factorización sin pérdida ninguna de generalidad, pués lo expuesto no sufriría alteración para el caso general $M = M_1 M_2$), podríamos multiplicar por ambos lados la matriz del sistema original A, y el sistema precondicionado tomaría la forma,

$$\tilde{A}\tilde{x} = \tilde{b} \begin{cases} \tilde{A} = L^{-1} A U^{-1} \\ \tilde{x} = Ux \\ \tilde{b} = L^{-1} b \end{cases}$$

Dado que $\tilde{A}^T = U^{-T} A^T L^{-T}$, el sistema auxiliar precondicionado sería en este caso,

$$U^{-T} A^T L^{-T} L^T x = U^{-T} b_2$$

De esta forma, las relaciones entre los vectores residuos de ambos sistemas quedan,

$$\tilde{r}_i = L^{-1} r_i$$
$$\tilde{\bar{r}}_i = U^{-T} \bar{r}_i$$

Con las transformaciones correspondientes, al igual que en los casos anteriores, resultan las siguientes relaciones,

Elegimos $\tilde{\bar{r}}_0$.

$\tilde{\rho}_0 = 1.$

$\tilde{p}_0 = \tilde{\bar{p}}_0 = 0.$

$\tilde{\rho}_i = \tilde{\bar{r}}_{i-1}^T \tilde{r}_{i-1} = \bar{r}_{i-1}^T U^{-1} L^{-1} r_{i-1} = \bar{r}_{i-1}^T M^{-1} r_{i-1}$, y con $Mz_{i-1} = r_{i-1}$, queda $\tilde{\rho}_i = \bar{r}_{i-1}^T z_{i-1}$.

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}.$

$\tilde{p}_i = \tilde{r}_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1} = L^{-1} r_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1}$; multiplicando por U^{-1} con $U^{-1}\tilde{p}_i = p_i$ queda $p_i = z_{i-1} + \tilde{\beta}_i p_{i-1}$.

$\tilde{\bar{p}}_i = \tilde{\bar{r}}_{i-1} + \tilde{\beta}_i \tilde{\bar{p}}_{i-1}$; multiplicando por L^{-T} con $L^{-T}\tilde{\bar{p}}_i = \bar{p}_i$ queda $\bar{p}_i = M^{-T}\bar{r}_{i-1} + \tilde{\beta}_i \bar{p}_{i-1}$; y haciendo $M^T y_{i-1} = \bar{r}_{i-1}$ resulta $\bar{p}_i = y_{i-1} + \tilde{\beta}_i \bar{p}_{i-1}$.

$\tilde{v}_i = \tilde{A}\tilde{p}_i = L^{-1}AU^{-1}\tilde{p}_i$; multiplicando por L con $L\tilde{v}_i = v_i$ queda $v_i = Ap_i$.

$\tilde{\alpha}_i = \tilde{\rho}_i / (\tilde{\bar{p}}_i^T \tilde{v}_i) = \tilde{\rho}_i / (\tilde{\bar{p}}_i^T L^{-1} AU^{-1} \tilde{p}_i) = \tilde{\rho}_i / (\bar{p}_i^T A p_i) = \tilde{\rho}_i / (\bar{p}_i^T v_i)$.

$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{p}_i$; multiplicando por U^{-1}, $x_i = x_{i-1} + \tilde{\alpha}_i p_i$.

$\tilde{r}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{v}_i$; multiplicando por L , $r_i = r_{i-1} - \tilde{\alpha}_i v_i$.

$\tilde{\bar{v}}_i = \tilde{A}^T \tilde{\bar{p}}_i = U^{-T} A^T L^{-T} L^T \bar{p}_i$; multiplicando por U^T, $\bar{v}_i = A^T \bar{p}_i$.

$\tilde{\bar{r}}_i = \tilde{\bar{r}}_{i-1} - \tilde{\alpha}_i \tilde{\bar{v}}_i$; multiplicando por U^T, $\bar{r}_i = \bar{r}_{i-1} - \tilde{\alpha}_i \bar{v}_i$.

Tomando las expresiones finales en las distintas transformaciones, resulta el siguiente **Algoritmo Bi-CG con precondicionamiento por ambos lados**:

Valor inicial $\mathbf{x_0}$, $\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$;

Elegimos $\tilde{\bar{\mathbf{r}}}_0$, tal que $\tilde{\bar{\mathbf{r}}}_0^{\mathrm{T}} \mathbf{r}_0 \neq \mathbf{0}$;

$\tilde{\rho}_0 = 1$;

$\tilde{\mathbf{p}}_0 = \tilde{\bar{\mathbf{p}}}_0 = \mathbf{0}$;

Si $\|\mathbf{r_i}\| / \|\mathbf{r_0}\| \geq \varepsilon, (i = 1,2,3,\ldots)$;

$\mathbf{M}\mathbf{z}_{i-1} = \mathbf{r}_{i-1}$;

$\tilde{\rho}_i = \bar{\mathbf{r}}_{i-1}^{\mathrm{T}} \mathbf{z}_{i-1}$;

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$;

$\mathbf{p}_i = \mathbf{z}_{i-1} + \tilde{\beta}_i \mathbf{p}_{i-1}$;

$\mathbf{M}^{\mathrm{T}} \mathbf{y}_{i-1} = \bar{\mathbf{r}}_{i-1}$;

$\bar{\mathbf{p}}_i = \mathbf{y}_{i-1} + \tilde{\beta}_i \bar{\mathbf{p}}_{i-1}$;

$\mathbf{v}_i = \mathbf{A}\mathbf{p}_i$;

$\tilde{\alpha}_i = \tilde{\rho}_i / (\bar{\mathbf{p}}_i^{\mathrm{T}} \mathbf{v}_i)$;

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \tilde{\alpha}_i \mathbf{p}_i;$$

$$\mathbf{r}_i = \mathbf{r}_{i-1} - \tilde{\alpha}_i \mathbf{v}_i;$$

$$\overline{\mathbf{v}}_i = \mathbf{A}^T \overline{\mathbf{p}}_i;$$

$$\overline{\mathbf{r}}_i = \overline{\mathbf{r}}_{i-1} - \tilde{\alpha}_i \overline{\mathbf{v}}_i;$$

fin

- Atendiendo a los requisitos que hemos estimado deben reunir estos algoritmos planteados al principio del capítulo, en los tres esquemas figuran relaciones de recurrencia respectivas para la solución x, del sistema $Ax = b$, a pesar de que en principio, estamos resolviendo el sistema precondicionado $\tilde{A}\tilde{x} = \tilde{b}$.

Asimismo, los vectores residuos del sistema original que figuran en las relaciones de recurrencia, son los del sistema sin precondicionar, estableciéndose el criterio de parada a partir de los mismos.

- Sin embargo, en cuanto a los vectores residuos del sistema auxiliar, cuya solución no buscamos, en el algoritmo relativo al precondicionamiento por ambos lados se establecen relaciones de recurrencia para los vectores del sistema sin precondicionar, mientras que en los otros dos casos figuran en las expresiones equivalentes los vectores residuos del sistema precondicionado.

Igual ocurre, aunque no influya en la aplicación del algoritmo, para las direcciones conjugadas. Se trabaja con las direcciones conjugadas de los sistemas original y auxiliar sin precondicionar en el caso del precondicionamiento por ambos lados y con las correspondientes de los sistemas precondicionados en los otros dos tipos.

- En la inicialización del algoritmo, además de elegir el valor inicial para las iteraciones de las soluciones aproximadas del sistema, x_0, que nos determinará el vector residuo inicial del sistema original $Ax = b$, hay que tomar

arbitrariamente el residuo inicial del sistema auxiliar, pudiéndose elegir el del sistema sin precondicionar $\bar{r}_0$, ó el del sistema precondicionado $\tilde{\bar{r}}_0$, ya que entre ambos, para cada una de las formas de precondicionamiento, existe una relación que nos permite obtener uno de ellos a partir del otro.

- En los algoritmos correspondientes a las formas de precondicionamiento por la derecha y por ambos lados, figuran dos productos añadidos matriz inversa por vector, mientras que en el del precondicionamiento por la izquierda, más costoso, se añaden al algoritmo sin precondicionar tres productos de este tipo.

- Aunque, en los dos primeros esquemas, las operaciones de remonte se realizan en fases distintas del algoritmo y con vectores, asimismo, distintos, es posible establecer una analogía completa entre ambos mediante transformaciones matriciales elementales. Para establecer comparaciones con otros métodos hemos tomado como referencia el algoritmo correspondiente al precondicionamiento por ambos lados que denotaremos por **algoritmo Bi-CG***.

- 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.

Como ya se ha observado, una forma cómoda de efectuar el producto es expresar M factorizada en el producto de dos matrices triangulares 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.

5.2 - CGS (GRADIENT CONJUGATE SQUARED)

Se obtendrán dos "familias" de algoritmos, comprobando que los esquemas correspondientes a las distintas formas de precondicionar en cada una de estas familias, solo difieren en la inicialización. Precisamente, la elección de este vector inicial va a proporcionar una nueva herramienta versátil y abierta a innovaciones futuras para el precondicionamiento de estos sistemas de ecuaciones en los métodos tipo Bi-CG.

De manera similar a lo efectuado en el Doble Gradiente, plantearemos las relaciones entre matrices, vectores residuos y solución del sistema antes y después de precondicionar y realizaremos los cambios adecuados para establecer los nuevos algoritmos.

- Precondicionamiento por la derecha

Sistema precondicionado,

$$\tilde{A}\tilde{x} = b \begin{cases} \tilde{A} = AM^{-1} \\ \tilde{x} = Mx \\ \tilde{b} = b \end{cases}$$

Aplicando el algoritmo del CGS expuesto en el Capítulo I a este nuevo sistema, los nuevos vectores y parámetros resultarían:

$$\tilde{\rho}_0 = 1$$

$$\tilde{p}_0 = \tilde{q}_0 = 0$$

$$\tilde{\rho}_i = \tilde{\tilde{r}}_0^T \tilde{r}_{i-1} = \hat{r}_0^T r_{i-1}, \text{ con } \hat{r}_0 = \tilde{\tilde{r}}_0$$

$$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$$

$$\tilde{u} = \tilde{r}_{i-1} + \tilde{\beta}_i \tilde{q}_{i-1} = r_{i-1} + \tilde{\beta}_i \tilde{q}_{i-1}$$

$\tilde{p}_i = \tilde{u} + \tilde{\beta}_i\left(\tilde{q}_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1}\right)$

$\tilde{v}_i = \tilde{A}\tilde{p}_i = AM^{-1}\tilde{p}_i$; haciendo $My_i = \tilde{p}_i$ queda $\tilde{v}_i = Ay_i$

$\tilde{\alpha}_i = \tilde{\rho}_i / \left(\tilde{\tilde{r}}_0^T \tilde{v}_i\right)$

$\tilde{q}_i = \tilde{u} - \tilde{\alpha}_i \tilde{v}_i$

$\tilde{w} = \tilde{u} + \tilde{q}_i$

$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{w}$; multiplicando por M^{-1}, $x_i = x_{i-1} + \tilde{\alpha}_i M^{-1}\tilde{w}$; y con $Mz = \tilde{w}$ queda $x_i = x_{i-1} + \tilde{\alpha}_i z$

$\tilde{r}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{A}\tilde{w} = r_{i-1} - \tilde{\alpha}_i AM^{-1}\tilde{w} = r_{i-1} - \tilde{\alpha}_i Az$

Esto conduce al **Algoritmo CGS con precondicionamiento por la derecha:**

Valor inicial $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;

Elegimos $\hat{\mathbf{r}}_0$;

$\tilde{\rho}_0 = 1$;

$\tilde{\mathbf{p}}_0 = \tilde{\mathbf{q}}_0 = \mathbf{0}$;

Si $\|\mathbf{r}_i\| / \|\mathbf{r}_0\| \geq \varepsilon, (i = 1,2,3,\ldots)$;

$\tilde{\rho}_i = \hat{\mathbf{r}}_0^{\mathbf{T}} \mathbf{r}_{i-1}$;

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$;

$\tilde{\mathbf{u}} = \mathbf{r}_{i-1} + \tilde{\beta}_i \tilde{\mathbf{q}}_{i-1}$;

$\tilde{\mathbf{p}}_i = \tilde{\mathbf{u}} + \tilde{\beta}_i\left(\tilde{\mathbf{q}}_{i-1} + \tilde{\beta}_i \tilde{\mathbf{p}}_{i-1}\right)$;

$\mathbf{M}\mathbf{y}_i = \tilde{\mathbf{p}}_i$;

$\tilde{\mathbf{v}}_i = \mathbf{A}\mathbf{y}_i$;

$\tilde{\alpha}_i = \tilde{\rho}_i / \left(\hat{\mathbf{r}}_0^{\mathbf{T}} \tilde{\mathbf{v}}_i\right)$;

$\tilde{\mathbf{q}}_i = \tilde{\mathbf{u}} - \tilde{\alpha}_i \tilde{\mathbf{v}}_i$;

$\tilde{\mathbf{w}} - \tilde{\mathbf{u}} + \tilde{\mathbf{q}}_i$;

$\mathbf{M}\mathbf{z} = \tilde{\mathbf{w}}$;

$\mathbf{x}_i = \mathbf{x}_{i-1} + \tilde{\alpha}_i \mathbf{z}$;

$\mathbf{r}_i = \mathbf{r}_{i-1} - \tilde{\alpha}_i \mathbf{A}\mathbf{z}$;

fin

- Precondicionamiento por la izquierda

Sistema precondicionado,

$$\tilde{A}\tilde{x} = \tilde{b} \quad \begin{cases} \tilde{A} = M^{-1}A \\ \tilde{x} = x \\ \tilde{b} = M^{-1}b \end{cases}$$

Aplicando el algoritmo y operando,

$\tilde{\rho}_0 = 1$

$\tilde{p}_0 = \tilde{q}_0 = 0$

$\tilde{\rho}_i = \tilde{\tilde{r}}_0^T \tilde{r}_{i-1} = \tilde{\tilde{r}}_0^T M^{-1} r_{i-1} = \left[M^{-T}\tilde{\tilde{r}}_0\right] r_{i-1}$; $\tilde{\rho}_i = \hat{r}_0^T r_{i-1}$ con $\hat{r}_0 = M^{-T}\tilde{\tilde{r}}_0$

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$

$\tilde{u} = \tilde{r}_{i-1} + \tilde{\beta}_i \tilde{q}_{i-1}$, multiplicando por M, $u = r_{i-1} + \tilde{\beta}_i q_{i-1}$

$\tilde{p}_i = \tilde{u} + \tilde{\beta}_i \left(\tilde{q}_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1}\right)$, multiplicando por M, $p_i = u + \tilde{\beta}_i \left(q_{i-1} + \tilde{\beta}_i p_{i-1}\right)$

$\tilde{v}_i = \tilde{A}\tilde{p}_i = M^{-1}AM^{-1}p_i$; $v_i = AM^{-1}p_i$; haciendo $My_i = p_i$ queda $v_i = Ay_i$

$$\tilde{\alpha}_i = \frac{\tilde{\rho}_i}{\tilde{\tilde{r}}_0^T \tilde{v}_i} = \frac{\tilde{\rho}_i}{\tilde{\tilde{r}}_0^T M^{-1} v_i} = \frac{\tilde{\rho}_i}{\left(M^{-T}\tilde{\tilde{r}}_0\right)^T v_i} = \frac{\tilde{\rho}_i}{\hat{r}_0^T v_i}$$

$\tilde{q}_i = \tilde{u} - \tilde{\alpha}_i \tilde{v}_i$; $q_i = u - \tilde{\alpha}_i v_i$

$\tilde{w} = \tilde{u} + \tilde{q}_i$; $w = u + q_i$

$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i w$; $x_i = x_{i-1} + \tilde{\alpha}_i M^{-1} w$; haciendo $Mz = w$ queda $x_i = x_{i-1} + \tilde{\alpha}_i z$

$\tilde{r}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{A}\tilde{w}$, $r_i = r_{i-1} - MM^{-1}AM^{-1}w = r_{i-1} - \tilde{\alpha}_i Az$

Queda entonces, para el **Algoritmo CGS con precondicionamiento por la izquierda:**

Valor inicial $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;

Elegimos $\hat{\mathbf{r}}_0$;

$\tilde{\rho}_0 = 1;$

$\tilde{\mathbf{p}}_0 = \tilde{\mathbf{q}}_0 = \mathbf{0};$

Si $\|\mathbf{r}_i\|/\|\mathbf{r}_0\| \geq \varepsilon, (i = 1,2,3,...);$

$\tilde{\rho}_i = \hat{\mathbf{r}}_0^{\mathrm{T}} \mathbf{r}_{i-1};$

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1};$

$\mathbf{u} = \mathbf{r}_{i-1} + \tilde{\beta}_i \mathbf{q}_{i-1};$

$\mathbf{p}_i = \mathbf{u} + \tilde{\beta}_i (\mathbf{q}_{i-1} + \beta_i \mathbf{p}_{i-1});$

$\mathbf{M}\mathbf{y}_i = \mathbf{p}_i;$

$\mathbf{v}_i = \mathbf{A}\mathbf{y}_i;$

$\tilde{\alpha}_i = \tilde{\rho}_i / (\hat{\mathbf{r}}_0^{\mathrm{T}} \mathbf{v}_i);$

$\mathbf{q}_i = \mathbf{u} - \tilde{\alpha}_i \mathbf{v}_i;$

$\mathbf{w} = \mathbf{u} + \mathbf{q}_i;$

$\mathbf{M}\mathbf{z} = \mathbf{w};$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \tilde{\alpha}_i \mathbf{z};$

$\mathbf{r}_i = \mathbf{r}_{i-1} - \tilde{\alpha}_i \mathbf{A}\mathbf{z};$

fin

- Precondicionamiento por ambos lados

El sistema precondicionado y la relación entre los vectores residuos serían,

$$\tilde{A}\tilde{x} = \tilde{b} \begin{cases} \tilde{A} = L^{-1} A U^{-1} \\ \tilde{x} = Ux \\ \tilde{b} = L^{-1} b \end{cases}$$

$\tilde{r}_i = L^{-1} r_i$

Aplicando el algoritmo CGS y operando,

$\tilde{\rho}_0 = 1$

$\tilde{p}_0 = \tilde{q}_0 = 0$

$\tilde{\rho}_i = \tilde{\tilde{r}}_0^T \tilde{r}_{i-1} = \tilde{\tilde{r}}_0^T L^{-1} r_{i-1} = \left[L^{-T}\tilde{\tilde{r}}_0\right]^T r_{i-1}$; $\tilde{\rho}_i = \hat{r}_0^T r_{i-1}$ con $\hat{r}_0 = L^{-T}\tilde{\tilde{r}}_0$

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$

$\tilde{u} = \tilde{r}_{i-1} + \tilde{\beta}_i \tilde{q}_{i-1}$, multiplicando por L, $u = r_{i-1} + \tilde{\beta}_i q_{i-1}$

$\tilde{p}_i = \tilde{u} + \tilde{\beta}_i\left(\tilde{q}_{i-1} + \tilde{\beta}_i \tilde{p}_{i-1}\right)$, multiplicando por L, $p_i = u + \tilde{\beta}_i\left(q_{i-1} + \tilde{\beta}_i p_{i-1}\right)$

$\tilde{v}_i - \tilde{A}\tilde{p}_i - L^{-1}AU^{-1}L^{-1}p_i$; $v_i = AM^{-1}p_i$; haciendo $My_i = p_i$ queda $v_i = Ay_i$

$$\tilde{\alpha}_i = \frac{\tilde{\rho}_i}{\tilde{\tilde{r}}_0^T \tilde{v}_i} = \frac{\tilde{\rho}_i}{\tilde{\tilde{r}}_0^T L^{-1} A U^{-1} L^{-1} p_i} = \frac{\tilde{\rho}_i}{\left[L^{-T}\tilde{\tilde{r}}_0\right]^T A y_i} = \frac{\tilde{\rho}_i}{\hat{r}_0^T A y_i} = \frac{\tilde{\rho}_i}{\hat{r}_0^T v_i}$$

$\tilde{q}_i = \tilde{u} - \tilde{\alpha}_i \tilde{v}_i$; $q_i = u - \tilde{\alpha}_i v_i$

$\tilde{w} = \tilde{u} + \tilde{q}_i$; $w = u + q_i$

$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{w} = \tilde{x}_{i-1} + \tilde{\alpha}_i L^{-1} w$;, multiplicando por U^{-1}, $\tilde{x}_i = x_{i-1} + \tilde{\alpha}_i M^{-1} w$;

haciendo $Mz = w$ queda $x_i = x_{i-1} + \tilde{\alpha}_i z$

$\tilde{r}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{A}\tilde{w} = \tilde{r}_{i-1} - \tilde{\alpha}_i L^{-1} A U^{-1} L^{-1} w$ multiplicando por L, $r_i = r_{i-1} - \tilde{\alpha}_i A z$

Con las expresiones finales de las operaciones anteriores, resulta el siguiente **Algoritmo CGS con precondicionamiento por ambos lados**:

Valor inicial $\mathbf{x_0}$, $\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$;

Elegimos $\hat{\mathbf{r}}_0$;

$\tilde{\rho}_0 = 1$;

$\tilde{\mathbf{p}}_0 = \tilde{\mathbf{q}}_0 = \mathbf{0}$;

Si $\|\mathbf{r}_i\| / \|\mathbf{r}_0\| > \varepsilon, (i = 1,2,3,\ldots)$;

$\tilde{\rho}_i = \hat{\mathbf{r}}_0^T \mathbf{r}_{i-1}$;

$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$;

$\mathbf{u} = \mathbf{r}_{i-1} + \tilde{\beta}_i \mathbf{q}_{i-1}$;

$\mathbf{p}_i = \mathbf{u} + \tilde{\beta}_i\left(\mathbf{q}_{i-1} + \tilde{\beta}_i \mathbf{p}_{i-1}\right)$;

$\mathbf{M}\mathbf{y}_i = \mathbf{p}_i$;

$$\mathbf{v}_i = \mathbf{A}\mathbf{y}_i;$$
$$\tilde{\alpha}_i = \tilde{\rho}_i / \left(\hat{\mathbf{r}}_0^{\mathrm{T}} \mathbf{v}_i\right);$$
$$\mathbf{q}_i = \mathbf{u} - \tilde{\alpha}_i \mathbf{v}_i;$$
$$\mathbf{w} = \mathbf{u} + \mathbf{q}_i;$$
$$\mathbf{M}\mathbf{z} = \mathbf{w};$$
$$\mathbf{x}_i = \mathbf{x}_{i-1} + \tilde{\alpha}_i \mathbf{z};$$
$$\mathbf{r}_i = \mathbf{r}_{i-1} - \tilde{\alpha}_i \mathbf{A}\mathbf{z};$$

fin

Comparando los algoritmos correspondientes a las diferentes formas de precondicionamiento, observamos que, a pesar de utilizar vectores distintos, (en el del precondicionamiento por la derecha figuran $\tilde{p}_i, \tilde{q}_i, \tilde{w}, \tilde{u}$, y en los dos restantes p, q, w, u), coinciden en todos sus pasos, a excepción de los valores de los parámetros $\tilde{\rho}_i = \hat{r}_0^T r_{i-1}$ y $\tilde{\alpha}_i = \dfrac{\tilde{\rho}_i}{\hat{r}_0^T v_i}$. Valores estos que dependen del vector inicial $\hat{r}_0$, cuyas expresiones respectivas para cada una de las mismas, en función del vector residuo inicial del sistema auxiliar precondicionado son,

$\hat{r}_0 = \tilde{\tilde{r}}_0$	Precondicionamiento derecha
$\hat{r}_0 = M^{-T}\tilde{\tilde{r}}_0$	Precondicionamiento izquierda
$\hat{r}_0 = L^{-T}\tilde{\tilde{r}}_0$	Precondicionamiento por ambos lados

Si eligiéramos el mismo vector $\hat{r}_0$ para los tres algoritmos, coincidirían todos exactamente. Aunque, en realidad, se estarían inicializando con vectores residuos $\tilde{\tilde{r}}_0$, distintos.

Si, de otra forma, partiéramos de un determinado $\tilde{\tilde{r}}_0$, idéntico para los tres tipos de precondicionamiento, por ejemplo $\tilde{\tilde{r}}_0 = r_0$, los algoritmos diferirían, pués los sucesivos valores del parámetro $\tilde{\rho}_i$ quedan,

$\tilde{\rho}_i = r_0 r_{i-1}$, Precondicionamiento derecha

$\tilde{\rho}_i = \left(M^{-T} r_0\right) r_{i-1}$, Precondicionamiento izquierda

$\tilde{\rho}_i = \left(L^{-T} r_0\right) r_{i-1}$, Precondicionamiento ambos lados

Pero, al ser $\tilde{\tilde{r}}_0$ arbitrario, podemos irlo variando dentro de cada una de las formas de precondicionamiento, existiendo vectores distintos que transforman unos algoritmos en otros. Así, por ejemplo, eligiendo:

$\tilde{\tilde{r}}_0 = r_0$, en Precondicionamiento derecha

$\tilde{\tilde{r}}_0 = M^T r_0$, en Precondicionamiento izquierda

$\tilde{\tilde{r}}_0 = L^T r_0$, en Precondicionamiento ambos lados

los valores de $\hat{r}_0$, son iguales, $\hat{r}_0 = r_0$, y, consiguientemente, coincidirían los parámetros $\tilde{\rho}_i$, y los tres algoritmos proporcionarían los mismos resultados.

En conclusión, el precondicionamiento depende directamente de la elección de $\tilde{\tilde{r}}_0$, y podemos considerar las diferentes formas de precondicionar, por la derecha, por la izquierda ó por ambos lados, tomando uno de los tres algoritmos y variando, bien este vector, bien el vector operacional $\hat{r}_0$. Notemos que estos vectores, a pesar de ser de inicialización, no afectan solo al comienzo del algoritmo, sino que intervienen en cada una de las iteraciones a través de las expresiones de los parámetros $\tilde{\rho}_i$ y $\tilde{\alpha}_i$.

Este razonamiento da lugar a un amplio campo de posibilidades, porque, en principio, la elección de este vector inicial no tiene porque quedar reducida a las tres posibilidades que conducen a las tres formas clásicas de precondicionamiento. Queda así abierto el espectro de expresiones para este vector, y sobre todo la cuestión de cual de ellas será la que proporcione mejores resultados en cada caso, pudiéndose escoger, entre otras, las opciones de:

- igualar $\hat{r}_0$ al producto de determinadas matrices, como la matriz de precondicionamiento ó cualquiera de las resultantes de su factorización, por r_0.

- igualar $\hat{r}_0$ al vector residuo que resulte después de un número determinado de iteraciones al aplicar algún método adecuado en la resolución de $Ax = b$.

Resulta así, **un único algoritmo,** representado por cualquiera de los esquemas anteriores, en el que figuran dos "procesos de remonte", como incremento del coste computacional respecto al algoritmo sin precondicionar y que, como en el Bi-CG, nos proporciona soluciones del sistema original, además de establecer un criterio de parada a partir de los vectores residuos del mismo sistema.

5.3 - CGS* (GRADIENT CONJUGATE SQUARED)

Utilizando las relaciones entre los vectores residuos iniciales del sistema auxiliar para cada una de las distintas formas de precondicionamiento derivadas de la aplicación del Bi-CG, que son, respectivamente,

$$\tilde{\bar{r}}_0 = M^{-T}\bar{r}_0; \quad \tilde{\bar{r}}_0 = \bar{r}_0; \quad \tilde{\bar{r}}_0 = U^{-T}\bar{r}_0$$

según precondicionemos por la derecha, por la izquierda ó por ambos lados, y, aplicando, igual que anteriormente, el CGS al sistema precondicionado, resultan unos algoritmos análogos, que, asimismo, sólo difieren en la aproximación inicial, dando lugar a un único algoritmo, representativo de las tres formas de precondicionar.

Escogiendo, por ejemplo, el Precondicionamiento por ambos lados para obtener este algoritmo, y desarrollando,

$$\tilde{\rho}_i = \tilde{\bar{r}}_0^T \tilde{r}_{i-1} = \left[U^{-T}\bar{r}_0\right]^T L^{-1} r_{i-1} = \bar{r}_0^T U^{-1} L^{-1} r_{i-1} = \bar{r}_0^T M^{-1} r_{i-1}; \; Mz_{i-1} = r_{i-1}; \; \tilde{\rho}_i = \bar{r}_0^T z_{i-1}$$

$$\tilde{\beta}_i = \tilde{\rho}_i / \tilde{\rho}_{i-1}$$

$$U^{-1}\left[\tilde{u}=\tilde{r}_{i-1}+\tilde{\beta}_i\tilde{q}_{i-1}\right];\ u=z_{i-1}+\tilde{\beta}_i q_{i-1}$$

$$U^{-1}\left[\tilde{p}_i=\tilde{u}+\tilde{\beta}_i(\tilde{q}_{i-1}+\tilde{\beta}_i\tilde{p}_{i-1}\right];\ p_i=u+\tilde{\beta}_i\left(q_{i-1}+\beta_i p_{i-1}\right)$$

$$\tilde{v}_i=\tilde{A}\tilde{p}_i=L^{-1}AU^{-1}Up_i;\ v_i=M^{-1}Ap_i;\ Mv_i=Ap_i$$

$$\tilde{\alpha}_i=\frac{\tilde{\rho}_i}{\left(\tilde{r}_0^T\tilde{v}_i\right)}=\frac{\tilde{\rho}_i}{\left(\tilde{r}_0^TU^{-1}Uv_i\right)}=\frac{\tilde{\rho}_i}{\left(\tilde{r}_0^Tv_i\right)}$$

$$U^{-1}\left[\tilde{q}_i=\tilde{u}-\tilde{\alpha}_i\tilde{v}_i\right];\ q_i=u-\tilde{\alpha}_i v_i$$

$$U^{-1}\left[\tilde{w}=\tilde{u}+\tilde{q}_i\right];\ w=u+q_i$$

$$U^{-1}\left[\tilde{x}_i=\tilde{x}_{i-1}+\tilde{\alpha}_i\tilde{w}\right];\ x_i=x_{i-1}+\tilde{\alpha}_i w$$

$$L\left[\tilde{r}_i=\tilde{r}_{i-1}-\tilde{\alpha}_i\tilde{A}\tilde{w}\right];\ r_i=r_{i-1}-\tilde{\alpha}_iLL^{-1}AU^{-1}Uw;\ r_i=r_{i-1}-\tilde{\alpha}_iAw$$

Generalizando el vector residuo que figura en la inicialización, resulta el siguiente algoritmo representativo de esta variante, **Algoritmo precondicionado CGS***, también con dos productos matriz inversa - vector y, con iteraciones x_i y residuos r_i, del sistema $Ax=b$:

Valor inicial $\mathbf{x_0}$, $\mathbf{r_0}=\mathbf{b}-\mathbf{Ax_0}$

Elegimos $\hat{\mathbf{r}}_0$:

$\tilde{\rho}_0=1$

$\tilde{\mathbf{p}}_0=\tilde{\mathbf{q}}_0=\mathbf{0}$

Si $\|\mathbf{r_i}\|/\|\mathbf{r_0}\|\geq\varepsilon,(i=1,2,3,\ldots)$

$\mathbf{Mz}=\mathbf{r_{i-1}}$;

$\tilde{\rho}_i=\hat{\mathbf{r}}_0^{\mathbf{T}}\mathbf{z}$;

$\tilde{\beta}_i=\tilde{\rho}_i/\tilde{\rho}_{i-1}$;

$\mathbf{u}=\mathbf{z}+\tilde{\beta}_i\mathbf{q_{i-1}}$;

$\mathbf{p_i}=\mathbf{u}+\tilde{\beta}_i\left(\mathbf{q_{i-1}}+\tilde{\beta}_i\mathbf{p_{i-1}}\right)$;

$\mathbf{Mv}=\mathbf{Ap_i}$;

$\tilde{\alpha}_i=\tilde{\rho}_i/\left(\hat{\mathbf{r}}_0^{\mathbf{T}}\mathbf{v_i}\right)$;

$$\mathbf{q_i} = \mathbf{u} - \tilde{\alpha}_i \mathbf{v};$$

$$\mathbf{w} = \mathbf{u} + \mathbf{q_i};$$

$$\mathbf{x_i} = \mathbf{x_{i-1}} + \tilde{\alpha}_i \mathbf{w};$$

$$\mathbf{r_i} = \mathbf{r_{i-1}} - \tilde{\alpha}_i \mathbf{A}\mathbf{w};$$

fin

Estos algoritmos que hemos llamado CGS y CGS* aparentemente difieren: los procesos de "remonte" afectan a distintos vectores y aparecen en distintas posiciones en el desarrollo de los mismos. Pero haciendo un estudio de ambos, comprobamos que es posible obtener uno a partir del otro variando el vector inicialización $\hat{r}_0$, (que no se identifica con ningún vector residuo inicial ni del sistema original ni del sistema auxiliar), mediante la relación $\hat{r}_0^* = M^T \hat{r}_0$, y efectuando transformaciones en aquellas expresiones donde figure. Es decir, que en aritmética exacta utilizando esta relación en uno de los algoritmos, los dos serían equivalentes, pero en aritmética finita, esta distinta secuencia de operaciones hace que tengan comportamiento distinto, como se comprueba además, en las aplicaciones numéricas.

Además, es interesante trabajar con ambos algoritmos por separado a efectos de una mayor operatividad en el precondicionamiento, porque nos va a permitir obtener vectores de inicialización distintos, que difieran en el producto por M^t, sin necesidad de almacenar esta matriz traspuesta. Así, haciendo $\hat{r}_0 = \tilde{\tilde{r}}_0$ en cada uno de ellos, para un cierto vector $\tilde{\tilde{r}}_0$, estamos efectuando un precondicionamiento por la derecha con el Algoritmo CGS* y un precondicionamiento por la izquierda con el CGS, pero sin utilizar la relación $\hat{r}_0 = M^{-T} \tilde{\tilde{r}}_0$, y, por consiguiente, sin recurrir al cálculo de la traspuesta de la matriz de precondicionamiento.

5.4 - Bi-CGSTAB

La introducción del parámetro w_i en el polinomio "reductor" $Q_i(A)$ que propone Van der Vorst, da lugar a que los algoritmos obtenidos para las distintas formas de precondicionamiento de manera similar a como se obtuvieron en el CGS, difieran, no sólo en el vector inicialización del algoritmo, sino también en la expresión que define este escalar. Formularemos una condición para obtenerlo tal que, al igual que en el CGS, la forma de precondicionar se identifique con una adecuada elección del vector residuo inicial.

Se presenta, asimismo, una variante para el Bi-CGSTAB, que en aritmética finita ofrece un comportamiento distinto al algoritmo precondicionado que se expone en [55], mejorándolo en ciertas ocasiones.

Obviaremos las relaciones entre matrices, vectores residuos y soluciones antes y después de precondicionar, que son las mismas expuestas en los métodos Bi-CG y CGS.

Precondicionando el sistema $Ax = b$ por la derecha, izquierda y por ambos lados, al igual que en los apartados anteriores, y aplicando el algoritmo del Bi-CGSTAB al sistema precondicionado $\tilde{A}\tilde{x} = \tilde{b}$, resultan los siguientes algoritmos en los que, asimismo, se han realizado las transformaciones correspondientes para obtener soluciones aproximadas del sistema original.

- Precondicionamiento por la derecha

$$\tilde{\rho}_0 = \tilde{\alpha}_0 = \tilde{w}_0 = 1$$

$$\tilde{v}_0 = \tilde{p}_0 = 0$$

$$\tilde{\rho}_i = \tilde{\tilde{r}}_0^T \tilde{r}_{i-1} = \hat{r}_0^T r_{i-1}, \text{ con } \hat{r}_0 = \tilde{\tilde{r}}_0$$

$$\tilde{\beta}_i = (\tilde{\rho}_i / \tilde{\rho}_{i-1})(\tilde{\alpha}_{i-1} / \tilde{w}_{i-1})$$

$$\tilde{p}_i = \tilde{r}_{i-1} + \tilde{\beta}_i(\tilde{p}_{i-1} - \tilde{w}_{i-1}\tilde{v}_{i-1}) = r_{i-1} + \tilde{\beta}_i(\tilde{p}_{i-1} - \tilde{w}_{i-1}\tilde{v}_{i-1})$$

$$\tilde{v}_i = \tilde{A}\tilde{p}_i = AM^{-1}\tilde{p}_i;\ My_i = \tilde{p}_i;\ \tilde{v}_i = Ay_i$$

$$\tilde{\alpha}_i = \tilde{\rho}_i / \left(\tilde{\tilde{r}}_0^T \tilde{v}_i\right)$$

$$\tilde{s} = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{v}_i = r_{i-1} - \tilde{\alpha}_i \tilde{v}_i$$

$$\tilde{t} = \tilde{A}\tilde{s} = AM^{-1}\tilde{s};\ Mz = s;\ \tilde{t} = Az$$

$$\tilde{w}_i = \left(\tilde{t}^T \tilde{s}\right) / \left(\tilde{t}^T \tilde{t}\right)$$

$$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{p}_i + \tilde{w}_i \tilde{s};\ x_i = x_{i-1} + \tilde{\alpha}_i M^{-1}\tilde{p}_i + \tilde{w}_i M^{-1}\tilde{s};\ x_i = x_{i-1} + \tilde{\alpha}_i y_i + \tilde{w}_i z$$

$$\tilde{r}_i = \tilde{s} - \tilde{w}_i \tilde{t};\ r_i = \tilde{s} - \tilde{w}_i \tilde{t}$$

Con estas expresiones, el esquema del **Algoritmo Bi-CGSTAB con precondicionamiento por la derecha** queda como sigue:

Valor inicial $\mathbf{x_0}$, $\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$;

Elegimos $\hat{\mathbf{r}}_0^{\mathbf{T}}$;

$\rho_0 = \alpha_0 = \tilde{w}_0 = 1$;

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

Si $\|\mathbf{r_i}\| / \|\mathbf{r_0}\| \geq \varepsilon$, $(i = 1,2,3,\ldots)$;

$\tilde{\rho}_i = \hat{\mathbf{r}}_0^{\mathbf{T}} \mathbf{r}_{\mathbf{i-1}}$;

$\tilde{\beta}_i = (\tilde{\rho}_i / \tilde{\rho}_{i-1})(\tilde{\alpha}_{i-1} / \tilde{w}_{i-1})$;

$\tilde{\mathbf{p}}_\mathbf{i} = \mathbf{r}_{\mathbf{i-1}} + \tilde{\beta}_i (\tilde{\mathbf{p}}_{\mathbf{i-1}} - \tilde{w}_{i-1} \tilde{\mathbf{v}}_{\mathbf{i-1}})$;

$\mathbf{M}\mathbf{y}_\mathbf{i} = \tilde{\mathbf{p}}_\mathbf{i}$;

$\tilde{\mathbf{v}}_\mathbf{i} = \mathbf{A}\mathbf{y}_\mathbf{i}$;

$\tilde{\alpha}_i = \tilde{\rho}_i / (\hat{\mathbf{r}}_0^{\mathbf{T}} \tilde{\mathbf{v}}_\mathbf{i})$;

$\tilde{\mathbf{s}} = \mathbf{r}_{\mathbf{i-1}} - \tilde{\alpha}_i \tilde{\mathbf{v}}_\mathbf{i}$;

$\mathbf{M}\mathbf{z} = \mathbf{s}$;

$\tilde{\mathbf{t}} = \mathbf{A}\mathbf{z}$;

$\tilde{w}_i = \left(\tilde{\mathbf{t}}^{\mathbf{T}} \tilde{\mathbf{s}}\right) / \left(\tilde{\mathbf{t}}^{\mathbf{T}} \tilde{\mathbf{t}}\right)$;

$\mathbf{x}_\mathbf{i} = \mathbf{x}_{\mathbf{i-1}} + \tilde{\alpha}_i \mathbf{y}_\mathbf{i} + \tilde{w}_i \mathbf{z}$;

$\mathbf{r}_i = \tilde{\mathbf{s}} - \tilde{w}_i \tilde{\mathbf{t}}$;

fin

- Precondicionamiento por la izquierda

$\tilde{\rho}_0 = \tilde{\alpha}_0 = \tilde{w}_0 = 1$

$\tilde{v}_0 = \tilde{p}_0 = 0$

$\tilde{\rho}_i = \tilde{\tilde{r}}_0^T \tilde{r}_{i-1} = \tilde{\tilde{r}}_0^T M^{-1} r_{i-1} = \left[M^{-T} \tilde{\tilde{r}}_0 \right]^T r_{i-1}$; $M^{-T} \tilde{\tilde{r}}_0 = \hat{r}_0$; $\tilde{\rho}_i = \hat{r}_0^T r_{i-1}$

$\tilde{\beta}_i = (\tilde{\rho}_i / \tilde{\rho}_{i-1})(\tilde{\alpha}_{i-1} / \tilde{w}_{i-1})$

$\tilde{p}_i = \tilde{r}_{i-1} + \tilde{\beta}_i (\tilde{p}_{i-1} - \tilde{w}_{i-1} \tilde{v}_{i-1})$, *multiplicando por* M, $p_i = r_{i-1} + \tilde{\beta}_i (p_{i-1} - \tilde{w}_{i-1} v_{i-1})$

$\tilde{v}_i = \tilde{A} \tilde{p}_i = M^{-1} A M^{-1} p_i$; $v_i = A M^{-1} p_i$; $M y_i = p_i$; $v_i = A y_i$

$$\tilde{\alpha}_i = \frac{\tilde{\rho}_i}{\tilde{\tilde{r}}_0^T \tilde{v}_i} = \frac{\tilde{\rho}_i}{\tilde{\tilde{r}}_0^T M^{-1} v_i} = \frac{\tilde{\rho}_i}{\left(M^{-T} \tilde{\tilde{r}}_0 \right)^T v_i} = \frac{\tilde{\rho}_i}{\hat{r}_0^T v_i}$$

$\tilde{s} = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{v}_i = M^{-1} r_{i-1} - \tilde{\alpha}_i M^{-1} A M^{-1} p_i$; $s = r_{i-1} - \tilde{\alpha}_i v_i$

$\tilde{t} = \tilde{A} \tilde{s} = M^{-1} A M^{-1} s$; $t = A M^{-1} s$; $M z = s$; $t = A z$

$$\tilde{w}_i = \frac{\tilde{t}^T \tilde{s}}{\tilde{t}^T \tilde{t}} = \frac{\left(M^{-1} t \right)^T \left(M^{-1} s \right)}{\left(M^{-1} t \right)^T \left(M^{-1} t \right)}$$

$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{p}_i + \tilde{w}_i \tilde{s}$; $x_i = x_{i-1} + \tilde{\alpha}_i M^{-1} p_i + \tilde{w}_i M^{-1} s = x_{i-1} + \tilde{\alpha}_i y_i + \tilde{w}_i z$

$\tilde{r}_i = \tilde{s} - \tilde{w}_i \tilde{t}$; $r_i = s - \tilde{w}_i t$

El correspondiente **Algoritmo Bi-CGSTAB precondicionado por la izquierda** sería:

Valor inicial $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;

Elegimos $\hat{\mathbf{r}}_0^T$;

$\tilde{\rho}_0 = \tilde{\alpha}_0 = \tilde{w}_0 = 1$;

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

Si $\|\mathbf{r}_i\| / \|\mathbf{r}_0\| \geq \varepsilon$, $(i = 1,2,3,...)$;

$$\tilde{\rho}_i = \hat{\mathbf{r}}_0^{\mathrm{T}}\mathbf{r}_{i-1};$$

$$\tilde{\beta}_i = (\tilde{\rho}_i/\tilde{\rho}_{i-1})(\tilde{\alpha}_{i-1}/\tilde{w}_{i-1});$$

$$\mathbf{p}_i = \mathbf{r}_{i-1} + \tilde{\beta}_i(\mathbf{p}_{i-1} - \tilde{w}_{i-1}\mathbf{v}_{i-1});$$

$$\mathbf{M}\mathbf{y}_i = \mathbf{p}_i;$$

$$\mathbf{v}_i = \mathbf{A}\mathbf{y}_i;$$

$$\tilde{\alpha}_i = \tilde{\rho}_i/(\hat{\mathbf{r}}_0^{\mathrm{T}}\mathbf{v}_i);$$

$$\mathbf{s} = \mathbf{r}_{i-1} - \tilde{\alpha}_i\mathbf{v}_i;$$

$$\mathbf{M}\mathbf{z} = \mathbf{s};$$

$$\mathbf{t} = \mathbf{A}\mathbf{z};$$

$$\tilde{w}_i = \frac{(\mathbf{M}^{-1}\mathbf{t})^{\mathrm{T}}(\mathbf{M}^{-1}\mathbf{s})}{(\mathbf{M}^{-1}\mathbf{t})^{\mathrm{T}}(\mathbf{M}^{-1}\mathbf{t})};$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \tilde{\alpha}_i\mathbf{y}_i + \tilde{w}_i\mathbf{z};$$

$$\mathbf{r}_i = \mathbf{s} - \tilde{w}_i\mathbf{t};$$

fin

- Precondicionamiento por ambos lados

$$\tilde{\rho}_0 = \tilde{\alpha}_0 = \tilde{w}_0 = 1$$

$$\tilde{v}_0 = \tilde{p}_0 = 0$$

$$\tilde{\rho}_i = \tilde{\tilde{r}}_0^T\tilde{r}_{i-1} = \tilde{\tilde{r}}_0^T L^{-1} r_{i-1} = \left[L^{-T}\tilde{\tilde{r}}_0\right]^T r_{i-1} = \hat{r}_0^T r_{i-1}$$

$$\tilde{\beta}_i = (\tilde{\rho}_i/\tilde{\rho}_{i-1})(\tilde{\alpha}_{i-1}/\tilde{w}_{i-1})$$

$$L\left[\tilde{p}_i = \tilde{r}_{i-1} + \tilde{\beta}_i(\tilde{p}_{i-1} - \tilde{w}_{i-1}\tilde{v}_{i-1})\right]\colon\ p_i = r_{i-1} + \tilde{\beta}_i(p_{i-1} - \tilde{w}_{i-1}v_{i-1})$$

$$\tilde{v}_i = \tilde{A}\tilde{p}_i = L^{-1}AU^{-1}L^{-1}p_i;\ v_i = AM^{-1}p_i;\ My_i = p_i;\ v_i = Ay_i$$

$$\tilde{\alpha}_i = \frac{\tilde{\rho}_i}{\tilde{\tilde{r}}_0^T\tilde{v}_i} = \frac{\tilde{\rho}_i}{\tilde{\tilde{r}}_0^T L^{-1}v_i} = \frac{\tilde{\rho}_i}{\left[L^{-T}\tilde{\tilde{r}}_0\right]^T v_i} = \frac{\tilde{\rho}_i}{\hat{r}_0^T v_i}$$

$$\tilde{s} = \tilde{r}_{i-1} - \tilde{\alpha}_i\tilde{v}_i = \tilde{r}_{i-1} - \tilde{\alpha}_i L^{-1}AU^{-1}L^{-1}p_i;\ s = r_{i-1} - \tilde{\alpha}_i AM^{-1}p_i;\ s = r_{i-1} - \tilde{\alpha}_i v_i$$

$$\tilde{t} = \tilde{A}\tilde{s} = L^{-1}AU^{-1}L^{-1}s;\ t = AM^{-1}s;\ Mz = s;\ t = Az$$

$$\tilde{w}_i = \frac{\tilde{t}^T\tilde{s}}{\tilde{t}^T\tilde{t}} = \frac{\left(L^{-1}t\right)^T\left(L^{-1}s\right)}{\left(L^{-1}t\right)^T\left(L^{-1}t\right)}$$

$$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i\tilde{p}_i + \tilde{w}_i\tilde{s};\ U^{-1}\left[\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i L^{-1}p_i + \tilde{w}_i L^{-1}s\right];\ x_i = x_{i-1} + \tilde{\alpha}_i y_i + \tilde{w}_i z$$

$$\tilde{r}_i = \tilde{s} - \tilde{w}_i\tilde{t};\ r_i = s - \tilde{w}_i t$$

Con las operaciones finales de las expresiones resulta el siguiente

Algoritmo Bi-CGSTAB con precondicionamiento por ambos lados:

Valor inicial $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;

Elegimos $\hat{\mathbf{r}}_0$;

$\tilde{\rho}_0 = \tilde{\alpha}_0 = \tilde{w}_0 = 1$;

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

Si $\|\mathbf{r}_i\|/\|\mathbf{r}_0\| \geq \varepsilon,\ (i = 1,2,3,\ldots)$;

$\tilde{\rho}_i = \hat{\mathbf{r}}_0^{\mathrm{T}}\mathbf{r}_{i-1}$;

$\tilde{\beta}_i = (\tilde{\rho}_i/\tilde{\rho}_{i-1})(\tilde{\alpha}_{i-1}/\tilde{w}_{i-1})$;

$\mathbf{p}_i = \mathbf{r}_{i-1} + \tilde{\beta}_i(\mathbf{p}_{i-1} - \tilde{w}_{i-1}\mathbf{v}_{i-1})$;

$\mathbf{M}\mathbf{y}_i = \mathbf{p}_i$;

$\mathbf{v}_i = \mathbf{A}\mathbf{y}_i$;

$\tilde{\alpha}_i = \tilde{\rho}_i/(\hat{\mathbf{r}}_0^{\mathrm{T}}\mathbf{v}_i)$;

$\mathbf{s} = \mathbf{r}_{i-1} - \tilde{\alpha}_i\mathbf{v}_i$;

$\mathbf{M}\mathbf{z} = \mathbf{s}$;

$\mathbf{t} = \mathbf{A}\mathbf{z}$;

$$\tilde{w}_i = \frac{\left(\mathbf{L}^{-1}\mathbf{t}\right)^{\mathrm{T}}\left(\mathbf{L}^{-1}\mathbf{s}\right)}{\left(\mathbf{L}^{-1}\mathbf{t}\right)^{\mathrm{T}}\left(\mathbf{L}^{-1}\mathbf{t}\right)};$$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \tilde{\alpha}_i\mathbf{y}_i + \tilde{w}_i\mathbf{z}$;

$\mathbf{r}_i = \mathbf{s} - \tilde{w}_i\mathbf{t}$;

fin

Si comparamos los algoritmos correspondientes a los distintos precondicionamientos, observamos que no sólo difieren en el vector de inicialización $\hat{r}_0$, de expresiones respectivas,

$$\hat{r}_0 = \tilde{\tilde{r}}_0$$
$$\hat{r}_0 = M^{-T}\tilde{\tilde{r}}_0$$
$$\hat{r}_0 = L^{-1}\tilde{\tilde{r}}_0$$

sino, también, en los valores del parámetro $\tilde{w}_i$ que interviene en la expresión polinómica del vector residuo que propone Van der Vorst.

La expresión de este escalar se ha obtenido imponiendo la condición de mínimo a la norma del vector residuo $\tilde{r}_i$, del sistema precondicionado, que, en principio, es el sistema al que se le aplica el algoritmo Bi-CGSTAB, aunque con las transformaciones realizadas se calculen vectores residuos y solución aproximada del sistema $Ax = b$. Su evaluación implica un "proceso de remonte" adicional en cada iteración para el precondicionamiento por la izquierda y dos en el precondicionamiento por ambos lados.

Si determinásemos su valor minimizando la misma norma para el vector residuo r_i, del sistema sin precondicionar $Ax = b$, quedaría, derivando esta norma e igualando a cero,

$$w_i = \left(t^T s\right) / \left(t^T t\right)$$

donde figuran vectores ya evaluados en pasos anteriores del algoritmo, eliminándose los productos matriz inversa por vector que había que efectuar para su cálculo y que incrementan el coste computacional en los algoritmos correspondientes a los precondicionamientos por la izquierda y por ambos lados. En el precondicionamiento por la derecha no se plantea esta cuestión, ya que en este caso los vectores residuos iniciales del sistema original y del sistema precondicionado coinciden.

Con esta consideración, la única diferencia entre los distintos algoritmos va a estribar, al igual que en el CGS, en el vector inicialización, con lo que

podremos escoger uno cualquiera representativo de los tres, donde elegiremos distintas opciones, bien para el vector $\hat{r}_0$, bien para $\tilde{\bar{r}}_0$, como efecto de precondicionamiento. Esto equivale a escoger el algoritmo correspondiente al precondicionamiento por la derecha, que es el menos costoso, y variar en el mismo el vector operacional inicial en el campo de posibilidades que ya hemos contemplado.

5.5 - ALGORITMO Bi-CGSTAB*

Procediendo de forma similar que en el CGS, es decir, utilizando las relaciones obtenidas en la inicialización del Bi-CG para los distintos modos de precondicionamiento,

$$\tilde{\bar{r}}_0 = M^{-T}\bar{r}_0; \quad \tilde{\bar{r}}_0 = \bar{r}_0; \quad \tilde{\bar{r}}_0 = U^{-T}\bar{r}_0$$

y aplicando otra vez el algoritmo del Bi-CGSTAB al sistema precondicionado, resultan otros tantos algoritmos, que difieren en el vector inicialización y en el valor de $\tilde{w}_i$. Conviniendo en calcular éste por minimización de $\|r_i\|_2$, para evitarnos nuevos productos matriz por vector, se obtienen, al igual que anteriormente, expresiones idénticas para el mismo, con lo que los algoritmos vuelven a diferir solamente en el vector $\hat{r}_0$, convirtiendo, una vez más, la elección del vector de inicialización en la forma general de precondicionar el sistema.

Escogiendo, por ejemplo, el precondicionamiento por ambos lados para obtener el algoritmo representativo:

$$\tilde{\rho}_i = \tilde{\bar{r}}_0^T\tilde{r}_{i-1} = \left[U^{-T}\bar{r}_0\right]^T L^{-1}r_{i-1} = \bar{r}_0^T U^{-1}L^{-1}r_{i-1} = \bar{r}_0^T M^{-1}r_{i-1}; \; Mz_{i-1} = r_{i-1}; \; \tilde{\rho}_i = \bar{r}_0^T z_{i-1}$$

$$\tilde{\beta}_i = (\tilde{\rho}_i/\tilde{\rho}_{i-1})(\tilde{\alpha}_{i-1}/\tilde{w}_{i-1})$$

$$U^{-1}\left[\tilde{p}_i = \tilde{r}_{i-1} + \tilde{\beta}_i(\tilde{p}_{i-1} - \tilde{w}_{i-1}\tilde{v}_{i-1})\right]; \; p_i = z_{i-1} + \tilde{\beta}_i(p_{i-1} - \tilde{w}_{i-1}v_{i-1})$$

$$\tilde{v}_i = \tilde{A}\tilde{p}_i = L^{-1}AU^{-1}\tilde{p}_i; \; v_i = M^{-1}Ap_i; \; Mv_i = Ap_i$$

$$\tilde{\alpha}_i = \frac{\tilde{\rho}_i}{\tilde{\bar{r}}_0^T\tilde{v}_i} = \frac{\tilde{\rho}_i}{\bar{r}_0^T U^{-1}L^{-1}AU^{-1}\tilde{p}_i} = \frac{\tilde{\rho}_i}{\bar{r}_0^T M^{-1}Ap_i} = \frac{\tilde{\rho}_i}{\bar{r}_0^T v_i}$$

$$U^{-1}\left[\tilde{s} = \tilde{r}_{i-1} - \tilde{\alpha}_i \tilde{v}_i\right];\ s = U^{-1}L^{-1}r_{i-1} - \tilde{\alpha}_i U^{-1}\tilde{v}_i = M^{-1}r_{i-1} - \tilde{\alpha}_i v_i = z_{i-1} - \tilde{\alpha}_i v_i$$

$$L\left[\tilde{t} = \tilde{A}\tilde{s}\right];\ t = LL^{-1}AU^{-1}Us = As$$

$$L\left[\tilde{r}_i = \tilde{s} - \tilde{w}_i \tilde{t}\right];\ r_i = LUs - \tilde{w}_i L\tilde{t} = Ms - \tilde{w}_i t;\ Ms = u;\ r_i = u - \tilde{w}_i t$$

Minimizando $\|r_i\|_2 = r_i^T r_i$, obtenemos $\tilde{w}_i = \left(t^T u\right)/\left(t^T t\right)$

$$U\left[\tilde{x}_i = \tilde{x}_{i-1} + \tilde{\alpha}_i \tilde{p}_i + \tilde{w}_i s\right];\ x_i = x_{i-1} + \tilde{\alpha}_i p_i + \tilde{w}_i s$$

Utilizando los tipos de precondicionamiento por la derecha ó por la izquierda, se obtienen expresiones finales similares, con un vector de inicialización diferente. Generalizando este vector residuo, cuyas posibilidades de elección nos proporcionan precisamente, en un único algoritmo, distintas opciones de precondicionar, resulta para esta variante el siguiente **Algoritmo precondicionado Bi-CGSTAB*:**

Valor inicial $\mathbf{x_0}$, $\mathbf{r_0} = \mathbf{b} - \mathbf{A}\mathbf{x_0}$;

Elegimos $\tilde{\mathbf{r}}_0$;

$\tilde{\rho}_0 = \tilde{\alpha}_0 = \tilde{w}_0 = 1$;

$\mathbf{v_0} = \mathbf{p_0} = \mathbf{0}$;

Si $\|\mathbf{r}_i\|/\|\mathbf{r}_0\| \geq \varepsilon$, $(i = 1,2,3,\ldots)$:

$\mathbf{Mz} = \mathbf{r}_{i-1}$;

$\tilde{\rho}_i = \hat{\mathbf{r}}_0^{\mathbf{T}}\mathbf{z}$;

$\tilde{\beta}_i = \left(\tilde{\rho}_i/\tilde{\rho}_{i-1}\right)\left(\tilde{\alpha}_{i-1}/\tilde{w}_{i-1}\right)$;

$\mathbf{p_i} = \mathbf{z} + \beta_i\left(\mathbf{p_{i-1}} - \tilde{w}_{i-1}\mathbf{v_{i-1}}\right)$;

$\mathbf{y} = \mathbf{A}\mathbf{p_i}$;

$\mathbf{M}\mathbf{v_i} = \mathbf{y}$;

$\tilde{\alpha}_i = \tilde{\rho}_i/\left(\hat{\mathbf{r}}_0^{\mathbf{T}}\mathbf{v_i}\right)$;

$\mathbf{s} = \mathbf{r_{i-1}} - \tilde{\alpha}_i\mathbf{y}$;

$\mathbf{u} = \mathbf{z} - \tilde{\alpha}_i\mathbf{v_i}$;

$\mathbf{t} = \mathbf{A}\mathbf{u}$;

$$\mathbf{t} = \mathbf{A}\mathbf{u};$$

$$\tilde{w}_i = \left(\mathbf{t}^{\mathrm{T}}\mathbf{s}\right)/\left(\mathbf{t}^{\mathrm{T}}\mathbf{t}\right);$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \tilde{\alpha}_i \mathbf{p}_i + \tilde{w}_i \mathbf{u};$$

$$\mathbf{r}_i = \mathbf{s} - \tilde{w}_i \mathbf{t};$$

fin

Para los algoritmos Bi-CGSTAB y Bi-CGSTAB* podemos hacer los mismos comentarios y conclusiones que se expusieron en el estudio del CGS. Se puede obtener, asimismo, un algoritmo a partir del otro, sustituyendo $\hat{r}_0^* = M^T \hat{r}_0$ en el segundo y realizando las operaciones adecuadas, con lo que en aritmética exacta coincidirían. Pero inicializando ambos algoritmos con vectores residuos que guarden esa relación, se obtendrían comportamientos distintos en aritmética finita, ya que se realizan las operaciones en distinta disposición y los vectores que intervienen en las mismas son diferentes.

De todas maneras, al igual que entonces, la utilización de ambos algoritmos por separado con la misma inicialización, por ejemplo $\hat{r}_0 = \tilde{\tilde{r}}_0$, nos permite efectuar precondicionamientos diferentes, de tal forma que en uno de ellos podremos precondicionar con el vector inicialización $\hat{r}_0 = M^{-T}\tilde{\tilde{r}}_0$, sin necesidad de utilizar M^T.

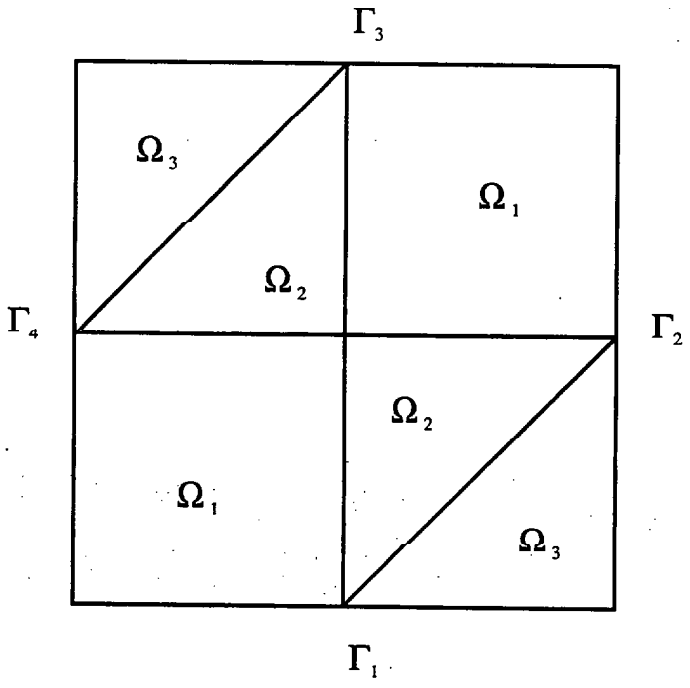
CAPÍTULO 6

APLICACIONES TEST

En este capítulo se aplican los algoritmos y técnicas detallados en apartados anteriores a la resolución de algunos problemas test clásicos, que se discretizan con MEF.

El objetivo primordial de la resolución de estos problemas, antes que obtener una solución como resultado a un problema físico concreto, es comprobar la efectividad de dichos algoritmos, formas de precondicionamiento y técnicas de renumeración en casos que se vean reflejados en distintos sistemas de ecuaciones apropiados, por lo que una vez hecha la formulación clásica del problema, elegiremos valores para las constantes físicas que figuren en ella, adecuados a esta circunstancia.

Con esta misma idea, todos los problemas son referidos a un dominio de geometría simple, cuadrado de lado unidad $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$, de frontera $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$.



El Código Neptuno [12], además de generar el sistema de ecuaciones, nos permite obtener un fichero de salida que luego será utilizado para representar el mallado correspondiente y la gráfica de la solución aproximada para cada una de estas aplicaciones test, mediante un software gráfico.

Para los problemas que se exponen a continuación se ha utilizado un indicador de error de tipo gradiente $\varepsilon_i = h_i |\nabla u_h|$, siendo h_i el diámetro del elemento δ_i y u_h la solución numérica correspondiente. Se han seguido estrategias de refinamiento similares a las propuestas en [33]. Este indicador mide la máxima variación de la solución numérica en el elemento δ_i (en el caso del problema de elasticidad se ha considerado el módulo del vector desplazamiento).

Aplicaremos los algoritmos sin precondicionar y las dos versiones de estos algoritmos precondicionados con las matrices de precondicionamiento, Diagonal, ILU(0) y SSOR.

En todos los casos, las matrices se almacenan en la forma compacta descrita en el Capítulo 3, y las operaciones que figuran en los algoritmos, además de las necesarias para la construcción de estos precondicionadores, se han programado ateniéndose a este almacenamiento.

Se exponen los resultados, recurriendo a diagramas de barras donde representamos número de iteraciones y tiempo de computación necesarios para alcanzar una determinada tolerancia y curvas de convergencia que nos dan información sobre la evolución más ó menos "suave" de la norma residual con el número de iteraciones. En aquellas ocasiones, en las que exista relación directa entre número de iteraciones y tiempos, sólo representaremos las iteraciones necesarias para alcanzar la tolerancia exigida. También hemos descartado los resultados de aquellos casos en los que la convergencia es muy lenta y la tolerancia deseada se alcanza para un número de iteraciones proporcionalmente elevado respecto a otros métodos ó técnicas.

En general se ha tomado como vector inicial de cada algoritmo el vector residuo del sistema original sin precondicionar $\hat{r}_0 = r_0$. En uno de los casos, sin embargo, se han elegido distintos vectores para la inicialización a fin de comprobar el efecto que produce esta circunstancia en el precondicionamiento.

En las resoluciones, realizadas en un ordenador HP 9000/730, con Sistema operativo HP-UX versión 9.01, se ha fijado un valor de tolerancia de 10^{-10}. Este valor de tolerancia, en ausencia de "reinicialización" por diferencia entre el residuo calculado por la relación de recurrencia y el residuo real, no afecta a las curvas de convergencia, aunque, evidentemente, tenga relación directa con el número de iteraciones ó el tiempo necesarios para alcanzarla. Se presentan, en distintos casos, a efectos de comprobación, resultados obtenidos con una tolerancia para la norma residual de 10^{-6}, que se adaptan a las curvas de convergencia correspondientes, obtenidas con tolerancia de 10^{-10}. Las gráficas nos van a permitir:

- Establecer comparaciones en la utilización de los algoritmos precondicionados, Bi-CG, CGS, Bi-CGSTAB y los que hemos denotado por Bi-CG*, CGS* y Bi-CGSTAB*.

- Contrastar la bondad de los distintos precondicionadores y comparar con la aplicación de los mismos algoritmos sin precondicionar. Hay que contemplar el coste inicial de la implementación del precondicionador y el coste por iteración, que puede hacer que en un determinado caso la utilización de un precondicionador de lugar a mayor número de iteraciones que otro, pero con menor tiempo de ejecución.

- Comprobar el efecto que producen las técnicas de renumeración en la aplicación del precondicionador ILU(0), comparando con los resultados obtenidos con el mismo precondicionador y con los precondicionadores Diagonal y SSOR sin reordenar.

6.1 - PROBLEMA 1

Se estudia la deformación de una laja elástica delgada Ω, *sometida a una carga constante en* Γ_2, ($L_2 = -1\ kg/m$), *y a otra que varía linealmente en* Γ_3, ($L_3 = -x/25\ kg/m$). *Se impone la condición de inmovilidad en* Γ_1 *y en el punto (1,1). Se considerarán nulas las fuerzas de volumen.*

La figura 1 representa el mallado correspondiente a una etapa intermedia del proceso de refinamiento adaptativo, incluyendo la deformación de forma acusada.

La solución para el sistema, precondicionado con el precondicionador Diagonal, aplicando los distintos algoritmos, se obtiene en tiempo y número de iteraciones similares a los que resultan con el sistema sin precondicionar. A pesar que el ILU(0) y el SSOR reducen sensiblemente el número de iteraciones, el coste de su aplicación hace que los tiempos de ejecución sean superiores a los que da lugar el Diagonal ó incluso, dado que el sistema está bien condicionado, a los que se obtiene resolviendo el sistema inicial sin precondicionar. Notar, asimismo, que, para todos los métodos, el precondicionador SSOR da lugar a un número de iteraciones mayor que en el ILU(0), pero con menores tiempos, debido al coste comparativamente más elevado de la "primera iteración" para éste.

En cuanto a las curvas de convergencia, cuando se utilizan los precondicionadores ILU(0) y Diagonal se suavizan en los algoritmos Bi-CG*, CGS* y BICGSTAB*, sobre todo en los dos primeros, frente a las correspondientes variantes Bi-CG, CGS y Bi-CGSTAB, según se comprueba en las figuras 7 y 9, respectivamente, mientras que en la aplicación del SSOR, no se logra este efecto de suavizado (figura 8), presentándose abundantes "picos" en las curvas correspondientes a todos los algoritmos.

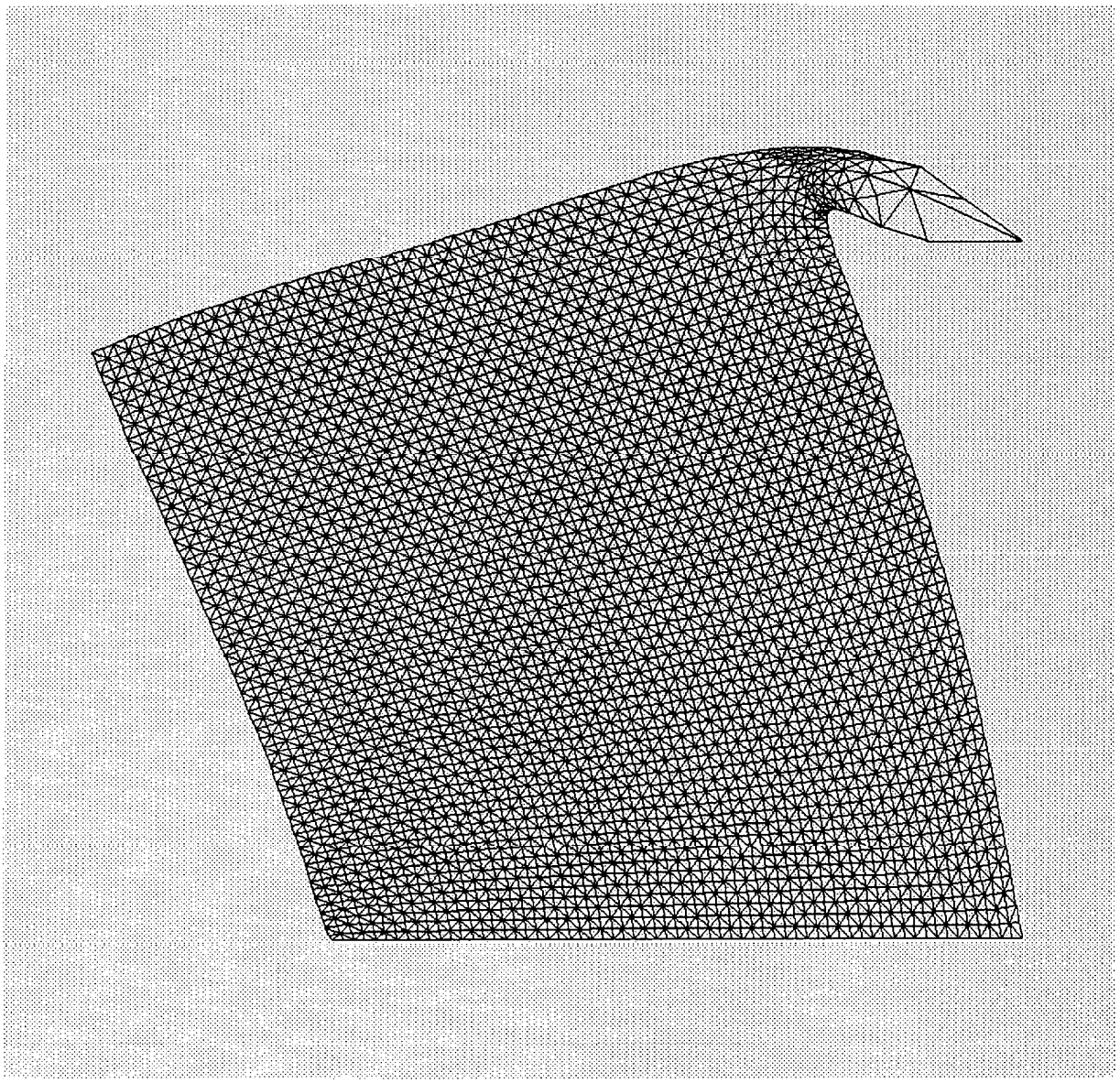


Figura 1: Problema 1. Mallado y desplazamiento.

Considerando la efectividad de los diferentes métodos para los mismos precondicionadores, los algoritmos de las dos versiones del CGS y Bi-CGSTAB proporcionan siempre mejores resultados que el correspondiente al Bi-CG.

Diagramas de barras

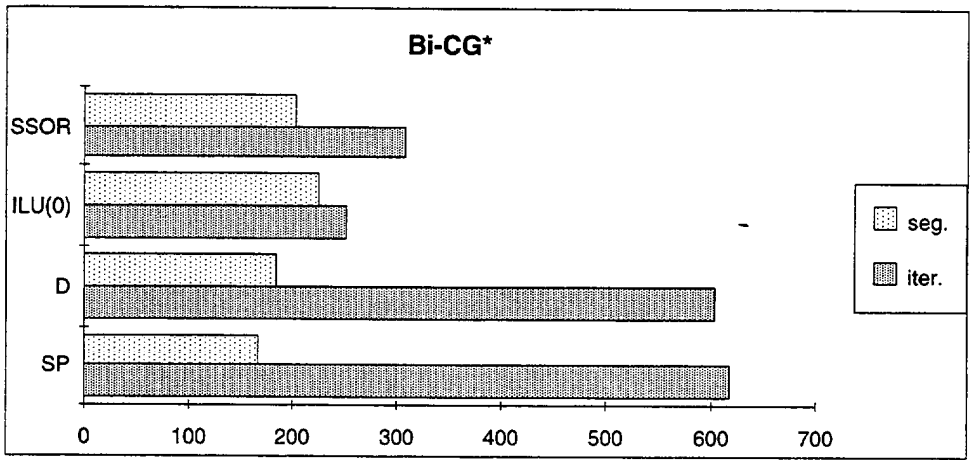


Figura 2: Problema 1 (8434 ecuaciones).

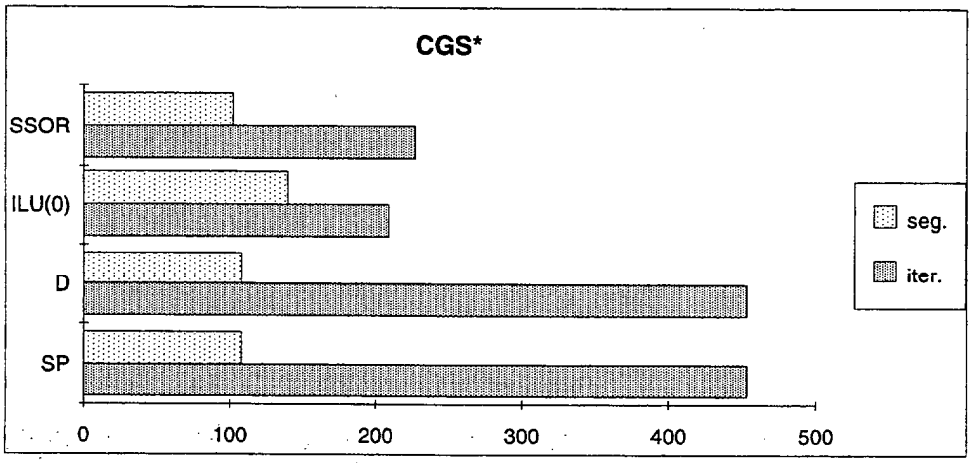


Figura 3: Problema 1 (8434 ecuaciones).

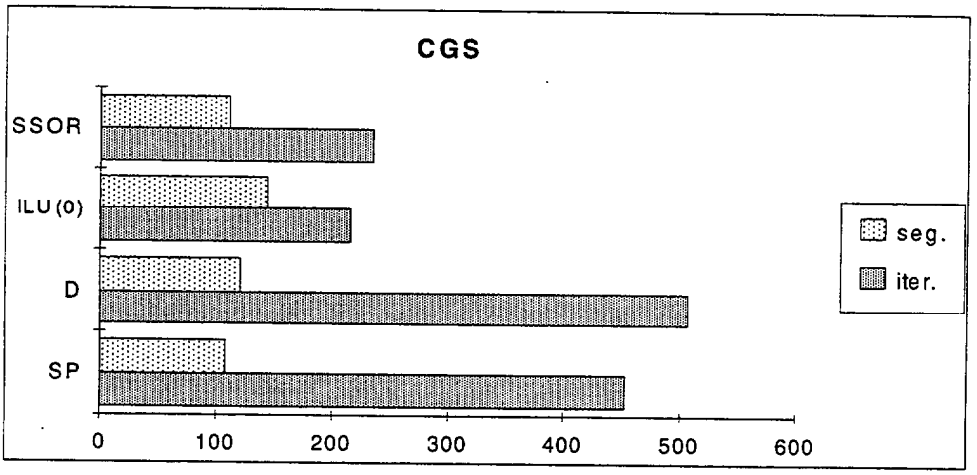


Figura 4: Problema 1 (8434 ecuaciones).

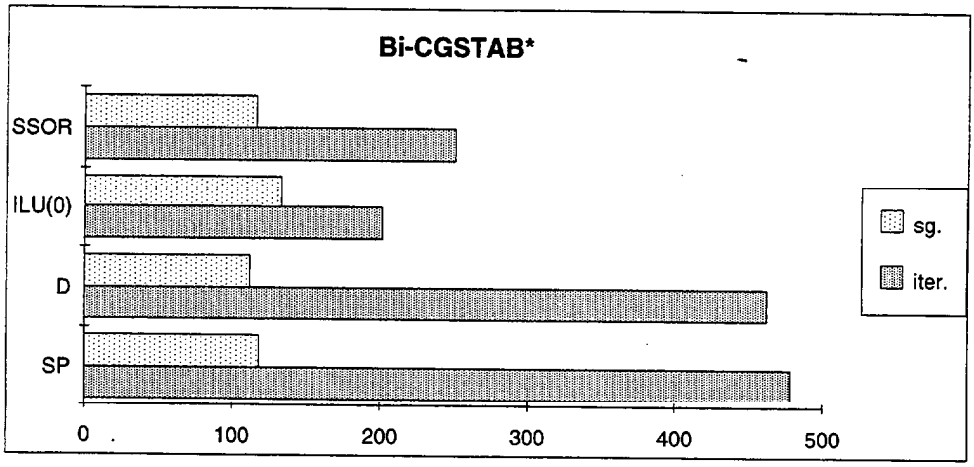


Figura 5: Problema 1 (8434 ecuaciones).

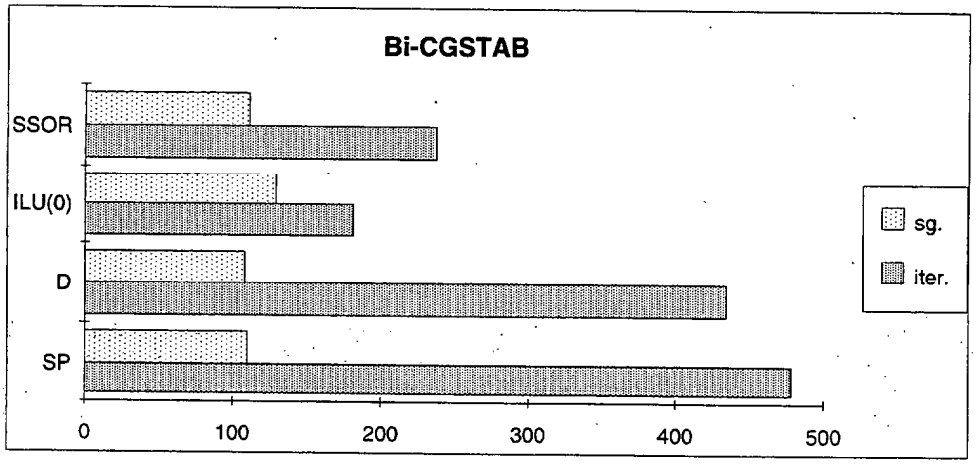


Figura 6: Problema 1 (8434 ecuaciones).

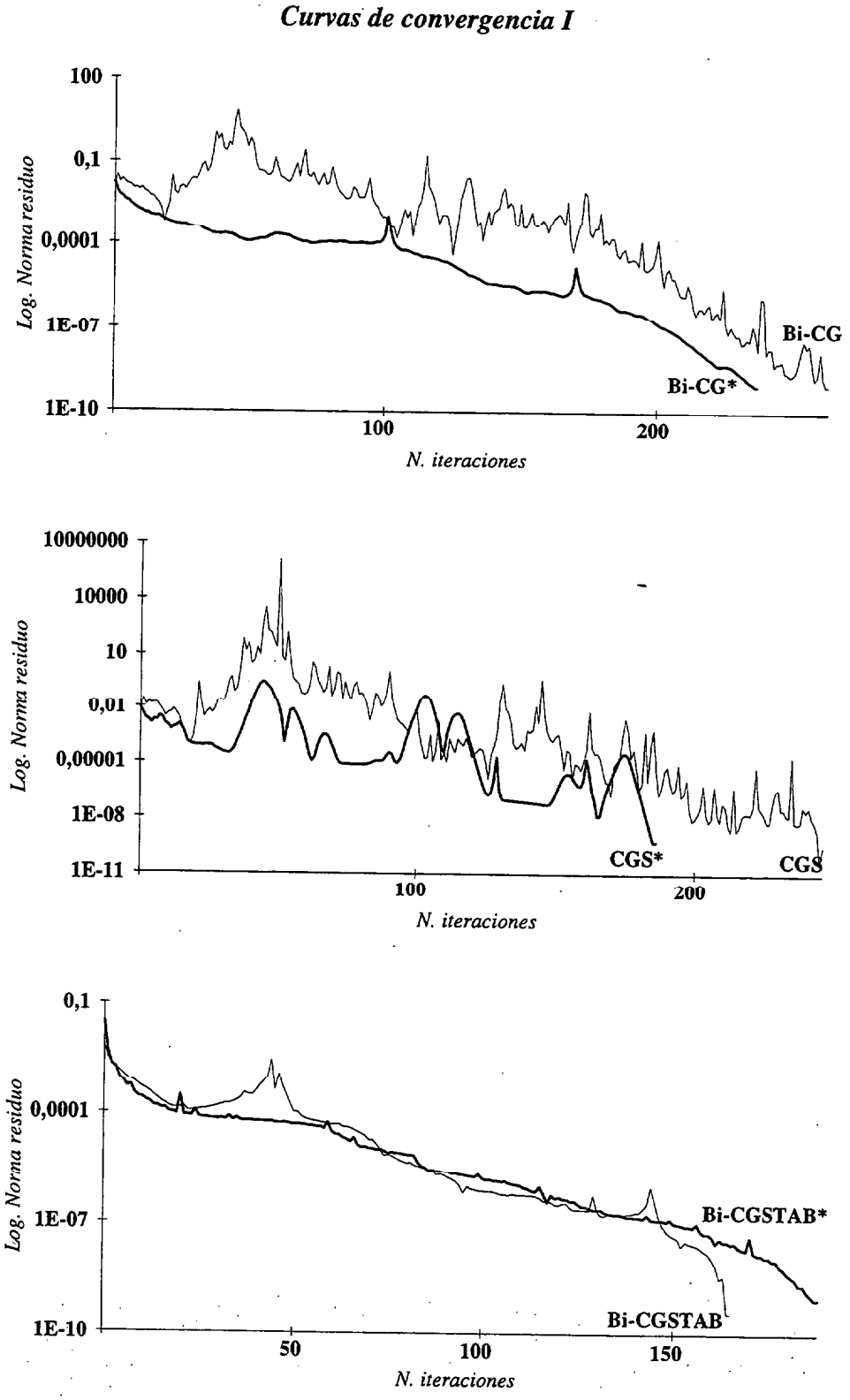


Figura 7: Problema 1 (8434 ecuaciones). Norma del residuo para ILU(0).

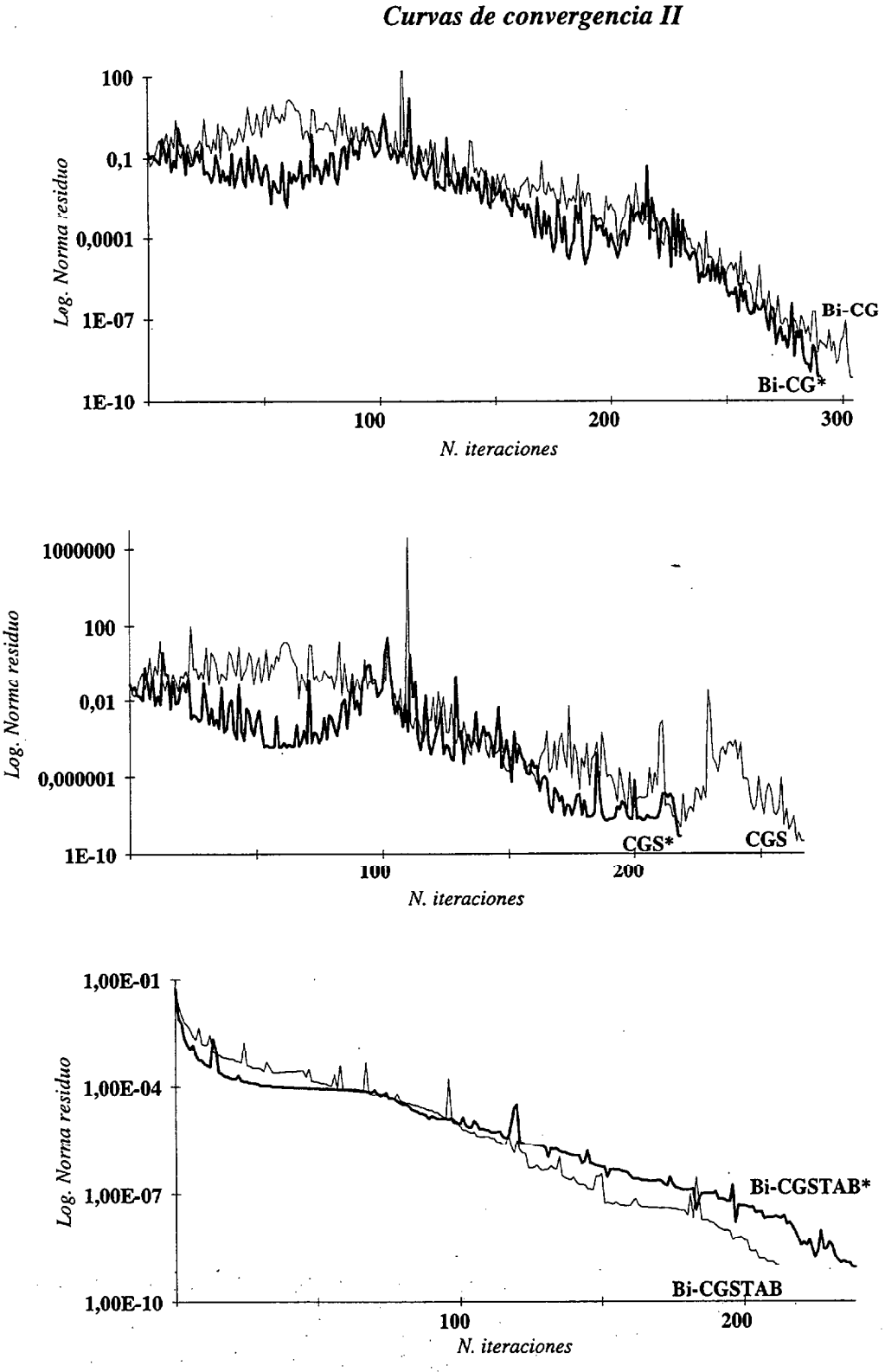


Figura 8: Problema 1 (8434 ecuaciones). Norma del residuo para SSOR

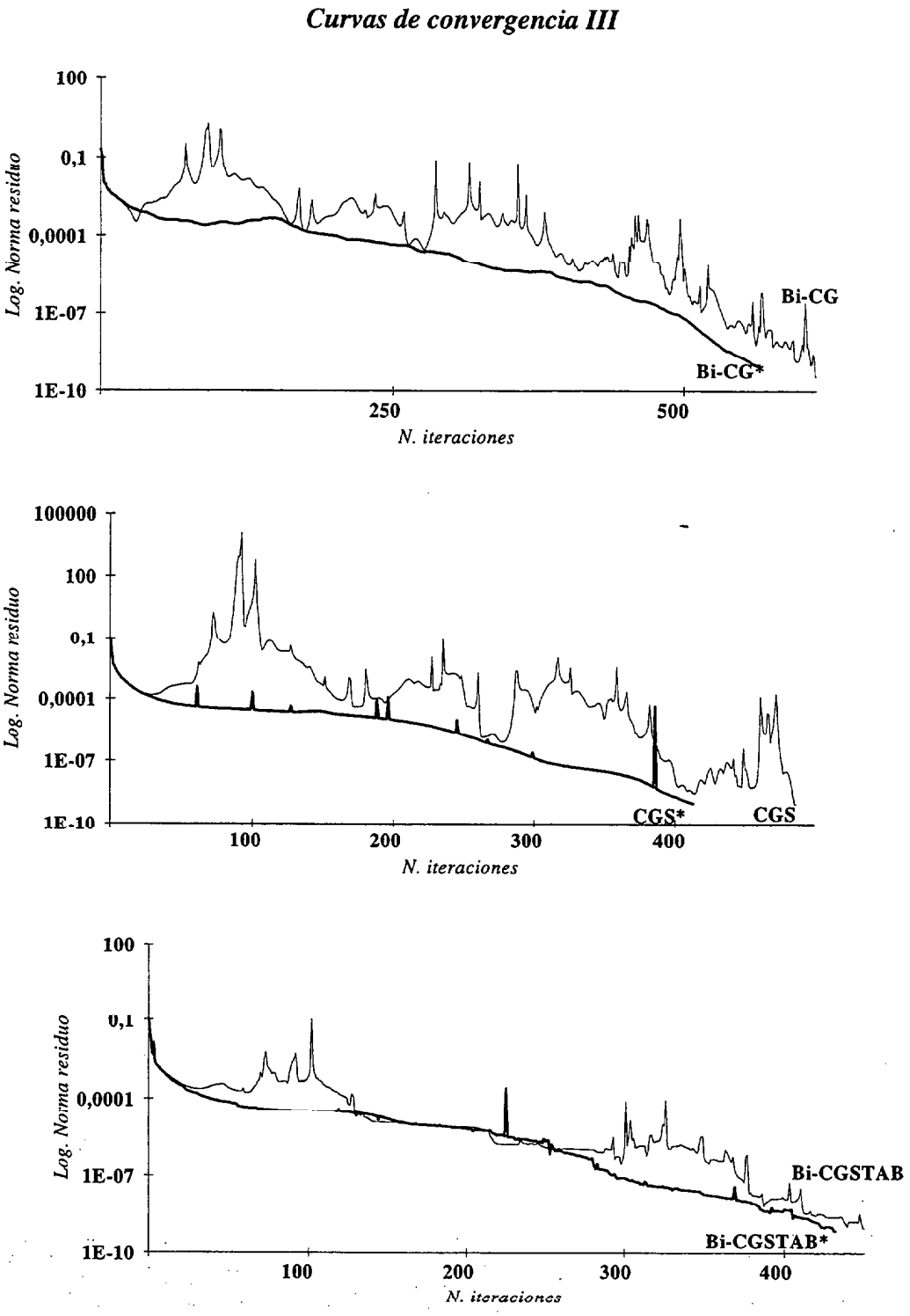


Figura 9: Problema 1 (8434 ecuaciones). Norma del residuo para el precondicionador Diagonal.

- Variación de la tolerancia

Aunque ya se ha comentado que en ausencia de "restarted", las curvas de convergencia para distintas tolerancias coinciden, se ha resuelto este sistema con una tolerancia de 10^{-6}, diferente a la exigida para los gráficos anteriores, a efectos de descartar la existencia de la reinicialización para ese valor de la misma y comprobar que en ese caso los datos obtenidos se ajustan a las mismas curvas de convergencia.

Presentamos los números de iteraciones correspondientes para los distintos métodos y precondicionadores en la tabla siguiente:

Tabla 1. Número de iteraciones para tolerancia de 10^{-6}

	Diagonal	*ILU(0)*	*SSOR*
*Bi-CG**	438	184	252
*CGS**	262	119	159
CGS	277	170	177
*Bi-CGSTAB**	261	108	132
Bi-CGSTAB	253	93	115

Para esta tolerancia exigida no ha existido reinicialización en ningún método con cualquier precondicionador, por lo que si representásemos las respectivas curvas de convergencia, se superpondrían con las ya expuestas, correspondientes al valor 10^{-10}.

6.2 - PROBLEMA 2

En nuestro segundo experimento, aplicamos FEM a un problema de transferencia de calor en Ω, *de 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* Γ_1 *y* Γ_4 *y* $u=u_2$ *en* Γ_2, *con un flujo en* Γ_3 , $-\lambda\frac{\partial u}{\partial x}=H(u-u_\infty)$. *Siendo,*

λ *: conductividad térmica,* $\left[\frac{kj}{h.m.^\circ C}\right]$.

f *: fuentes volumétricas externas,* $\left[\frac{kj}{hm^3}\right]$.

H *: coeficiente de convección,* $\left[\frac{kj}{hm^2\,^\circ C}\right]$.

Para la aplicación práctica se han tomado en las ecuaciones los valores, $\lambda=10^{-5}$, $f=1$, $H=20$, y como condiciones de temperatura $u_1=0$, $u_2=100$, $u_\infty=1000$.

Las figuras 10 y 11 muestran a modo ilustrativo el mallado y la solución para una etapa intermedia del proceso de refinamiento. El último refinamiento ha dado lugar a un sistema simétrico de 16.412 ecuaciones.

Los diagramas de barras muestran solamente el número de iteraciones y tiempos para aquellos casos en los que el algoritmo correspondiente converge. Paradójicamente, el único método que converge sin precondicionar en un número relativamente no muy elevado de iteraciones es el Bi-CG, pero con un tiempo de ejecución diez veces más alto que los que se obtienen con otros algoritmos precondicionados.

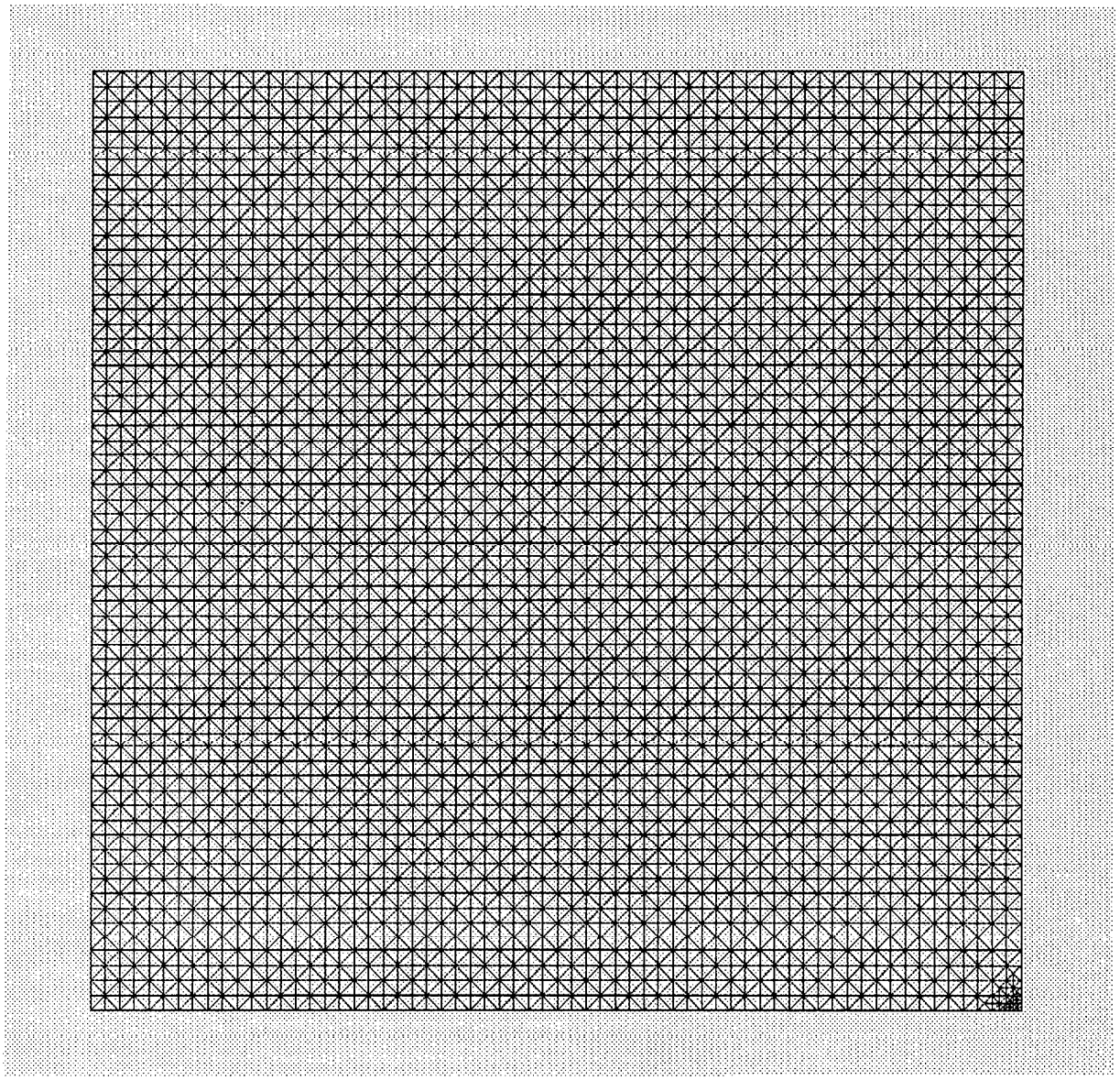


Figura 10: Problema 2. Mallado

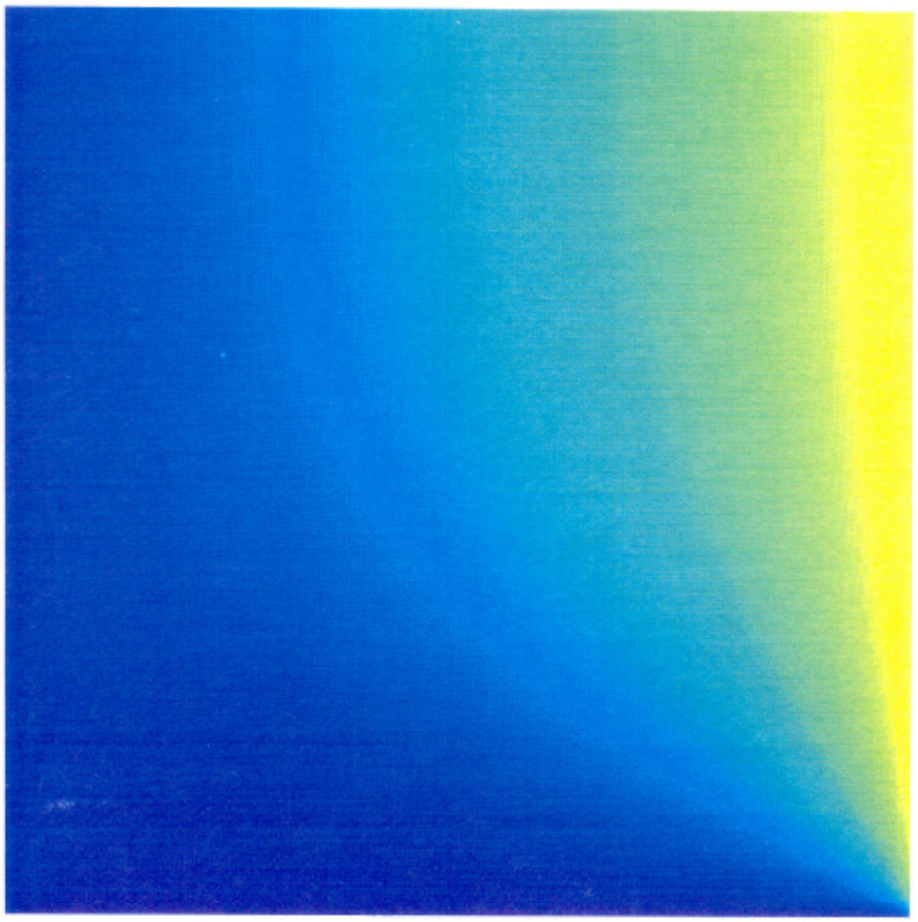


Figura 11: Problema 2. Solución.

Al igual que en el *Problema 1*, el bajo número de iteraciones a que conduce la utilización del ILU(0) frente a otros precondicionadores en la resolución del sistema, no se traduce en mínimos tiempos, debido a que el coste de obtención de la matriz ILU en un sistema de este orden es grande, y, dado que el sistema no tiene muy mal condicionamiento, es el precondicionador Diagonal, de bajo coste, aplicado en este caso en los algoritmos Bi-CGSTAB y Bi-CGSTAB* , el que da lugar a los menores tiempos de ejecución. Son también estos dos métodos los que presentan mejores resultados para los otros precondicionadores.

En este caso, hemos representado las curvas de convergencia para ILU(0), apreciándose también en este problema el comportamiento más suave de Bi-CG* y CGS*, frente a las algoritmos Bi-CG y CGS (fig. 4). Las curvas correspondientes a las dos versiones del Bi-CGSTAB presentan una convergencia suave, muy regular, con características análogas.

Diagramas

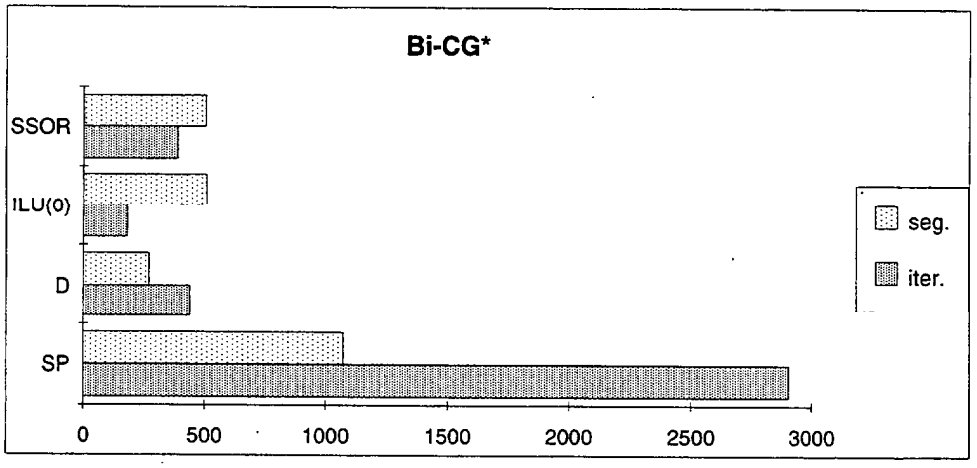


Figura 12: Problema 2 (16412 ecuaciones)

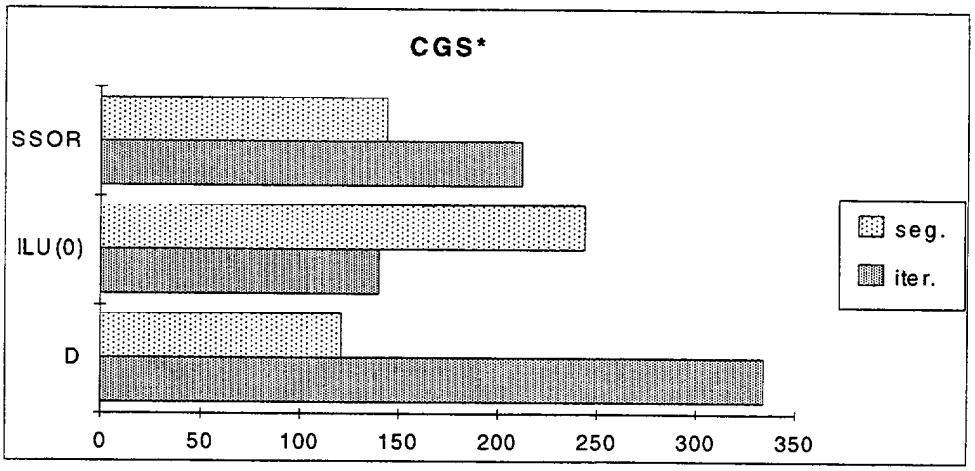


Figura 13: Problema 2 (16412 ecuaciones).

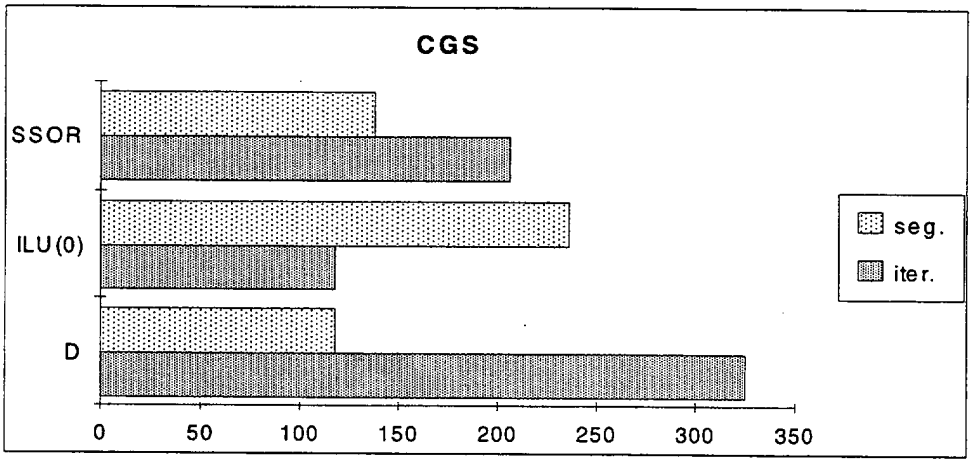


Figura 14: Problema 2 (16412 ecuaciones).

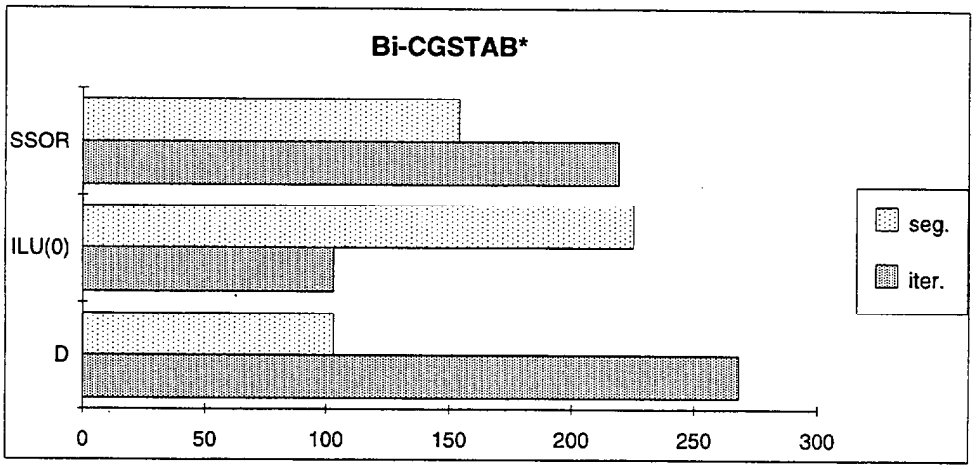


Figura 15: Problema 2 (16412 ecuaciones).

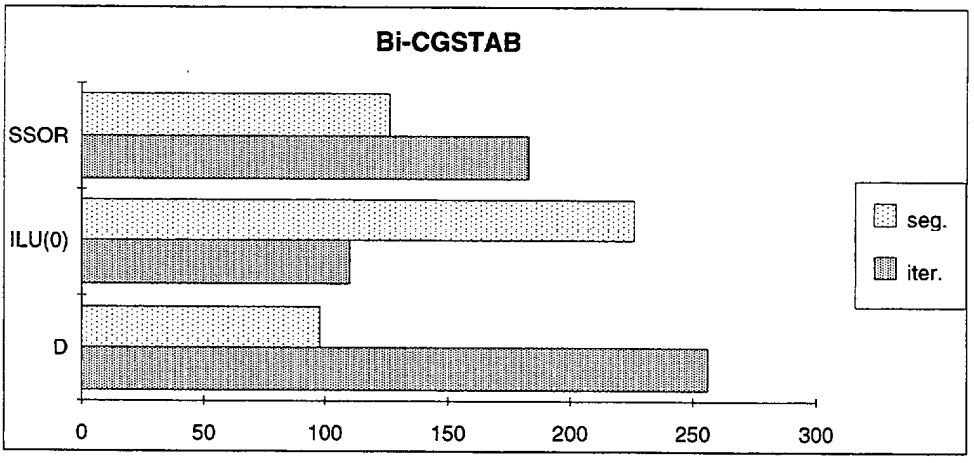


Figura 16: Problema 2 (16412 ecuaciones).

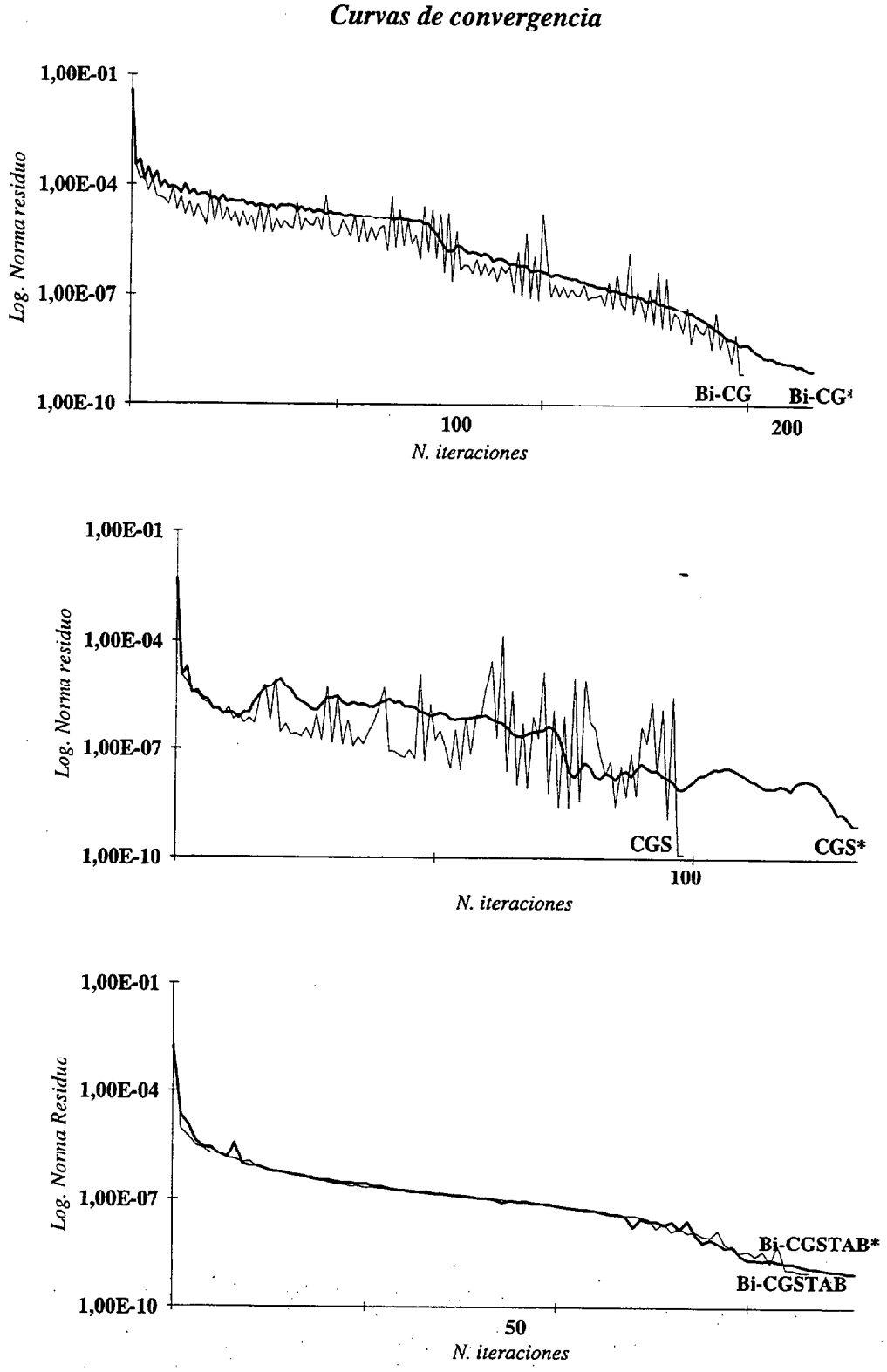


Figura 17: Problema 2 (16412 ecuaciones). Norma del residuo para ILU(0).

6.3 - PROBLEMA 3

Se estudia un problema de convección-difusión estacionario de un fluído 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* Ω, *con condiciones de Dirichlet en* Γ_2 *y* Γ_4, $(u=0 \text{ y } u=1)$, *y condiciones de Neumann,* $\frac{\partial u}{\partial n}=0$ *en el resto de la frontera . Siendo,*

$K=\frac{\lambda}{c\rho}$: *coeficiente de difusión,* $\left[\frac{m^2}{h}\right]$.

v_1, v_2: *componentes del campo de velocidades, de expresiones respectivas,*

$v_1=C\left(y-\frac{1}{2}\right)\left(x-x^2\right)$, $v_2=C\left(\frac{1}{2}-x\right)\left(y-y^2\right)$.

Se ha tomado $K=1$ y para el coeficiente C de velocidad un valor de 10^5. Con distintos mallados se obtienen, respectivamente, sistemas no simétricos de 255, 1023, 4095 y 7520 ecuaciones. Las figuras 18 y 19 representan el mallado y solución correspondiente al sistema de 4095 ecuaciones.

La resolución de los correspondientes sistemas de ecuaciones, generados por estas diferentes mallas, sin técnicas de precondicionamiento, da lugar en muchos casos a convergencias muy lentas que nos hacen descartar el método utilizado. En bastantes ocasiones, asimismo, los precondicionadores D y SSOR no conducen a mejoras sustanciales en esta convergencia.

Para este problema, se han representado también los resultados obtenidos utilizando renumeración previa con los algoritmos del Mínimo Vecino y el Inverso de Cuthill-Mckee en la implementación del precondicionador ILU(0).

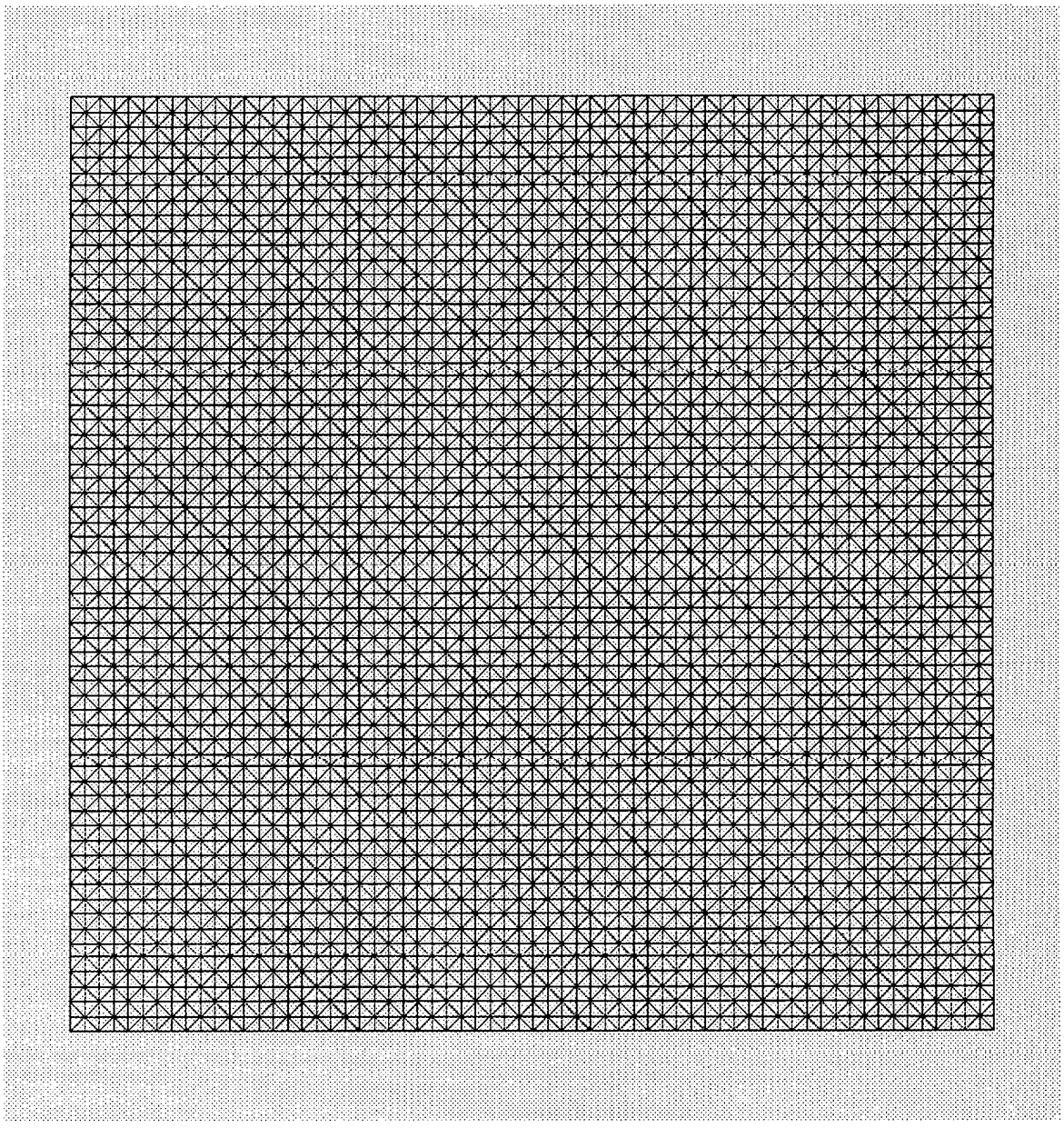


Figura 18: Problema 3 (4095 ecuaciones). Mallado.

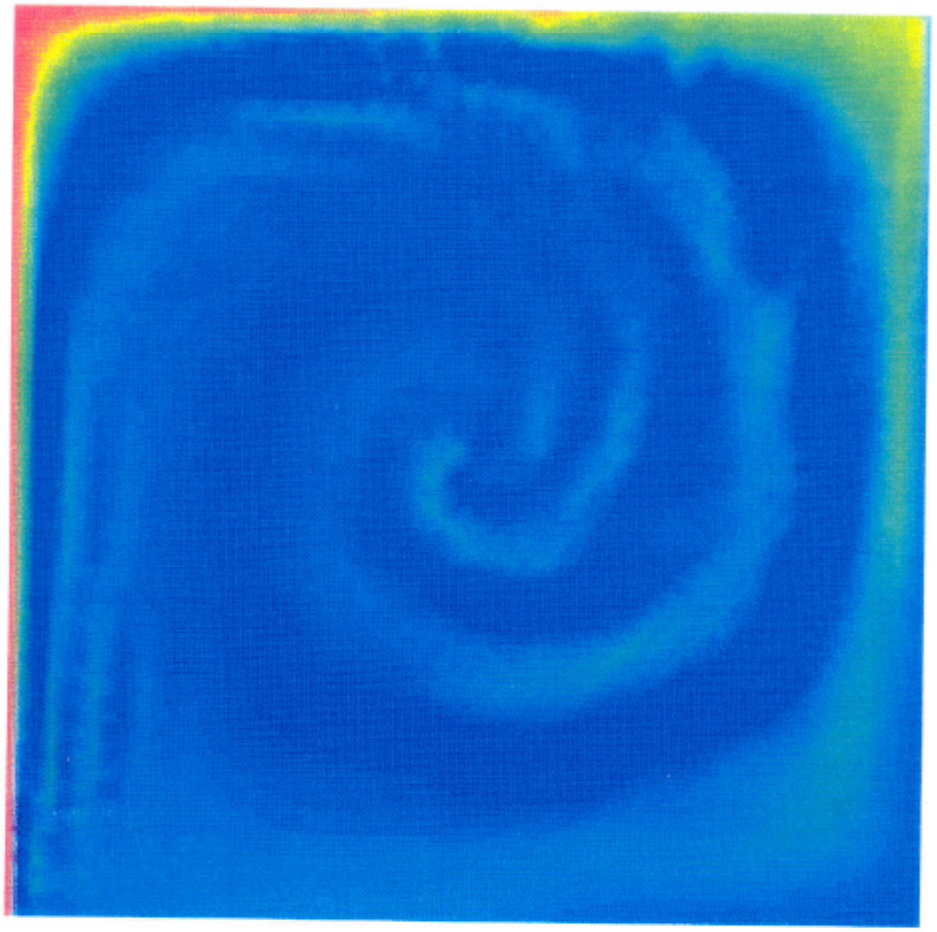


Figura 19: Problema 3 (4095 ecuaciones). Solución.

Diagramas para el sistema de 255 ecuaciones

Ninguno de los algoritmos converge cuando se aplica al sistema sin precondicionar. Asimismo, la utilización de los precondicionadores Diagonal ó SSOR, tampoco proporcionan solución en un número prudencial de iteraciones.

Con el ILU(0), se alcanza la tolerancia deseada en un número de iteraciones relativamente bajo, pero aproximado al orden de la matriz del sistema. Sin embargo, las técnicas de renumeración, sobre todo el algoritmo ICM, reducen considerablemente estas iteraciones en todos los métodos. Y, lo que es más importante, el tiempo computacional en la aplicación de estas técnicas, incluyendo el de la renumeración, es inferior al tiempo de ejecución utilizando el ILU(0) en la solución del sistema sin reordenar.

En cuanto a la efectividad de los métodos, las dos versiones del CGS y algoritmo del Bi-CGSTAB*, son las que proporcionan mejores resultados en general.

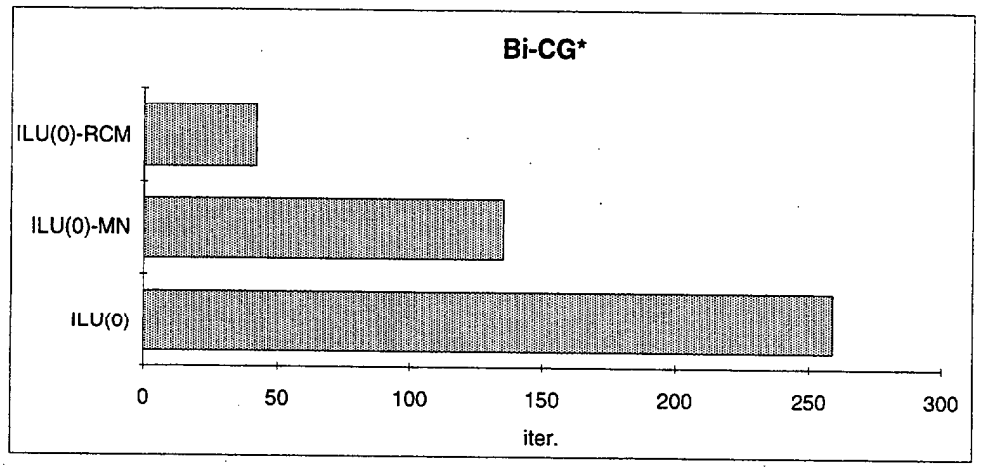


Figura 20: Problema 3 (255 ecuaciones).

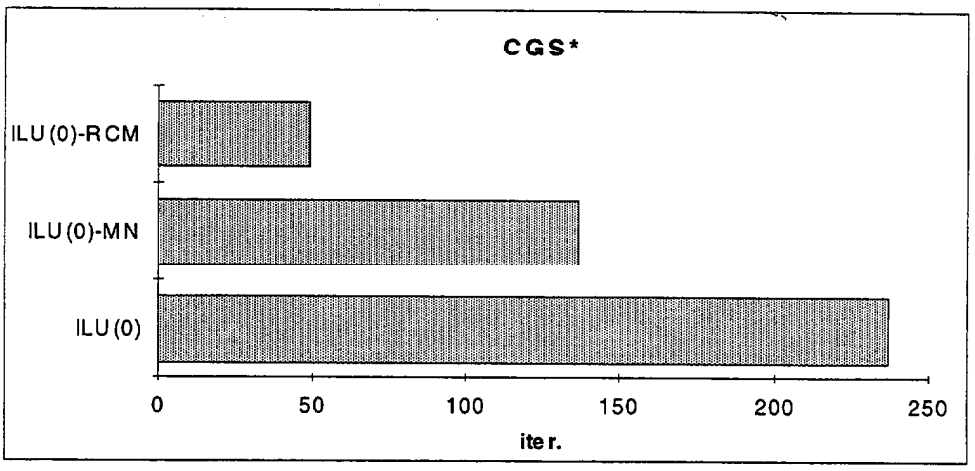


Figura 21: Problema 3 (255 ecuaciones).

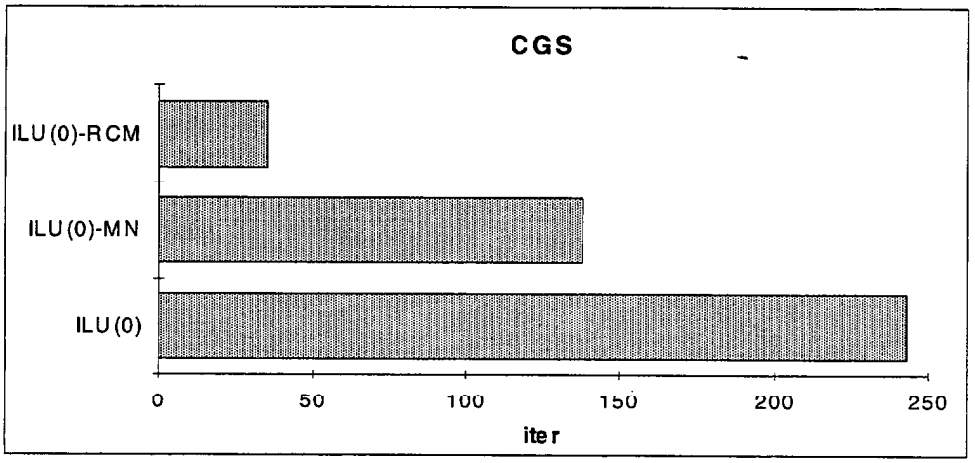


Figura 22: Problema 3 (255 ecuaciones).

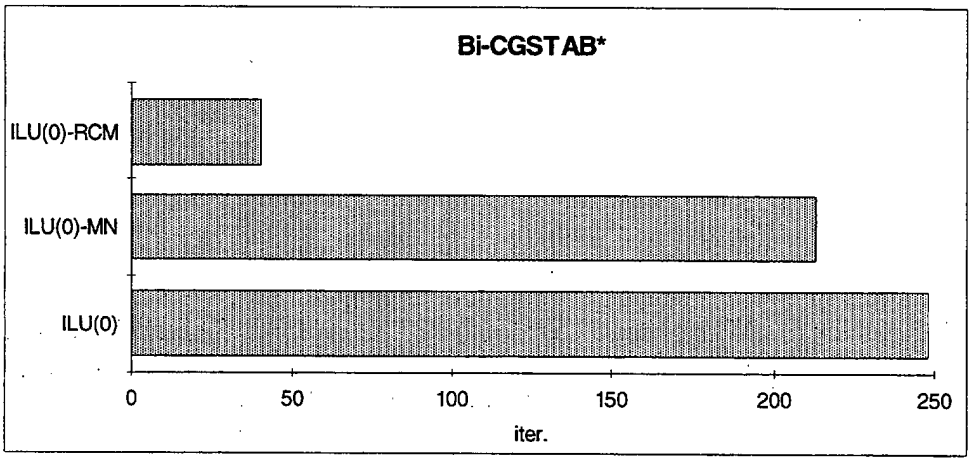


Figura 23: Problema 3 (255 ecuaciones).

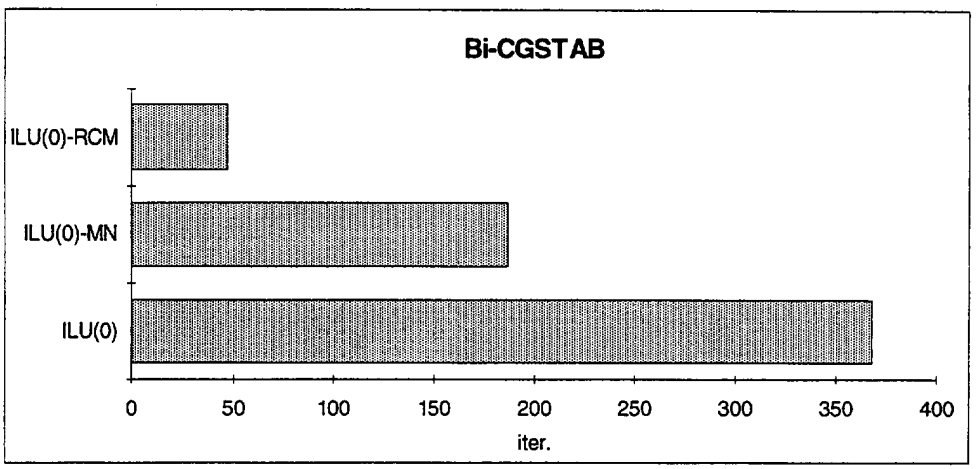


Figura 24: Problema 3 (255 ecuaciones).

Diagramas para el sistema de 1023 ecuaciones

En este sistema, mejor condicionado que el anterior, las dos versiones del Bi-CGSTAB convergen sin precondicionamiento en un número de iteraciones relativamente bajo.

El precondicionador SSOR se muestra tan efectivo como el ILU(0), incluso con menores tiempos de ejecución, pero utilizando las técnicas de reordenación es, con diferencia, el ILU(0) el que da lugar a menor número de iteraciones y tiempo en todos los métodos.

Los algoritmos de los métodos Bi-CGSTAB y Bi-CGSTAB* son los más ventajosos en todos los casos con los distintos precondicionadores.

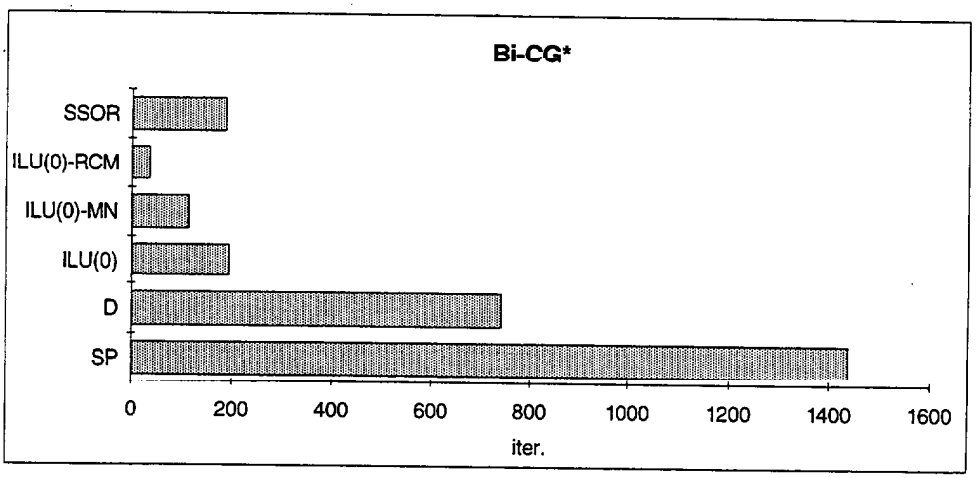


Figura 25: Problema 3 (1023 ecuaciones).

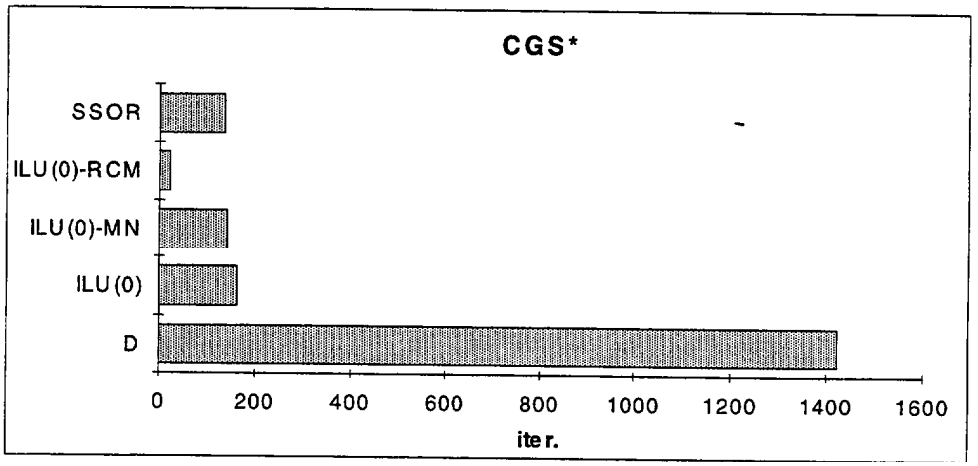


Figura 26: Problema 3 (1023 ecuaciones).

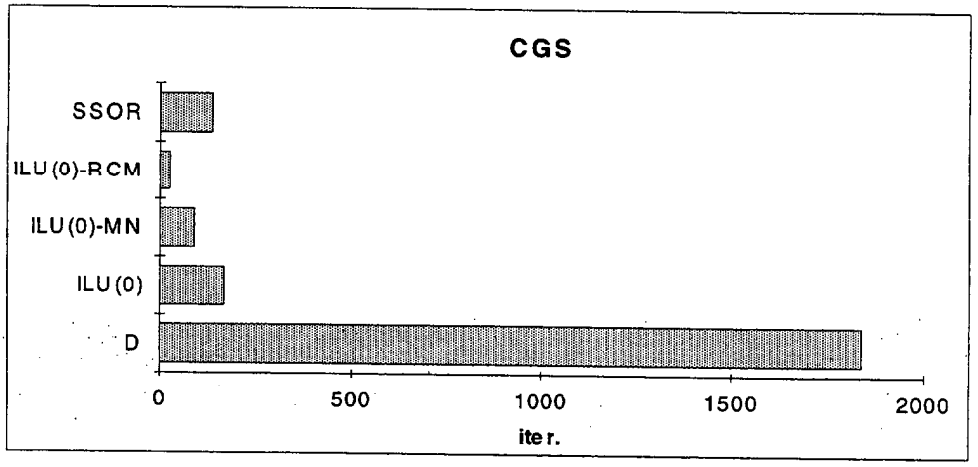


Figura 27: Problema 3 (1023 ecuaciones).

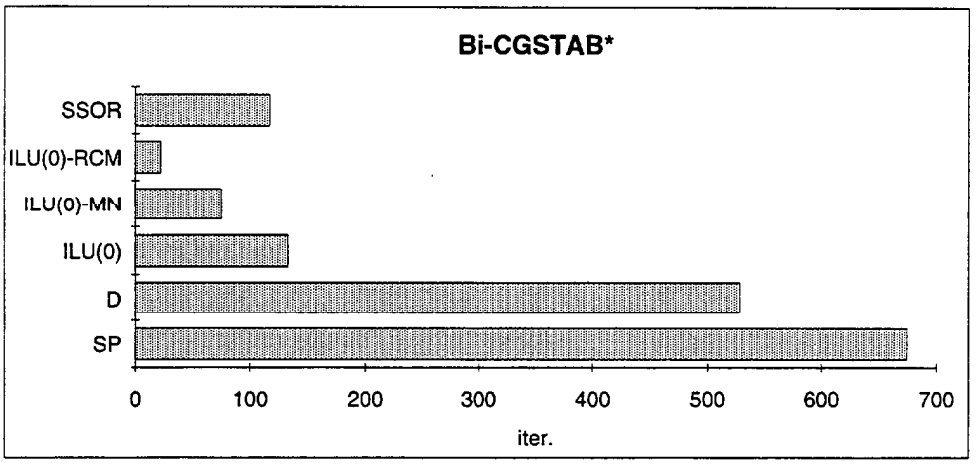


Figura 28: Problema 3 (1023 ecuaciones).

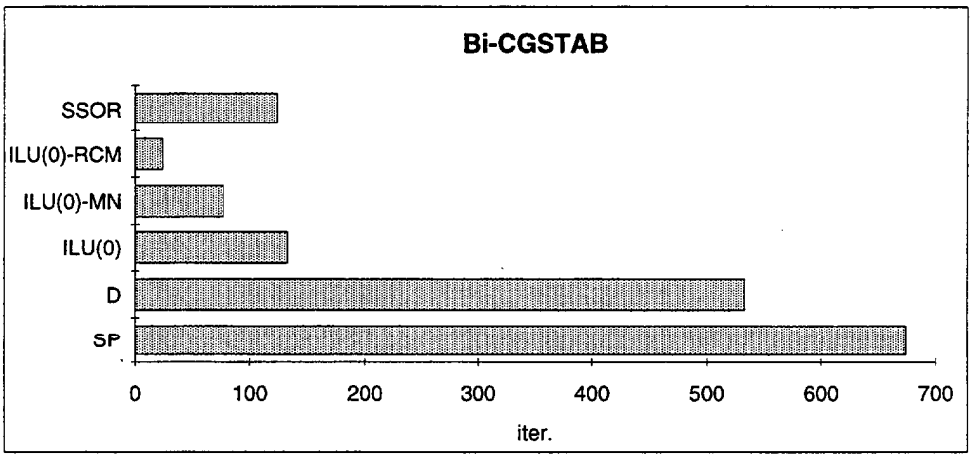


Figura 29: Problema 3 (1023 ecuaciones).

Diagramas para el sistema de 4095 ecuaciones

Sólo los algoritmos de las dos versiones del Bi-CGSTAB convergen sin precondicionar ó con el precondicionador Diagonal.

En este sistema de ecuaciones, al igual que en el anterior, el precondicionador SSOR se muestra más robusto y de menor coste que el ILU(0), características estas que se invierten con la reordenación, ya que tanto con el algoritmo

del Mínimo Vecino, como con el algoritmo ICM, se reducen tiempos e iteraciones en todos los métodos aplicando este precondicionador.

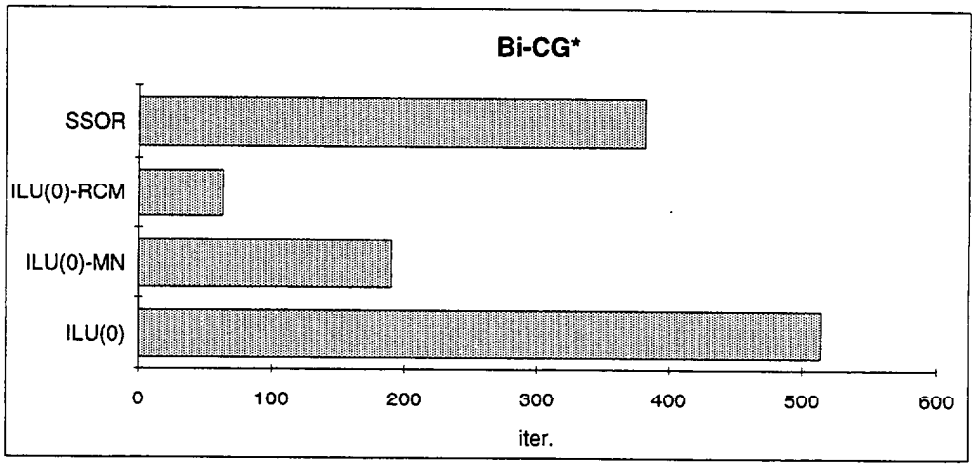


Figura 30: Problema 3 (4095 ecuaciones).

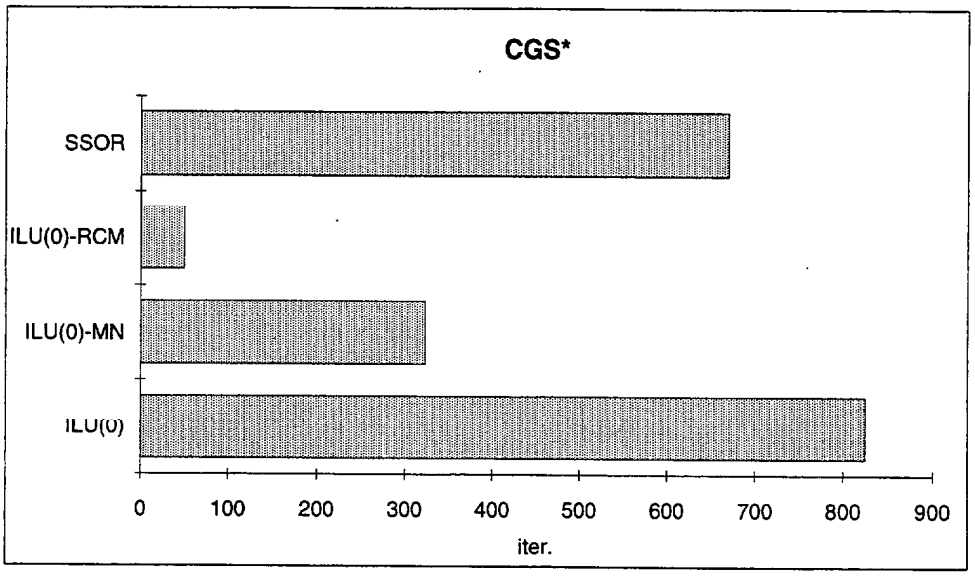


Figura 31: Problema 3 (4095 ecuaciones).

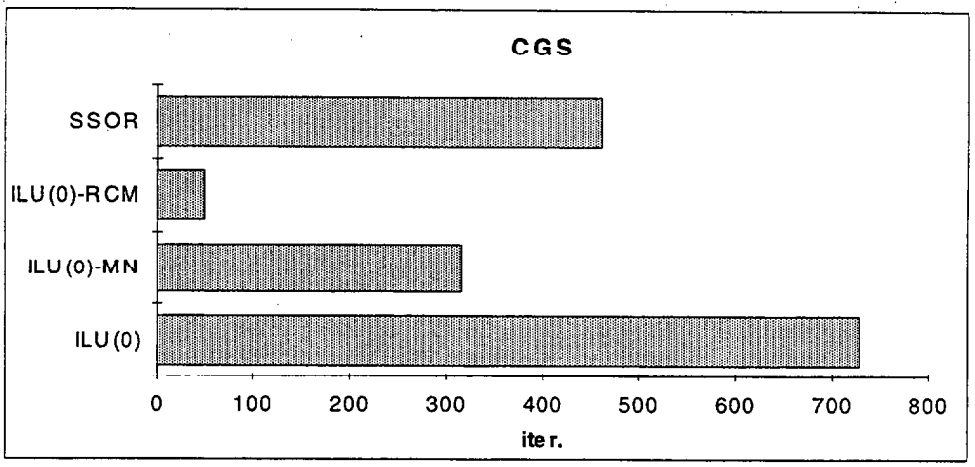


Figura 32: Problema 3 (4095 ecuaciones).

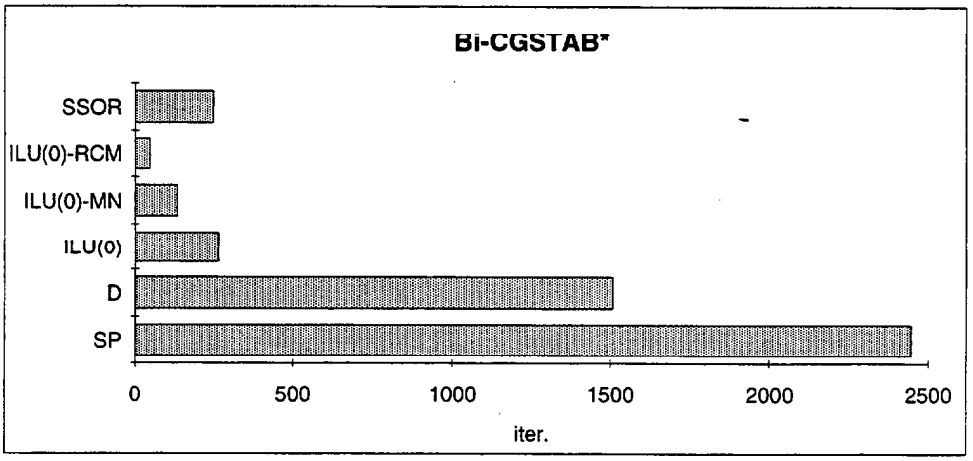


Figura 33: Problema 3 (4095 ecuaciones).

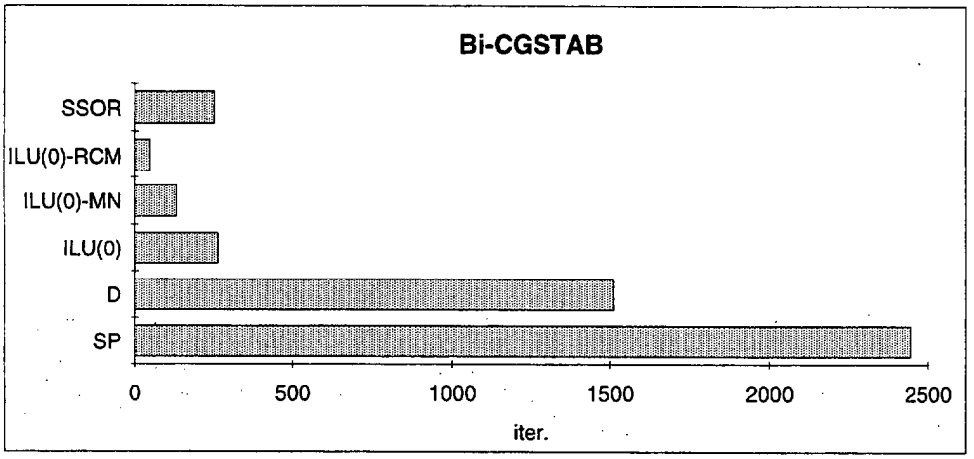


Figura 34: Problema 3 (4095 ecuaciones).

Diagramas para el sistema de 7520 ecuaciones

Los algoritmos Bi-CGSTAB vuelven a converger sin precondicionar, pero, de forma atípica, no lo hacen con el precondicionador Diagonal.

La aplicación de ILU(0), incluso sin reordenación, es mucho más efectiva que la aplicación del SSOR en este caso, efecto que se incrementa con la implementación de esta técnica. Es de destacar, que el algoritmo del Bi-CGSTAB no converge con el precondicionador ILU(0) sin la reordenación previa, mientras que el Bi-CGSTAB* lo hace en número de iteraciones y tiempo casi mitad de los correspondientes a los algoritmos CGS.

En la figura 40, se representan las curvas de convergencia para los métodos Bi-CG*, CGS* y Bi-CGSTAB*, precondicionados con ILU(0), observándose la ventaja que ofrece en este caso, este último algoritmo frente a la aplicación de los otros.

Para comprobar los efectos de la renumeración en el suavizado de las curvas de convergencia, se presentan, en la figura 42, estas curvas correspondientes a los algoritmos CGS* y Bi-CGSTAB* obtenidas con la aplicación de ILU(0).

Por último, en la figura 41, se representan las curvas de convergencia que se obtienen para el Bi-CGSTAB*, precondicionado con ILU(0) y con reordenación debida al algoritmo ICM, tomando como efecto de precondicionamiento distintas elecciones para el vector incialización. Algunos de los vectores adoptados, casos (a), (b) y (d), se han determinado utilizando los resultados obtenidos en las primeras iteraciones para otros métodos . En el caso (c), se considera un vector inicial aleatorio, calculado multiplicando las componentes del vector residuo inicial del sistema original por unas constantes aleatorias comprendidas entre 0 y 1.

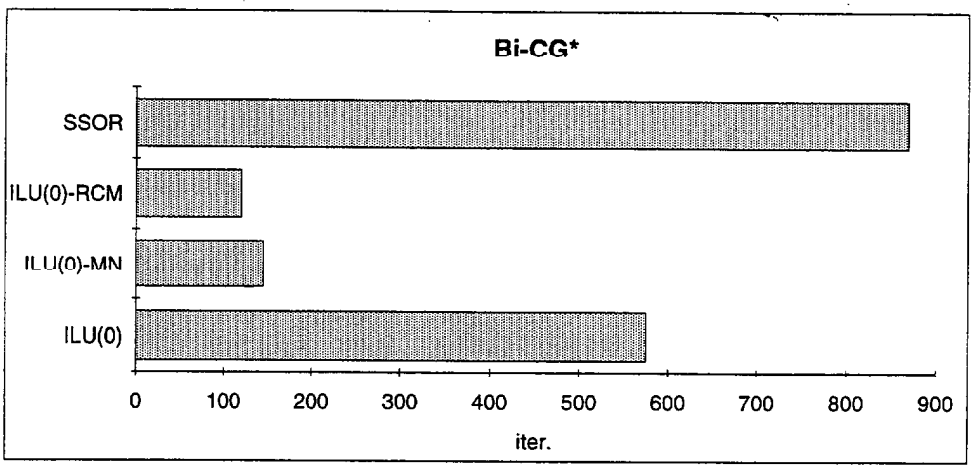


Figura 35: Problema 3 (7520 ecuaciones).

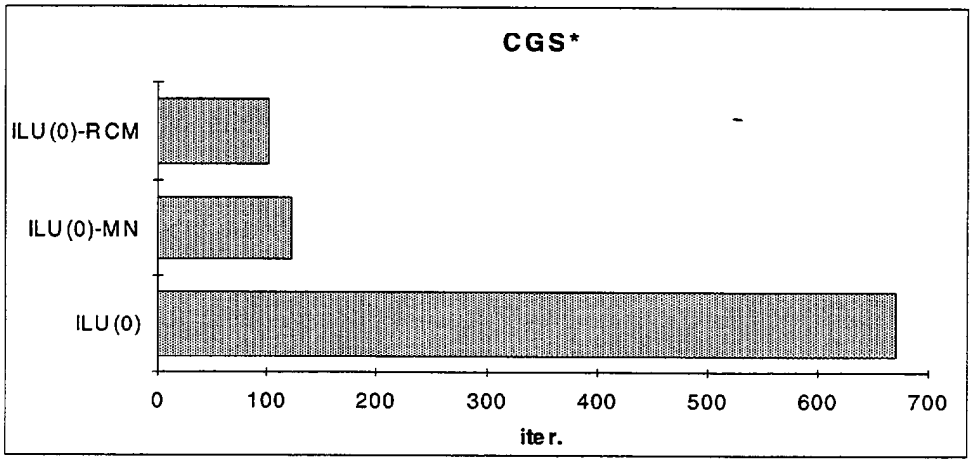


Figura 36: Problema 3 (7520 ecuaciones).

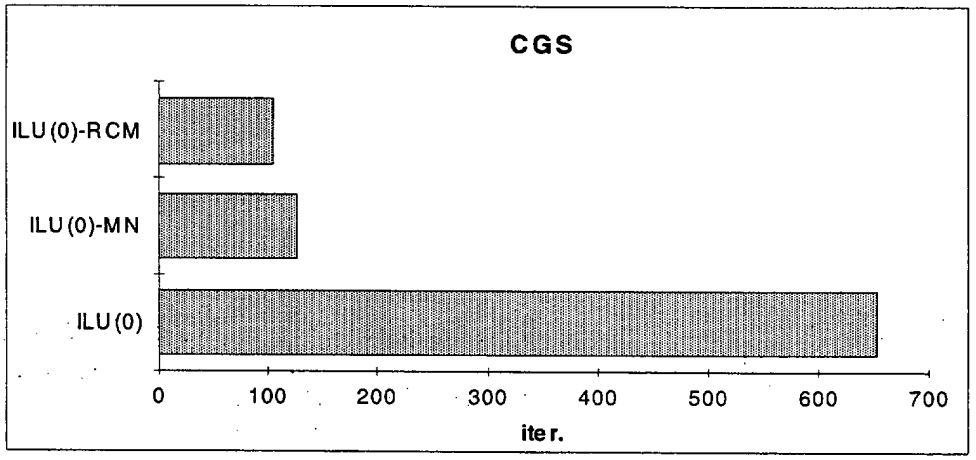


Figura 37: Problema 3 (7520 ecuaciones).

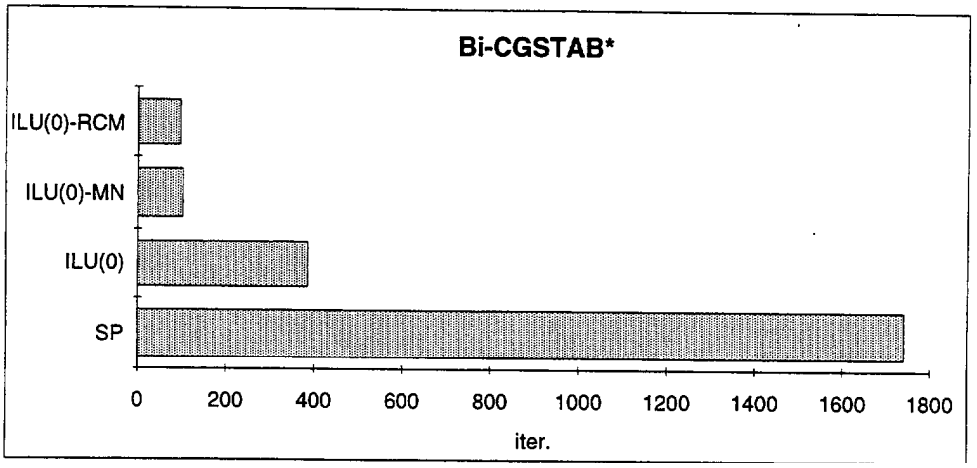


Figura 38: Problema 3 (7520 ecuaciones).

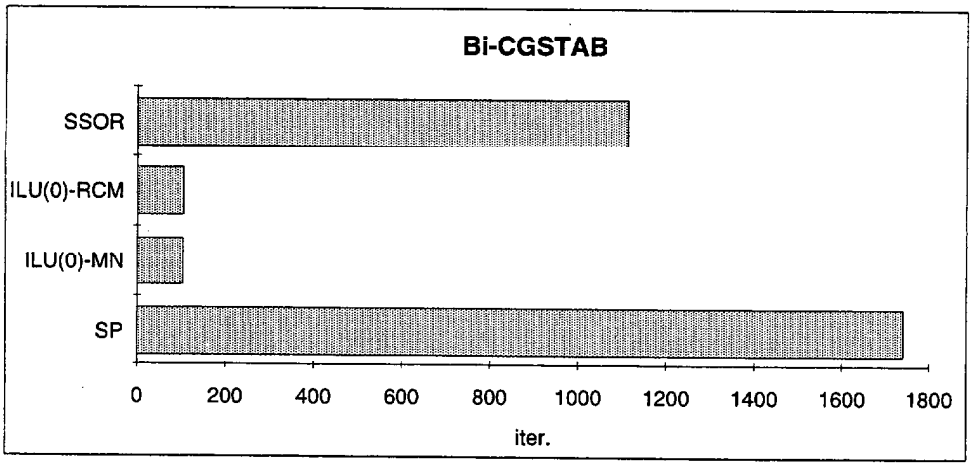


Figura 39: Problema 3 (7520 ecuaciones).

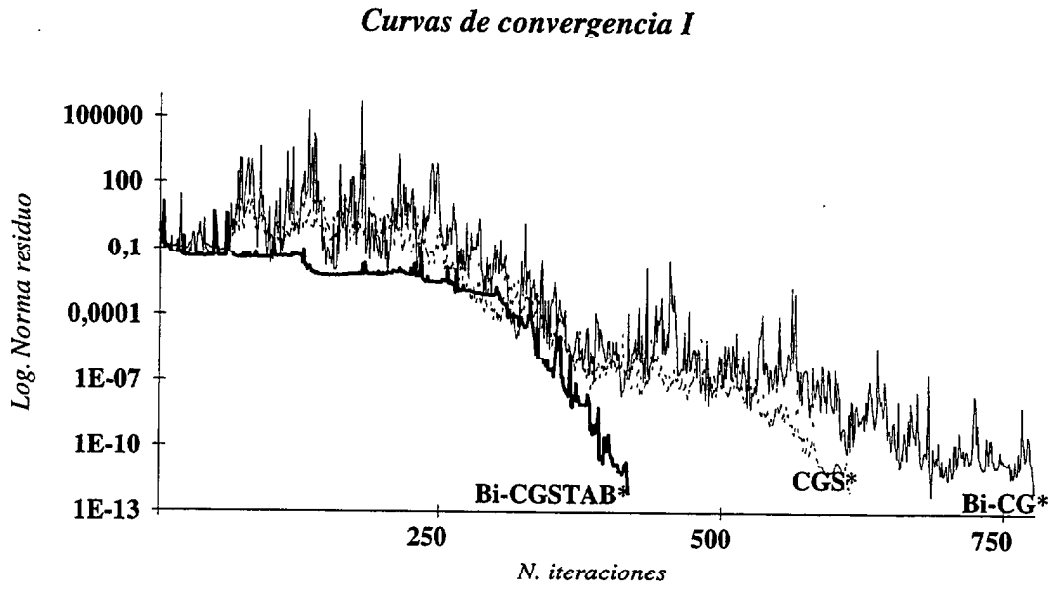


Figura 40: Problema 3 (7520 ecuaciones). Norma del residuo para ILU(0).

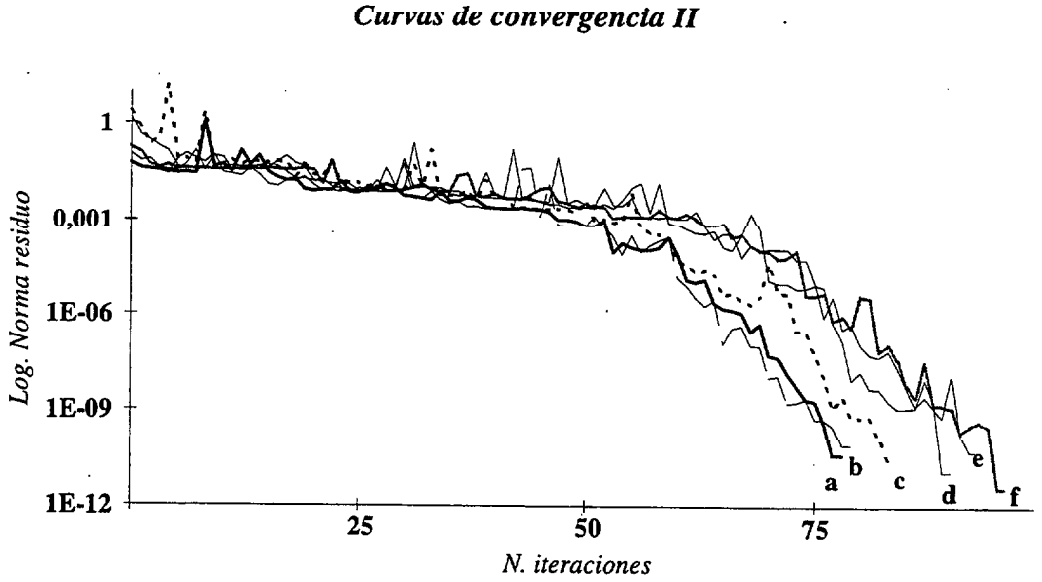


Figura 41: Problema 3 (7520 ecuaciones). Efecto de la variación del vector $\bar{r}_0$.

(a): x_0 =Bi-CGSTAB(5), $\bar{r}_0 = r_0$; **(b):** x_0 =QMRCGSTAB(5), $\bar{r}_0 = r_0$; **(c):** $\bar{r}_0 = D_r r_0$ (siendo D_r una matriz diagonal aleatoria); **(d):** $\bar{r}_0$= Bi-CGSTAB(5); **(e):** $x_0 = M^{-1}b$, $\bar{r}_0 = r_0$; **(f):** $\bar{r}_0 = r_0$.

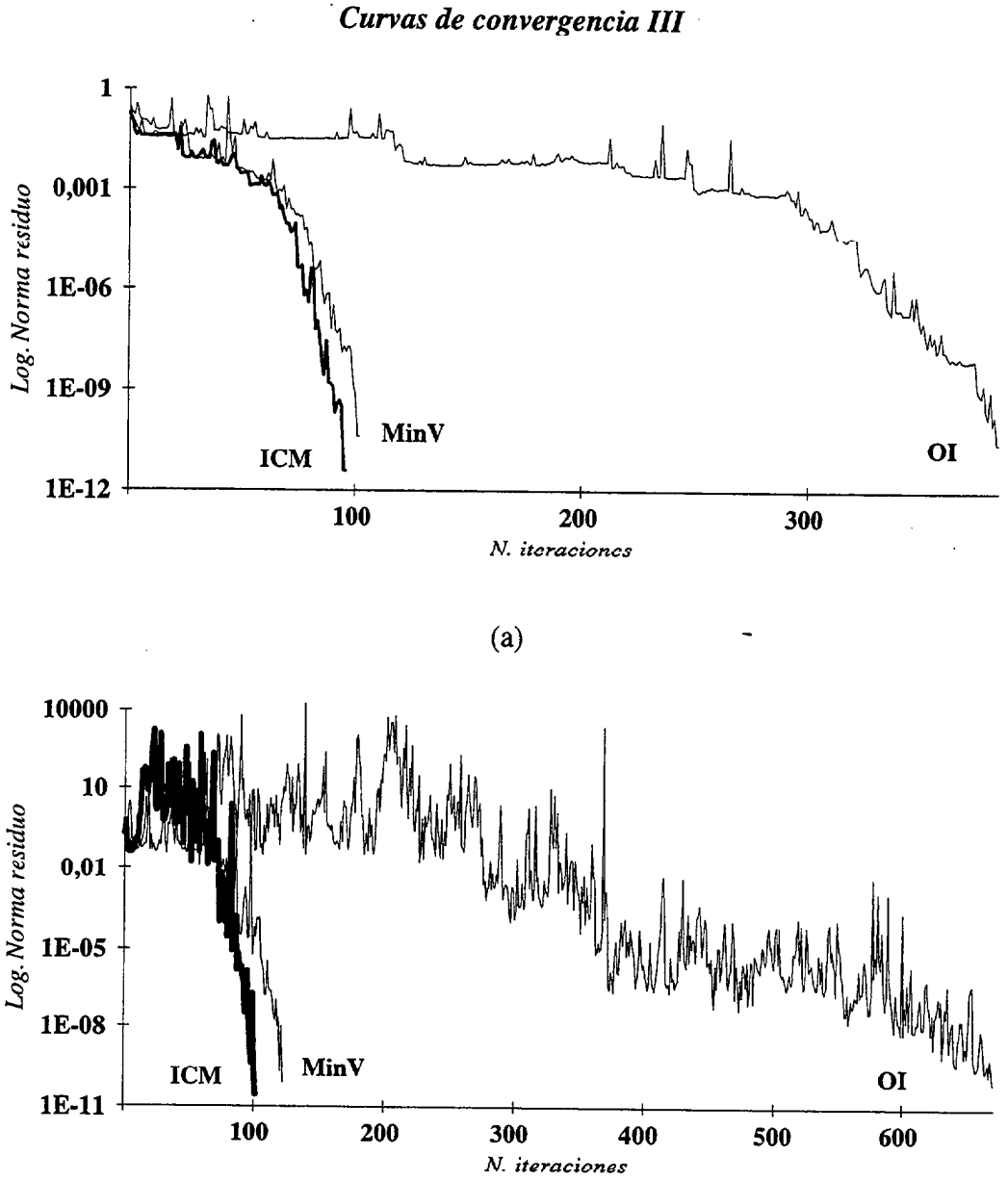


Figura 42: Problema 3 (7520 ecuaciones). Efecto de la renumeración con ICM (Algoritmo inverso de Cuthill-Mckee), MinV (Algoritmo del Mínimo Vecino), OI (Ordenación inicial) en ILU(0), con los métodos, (a): Bi-CGSTAB* y (b): CGS*.

6.4 - PROBLEMA 4

Finalmente, estudiamos un problema de convección-difusión similar al anterior, definido 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 definido por* $v_1 = 10^4\left(y - \frac{1}{2}\right)\left(x - x^2\right)\left(\frac{1}{2} - x\right)$. *Siendo,*

K: coeficiente de difusión, $\left[\frac{m^2}{h}\right]$.

$F = \frac{f}{c\rho}$: *coeficiente que depende de las fuentes volumétricas y de las características del medio,* $\left[\frac{^\circ C}{h}\right]$.

En este caso, hemos tomado para K un valor de 10^{-5} en Ω_2, y de 10^2 en el resto del dominio y para F el valor 10^3 en Ω_3 y de 1 en el resto. Las condiciones de contorno son las mismas que en el ejemplo 3. Los mallados utilizados dan lugar a sistemas no simétricos, respectivamente, de 3.423, 6.585 y 13.190 ecuaciones. Las figuras muestran el mallado y la solución para el caso de 3.423 ecuaciones.

El comportamiento de los sistemas de 3.423 y 6.585 ecuaciones es similar para todos los algoritmos, por lo que comentaremos conjuntamente los diagramas de barras respectivos, correspondientes a cada uno.

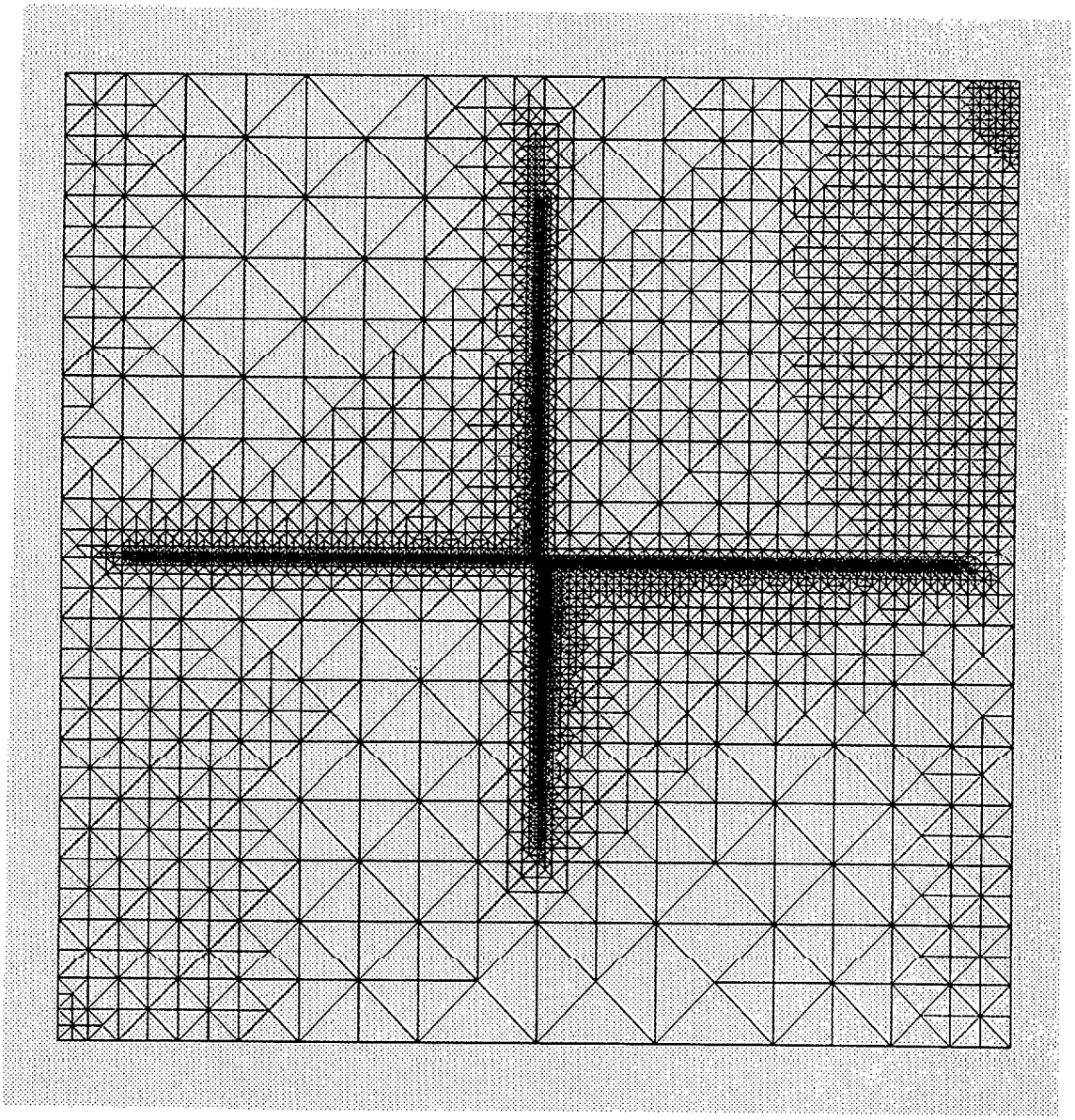


Figura 43: Problema 4 (3423 ecuaciones). Mallado

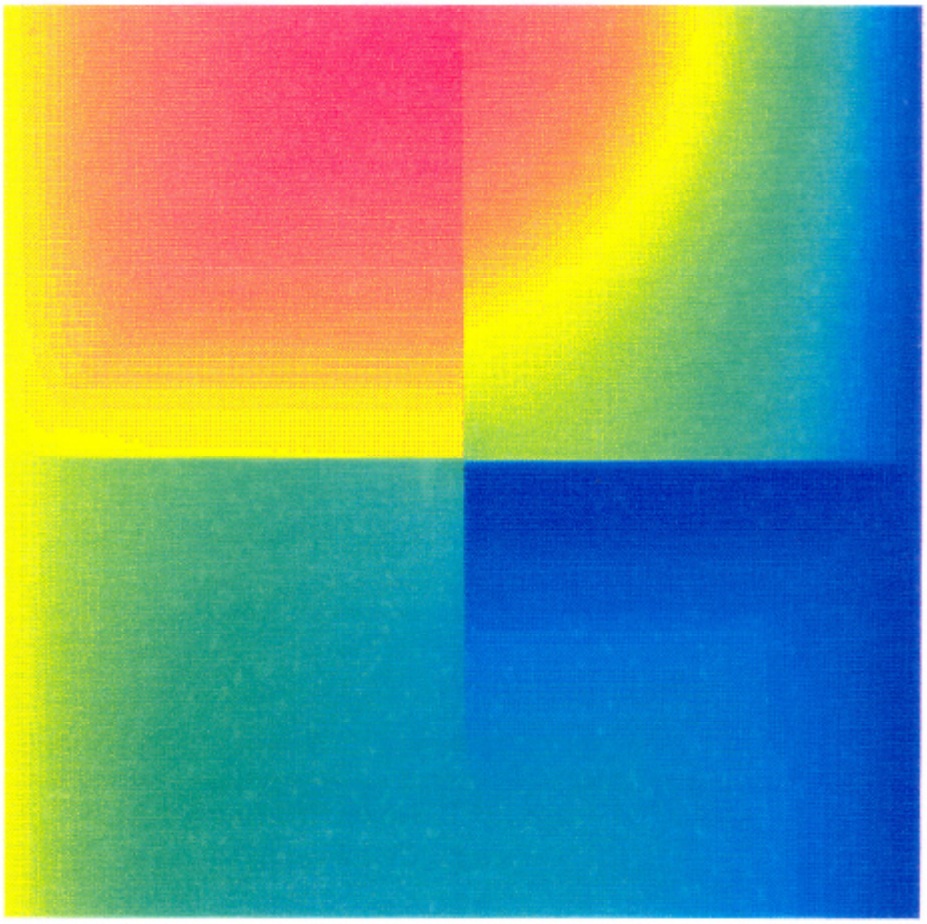


Figura 44: Problema 4 (3423 ecuaciones). Solución.

Diagramas para los sistemas de 3423 y 6585 ecuaciones.

Ningún método converge sin precondicionamiento. Es el precondicionador ILU(0), el que, en general, proporciona mejores resultados, tanto en rapidez como en iteraciones.

Los algoritmos de reordenación reducen de forma aproximada en un 50 % el número de iteraciones necesario para alcanzar la convergencia, sin diferencia significativa, en este caso, entre los resultados de aplicar el algoritmo del Mínimo Vecino ó el algoritmo ICM.

En cuanto a la bondad de los distintos métodos, vuelven a ser los dos algoritmos del Bi-CGSTAB los que destacan por su robustez y tiempos de ejecución.

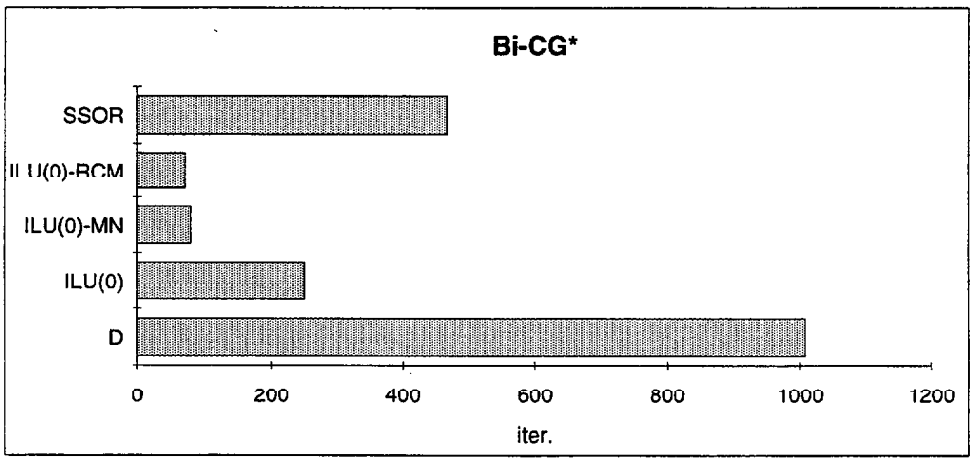


Figura 45: Problema 4 (3423 ecuaciones).

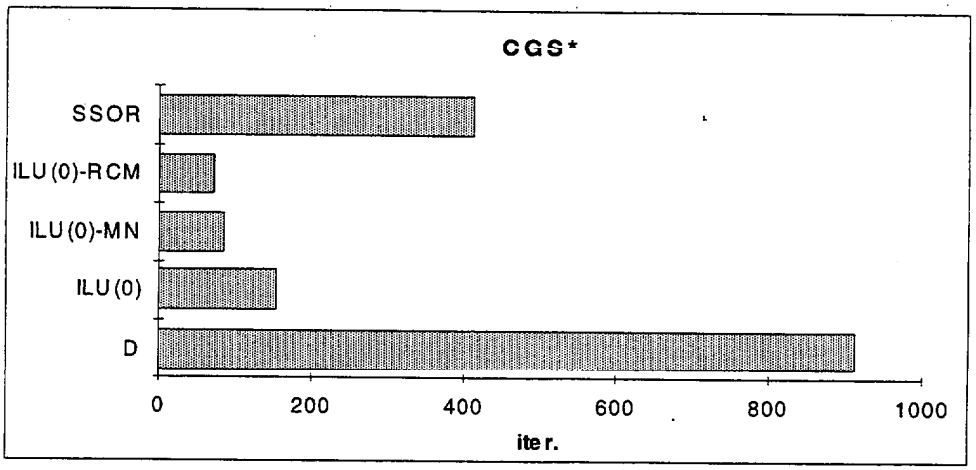


Figura 46: Problema 4 (3423 ecuaciones)

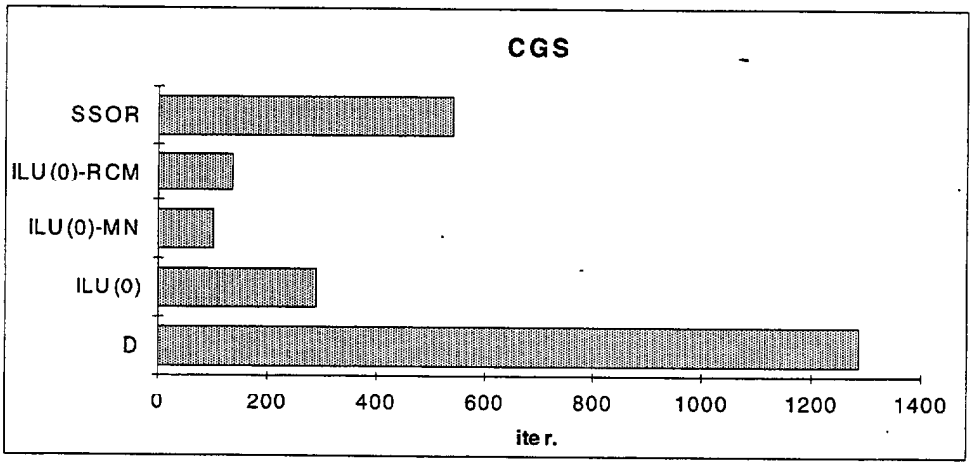


Figura 47: Problema 4 (3423 ecuaciones).

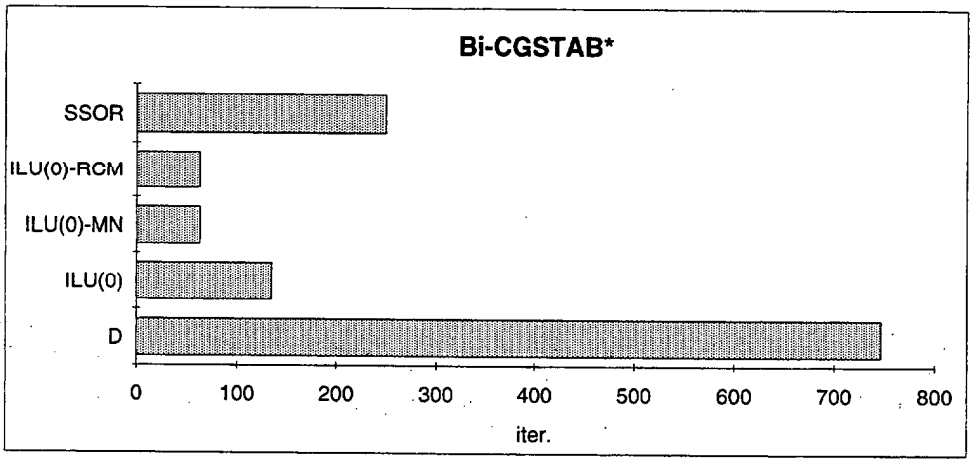


Figura 48: Problema 4 (3423 ecuaciones).

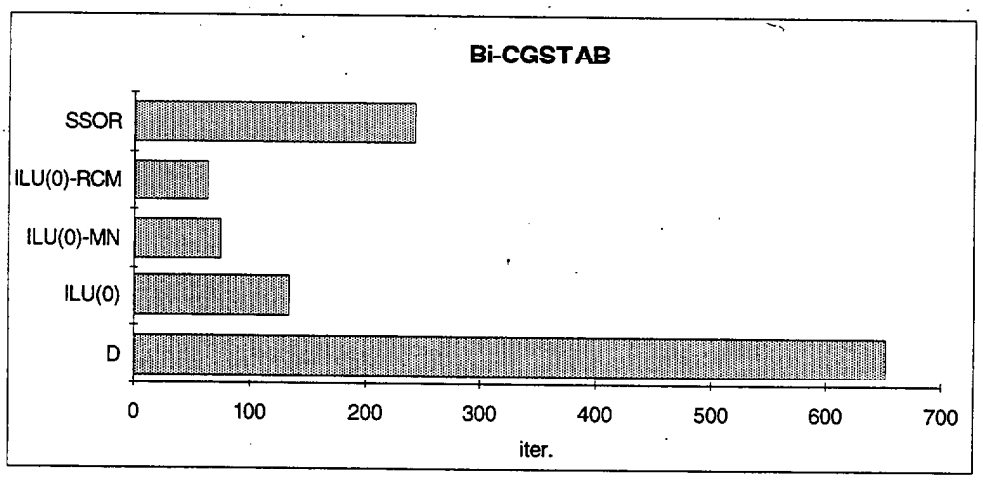


Figura 49: Problema 4 (3423 ecuaciones).

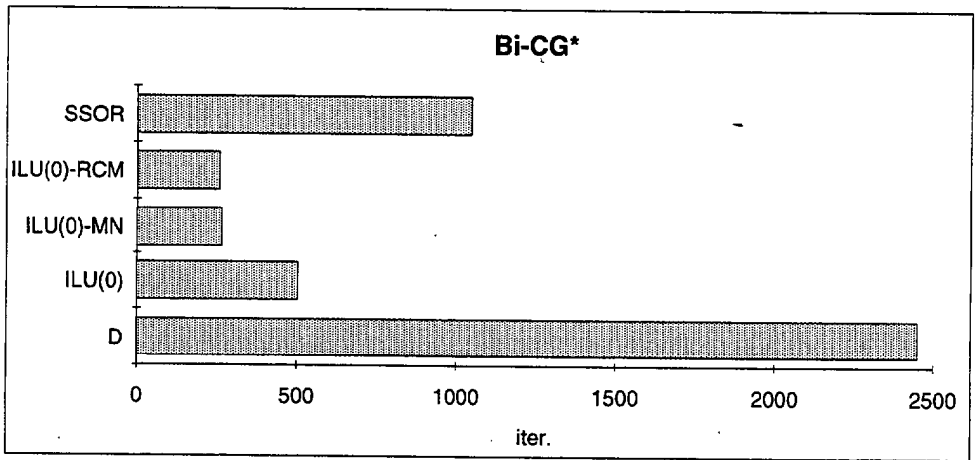


Figura 50: Problema 4 (6585 ecuaciones).

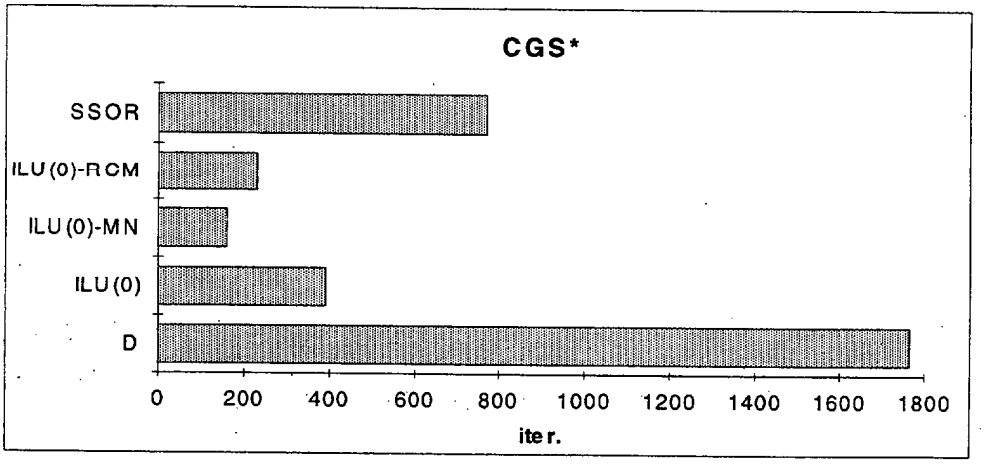


Figura 51: Problema 4 (6585 ecuaciones).

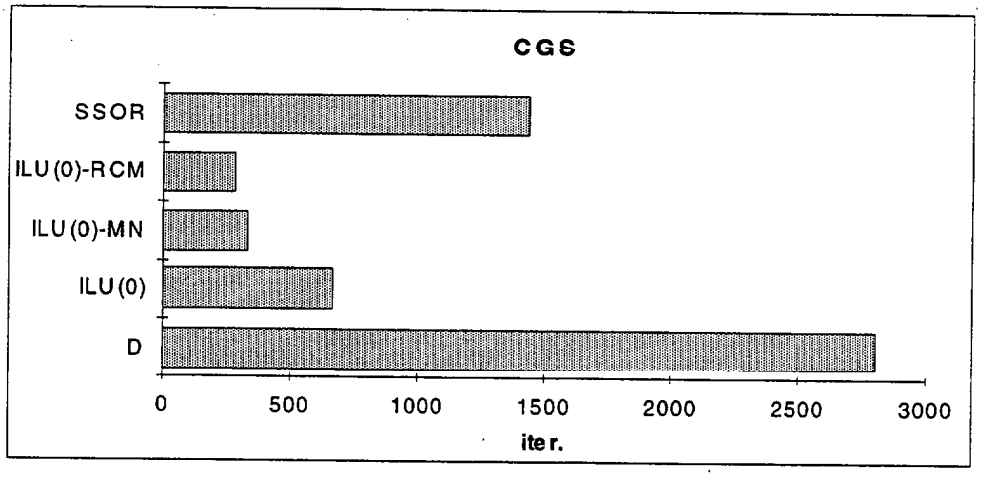


Figura 52: Problema 4 (6585 ecuaciones).

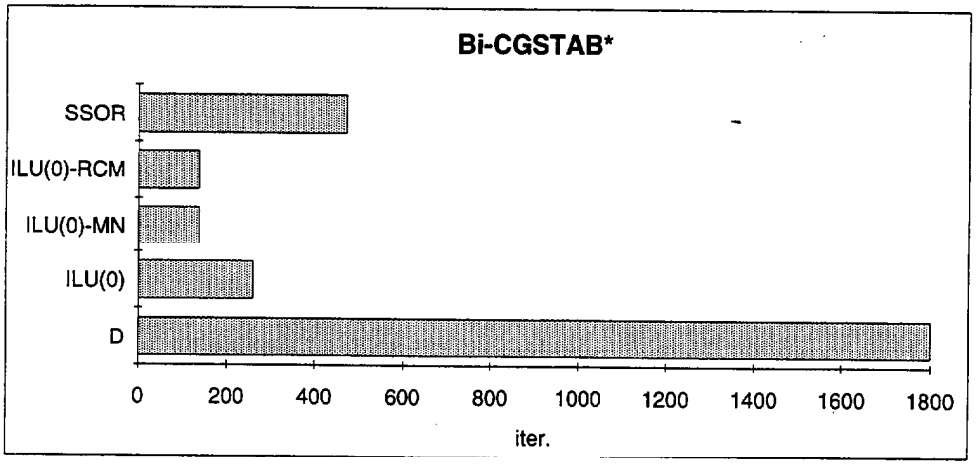


Figura 53: Problema 4 (6585 ecuaciones).

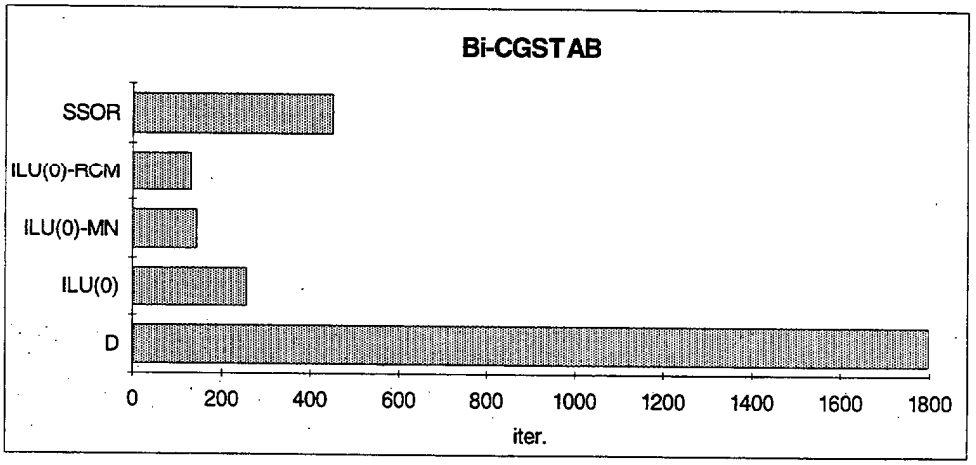


Figura 54: Problema 4 (6585 ecuaciones).

Diagramas para el sistema de 13190 ecuaciones.

Los algoritmos convergen solamente cuando se aplican los precondicionadores ILU(0) y SSOR, y es, en el caso de utilización del ILU(0), una vez más, cuando se alcanza la tolerancia exigida en menor número de iteraciones y menor tiempo de ejecución, para todos los métodos.

Asimismo, las dos versiones del Bi-CGSTAB. al igual que anteriores casos, vuelven a ser los algoritmos más eficaces.

En las curvas de convergencia comparamos, resolviendo el sistema precondicionado con ILU(0), los métodos Bi-CG, CGS y Bi-CGSTAB, con las correspondientes variantes Bi-CG*, CGS* y Bi-CGSTAB*, respectivamente. En este caso, es destacable el mejor comportamiento de los algoritmos Bi-CG* y CGS*, mientras las dos versiones del Bi-CGSTAB se comportan de forma similar.

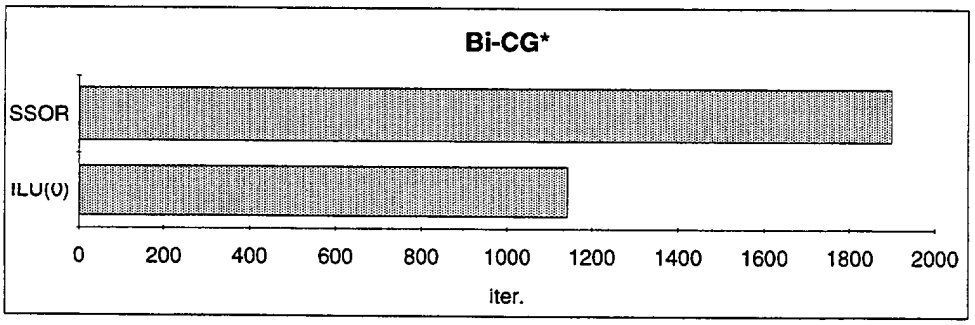


Figura 55: Problema 4 (13190 ecuaciones).

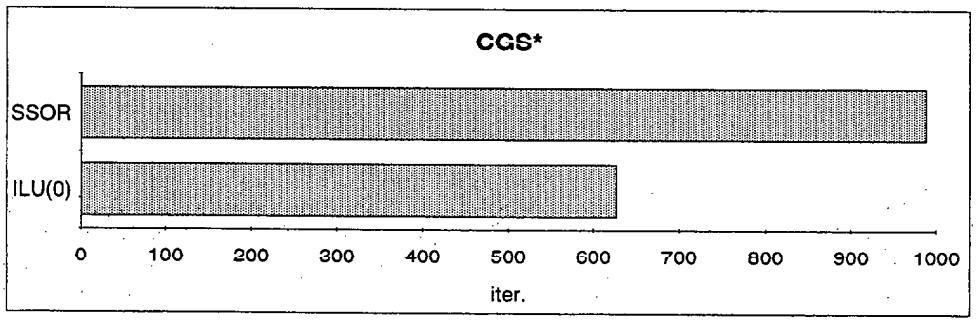


Figura 56: Problema 4 (13190 ecuaciones).

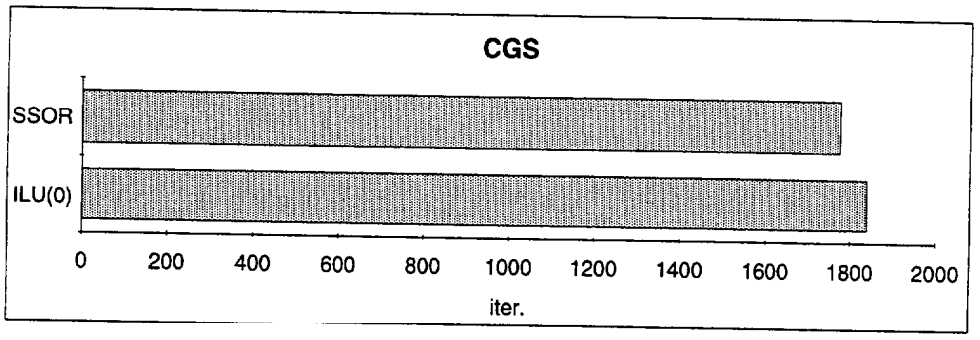


Figura 57: Problema 4 (13190 ecuaciones).

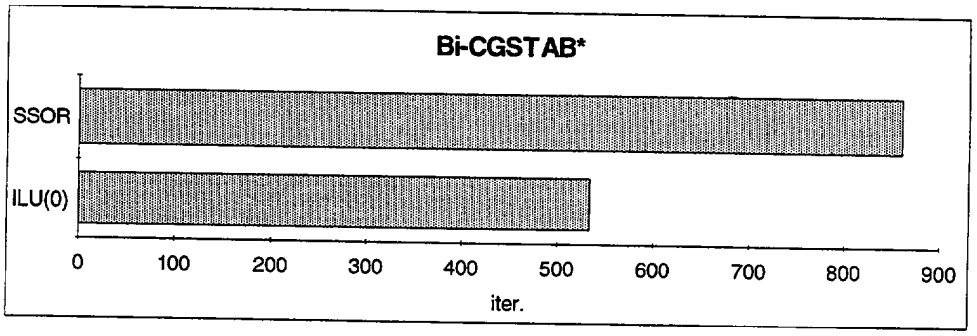


Figura 58: Problema 4 (13190 ecuaciones).

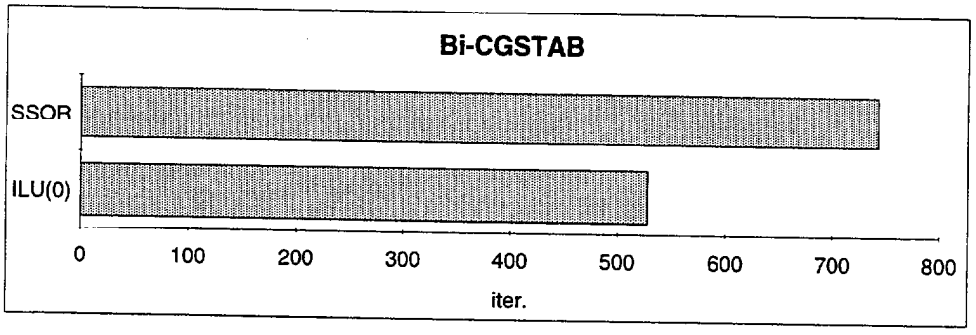


Figura 59: Problema 4 (13190 ecuaciones).

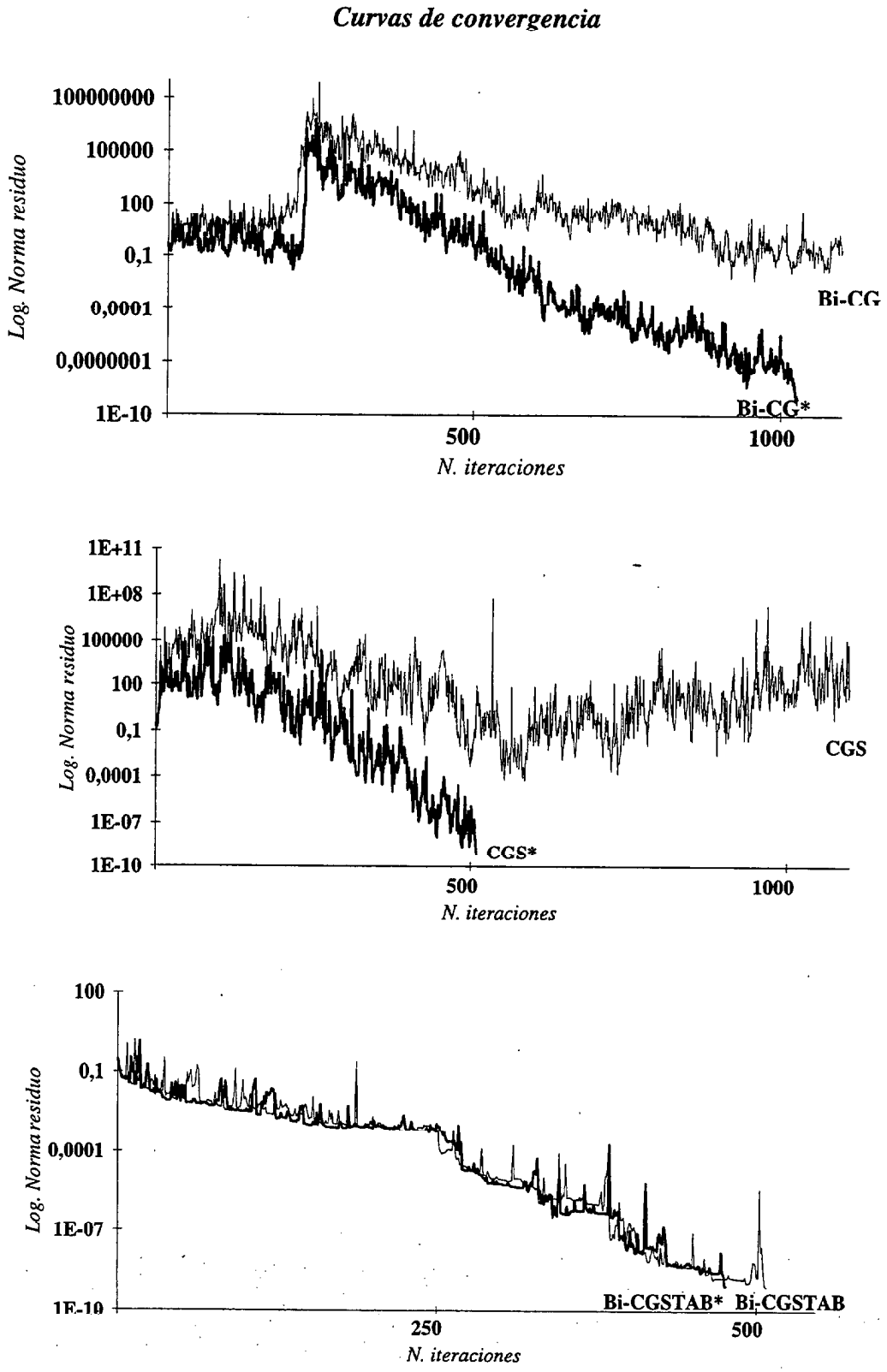


Figura 60: Problema 4 (13190 ecuaciones). Norma del residuo para ILU(0).

6.5 - OBSERVACIONES GENERALES

- Es notoria la necesidad del precondicionamiento. En la mayoría de los casos, cuando se aplican los métodos estudiados a los sistemas sin precondicionar, ó no convergen ó lo hacen en un número alto de iteraciones.

- Todos los algoritmos convergen con el precondicionador ILU.

- Los precondicionadores Diagonal y SSOR convergen en ocasiones, pero, salvo problemas particulares como sistemas simétricos y bien condicionados, en un número de iteraciones superior al que resulta con ILU(0), y además, con tiempo de ejecución mayor. Los casos en los que este tiempo para los primeros es inferior, son aquellos en los que se presenta conjuntamente en las gráficas con el número de iteraciones. En los restantes, es el precondicionador basado en la descomposición incompleta el que proporciona resultados más favorables, no solo en número de iteraciones sino también en cuanto al tiempo computacional se refiere.

Hay que hacer notar que el "coste" de los precondicionadores D y SSOR, es inferior al del ILU(0) y, consiguientemente, el tiempo para obtener la primera iteración es mayor en éste.

- Con la excepción del Problema 3 (255 ecuaciones), son las dos versiones del Bi-CGSTAB las que proporcionan los mejores resultados.

- En el Problema 3 (7520 ecuaciones), no converge el algoritmo BI-CGSTAB con ILU(0), haciéndolo sin embargo en 371 iteraciones el correspondiente al Bi-CGSTAB* con el mismo precondicionador.

- En el Problema 4 (13190 ecuaciones), la convergencia del algoritmo CGS es mucho más lenta que la del CGS* con los mismos precondicionadores.

- En cuanto a la renumeración, se comprueba, en cualquier caso, como las técnicas utilizadas, preferentemente el algoritmo inverso de Cuthill-Mckee, disminuyen ostensiblemente el número de iteraciones requeridas para resolver estos sistemas de ecuaciones con una determinada tolerancia, y además, de tal forma que el nuevo coste computacional con el añadido del de la reordenación, es aún sensiblemente inferior.

Es esta circunstancia, analizando conjuntamente iteraciones y tiempo requeridos, la que justifica la reordenación previa a la aplicación del algoritmo precondicionado, porque pudiera suceder que la disminución en el número de iteraciones fuese a costa de un incremento el tiempo de computación total debido al trabajo adicional que implica la utilización del algoritmo correspondiente de renumeración.

CAPÍTULO 7.

APLICACIONES EN LA RESOLUCIÓN DE PROBLEMAS DE CONTROL EN LA FRONTERA CON OPERADOR DE LAPLACE.

Se aplican técnicas y algoritmos desarrollados en capítulos anteriores a los sistemas de ecuaciones lineales, que resultan de discretizar la condición de optimización y la ecuación que define el problema, en un ejercicio concreto de control en la frontera.

7.1 - DEFINICIÓN DEL PROBLEMA

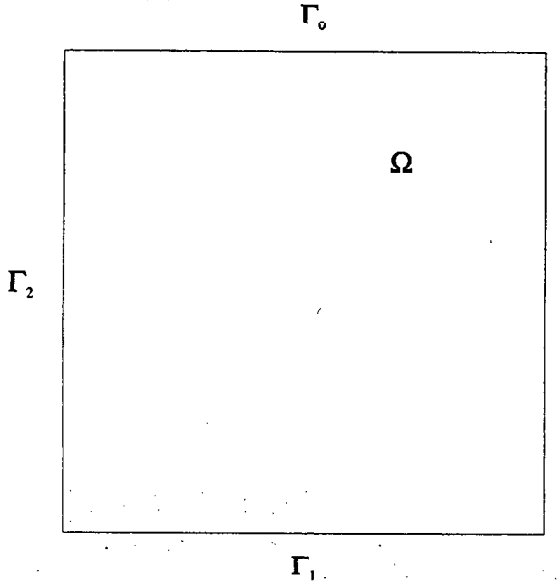
Encontrar v_{opt} solución de

$$v_{opt} = ArgMin\, j(v) \tag{1}$$

siendo

$$j(v) = \tfrac{1}{2}\int_{\Gamma_0} (u(v) - u_{target})^2\, d\Gamma_0 \tag{2}$$

y donde $u(v)$ es solución de



$$\left.\begin{array}{l} \Delta u = 0 \quad en \quad \Omega \\ \frac{\partial u}{\partial n} = 0 \quad en \quad \Gamma_0 \cup \Gamma_2 \cup \Gamma_3 \\ u = v \quad en \quad \Gamma_1 \end{array}\right\} \tag{3}$$

siendo $\Omega = (0,1) \times (0,1)$

7.2 - DISCRETIZACIÓN

Se ha aplicado diferencias finitas para la discretización de este problema, sin que se pierda generalidad en cuanto a la estrategia de resolución si utilizáramos otra técnica como, ejemplo, el método de elementos finitos.

Si consideramos una malla de N=n×n puntos, los vectores que intervienen en el planteamiento del problema se pueden escribir de la forma

$$\mathbf{v} = \{v_1, v_2, \ldots, v_n\} \text{ en } \Gamma_1$$

$$\mathbf{u}_{target} = \{u_1^t, u_2^t, \ldots, u_n^t\} \text{ en } \Gamma_0$$

$$\mathbf{Cu} = \{u_{1n}, u_{2n}, \ldots, u_{nn}\} \text{ en } \Gamma_0$$

siendo $\mathbf{C}$ la matriz nxN proyección del vector $\mathbf{u}$ definido en Ω, sobre Γ_0.

La ecuación (2) equivale ahora, por tanto,

$$j(\mathbf{v}) = \tfrac{1}{2}\sum_{i=1}^{n}(u_{in} - u^t)^2 \tag{4}$$

La discretización de (3) se puede expresar en forma matricial como:

$$\mathbf{Au} = \mathbf{Bv} \tag{5}$$

donde $\mathbf{A}$ es la matriz N×N clásica dada por el esquema en diferencias centradas para las segundas derivadas y descentradas para las primeras y $\mathbf{B}$ es la matriz N×n de prolongación de $\mathbf{v}$ en el segundo miembro de dimensión N.

7.3 - OPTIMIZACIÓN

La expresión (4) se puede escribir en forma vectorial como $j(\mathbf{v}) = \frac{1}{2}\left(\mathbf{Cu} - \mathbf{u}_{target}\right)^t\left(\mathbf{Cu} - \mathbf{u}_{target}\right)$. Sustituyendo el vector $\mathbf{u}$ en función del vector $\mathbf{v}$ de (5), $j(\mathbf{v}) = \frac{1}{2}\left(\mathbf{CA}^{-1}\mathbf{Bv} - \mathbf{u}_{target}\right)^t\left(\mathbf{CA}^{-1}\mathbf{Bv} - \mathbf{u}_{target}\right)$. Derivando respecto $\mathbf{v}$ en todo el dominio Ω, e imponiendo la condición de mínimo,

$$j'(\mathbf{v}) = \mathbf{A}^{-T}\mathbf{C}^T\left(\mathbf{CA}^{-1}\mathbf{Bv} - \mathbf{u}_{target}\right) = \mathbf{A}^{-T}\mathbf{C}^T\left(\mathbf{Cu} - \mathbf{u}_{target}\right)$$

haciendo $\tau = \mathbf{A}^{-\mathrm{T}}\mathbf{C}^{\mathrm{T}}\left(\mathbf{C}\mathbf{u} - \mathbf{u}_{target}\right)$, se puede escribir,

$$\mathbf{A}^{\mathrm{T}}\tau = \mathbf{C}^{\mathrm{T}}\left(\mathbf{C}\mathbf{u} - \mathbf{u}_{target}\right) \tag{6}$$

quedando en Ω,

$$j'(\mathbf{v}) = \tau,$$

y, proyectando en Γ_0,

$$j'(\mathbf{v}) - \mathbf{C}\tau .$$

Por tanto la condición de ortogonalidad resulta,

$$\mathbf{C}\tau = 0. \tag{7}$$

En resumen, hay que obtener una solución $\mathbf{v}$ que verifique la condición $\mathbf{C}\tau = \mathbf{0}$, estando relacionados los vectores $\mathbf{u}$ y $\mathbf{v}$ por el sistema $\mathbf{A}\mathbf{u} = \mathbf{B}\mathbf{v}$ y los vectores $\mathbf{u}$ y τ por $\mathbf{A}^{\mathrm{T}}\tau = \mathbf{C}^{\mathrm{T}}\left(\mathbf{C}\mathbf{u} - \mathbf{u}_{target}\right)$.

La ecuación matricial (6) equivale a discretizar por diferencias finitas el problema de contorno:

$$\left.\begin{array}{ll} \Delta\tau = 0 & en\ \ \Omega \\ \frac{\partial\tau}{\partial\mathbf{n}} = 0 & en\ \ \Gamma_2 \cup \Gamma_3 \\ \frac{\partial\tau}{\partial\mathbf{n}} = u - u_{target} & en\ \ \Gamma_0 \\ \tau = 0 & en\ \ \Gamma_1 \end{array}\right\} \tag{8}$$

de forma equivalente a la discretización de (3).

7.4 - MÉTODOS DE RESOLUCIÓN

Para resolver los sistemas de ecuaciones (5) y (6), hemos aplicado el Bi-CGSTAB* con precondicionador ILU y para el problema de optimización general los métodos CG, el Bi-CGSTAB, el QMRCGS y el VGMRES.

A efectos de determinar el vector residuo inicial $\mathbf{r}_0$ del sistema (7), hagamos notar que en la resolución de un hipotético sistema $Av = b$, siendo $j(v)$ una funcional cuadrática, se cumple

$$grad\ j(v) = j'(v)$$
$$grad\ j(v) = Av - b$$

con lo que $\mathbf{r} = b - Av = -j'(v)$, y de (7), $\mathbf{r} = -\mathbf{C}\tau$.

En lo sucesivo denotaremos $\mathbf{u} = LAP(\mathbf{A},\mathbf{B},\mathbf{v})$ a la solución de (5) y $\tau = RES(\mathbf{A},\mathbf{C},\mathbf{u},\mathbf{u}_{target})$ a la solución de (6). Hagamos notar que los productos matriz por vector que aparecen en los algoritmos, al igual que el cálculo del residuo, se obtienen resolviendo estos sistemas (5) y (6).

Algoritmo CG

Aproximación inicial $\mathbf{v}_0$;

Resolver $\mathbf{u} = LAP(\mathbf{A},\mathbf{B},\mathbf{v}_0)$; Resolver $\tau = RES(\mathbf{A},\mathbf{C},\mathbf{u},\mathbf{u}_{target})$; $\mathbf{r}_0 = -\mathbf{C}\tau$;

elegir $\underline{\mathbf{r}}_0$ arbitrario, tal que $\underline{\mathbf{r}}_0^{\mathrm{T}}\,\mathbf{r}_0 \neq 0$. e.g., $\underline{\mathbf{r}}_0 = \mathbf{r}_0$;

$\rho_0 = 1$; $\mathbf{p}_0 = \mathbf{0}$;

hacer mientras $\|\mathbf{r}_{i-1}\|/\|\mathbf{r}_0\| \geq \varepsilon$ (i=1,2,3,....),

$\rho_i = \mathbf{r}_{i-1}^{\mathrm{T}}\,\mathbf{r}_{i-1}$; $\beta_i = \rho_i/\rho_{i-1}$; $\mathbf{p}_i = \mathbf{r}_{i-1} + \beta_i\,\mathbf{p}_{i-1}$;

Resolver $\mathbf{x} = LAP(\mathbf{A},\mathbf{B},\mathbf{p}_i)$; Resolver $\sigma = RES(\mathbf{A},\mathbf{C},\mathbf{x},\mathbf{0})$; $\mathbf{y}_i = \mathbf{C}\sigma$;

$\alpha_i = \rho_i/(\mathbf{p}_i^{\mathrm{T}}\,\mathbf{y}_i)$; $\mathbf{v}_i = \mathbf{v}_{i-1} + \alpha_i\,\mathbf{p}_i$; $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i\,\mathbf{y}_i$;

fin

Algoritmo Bi-CGSTAB

Aproximación inicial $\mathbf{v}_0$;

Resolver $\mathbf{u} = LAP(\mathbf{A},\mathbf{B},\mathbf{v}_0)$; Resolver $\tau = RES(\mathbf{A},\mathbf{C},\mathbf{u},\mathbf{u}_{target})$; $\mathbf{r}_0 = -\mathbf{C}\tau$;

elegir $\underline{\mathbf{r}}_0$ arbitrario, tal que $\underline{\mathbf{r}}_0^{\mathrm{T}}\mathbf{r}_0 \neq 0$. e.g., $\underline{\mathbf{r}}_0 = \mathbf{r}_0$;

$\rho_0 = \alpha_0 = \omega_0 = 1$; $\mathbf{p}_0 = \mathbf{y}_0 = \mathbf{0}$;

hacer mientras $\|\mathbf{r}_{i-1}\|/\|\mathbf{r}_0\| \geq \varepsilon$ (i=1,2,3,....),

$\rho_i = \underline{\mathbf{r}}_0^{\mathrm{T}}\mathbf{r}_{i-1}$; $\beta_i = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$; $\mathbf{p}_i = \mathbf{r}_{i-1} + \beta_i(\mathbf{p}_{i-1} - \omega_{i-1}\mathbf{y}_{i-1})$;

Resolver $\mathbf{x} = LAP(\mathbf{A},\mathbf{B},\mathbf{p}_i)$; Resolver $\sigma = RES(\mathbf{A},\mathbf{C},\mathbf{x},\mathbf{0})$; $\mathbf{y}_i = \mathbf{C}\sigma$;

$\alpha_i = \rho_i/(\underline{\mathbf{r}}_0^{\mathrm{T}}\mathbf{y}_i)$; $\mathbf{s} = \mathbf{r}_{i-1} - \alpha_i\mathbf{y}_i$;

Resolver $\mathbf{x} = LAP(\mathbf{A},\mathbf{B},\mathbf{s})$; Resolver $\sigma = RES(\mathbf{A},\mathbf{C},\mathbf{x},\mathbf{0})$; $\mathbf{t} = \mathbf{C}\sigma$;

$\omega_i = (\mathbf{t}^{\mathrm{T}}\mathbf{s})/(\mathbf{t}^{\mathrm{T}}\mathbf{t})$; $\mathbf{v}_i = \mathbf{v}_{i-1} + \alpha_i\mathbf{p}_i + \omega_i\mathbf{s}$; $\mathbf{r}_i = \mathbf{s} - \omega_i\mathbf{t}$;

fin

Algoritmo QMRCGSTAB

Aproximación inicial $\mathbf{v}_0$;

Resolver $\mathbf{u} = LAP(\mathbf{A},\mathbf{B},\mathbf{v}_0)$; Resolver $\tau = RES(\mathbf{A},\mathbf{C},\mathbf{u},\mathbf{u}_{target})$; $\mathbf{r}_0 = -\mathbf{C}\tau$;

elegir $\underline{\mathbf{r}}_0$ arbitrario, tal que $\underline{\mathbf{r}}_0^{\mathrm{T}}\mathbf{r}_0 \neq 0$. e.g., $\underline{\mathbf{r}}_0 = \mathbf{r}_0$;

$\rho_0 = \alpha_0 = \omega_0 = 1$; $\theta_0 = \eta_0 = 0$; $\gamma = \|\mathbf{r}_0\|$; $\mathbf{p}_0 = \mathbf{y}_0 = \mathbf{d}_0 = \mathbf{0}$;

hacer mientras $\|\mathbf{r}_{i-1}\|/\|\mathbf{r}_0\| \geq \varepsilon$ (i=1,2,3,....),

$\rho_i = \underline{\mathbf{r}}_0^{\mathrm{T}}\mathbf{r}_{i-1}$; $\beta_i = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$; $\mathbf{p}_i = \mathbf{r}_{i-1} + \beta_i(\mathbf{p}_{i-1} - \omega_{i-1}\mathbf{y}_{i-1})$;

Resolver $\mathbf{x} = LAP(\mathbf{A},\mathbf{B},\mathbf{p}_i)$; Resolver $\sigma = RES(\mathbf{A},\mathbf{C},\mathbf{x},\mathbf{0})$; $\mathbf{y}_i = \mathbf{C}\sigma$;

$\alpha_i = \rho_i/(\underline{\mathbf{r}}_0^{\mathrm{T}}\mathbf{y}_i)$; $\mathbf{s}_i = \mathbf{r}_{i-1} - \alpha_i\mathbf{y}_i$;

Primera cuasi-minimización y solución intermedia

$\underline{\theta}_i = \|\mathbf{s}_i\|/\gamma$; $c = 1/\sqrt{1+\underline{\theta}_i^2}$; $\underline{\gamma} = \gamma\theta_i c$; $\underline{\eta}_i = c^2\alpha_i$;

$\underline{\mathbf{d}}_i = \mathbf{p}_i + \frac{\theta_{i-1}^2\eta_{i-1}}{\alpha_i}\mathbf{d}_{i-1}$; $\underline{\mathbf{y}}_i = \mathbf{v}_{i-1} + \underline{\eta}_i\underline{\mathbf{d}}_i$

Cálculo de $\mathbf{t}_i, \omega_i, \mathbf{r}_i$

Resolver $\mathbf{x} = LAP(\mathbf{A},\mathbf{B},\mathbf{s}_i)$; Resolver $\sigma = RES(\mathbf{A},\mathbf{C},\mathbf{x},\mathbf{0})$; $\mathbf{t}_i = \mathbf{C}\sigma$;

$\omega_i = (\mathbf{t}_i^T\mathbf{s}_i)/(\mathbf{t}_i^T\mathbf{t}_i)$; $\mathbf{r}_i = \mathbf{s}_i - \omega_i\,\mathbf{t}_i$;

Segunda cuasi-minimización y solución

$\theta_i = \|\mathbf{r}_i\|/\underline{\gamma}$; $c = 1/\sqrt{1+\theta_i^2}$; $\gamma = \underline{\gamma}\,\theta_i\,c$; $\eta_i = c^2\omega_i$;

$\mathbf{d}_i = \mathbf{s}_i + \frac{\theta_i^2\eta_i}{\omega_i}\underline{\mathbf{d}}_i$; $\mathbf{v}_i = \underline{\mathbf{v}}_i + \eta_i\,\mathbf{d}_i$;

fin

Algoritmo VGMRES

Aproximación inicial $\mathbf{v}_0$; $k = 0$

Elegir k_{max}, dimensión máxima del subespacio de Krylov y la subtolerancia definida por $\delta \in]0,1]$;

Resolver $\mathbf{u} = LAP(\mathbf{A},\mathbf{B},\mathbf{v}^{(0)})$; Resolver $\tau = RES(\mathbf{A},\mathbf{C},\mathbf{u},\mathbf{u}_{target})$; $\mathbf{r}^{(0)} = -\mathbf{C}\tau$;

hacer mientras $\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\| \geq \varepsilon$ (n=1,2,3,....),

$\beta^{(n-1)} = \|\mathbf{r}^{(n-1)}\|$; $\mathbf{y}_1 = \mathbf{r}^{(n-1)}/\beta^{(n-1)}$;

Si $\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\| \geq \varepsilon^{\delta}$ y $k < k_{max}$ hacer $k = k+1$;

Para j=1,2,...,k

Resolver $\mathbf{x} = LAP(\mathbf{A},\mathbf{B},\mathbf{y}_j)$; Resolver $\sigma = RES(\mathbf{A},\mathbf{C},\mathbf{x},\mathbf{0})$; $\mathbf{w} = \mathbf{C}\sigma$;

$h_{ij} = \mathbf{w}^T\mathbf{y}_i$, i=1,...,j

$$\mathbf{w} = \mathbf{w} - \sum_{i=1}^{j} h_{ij}\mathbf{y}_i;\quad h_{j+1j} = \|\mathbf{w}\|;\quad \mathbf{y}_{j+1} = \frac{\mathbf{w}}{h_{j+1j}};$$

fin

Resolver $\mathbf{M}_k^T\underline{\mathbf{z}} = \mathbf{d}_k$ y $\mathbf{M}_k\mathbf{z} = \underline{\mathbf{z}}$; siendo $\begin{cases} (\mathbf{d}_k)_i = h_{1i} \\ (\mathbf{M}_k)_{ij} = h_{i+1j} \end{cases}$;

$$\lambda^{(n)} = \frac{\beta^{(n-1)}}{1+\mathbf{d}_k^T\mathbf{z}};\quad \mathbf{p}_k = \lambda^{(n)}\,\mathbf{z};\quad \mathbf{v}^{(n)} = \mathbf{v}^{(n-1)} + \mathbf{Y}_k\,\mathbf{p}_k;\quad \text{siendo } \mathbf{Y}_k = [\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_k]$$

Resolver $\mathbf{u} = LAP(\mathbf{A},\mathbf{B},\mathbf{v}^{(n)})$; Resolver $\tau = RES(\mathbf{A},\mathbf{C},\mathbf{u},\mathbf{u}_{target})$; $\mathbf{r}^{(n)} = -\mathbf{C}\tau$;

fin

7.5 - RESULTADOS NUMÉRICOS

Las figuras 61-67 muestran el comportamiento de los métodos anteriores al aplicarlos al problema considerando diferentes niveles de discretización. En la aplicación del Bi-CGSTAB* para la resolución de (5) y (6) hemos exigido una tolerancia alta, (10^{-14}), para asegurarnos en lo posible la exactitud de los productos matriz por vector que figuran en el algoritmo que resuelve el problema de optimización. Para el problema global se ha elegido una tolerancia de 10^{-13} en los algoritmos que resuelven los problemas para n= 10, 20, 30 y 40, y de 10^{-9} para las discretizaciones de n= 75, 100 y 150.

Se presentan, asimismo, respectivas tablas, donde figuran los tiempos requeridos por cada método, a efectos de comparar no sólo la robustez de los mismos en cuanto al número de iteraciones se refiere, sino también para evaluar el coste computacional que requiere cada uno.

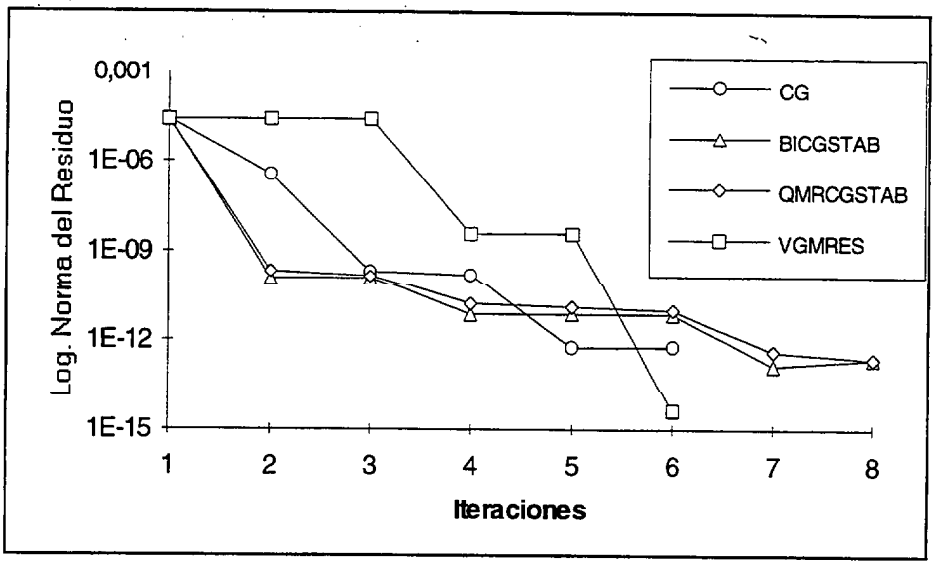


Figura 61. $Log\left[\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\|\right]$ para n=10.

Tabla 2. Tiempos de CPU para n=10

ALGORITMO	*TIEMPO DE CPU (s)*
CG	0.88
Bi-CGSTAB	2.52
QMRCGSTAB	2.61
VGMRES	5.66

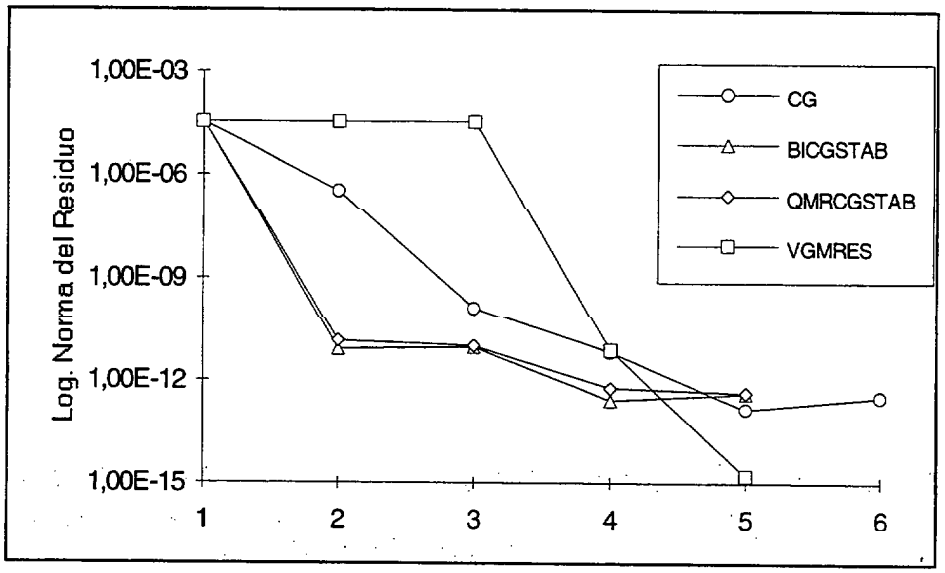


Figura 62. $Log\left[\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\|\right]$ para n=20.

Tabla 3. Tiempos de CPU para n=20

ALGORITMO	*TIEMPO DE CPU (s)*
CG	57.42
Bi-CGSTAB	33.72
QMRCGSTAB	24.44
VGMRES	67.74

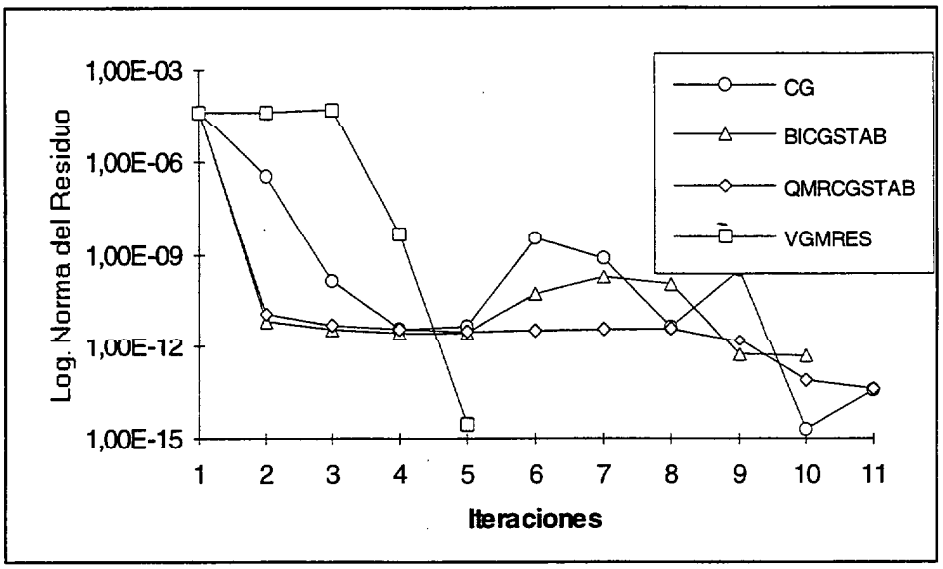


Figura 63. $Log\left[\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\|\right]$ para n=30.

Tabla 4. Tiempos de CPU para n=30

ALGORITMO	*TIEMPO DE CPU (s)*
CG	285.92
Bi-CGSTAB	240.75
QMRCGSTAB	157.11
VGMRES	524.25

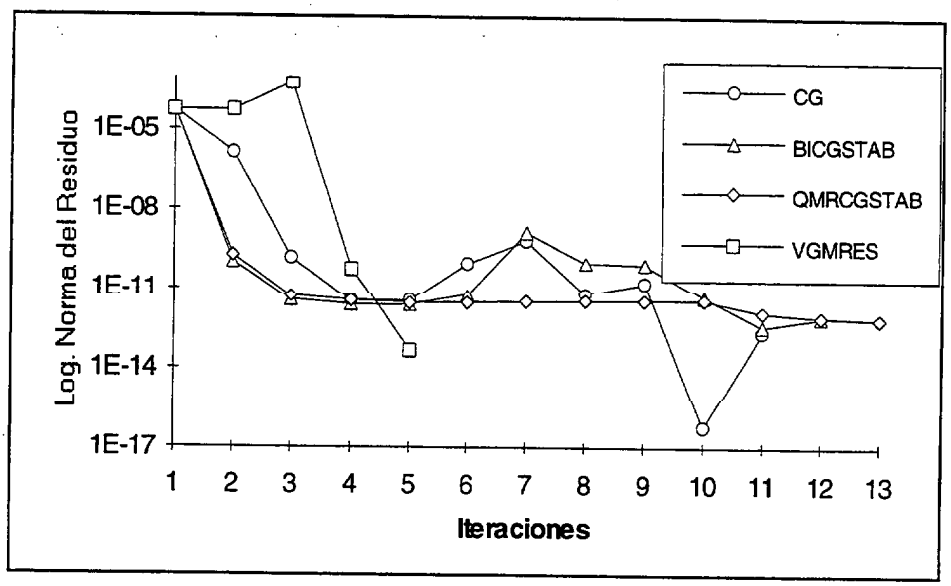


Figura 64. $Log\left[\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\|\right]$ para n=40.

Tabla 5. Tiempos de CPU para n=40 -

ALGORITMO	*TIEMPO DE CPU (s)*
CG	1046.88
Bi-CGSTAB	469.00
QMRCGSTAB	490.36
VGMRES	1302.33

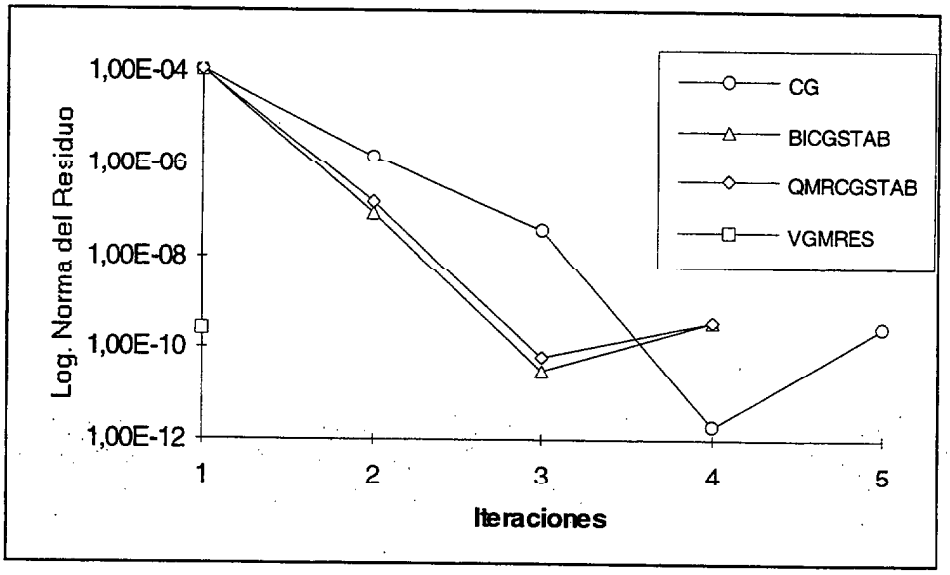


Figura 65. $Log\left[\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\|\right]$ para n=75.

Tabla 6. Tiempos de CPU para n=75

ALGORITMO	*TIEMPO DE CPU (s)*
CG	5985.22
Bi-CGSTAB	4165.31
QMRCGSTAB	4163.57
VGMRES	8370.23

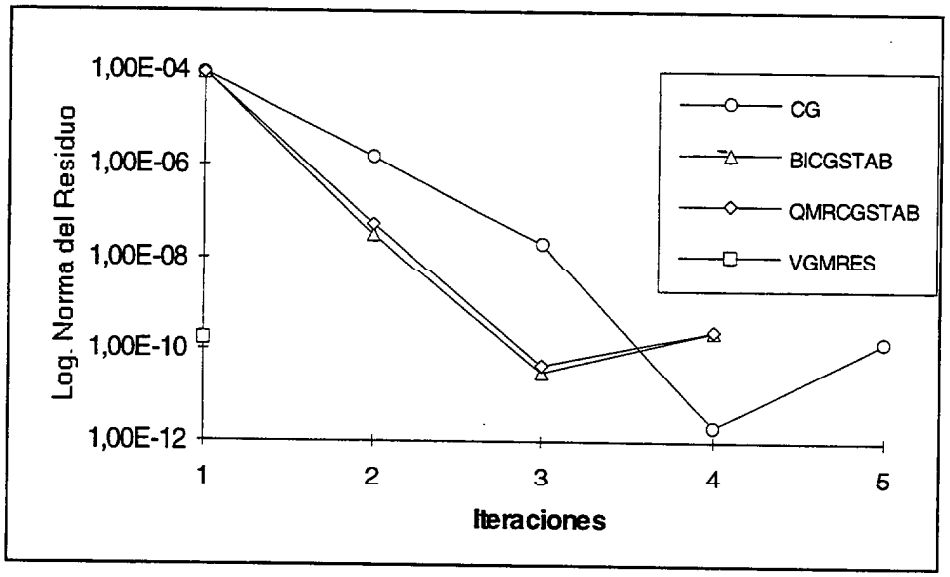


Figura 66. $Log\left[\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\|\right]$ para n=100.

Tabla 7. Tiempos de CPU para n=100

ALGORITMO	*TIEMPO DE CPU (s)*
CG	19115.58
Bi-CGSTAB	13172.55
QMRCGSTAB	13227.87
VGMRES	25290.57

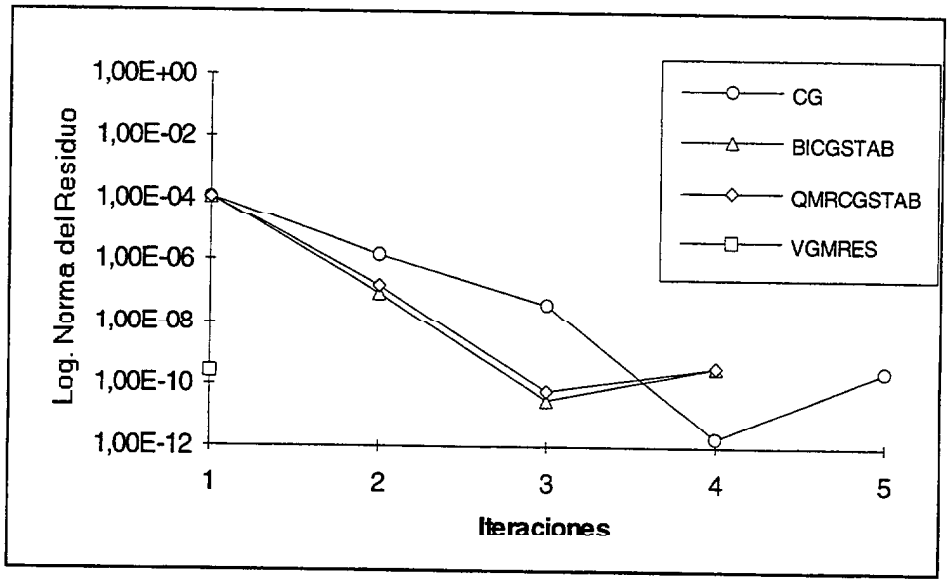


Figura 67. $Log\left[\|\mathbf{r}^{(n-1)}\|/\|\mathbf{r}^{(0)}\|\right]$ para n=150.

Para n=150, los tiempos de ejecución siguen la misma tónica que para el caso n=100, siendo el método más rápido, el Bi-CGSTAB con 12424.81 s. de CPU consumidos.

Observaciones

Se observa, en general, un comportamiento numérico similar para el Bi-CGSTAB y el QMRCGSTAB, con coste computacional también del mismo orden. Los cambios de pendiente de las rectas representativas de la convergencia son debidas al proceso de reinicio inherente a estos métodos al recalcular los vectores residuos directamente en lugar de utilizar la fórmula de recurrencia. El VGMRES presenta una convergencia más rápida que se acentúa al aumentar n (mallas más finas). Así, para los casos de n= 75 y n= 100 converge en una sola iteración, pero de cualquier modo, a costa de requerir un elevadó tiempo de computación. Los resultados obtenidos para el GC son también satisfactorios pero su eficacia está por debajo de la de los otros métodos.

Se observa, además, que el coste por iteración de cada método no sólo depende del número teórico de operaciones por iteración, sino también de la calidad de las soluciones intermedias que va generando cada algoritmo, que influyen en la velocidad de resolución de (5) y (6). Así, una iteración de GC, teóricamente más barato que el Bi-CGSTAB ó el QMRCGSTAB, puede ser más costosa que una de estos últimos.

CONCLUSIONES Y LÍNEAS FUTURAS

El análisis de los algoritmos precondicionados establecidos, con las aportaciones de las variantes propuestas, y las aplicaciones prácticas realizadas para las formas de precondicionamiento y técnicas de renumeración que se exponen en los distintos capítulos, nos permiten obtener las siguientes conclusiones:

- Una adecuada elección del vector inicialización $\hat{r}_0$ es, en general, determinante del comportamiento de los algoritmos en cuanto a la convergencia. Se abre, pues, un amplio campo de elección que no sólo da lugar a precondicionamiento por la izquierda, por la derecha o por ambos lados, sino que existirán otras diversas formas de precondicionar dependiendo del vector $\hat{r}_0$.

- Las aplicaciones numéricas sugieren que no sólo es fundamental utilizar precondicionamiento para la resolución de un sistema de forma eficiente, sino que la adecuada elección del mismo, dependiendo de cada problema, puede ser incluso más importante.

- Se ha comprobado que el efecto de la renumeración cuando se utiliza el precondicionador ILU ha sido siempre positivo tanto desde el punto de vista de la convergencia (suavidad y número de iteraciones), como, y principalmente, desde el del coste computacional, ya que el tiempo requerido para la renumeración fue siempre de orden muy inferior al total del proceso.

- Las versiones que aquí se han propuesto para los algoritmos, Bi-CG, CGS y Bi-CGSTAB, aportan mejoras en la convergencia. Éstas son más acentuadas para los casos del Bi-CG y CGS, donde se obtienen curvas de convergencia más suaves. Las diferencias relativas al Bi-CGSTAB no son tan espectaculares, si bien en algún caso se obtuvo convergencia con nuestro algoritmo y no con la versión clásica.

- Se han obtenido resultados satisfactorios en la aplicación de los métodos de Krylov al problema de control en la frontera. Aunque el GMRES (en la versión variable

que hemos utilizado) se presenta como el método más robusto y seguro, el aumento de memoria y el elevado coste computacional requeridos, indican que para los casos de dimensiones menores parece más adecuada la utilización de métodos de biortogonalización.

Como perspectivas ó líneas futuras para las cuestiones planteadas en estas conclusiones podíamos señalar:

- Estudiar las distintas formas de precondicionamiento y la utilización de los diversos precondicionadores en el QMR y otros métodos derivados de éste.

- Profundizar en técnicas de elección del vector inicialización $\hat{r}_0$ considerando su influencia en la forma de precondicionar.

- Ampliar el campo de posibles precondicionadores a utilizar que supongan mejoras en la convergencia con el menor coste computacional añadido posible.

- Aplicar los métodos basados en los subespacios de Krylov en problemas de control óptimo no lineales.

REFERENCIAS

[1] L. Adams, **m-Step preconditioned conjugate gradient methods**, *SIAM J. Sci. Stat. Comput.*, 6, 2, 453 - 463, (1985).

[2] P. Almeida, **Resolución directa de sistemas sparse por grafos,** *Tesis doctoral*, ULPGC, (1989).

[3] S. F. Ashby, T. A. Manteuffel and P. E. Saylor, **A taxonomy for conjugate gradient methods,** *SIAM J. Numer. Anal.*, 27, 1542-1568, (1990).

[4] O. Axelson, **Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations**, *Linear Algebra and its applications,* 29, 1-16, (1980).

[5] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H.A. van der Vorst, **Templates for the solution of linear systems**, SIAM, Philadelphia - (1994).

[6] T.F. Chan, E. Gallopoulos, V. Simonsini, T. Szeto and C.H. Tong, **A Quasi-Minimal residual variant of the BI-CGSTAB algorithm for nonsymmetric systems,** *SIAM J. Sci.Comput.*, 15, 2, 338-347, (1994).

[7] C. Conde and G. Winter, **Métodos y algoritmos básicos del álgebra numérica**, Editorial Reverté, Barcelona - (1990).

[8] E. H. Cuthill and J. M. McKee, **Reducing the bandwidth of sparse symmetric matrices**, *Proc. 24th National Conference of the Association for Computing Machinery*, Brondon Press, New Jersey, 157-172, (1969).

[9] Q. V. Dinh, V. Mantel, J. Periaux and B. Stoufslet, **Contribution to problems T4 and T6 finite element GMRES and conjugate gradient solvers**, *Technical Report, Dassault Aviation*, (1993).

[10] J. J. Dongarra, I. S. Duff, D. C. Sorensen and H.A. van der Vorst, **Solving Linear Systems on Vector and Shared Memory Computers**, *SIAM*, (1991).

[11] L.C. Dutto, **The effect of ordering on preconditioned GMRES algorithm**, *Int. Jour. Num. Meth. Eng.*, 36, 457-497, (1993).

[12] L. Ferragut, **NEPTUNO, un Sistema adaptativo de Elementos Finitos**, Dpto. de Matemática Aplicada y Métodos Informáticos, ETSI de Minas, Madrid, (1987).

[13] S. Finzi Vita, **A numerical study of relaxed shape optimization for Dirichlet problems**, *Numerical Methods in Engineering'92*, Els. Sci. Publ. B.V. (1992).

[14] R. Fletcher, **Conjugate gradient methods for indefinite systems**, *Lectures Notes in Math.*, 506, 73-89, (1976).

[15] R.W. Freund, **A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems**, *SIAM J. Sci. Comput.*, 14, 470-482, (1993).

[16] R. W. Freund, M. H. Gutknecht and N. M. Natchigal, **An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices**, *SIAM J. Sci. Comput.*, 14, 137-158, (1993).

[17] R.W. Freund and N.M. Nachtigal, **QMR: a quasi-minimal residual method for non-Hermitian linear systems**, *Numerische Math.*, 60, 315-339, (1991).

[18] R.W. Freund and N.M. Natchigal, **An implementation of the QMR method based on coupled two-term recurrences**, *SIAM J. Sci. Comp.*, 15, 2, 313-337, (1994).

[19] R.W. Freund and T. Szeto, **A quasi-minimal residual squared algorithm for non-Hermitian linear systems**, *Tech. Report. 91.26.* Research Institute for Advanced Computer Science, NASA Ames Research Center. Moffett Field. C.A.,(1991).

[20] M. Galán, G. Montero and G. Winter, **A direct solver for the least square problem arising from GMRES(k)**, *Com. Num. Meth. Eng.*, 10, 743-749, (1994).

[21] A. George, **Computer implementation of the finite element method**, *Report stan CS-71-208*, (1971).

[22] A. George and J. W. Liu, **The evolution of the minimum degree ordering algorithm**, *SIAM Rev.*, 31, 1-19, (1989).

[23] G. H. Golub et G. A. Meurant, **Résolution numérique des grands systèmes linéaires**, Editions Eyrolles, París - (1983).

[24] M. Grote and H. Simon, **Parallel Preconditioning and Approximate Inverses on the Connection Machine,** *NASA Contract No. NAS2-12961.*

[25] I. Gustafsson, **A class of first order factorization methods**, *BIT*, 18, 142-156, (1978).

[26] M.R. Hestenes and E. Stiefel, **Methods of conjugate gradients for solving linear systems.** *Jour. Res. Nat. Bur. Standards*, 49, 409 436, (1952).

[27] A. Jennings and G.M. Malik, **The solution of sparse linear equations by conjugate gradient method,** *Int. Jour. Num. Meth. Eng.*, 12, 141-158, (1978).

[28] J. J. Júdice and J. M. Patrício, **Truncated envelope preconditioning technique**, *Conmunications in numerical methods in Engineering*, Vol. 10, 149-154, (1994).

[29] P. Lascaux et R. Théodor, **Analyse Numérique matricielle appliquée a l'art de l'ingénieur**, Tomos 1 y 2, Edit. Massson, París - (1987).

[30] G. Martin, **Méthodes de préconditionnement par factorisation incomplète**, *Mémoire de Maitrise*, Université Laval, Québec, Canada - (1991).

[31] J.A. Meijerink and H.A. van der Vorst, **An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix**, *Math. Comp.*, 31, 148-162, (1977).

[32] G. Montero, **Aplicación de esquemas elemento a elemento de resolución de sistemas de ecuaciones asociados a métodos de elemento finitos adaptativos**, *Tesis doctoral*, ULPGC, (1989).

[33] G. Montero, R. Montenegro, G. Winter and L. Ferragut, **Aplicación de esquemas EBE en procesos adaptativos**, *Rev. Int. Met. Num. Cal. Dis. Ing.*, 6, 311-332, (1990).

[34] G. Montero y A. Suárez, **Precondicionamiento de matrices en la resolución de sistemas de ecuaciones lineales. Aplicaciones en métodos tipo-gradiente**, *Encuentro de Análisis matricial y aplicaciones*, Vitoria, (1994).

[35] A. Muller and T. J. Hughes, **Precondicionadores elemento-por-elemento y globales. Una perspectiva**, *Rev. Int. Met. Num. Cal. Dis. Ing.*,2, 1, 27-41, (1986).

[36] N.M. Nachtigal, S.C. Reddy and L.N. Trefethen, **How fast are nonsymmetric matrix iterations?**, *SIAM J. Matr. Anal. Appl.*, 13, 3, 778-795, (1992).

[37] B. Nour-Omid and B.N. Parlett, **Element preconditioning using splitting techniques**, *SIAM J. Sci. Stat. Comput.*, 6, 3, 761-771, (1985).

[38] C. P. Paige and M. A. Saund4ers, **LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares**, *ACM Transactions on Mathematical Software*, 8, 1, 43-71, (1982).

[39] A. Peters, **Non-symmetric CG-like schemes and the finite element solution of the advection-dispersion equation**, *Int. Jour. Num. Meth. Fl.*, 9, 67-79, (1993).

[40] O. Pironneau and A. Vossinis, **Comparison of some optimization algorithms for optimun shape design in aerodinamics**, *INRIA Report*, (1991).

[41] K.V.G. Prakhya, **Some Conjugate Gradient methods for symmetric and nonsymmetric systems**, *Com. Appl. Num. Meth.*, 4, 531-539, (1988).

[42] G. Radicati and M. Vitaletti, **Sparse matrix-vector product and storage representations on the IBM 3090 with Vector Facility**, *Report G513-4098, IBM-ECSEC,* Roma - (1986).

[43] G. Radicati and Y. Robert, **Parallel conjugate gradient-like algorithms for solving sparse non-symmetric systems on a vector multiprocessor**, *Parallel Comput.,* 11, 223-239, (1989).

[44] G. Radicati and M. Vitaletti, **Sparse matrix-vector product and storage representations on the IBM 3090 with Vector Facility**, Report G513-4098, IBM-ECSEC, Rome, (1986).

[45] Y. Saad, **Highly Hparallel preconditioners for general sparse matrices**, Tech. Rep. 92-087, Army High Performance Computing Research Center, Minneapolis, MN, (1992).

[46] Y. Saad, **A flexible inner-outer preconditioned GMRES algorithm**, *SIAM J. Sci. Statist. Comput.*, 14, 2, 461-469, (1993).

[47] Y. Saad and M.H. Schultz, GMRES: **A generalized minimal residual algorithm for solving nonsymmetric linear systems**, *SIAM J. Sci. Statist. Comput.*, **7**, 856-869, (1986).

[48] J. N. Shadid and R.S. Tuminaro, **A comparison of preconditioned nonsymmetric Krylov methods on a large-scale mind machine**, *SIAM J. Sci. Statist. Comput.*,15, 2, 440 459, (1994).

[49] P. Sonneveld, **CGS: a fast Lanczos-type solver for nonsymmetric linear systems,** *SIAM J. Sci. Statist. Comput.,* 10, 36-52, (1989).

[50] A. Suárez, G. Montero y M. Galán, **Estudio del comportamiento de diferentes precondicionadores para doble gradiente conjugado en problemas de convección-difusión**, *XIII C.E.D.Y.A. / III C.M.A.*, Madrid, (1993).

[51] Tsun-zee Mai, **Modified Lanczos method for solving large sparse linear systems,** *Com. Num. Meth. Eng.*, 9, 67-79, (1993).

[52] A. van der Sluis and H. A. van der Vorst, **The rate of convergence of conjugate gradients**, *Numer. Math.*, 48, 543-560, (1986).

[53] H.A. van der Vorst, **The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors**, *Lectures Notes in Math.*, 1457, 126-136, (1990).

[54] H. A. van der Vorst, **Conjugate gradient type methods for nonsymmetric linear systems**, *Iterative methods in linear algebra,* Proceedings of the IMACS International Symposium, Bruselas, (1991).

[55] 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. Statist. Comput.*, **13**, 2, 631-644, (1992).

[56] H.A. van der Vorst and C. Vuik, **GMRESR: A family of nested GMRES methods,** Tech. Rep. 91-80, Delft University of Technology, Mathematics and Informatics, Delft, The Netherlands, (1991).

[57] P. M. de Zeeuw, **Incomplete line LU for discretized coupled PDEs as preconditioner in BI-CGSTAB**, *Report*, Centre for Mathematics and Computer Science, Amsterdam.

[58] P. M. de Zeeuw, **Incomplete Line LU as smoother and as preconditioner**, *Report NM-R9213*, Centre for Mathematics and Computer Science, Amsterdam.

[59] L. Zhou and H. F. Walker, **Residual smoothing techniques for iterative methods,** *SIAM J. Sci. Comput.*, 15, 2, 297-312, (1994).

[60] O. C. Zienkiewicz, **El método de los elementos finitos**, Editorial Reverté, S.A., Barcelona, 1980.