CMNE/CILAMCE 2007
Porto, 13 a 15 de Junho, 2007


# PRECONDICIONAMIENTO Y REORDENACIÓN EN SISTEMAS DE ECUACIONES LINEALES VARIABLES

**Antonio Suárez*, Hector Sarmiento, M. Dolores García, Elizabeth Flórez y Gustavo Montero**

Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería
Universidad de Las Palmas de Gran Canaria
Edificio Central del Parque Científico Tecnológico. Campus Universitario de Tafira
35017 - Las Palmas de Gran Canaria
e-mail: asuarez,hsarmiento,lgarcia,eflorez,gustavo@dma.ulpgc.es
web: http://www.iusiani.ulpgc.es

**Palabras clave:** Factorización Incompleta, Inversas Aproximadas, Sistemas de ecuaciones variables, Precondicionamiento, Gradiente Conjugado, Reordenación.

**Resumen.** *La aplicación de técnicas de discretización para resolver diversos problemas definidos por ecuaciones en derivadas parciales da lugar a sistemas de ecuaciones lineales cuyas respectivas matrices se pueden escribir de la forma $A_\varepsilon = M + \varepsilon N$ con $M$ y $N$ matrices simétricas y definidas positivas. Para precondicionar estos sistemas para distintos valores de $\varepsilon$, podríamos, por un lado,construir un precondicionador diferente para cada $\varepsilon$ lo que conllevaría un alto coste computacional, o bien, en el extremo opuesto, obtener un precondicionador para un cierto valor del parámetro y aplicarlo para los restantes valores, lo que daría lugar a un empeoramiento de la convergencia. En este trabajo se propone una solución intermedia para estos sistemas de forma similar a las propuestas por Meurant y Benzi, construyendo un único precondicionador que se pueda adaptar para cada valor del parámetro con un bajo coste computacional. Asimismo, se exponen distintos experimentos numéricos para ilustrar el efecto de las técnicas de reordenación en la convergencia de estos sistemas cuando se utilizan los precondicionadores propuestos.*

## 1. INTRODUCCIÓN

La aplicación de técnicas de discretización para resolver problemas definidos por ecuaciones en derivadas parciales, obtenidos de la modelización de fenómenos físicos, da lugar a la solución de sistemas lineales de ecuaciones cuya matriz está dada en función de un parámetro,

$$(M + \varepsilon N)\, x_\varepsilon = b_\varepsilon \tag{1}$$

Esta situación se encuentra, por ejemplo, en la simulación numérica de modelos de masa consistente de un campo de viento [1, 2]. Aquí consideramos que $M$ y $N$ son constantes para cada nivel de discretización, simétricas y definidas positivas (SPD) . Se sabe además, que el algoritmo del Gradiente Conjugado Precondicionado (PCG) [3, 4] nos da los mejores resultados de convergencia en la resolución de estos tipos de sistemas lineales. Para cada valor de $\varepsilon$ tenemos diferentes sistemas lineales. Podemos obtener un precondicionador diferente para cada uno y mejorar la convergencia del PCG a un alto coste computacional debido a la construcción de cada precondicionador, o usar un precondicionador obtenido con un valor dado de $\varepsilon$ para todos los sistemas. En este último caso la convergencia empeorará a medida que el valor del parámetro se aleje del valor inicial. Benzi [5] y Meurant [6] proponen una solución intermedia usando dos tipos de precondicionadores, respectivamente, que se construyen una vez al principio y se actualizan para cada $\varepsilon$ a un bajo coste computacional. Con estas estrategias obtenemos una convergencia intermedia entre las dos opciones extremas expuestas anteriormente. Benzi desarrolla el estudio de inversas aproximadas factorizadas usando el algoritmo SAINV [7] para el caso especial $A_\varepsilon = M + \varepsilon I$, donde $I$ es la matriz identidad. Sin embargo Meurant lo hace para $A_\varepsilon = M + \varepsilon D$ con $D$ diagonal, construyendo el precondicionador a partir de una factorización incompleta de Cholesky de $M$. En el presente trabajo, generalizamos ambos algoritmos para el caso de $A_\varepsilon = M + \varepsilon N$, con $M$ y $N$ matrices SPD. Por otra parte, muchos autores [8, 9, 10, 11, 12] muestran el efecto de la reordenación en la convergencia de los métodos iterativos basados en los subespacios de Krylov precondicionados [13, 14], y, en particular en la convergencia del PCG para problemas simétricos. Proponemos el estudio de este efecto cuando resolvemos sistemas con matrices de la forma $A_\varepsilon = M + \varepsilon N$ usando los precondicionadores propuestos en [15, 16] basados en los trabajos de Benzi y Meurant. Las técnicas de reordenación, reducen el perfil de la matriz y también tratan de evitar el efecto fill-in. Estas características las hacen útiles para mejorar, por una parte el comportamiento de los precondicionadores basados en inversas aproximadas factorizadas y, por otra, la calidad de los precondicionadores basados en la factorización incompleta de Cholesky . Los precondicionadores se construyen considerando solamente un pequeño número de diagonales superiores en la matriz $N$ y de la matriz triangular superior que se obtiene en la factorización SAINV de $A^{-1}$, así la factorización mejorará a medida que se reduce el perfil. Además, los últimos precondicionadores mejorarán si se reduce el efecto fill-in después de la reordenación. Los algoritmos de reordenación aplicados en los experimentos son: Reverse Cuthill-Mckee (RCM) [17, 18], Mínimo Vecino (MN) [19, 10] y Multicoloring (MC)

[14]. Para iniciar estos algoritmos y encontrar el nodo pseudo-periférico, usamos el algoritmo de George[19] que tiene un comportamiento similar a otros algoritmos propuestos posteriormente por Grimes [20] y Paulino [21]. En la sección 2 resumimos la construcción de los precondicionadores basados en la inversa aproximada factorizada y la factorización incompleta de Cholesky que fueron usados en los experimentos numéricos. La sección 3 muestra el efecto de estos precondicionadores y de la reordenación en sistemas lineales variables obtenidos de dos problemas de ingeniería diferentes. Finalmente, se presentan las conclusiones en la sección 4.

## 2. PRECONDICIONADORES DE SISTEMAS VARIABLES

En esta sección describimos brevemente dos tipos de precondicionadores utilizados para mejorar la convergencia del algoritmo del gradiente conjugado: las inversas aproximadas factorizadas y la factorizacion incompleta de Cholesky.

### 2.1. Precondicionamiento con inversas aproximadas factorizadas

Buscamos un precondicionador que mejore la resolución del sistema de ecuaciones lineales $(M + \varepsilon N)\, x = b$, con matriz variable SPD usando el gradiente conjugado, siguiendo un camino similar al propuesto por Benzi [5] para el sistema $(M + \varepsilon I)\, x = b$. El algoritmo SAINV nos da una inversa aproximada factorizada de $M$,

$$M^{-1} \approx \tilde{Z}\tilde{D}^{-1}\tilde{Z}^T = P^{-1}$$

Entonces consideramos un precondicionador para $A_\varepsilon = M + \varepsilon N$ de la forma,

$$P_\varepsilon^{-1} = \tilde{Z}\left(\tilde{D} + \varepsilon E\right)^{-1}\tilde{Z}^T, \tag{2}$$

donde $E$ es la matriz simétrica a calcular. Además debe ser de fácil obtención tal que $\left(\tilde{D} + \varepsilon E\right)$ sea SPD y que los productos matriz-vector a realizar en el algoritmo PCG, con $P_\varepsilon^{-1}$, no tengan un coste computacional alto. Para definirlo y, considerando que $P^{-1} = ZD^{-1}Z^T$ es la matriz inversa exacta, calculamos la diferencia,

$$P_\varepsilon - A_\varepsilon = Z^{-T}\left(D + \varepsilon E\right)Z^{-1} - \left(M + \varepsilon N\right) = \varepsilon\left(Z^{-T}EZ^{-1} - N\right). \tag{3}$$

Si tomamos $E = Z^T N Z$, obtendremos el precondicionador ideal $P_\varepsilon^{-1} = A_\varepsilon^{-1}$. Evidentemente, no es una elección práctica puesto que no conocemos la matriz $Z$ exacta, solo conocemos la aproximada $\tilde{Z}$. Sin embargo, este resultado sugiere el uso de la siguiente matriz $E$ en $P_\varepsilon^{-1}$,

$$E = \tilde{Z}^T N \tilde{Z} \tag{4}$$

tal que $E$ satisface las condiciones necesarias anteriores. En lugar de comenzar por una inversa aproximada de $M$, correspondiente a $\varepsilon = 0$ en $A_\varepsilon = M + \varepsilon N$, podemos obtener

una inversa aproximada de $A_{\varepsilon_0} = M + \varepsilon_0 N$. Las matrices subsecuentes para los valores correspondientes de $\varepsilon$ deben escribirse como $A_\varepsilon = M + \varepsilon N = A_{\varepsilon_0} + \Delta\varepsilon N$, donde $\Delta\varepsilon = \varepsilon - \varepsilon_0$. Como $\varepsilon$ es siempre una cantidad positiva, la matriz $A_\varepsilon$ obviamente es definida positiva aún cuando $\Delta\varepsilon$ sea negativo. Así, tenemos

$$A_{\varepsilon_0}^{-1} \approx \tilde{Z}\tilde{D}^{-1}\tilde{Z}^T = P_{\varepsilon_0}^{-1}$$

y

$$P_\varepsilon^{-1} = \tilde{Z}\left(\tilde{D} + \Delta\varepsilon E\right)^{-1}\tilde{Z}^T,$$

con $E = \tilde{Z}^T N \tilde{Z}$ siendo la misma que antes y con las mismas condiciones. Una opción para definir la matriz $E$ con estos requerimientos es tomar una aproximación de $\tilde{Z}$ a la que denotamos por $\tilde{Z}_k$, con $k > 1$, que se obtiene seleccionando las $k-1$ diagonales superiores de $\tilde{Z}$, y para $N$, la aproximación $N_h$, considerando su diagonal principal si $h = 1$ y también las $h-1$ diagonales secundarias si $h > 1$. Así,

$$E_{h,k} = \tilde{Z}_k^T N_h \tilde{Z}_k \tag{5}$$

En la práctica, para no aumentar el coste por iteración del PCG, los pares $h = 1$, $k = 2$ y $h = 2$, $k = 1$, nos llevan a las matrices tridiagonales $E_{1,2}$ y $E_{2,1}$, respectivamente, lo que parece ser la mejor elección. Puede obtenerse un precondicionador más simple con $h = 1$, $k = 1$, que nos lleva a la matriz diagonal $E_{1,1}$.

### 2.2. La factorización incompleta de Cholesky

Generalizaremos la factorización incompleta propuesta por Meurant [6] para el caso de matrices $A_\varepsilon = M + \varepsilon D$, con $D$ diagonal, a matrices $A_\varepsilon = M + \varepsilon N$, con $M$ y $N$ dos $n \times n$ matrices SPD. Podemos escribir $A_\varepsilon$ como sigue,

$$A_\varepsilon = (m_i j) + \varepsilon\,(n_i j) = \begin{pmatrix} m_{11} + \varepsilon n_{11} & (f_{1M} + \varepsilon f_{1N})^T \\ f_{1M} + \varepsilon f_{1N} & M_2 + \varepsilon N_2 \end{pmatrix}$$

donde $f_{1M}, f_{1N}$ representa $(n-1) \times 1$ matrices columnas y $M_2, N_2$, $(n-1) \times (n-1)$ matrices.
$A_\varepsilon$ se factoriza como sigue,

$$A_\varepsilon = L_1 Z_1 L_1^T$$

donde,

$$L_1 = \begin{pmatrix} m_{11} + \varepsilon n_{11} & \mathbf{0} \\ l_{1M} + \varepsilon l_{1N} & \mathbf{I} \end{pmatrix} \quad ; \quad Z_1 = \begin{pmatrix} (m_{11} + \varepsilon n_{11})^{-1} & \mathbf{0} \\ \mathbf{0} & C_2 \end{pmatrix}$$

con $l_{1M} = f_{1M}$ y $l_{1N} = f_{1N}$.
Entonces, identificando término a término, obtenemos para la matriz $C_2$,

$$C_2 = M_2 + \varepsilon N_2 - \frac{1}{m_{11} + \varepsilon n_{11}}\,(l_{1M} + \varepsilon l_{1N})\,(l_{1M} + \varepsilon l_{1N})^T \tag{6}$$

Eliminando los productos por el parámetro $\varepsilon$ para obtener el precondicionador de forma recursiva, tenemos,

$$C_2 = \varepsilon N_2 + M_2 - \frac{1}{m_{11}} l_{1M} l_{1M}^T \tag{7}$$

Por lo que las entradas de $C_2$ se calculan añadiendo $\varepsilon N_2$ a lo que obtendríamos de la factorización incompleta de $M$.
De las entradas de la diagonal de $N_2$ podemos obtener otra simplificación en lugar de la matriz total $N_2$,

$$C_2 = \varepsilon D_2 + M_2 - \frac{1}{m_{11}} l_{1M} l_{1M}^T \tag{8}$$

y así, en forma matricial,

$$C_2 = \varepsilon N_2 + \begin{pmatrix} m_{22}^{(2)} & f_{2M}^T \\ f_{2M} & M_3 \end{pmatrix} = \begin{pmatrix} m_{22}^{(2)} + \varepsilon n_{22} & (f_{2M} + \varepsilon l_{2N}) \\ f_{2M} + \varepsilon l_{2N} & M_3 + \varepsilon N_3 \end{pmatrix}$$

De esta forma, después de obtener todas las matrices $C_i$, la factorización incompleta de $A_\varepsilon$ es,

$$A_\varepsilon \approx L_1 Z_1 L_1^T = L_1 L_2 Z_2 L_2^T L_1^T = (L_1 L_2 \cdots L_n)\, Z_n\, (L_1 L_2 \cdots L_n)^T \tag{9}$$

siendo $Z_n$ la matriz diagonal cuyas entradas son de la forma $\left(m_{ii}^{(i)} + \varepsilon n_{ii}\right)^{-1}$. Las entradas de la diagonal de la matriz triangular inferior $L_1 L_2 \cdots L_n$ son $m_{ii}^{(i)} + \varepsilon n_{ii}$. Las respectivas columnas debajo de la diagonal principal se definen por $(n-i) \times 1$ matrices $l_{jM} + \varepsilon l_{jN}$.

## 3. EXPERIMENTOS NUMÉRICOS

En esta sección presentamos los resultados obtenidos utilizando el Gradiente Conjugado (CG) con los diferentes precondicionadores propuestos para resolver los sistemas lineales de ecuaciones que se obtienen en la discretización de dos problemas en derivadas parciales. Se incluyen asimismo, las tablas que muestran el efecto de diferentes técnicas de reordenación cuando se aplican estos precondicionadores. Todos los experimentos se ejecutaron en un XEON Precision 530 con Fortran de Doble Precisión. Siempre comenzamos la resolución desde el vector nulo e interrumpimos la ejecución si $\|r_k\|_2 \leq 10^{-10} \|r_0\|_2$ o si el número de iteraciones es mayor que 5000. Los resultados de los diferentes problemas se presentan en tablas para un amplio rango de valores de $\varepsilon$, incluyendo el tiempo de reordenación en segundos, el número de iteraciones y los tiempos de convergencia en cada caso. En las tablas, ICHOL$_D$, ICHOL$_N$, SAINV$_{11}$, SAINV$_{12}$ y SAINV$_{21}$ representan los precondicionadores obtenidos con los procedimientos descritos en la sección 2, siendo la tolerancia $\delta = 0{,}1$ para todos los experimentos.

### 3.1. Ejemplo 1: Vthcond

Se trata de un problema de transferencia de calor en 2-D modelado con una ecuación en derivadas parciales parabólica,

$$\frac{\partial u}{\partial t} - (c + \delta c)\Delta u = f \tag{10}$$

$$u = u_d \qquad en \quad \Gamma \times (0, T] \tag{11}$$

$$u(x, 0) = u^0 \qquad en \quad \Omega \tag{12}$$

donde $u$ es la temperatura, $c$ la conductividad térmica, $\delta c$ es una perturbación en $c$ y $f$ una fuente. Hemos resuelto este problema para un dominio en 2-D definido por $A(0, 0)$, $B(3, 0)$, $C(3, 3)$, $D(2, 3)$, $E(2, 2)$ y $F(0, 2)$. Discretizamos en el espacio en diferencias finitas con un intervalo $h$ y un esquema de tiempo implícito con un paso de tiempo $k$. Entonces obtenemos,

$$\left(\frac{1}{k}I + \frac{c}{h^2}R + \frac{\delta c}{h^2}R\right) u^{n+1} = \frac{u^n}{k} + f^{n+1}$$

Si definimos

$$M = \frac{1}{k}I + \frac{c}{h^2}R, \qquad N = \frac{1}{h^2}R,$$

la matriz del problema es,

$$A_{\delta c} = M + \delta c N$$

Por tanto en este experimento $\varepsilon$ se identifica con $\delta c$. El tamaño del intervalo es $h = 0{,}02$, el paso de tiempo $k = 0{,}001$, la fuente $f = 1$ y $u_d = u^0 = 0$. Hemos trabajado con un sistema lineal correspondiente a un paso de tiempo intermedio del proceso con 17201 incógnitas y 85409 entradas no nulas, cambiando la perturbacion de $c = 0{,}1$ de $10^{-6}$ a $10^6$. Las tablas 1 y 2 muestran el comportamiento de los precondicionadores propuestos para este problema. En este tipo de matrices con reducido número de entradas no nulas (máximo de cinco elementos distintos de cero por fila), el coste computacional de la factorización incompleta de Cholesky es bajo. El efecto se muestra en la tabla 1. Para $\varepsilon < 1$,todas las estrategias presentan un comportamiento similar para los diferentes precondicionadores. Para $\varepsilon \geq 1$ el Full-ICHOL proporciona los mejores resultados de convergencia. Solamente para los valores de $\varepsilon$ comprendidos entre 1 y 10 $\text{ICHOL}_N$ puede competir con esta alternativa. Otro resultado destacable es que $\text{ICHOL}_D$ presenta malos resultados para $\varepsilon \geq 1$, incluso peores que $\text{ICHOL}(A_{\varepsilon_0})$, lo que se puede explicar ya que añadir $\varepsilon D$ con $\varepsilon \geq 1$ a la factorización incompleta de Cholesky de $M$ puede suponer la pérdida de la propiedad de diagonal dominante de la matriz precondicionada, que se traduce en una disminución en la calidad del precondicionador. En la tabla 2 se observa que para las variantes derivadas de la inversa aproximada es el precondicionador $\text{SAINV}_{11}$ el que ofrece los mejores resultados. Hemos seleccionado $\text{ICHOL}_D$ y $\text{SAINV}_{12}$, para representar ambas familias de precondicionadores en el estudio del efecto de la reordenación. En las tablas 3 y 4 observamos que la reordenación nos lleva a una disminución del coste computacional para valores de $\varepsilon$ mayores que 10.

| $\varepsilon$ | | Full-ICHOL | ICHOL$_D$ | ICHOL$_N$ | ICHOL$\left(A_{\varepsilon_0}\right)$ |
|---|---|---|---|---|---|
| 0 | n$^o$Iter. | 6 | – | – | – |
| | t(s) | 0.03 | – | – | – |
| $10^{-6}$ | n$^o$Iter. | 6 | 6 | 6 | 6 |
| | t(s) | 0.04 | 0.03 | 0.03 | 0.03 |
| $10^{-5}$ | n$^o$Iter. | 6 | 6 | 6 | 6 |
| | t(s) | 0.04 | 0.03 | 0.03 | 0.03 |
| $10^{-4}$ | n$^o$Iter. | 6 | 6 | 6 | 6 |
| | t(s) | 0.03 | 0.03 | 0.03 | 0.03 |
| $10^{-3}$ | n$^o$Iter. | 6 | 6 | 6 | 6 |
| | t(s) | 0.03 | 0.03 | 0.03 | 0.03 |
| $10^{-2}$ | n$^o$Iter. | 6 | 6 | 6 | 6 |
| | t(s) | 0.03 | 0.03 | 0.03 | 0.03 |
| $10^{-1}$ | n$^o$Iter. | 7 | 7 | 7 | 7 |
| | t(s) | 0.04 | **0.03** | 0.04 | 0.04 |
| 1 | n$^o$Iter. | 8 | 15 | 9 | 11 |
| | t(s) | **0.04** | 0.07 | **0.04** | 0.05 |
| 10 | n$^o$Iter. | 17 | 49 | 19 | 31 |
| | t(s) | **0.08** | 0.19 | **0.08** | 0.13 |
| $10^2$ | n$^o$Iter. | 48 | 159 | 57 | 95 |
| | t(s) | **0.21** | 0.62 | 0.23 | 0.40 |
| $10^3$ | n$^o$Iter. | 118 | 395 | 141 | 234 |
| | t(s) | **0.48** | 1.53 | 0.54 | 0.95 |
| $10^4$ | n$^o$Iter. | 152 | 512 | 181 | 302 |
| | t(s) | **0.64** | 1.96 | 0.70 | 1.23 |
| $10^5$ | n$^o$Iter. | 158 | 523 | 188 | 313 |
| | t(s) | **0.66** | 2.04 | 0.72 | 1.27 |
| $10^6$ | n$^o$Iter. | 159 | 535 | 189 | 314 |
| | t(s) | **0.65** | 2.06 | 0.73 | 1.28 |

Cuadro 1: Ejemplo 1: Vthcond. Numero de iteraciones y coste computacional (en s.) del Gradiente Conjugado con diferentes precondicionadores ICHOL

| $\varepsilon$ | | full SAINV | SAINV-span | SAINV$_{11}$ | SAINV$_{12}$ | SAINV$_{21}$ | SAINV$\left(A_{\varepsilon_0}\right)$ |
|---|---|---|---|---|---|---|---|
| 0 | n$^o$Iter. | 8 | – | – | – | – | – |
| | t(seg.) | 27.24 | – | – | – | – | – |
| $10^{-6}$ | n$^o$Iter. | 8 | 8 | 8 | 8 | 8 | 8 |
| | t(seg.) | 24.58 | 0.06 | **0.04** | **0.04** | 0.05 | **0.04** |
| $10^{-5}$ | n$^o$Iter. | 8 | 8 | 8 | 8 | 8 | 8 |
| | t(seg.) | 27.16 | 0.06 | **0.03** | 0.04 | 0.04 | 0.04 |
| $10^{-4}$ | n$^o$Iter. | 8 | 8 | 8 | 8 | 8 | 8 |
| | t(seg.) | 26.93 | 0.07 | 0.04 | 0.04 | 0.04 | **0.03** |
| $10^{-3}$ | n$^o$Iter. | 8 | 8 | 8 | 8 | 8 | 8 |
| | t(seg.) | 27.06 | 0.09 | **0.04** | 0.05 | **0.04** | **0.04** |
| $10^{-2}$ | n$^o$Iter. | 8 | 8 | 8 | 9 | 8 | 8 |
| | t(seg.) | 26.99 | 0.08 | **0.04** | 0.05 | 0.05 | **0.04** |
| $10^{0-1}$ | n$^o$Iter. | 9 | 8 | 9 | 9 | 9 | 9 |
| | t(seg.) | 27.06 | 0.07 | **0.04** | 0.06 | 0.05 | **0.04** |
| $10^0$ | n$^o$Iter. | 12 | 9 | 13 | 14 | 12 | 13 |
| | t(seg.) | 27.03 | 0.07 | 0.06 | 0.07 | 0.06 | **0.05** |
| $10^1$ | n$^o$Iter. | 27 | >5000 | 32 | 40 | 30 | 32 |
| | t(seg.) | 26.03 | – | **0.13** | 0.20 | 0.16 | 0.14 |
| $10^2$ | n$^o$Iter. | 82 | >5000 | 99 | 125 | 90 | 99 |
| | t(seg.) | 27.35 | – | **0.38** | 0.60 | 0.43 | 0.43 |
| $10^3$ | n$^o$Iter. | 203 | >5000 | 244 | 312 | 221 | 243 |
| | t(seg.) | 27.63 | – | **0.95** | 1.49 | 1.05 | 0.99 |
| $10^4$ | n$^o$Iter. | 263 | >5000 | 315 | 403 | 286 | 315 |
| | t(seg.) | 27.88 | – | **1.21** | 1.90 | 1.36 | 1.26 |
| $10^5$ | n$^o$Iter. | 272 | >5000 | 326 | 417 | 297 | 326 |
| | t(seg.) | 27.86 | – | **1.27** | 1.98 | 1.41 | **1.27** |
| $10^6$ | n$^o$Iter. | 273 | >5000 | 328 | 420 | 298 | 328 |
| | t(seg.) | 28.21 | – | **1.27** | 1.99 | 1.41 | 1.29 |

Cuadro 2: Ejemplo 1: Vthcond. Numero de iteraciones y coste computacional (en s.) del Gradiente Conjugado con diferentes precondicionadores SAINV

### 3.2. Ejemplo 2: Windfield

Este modelo de viento [1, 2] está basado en la ecuación de continuidad para un fluído incompresible con densidad del aire constante en el dominio $\Omega$ y condiciones de contorno

| $\varepsilon$ | | Orden Inicial | MN (1,97$s$) | RCM (0,03$s$) | MC (0,02$s$) |
|---|---|---|---|---|---|
| 0 | n$^o$Iter. | 6 | 6 | 6 | 9 |
| | t(s) | 0.04 | **0.03** | **0.03** | 0.04 |
| $10^{-6}$ | n$^o$Iter. | 6 | 6 | 6 | 9 |
| | t(s) | 0.03 | 0.03 | **0.02** | 0.04 |
| $10^{-5}$ | n$^o$Iter. | 6 | 6 | 6 | 9 |
| | t(s) | **0.03** | **0.03** | **0.03** | 0.04 |
| $10^{-4}$ | n$^o$Iter. | 6 | 6 | 6 | 9 |
| | t(s) | **0.03** | **0.03** | **0.03** | 0.04 |
| $10^{-3}$ | n$^o$Iter. | 6 | 6 | 6 | 9 |
| | t(s) | **0.03** | **0.03** | **0.03** | 0.04 |
| $10^{-2}$ | n$^o$Iter. | 6 | 6 | 6 | 9 |
| | t(s) | **0.03** | **0.03** | **0.03** | 0.05 |
| $10^{-1}$ | n$^o$Iter. | 7 | 7 | 7 | 9 |
| | t(s) | 0.03 | 0.03 | 0.03 | 0.03 |
| 1 | n$^o$Iter. | 15 | 15 | 15 | 15 |
| | t(s) | 0.07 | **0.06** | 0.07 | **0.06** |
| 10 | n$^o$Iter. | 49 | 49 | 49 | 49 |
| | t(s) | **0.19** | 0.20 | **0.19** | 0.20 |
| $10^2$ | n$^o$Iter. | 159 | 159 | 159 | 159 |
| | t(s) | 0.62 | 0.63 | **0.59** | 0.62 |
| $10^3$ | n$^o$Iter. | 395 | 395 | 395 | 395 |
| | t(s) | 1.53 | 1.52 | **1.46** | 1.52 |
| $10^4$ | n$^o$Iter. | 512 | 512 | 512 | 512 |
| | t(s) | 1.96 | 1.97 | **1.89** | 1.97 |
| $10^5$ | n$^o$Iter. | 523 | 529 | 529 | 529 |
| | t(s) | 2.04 | 2.03 | **1.96** | 2.04 |
| $10^6$ | n$^o$Iter. | 535 | 535 | 535 | 535 |
| | t(s) | 2.06 | 2.04 | **1.97** | 2.06 |

Cuadro 3: Ejemplo 1: Vthcond. Número de iteraciones y coste computacional (en s.) del Gradiente Conjugado con precondicionador ICHOL$_D$ para diferentes reordenaciones

| $\varepsilon$ | | Orden Inicial | MN (1,97$s$) | RCM (0,03$s$) | MC (0,02$s$) |
|---|---|---|---|---|---|
| 0 | n$^o$Iter. | 8 | 8 | 8 | 9 |
| | t(s) | 26.39 | 30.53 | **26.24** | 36.58 |
| $10^{-6}$ | n$^o$Iter. | 8 | 8 | 8 | 9 |
| | t(s) | **0.04** | **0.04** | **0.04** | 0.05 |
| $10^{-5}$ | n$^o$Iter. | 8 | 8 | 8 | 9 |
| | t(s) | **0.04** | **0.04** | **0.04** | 0.05 |
| $10^{-4}$ | n$^o$Iter. | 8 | 8 | 8 | 9 |
| | t(s) | **0.04** | 0.05 | **0.04** | 0.06 |
| $10^{-3}$ | n$^o$Iter. | 8 | 8 | 8 | 9 |
| | t(s) | 0.05 | 0.05 | 0.05 | 0.05 |
| $10^{-2}$ | n$^o$Iter. | 9 | 9 | 8 | 9 |
| | t(s) | **0.05** | 0.06 | **0.05** | **0.05** |
| $10^{-1}$ | n$^o$Iter. | 9 | 9 | 9 | 9 |
| | t(s) | 0.06 | **0.05** | **0.05** | **0.05** |
| 1 | n$^o$Iter. | 14 | 14 | 13 | 13 |
| | t(s) | 0.07 | 0.07 | 0.07 | 0.07 |
| 10 | n$^o$Iter. | 40 | 40 | 33 | 34 |
| | t(s) | 0.20 | 0.20 | **0.17** | **0.17** |
| $10^2$ | n$^o$Iter. | 125 | 126 | 100 | 103 |
| | t(s) | 0.60 | 0.60 | **0.48** | 0.50 |
| $10^3$ | n$^o$Iter. | 312 | 312 | 245 | 255 |
| | t(s) | 1.49 | 1.48 | **1.16** | 1.22 |
| $10^4$ | n$^o$Iter. | 403 | 403 | 318 | 330 |
| | t(s) | 1.90 | 1.90 | **1.50** | 1.58 |
| $10^5$ | n$^o$Iter. | 417 | 417 | 331 | 343 |
| | t(s) | 1.98 | 1.96 | **1.57** | 1.64 |
| $10^6$ | n$^o$Iter. | 420 | 420 | 342 | 349 |
| | t(s) | 1.99 | 1.98 | **1.62** | 1.67 |

Cuadro 4: Ejemplo 1: Vthcond. Número de iteraciones y coste computacional (en s.) del Gradiente Conjugado con precondicionador SAINV$_{12}$ para diferentes reordenaciones

en el terreno $\Gamma_b$,

$$\vec{\nabla} \cdot \vec{u} \;=\; 0 \qquad en \quad \Omega \tag{13}$$

$$\vec{n} \cdot \vec{u} \;=\; 0 \qquad en \quad \Gamma_b \tag{14}$$

El problema se formula como un problema de mínimos cuadrados en $\Omega$, para ajustar $\vec{u}(\widetilde{u}, \widetilde{v}, \widetilde{w})$

$$E(\vec{u}) = \int_\Omega \left[\alpha_1^2 \left((\widetilde{u} - u_0)^2 + (\widetilde{v} - v_0)^2\right) + \alpha_2^2 \left(\widetilde{w} - w_0\right)^2\right] \, d\Omega \tag{15}$$

donde el tiempo interpolado $\vec{v}_0 = (u_0, v_0, w_0)$ se obtiene de medidas experimentales y consideraciones físicas, y $\alpha_1$, $\alpha_2$ representan el módulo de precisión de Gauss. En la práctica utilizamos el llamado parámetro de estabilidad para el modelo de viento,

$$\alpha = \frac{\alpha_1}{\alpha_2} \tag{16}$$

dado que el mínimo de la funcional dada por (15) es el mismo que si la dividimos por $\alpha_2^2$. Aplicando cálculo variacional resulta el siguiente problema elíptico,

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \alpha^2 \frac{\partial^2 \phi}{\partial z^2} = -2\alpha_1^2 \left(\frac{\partial u_0}{\partial x} + \frac{\partial v_0}{\partial y} + \frac{\partial w_0}{\partial z}\right) \quad en \;\; \Omega \tag{17}$$

Con condiciones de contorno,

$$\phi \;=\; 0 \quad en \quad \Gamma_a \tag{18}$$

$$\vec{n} \cdot T \, \vec{\nabla} \mu \;=\; -\vec{n} \cdot \vec{v}_0 \quad en \;\; \Gamma_b \tag{19}$$

con $T = diag\left[\frac{1}{2\alpha_1^2}, \frac{1}{2\alpha_1^2}, \frac{1}{2\alpha_2^2}\right]$. Obsérvese que en este experimento $\varepsilon = \alpha^2$. Este ejemplo es una simulación del viento en la región de La Palma. Las matrices $M$ y $N$ provienen de la modelización numérica con el modelo de masa consistente anterior para el ajuste de un campo de viento. Hemos usado una malla que produce un sistema lineal de 98999 ecuaciones con 1374089 entradas no nulas, y $M$, $N$ matrices SPD. Las tablas 5 y 6 muestran el comportamiento de los diferentes precondicionadores. En la tabla 5 se observa que el precondicionador $\text{ICHOL}_N$ presenta los mejores resultados para los basados en la factorización incompleta de Cholesky y que para los precondicionadores basados en la inversa aproximada, se observa una convergencia extremadamente lenta para valores de $\varepsilon \geq 10^4$. Es notoria la reducción del coste computacional obtenido con los precondicionadores $\text{SAINV}_{nk}$ respecto a $\text{SAINV}(A_{\varepsilon_0})$ (ver tabla 6). En las tablas 7 y 8 presentamos los resultados obtenidos para $\text{ICHOL}_N$ y $\text{SAINV}_{11}$. En todos los casos, la reordenación con los algoritmos MN y RCM mejoran la convergencia del PCG. Sin embargo, el algoritmo MC no produce ninguna mejora en la ejecución del método iterativo precondicionado.

## 4. CONCLUSIONES

- Por lo que a los precondicionadores se refiere, podemos concluir que para pequeños valores de $\varepsilon$ no parece rentable la construcción de un precondicionador variable,

| $\varepsilon$ | | Full-ICHOL | ICHOL$_D$ | ICHOL$_N$ | ICHOL$\left(A_{\varepsilon_0}\right)$ |
|---|---|---|---|---|---|
| 0 | n$^o$Iter. | 201 | – | – | – |
| | t(s) | 18.17 | – | – | – |
| $10^{-6}$ | n$^o$Iter. | 201 | 201 | 201 | 201 |
| | t(s) | 17.95 | 17.22 | **17.21** | 18.18 |
| $10^{-5}$ | n$^o$Iter. | 201 | 201 | 201 | 201 |
| | t(s) | 18.06 | 17.22 | **17.21** | 18.39 |
| $10^{-4}$ | n$^o$Iter. | 201 | 201 | 201 | 201 |
| | t(s) | 17.39 | **17.22** | **17.22** | 18.39 |
| $10^{-3}$ | n$^o$Iter. | 200 | 201 | 200 | 201 |
| | t(s) | 18.11 | **15.60** | 17.13 | 18.37 |
| $10^{-2}$ | n$^o$Iter. | 189 | 191 | 189 | 188 |
| | t(s) | 17.18 | 16.32 | **16.20** | 17.09 |
| $10^{-1}$ | n$^o$Iter. | 151 | 157 | 155 | 225 |
| | t(s) | 13.33 | 13.49 | **13.32** | 19.12 |
| 1 | n$^o$Iter. | 132 | 211 | 148 | 483 |
| | t(s) | **12.22** | 18.08 | 12.73 | 42.56 |
| 10 | n$^o$Iter. | 236 | 540 | 259 | 1350 |
| | t(s) | **21.11** | 46.04 | 22.15 | 119.56 |
| $10^2$ | n$^o$Iter. | >5000 | 1466 | 593 | 3973 |
| | t(s) | – | 124.71 | **50.44** | 355.98 |
| $10^3$ | n$^o$Iter. | >5000 | 3468 | 1269 | >5000 |
| | t(s) | – | 295.20 | **107.91** | – |

Cuadro 5: Ejemplo 2: Windfield. Número de iteraciones y coste computacional (en s.) del Gradiente Conjugado con diferentes precondicionadores ICHOL

| $\varepsilon$ | | Full-SAINV | SAINV-span | SAINV$_{11}$ | SAINV$_{12}$ | SAINV$_{21}$ | SAINV$\left(A_{\varepsilon_0}\right)$ |
|---|---|---|---|---|---|---|---|
| 0 | Iter. | 279 | – | – | – | – | – |
| | Time | 3627.00 | – | – | – | – | – |
| $10^{-6}$ | Iter. | 279 | 279 | 278 | 279 | 279 | 279 |
| | Time | 3648.14 | 36.48 | **18.99** | 20.48 | 20.52 | 20.31 |
| $10^{-5}$ | Iter. | 278 | 279 | 279 | 279 | 279 | 279 |
| | Time | 3643.49 | 34.74 | **19.12** | 20.44 | 20.51 | 20.13 |
| $10^{-4}$ | Iter. | 277 | 279 | 279 | 278 | 279 | 278 |
| | Time | 3662.09 | 36.47 | **19.13** | 20.38 | 20.51 | 19.30 |
| $10^{-3}$ | Iter. | 275 | 276 | 276 | 276 | 275 | 276 |
| | Time | 3660.47 | 36.22 | **18.89** | 20.55 | 20.22 | 20.11 |
| $10^{-2}$ | Iter. | 253 | 251 | 252 | 253 | 252 | 250 |
| | Time | 3631.91 | 32.47 | **17.31** | 18.58 | 18.54 | 18.23 |
| $10^{-1}$ | Iter. | 210 | >5000 | 224 | 223 | 219 | 292 |
| | Time | 3675.64 | – | **15.36** | 16.36 | 16.12 | 19.68 |
| $10^0$ | Iter. | 191 | >5000 | 224 | 223 | 219 | 292 |
| | Time | 3649.99 | – | **15.36** | 16.36 | 16.12 | 19.68 |
| $10^1$ | Iter. | 257 | >5000 | 594 | 561 | 580 | 1626 |
| | Time | 3731.01 | – | **40.55** | 40.96 | 42.57 | 117.09 |
| $10^2$ | Iter. | 548 | >5000 | 1739 | 1687 | 1696 | 4763 |
| | Time | 3939.68 | – | **118.55** | 122.98 | 124.19 | 339.72 |
| $10^3$ | Iter. | 1292 | >5000 | 4075 | >5000 | 4071 | >5000 |
| | Time | 4150.38 | – | **278.11** | – | 297.14 | – |

Cuadro 6: Ejemplo 2: Windfield. Número de iteraciones y coste computacional (en s.) del Gradiente Conjugado con diferentes precondicionadores SAINV

bastaría con aplicar a los sucesivos sistemas el precondicionador SAINV($A_{\varepsilon_0}$) o el ICHOL($A_{\varepsilon_0}$). Para los valores de $\varepsilon > 1$, los precondicionadores SAINV$_{11}$ o ICHOL$_N$, presentan notables ventajas sobre los basados en $A_{\varepsilon_0}$, debido a su bajo coste computacional y obviamente sobre los precondicionadores Full-SAINV o Full-ICHOL.

- El efecto de la reordenación es beneficioso para resolver sistemas lineales variables usando PCG con ambos tipos de precondicionadores, la factorización incompleta de Cholesky y la inversa aproximada factorizada. Para sistemas de ecuaciones mayores, el coste computacional fue reducido considerablemente para todos los valores del parámetro $\varepsilon$, mientras que para sistemas lineales pequeños, la convergencia mejora

| $\varepsilon$ | | Orden Inicial | MN (69,43$s$) | RCM (0,70$s$) | MC (0,45$s$) |
|---|---|---|---|---|---|
| 0 | n$^o$Iter. | 201 | 158 | 175 | 223 |
| | t(s) | 18.00 | **12.86** | 13.84 | 24.01 |
| $10^{-6}$ | n$^o$Iter. | 201 | 159 | 175 | 223 |
| | t(s) | 17.21 | **12.53** | 13.46 | 22.51 |
| $10^{-5}$ | n$^o$Iter. | 201 | 159 | 175 | 222 |
| | t(s) | 17.21 | **12.54** | 13.45 | 22.41 |
| $10^{-4}$ | n$^o$Iter. | 201 | 158 | 176 | 223 |
| | t(s) | 17.22 | **12.47** | 13.53 | 22.51 |
| $10^{-3}$ | n$^o$Iter. | 200 | 157 | 174 | 220 |
| | t(s) | 17.13 | **12.37** | 13.38 | 22.21 |
| $10^{-2}$ | n$^o$Iter. | 189 | 143 | 160 | 207 |
| | t(s) | 16.20 | 11.31 | 12.30 | 20.90 |
| $10^{-1}$ | n$^o$Iter. | 155 | 116 | 131 | 170 |
| | t(s) | 13.32 | **9.19** | 10.12 | 17.21 |
| 1 | n$^o$Iter. | 148 | 123 | 129 | 160 |
| | t(s) | 12.73 | **9.73** | 9.97 | 16.20 |
| 10 | n$^o$Iter. | 259 | 227 | 240 | 277 |
| | t(s) | 22.15 | **17.88** | 18.38 | 27.91 |
| $10^2$ | n$^o$Iter. | 593 | 533 | 536 | 632 |
| | t(s) | 50.44 | 41.61 | **40.73** | 63.43 |
| $10^3$ | n$^o$Iter. | 1269 | 1212 | 1177 | 1395 |
| | t(s) | 107.91 | 94.45 | **89.23** | 139.79 |

Cuadro 7: Ejemplo 2: Windfield. Número de iteraciones y coste computacional (en s.) del Gradiente Conjugado con precondicionador ICHOL$_N$ para diferentes reordenaciones

| $\varepsilon$ | | Orden Inicial | MN (69,43$s$) | RCM (0,70$s$) | MC (0,45$s$) |
|---|---|---|---|---|---|
| 0 | n$^o$Iter. | 279 | 264 | 273 | 263 |
| | t(s) | 3450.68 | 3092.93 | **2757.27** | 4626.74 |
| $10^{-6}$ | n$^o$Iter. | 278 | 265 | 274 | 263 |
| | t(s) | 18.99 | 16.35 | **16.25** | 20.13 |
| $10^{-5}$ | n$^o$Iter. | 279 | 264 | 274 | 263 |
| | t(s) | 19.12 | 16.29 | **16.22** | 20.15 |
| $10^{-4}$ | n$^o$Iter. | 279 | 264 | 273 | 263 |
| | t(s) | 19.13 | 16.27 | **16.18** | 20.15 |
| $10^{-3}$ | n$^o$Iter. | 276 | 262 | 272 | 260 |
| | t(s) | 18.89 | 16.16 | **16.11** | 19.93 |
| $10^{-2}$ | n$^o$Iter. | 252 | 236 | 247 | 240 |
| | t(s) | 17.31 | **14.61** | 14.66 | 28.40 |
| $10^{-1}$ | n$^o$Iter. | 224 | 208 | 212 | 219 |
| | t(s) | 15.36 | 12.89 | **12.58** | 16.81 |
| 1 | n$^o$Iter. | 275 | 261 | 265 | 270 |
| | t(s) | 18.85 | 16.15 | **15.71** | 20.69 |
| 10 | n$^o$Iter. | 594 | 520 | 549 | 553 |
| | t(s) | 40.55 | **32.04** | 32.32 | 42.24 |
| $10^2$ | n$^o$Iter. | 1739 | 1512 | 1589 | 1623 |
| | t(s) | 118.55 | 92.96 | **92.23** | 123.67 |
| $10^3$ | n$^o$Iter. | 4075 | 3701 | 3756 | 3924 |
| | t(s) | 278.11 | 227.79 | **220.51** | 294.04 |

Cuadro 8: Ejemplo 3: Windfield. Número de iteraciones y coste computacional (en s.) del Gradiente Conjugado con precondicionador SAINV$_{11}$ para diferentes reordenaciones

sólo para altos valores de $\varepsilon$. Para sistemas grandes los algoritmos MN y RCM dan los mejores resultados. La reordenación usando MC presentó el mejor comportamiento en sistemas pequeños cuando se usó la inversa aproximada factorizada, pero no mejoraba la convergencia del CG precondicionado con la factorización incompleta de Cholesky o, en general, cuando el orden de la matriz aumentaba.

## REFERENCIAS

[1] G. Montero, R. Montenegro, J.M. Escobar. A 3-D diagnostic model for wind field adjustment, *J. Wind Eng. Ind. Aer.*, Vol. **74,76**, pp. 249–261, (1998).

[2] G. Montero, E. Rodríguez, R. Montenegro, J.M. Escobar, J.M. González- Yuste. Genetic algorithms for an improved parameter estimation with local refinenent of tetrahedral meshes in a wind model, *Adv. Engng. Soft.*, Vol. **36**, pp. 3–10, (2005).

[3] M.R. Hestenes y E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems, *Jour. Res. Nat. Bur. Sta.*, Vol. **49,6**, pp. 409–436, (1952).

[4] M. Benzi. Preconditioning techniques for large linear systems: a survey, *J. Comput Physics*, Vol. **182**, pp. 418–477, (2002).

[5] M. Benzi and D. Bertaccini. Approximate inverse preconditioning for shifted linear systems, *BIT Num. Math.*, Vol. **43**, pp. 231–244, (2003).

[6] G. Meurant. On the incomplete Cholesky decomposition of a class of perturbed matrices, *SIAM J. Sci. Comput.*, Vol. **23(2)**, pp. 419–429, (2001).

[7] M. Benzi, J.K. Cullum and M. Tůma,. Robust approximate inverse preconditioning for the conjugate gradient method, *SIAM J. Sci. Comput.*, Vol. **22**, pp. 1318–1332, (2000).

[8] I.S. Duff and G. Meurant. The effect of ordering on preconditioned conjugate gradients, *BIT*, Vol. **29**, pp. 635–657, (1989).

[9] M. Benzi, D.B. Szyld and A. Duin. Orderings for incomplete factorization preconditioning of nonsymmetric problems, *SIAM J. Sci. Comput.*, Vol. **20(5)**, pp. 1652–1670, (1999).

[10] L.C. Dutto. The Effect of Ordering on Preconditioned GMRES Algorithm, *Int. Jour. Num, Meth. Eng.*, Vol. **36**, pp. 457–497, (1993).

[11] M. Benzi and M. Tůma. Orderings for factorized sparse approximate inverse preconditioners, *SIAM J. Sci. Comput.*, Vol. **30(5)**, pp. 1851–1868, (2000).

[12] E. Florez, M.D. Garcia, L. Gonzalez and G. Montero. The Effect of Ordering on Sparse Approximate Inverse Preconditioners for Non-symmetric Problems, *Advances in Engineering Software*, Vol. **33**, pp 611–619, (2002).

[13] N.M. Nachtigal, S.C. Reddy and L.N. Trefethen. How Fast Are Nonsymmetric Matrix Iterations?, *SIAM J. Matr. Anal. Appl.*, Vol. **13, 3**, pp. 796–825, (1992).

[14] Y. Saad, *Iterative methods for sparse linear systems*, PWE Publishing Company, Boston, (1996).

[15] G. Montero, A. Suárez, E. Flórez and M.D. García, *Preconditioning shifter linear sistems arising in a win model*, *The Tenth Intern. Conf. on Civil, Structural and Environmental Engineering Computing, Roma 2005.*

[16] H. Sarmiento, A. Suárez and G. Montero, *Incomplete Factorization for Preconditioning Shifted Linear Systems*, *The Fifth International Conference on Engineering Computational Techonology, Las Palmas G.C. 2006.*

[17] E.H. Cuthill and J.M. Mckee, *Reducing the Bandwidth of Sparse Symmetric Matrices*, Brondon Press, *Proc. 24th National Conference of the Association for Computing Machinery, 1969, pp. 157–172.*

[18] A. George, *Computer Implementation of the Finite Element Method*, Report Stan CS-71-208, (1971).

[19] A. George and J.W. Liu,. The Evolution of the Minimum Degree Ordering Algorithms, *SIAM Rev.*, Vol. **31**, pp. 1–19, (1989).

[20] R.G. Grimes, D.J. Pierce and H.D. Simon. A new algorithm for finding a pseudoperipheral node in graph, *SIAM J. Matrix Anal Appl.*, Vol. **11(2)**, pp. 323-334, (1976).

[21] G.H. Paulino, I.F.M. Menezes, M. Gattas and S. Mukherjee. A New Algorithm for Finding a Pseudoperipheral Vertex or the Endpoints of a Pseudodiameter in a Grapf, *Num. Meth. Eng.*, Vol. **10**, pp. 913–926, (1994).