DESARROLLO DE UN CONJUNTO DE RUTINAS PARA LA RESOLUCION DE SISTEMAS CON MATRICES RALAS NO SIMETRICAS. COMPARACION DE VARIOS METODOS ITERATIVOS BASADOS EN EL DE GRADIENTES CONJUGADOS

Axel E. Larreteguy
Pablo M. Carrica
Laboratorio de Termohidráulica
CNEA - Centro Atómico Bariloche

Resumen

Frecuentemente los sistemas matriciales resultantes de las discretizaciones de ecuaciones por elementos finitos son no simétricos. La resolución de estos sistemas puede hacerse por métodos directos o iterativos. Los métodos directos resuelven el sistema mediante una factorización completa de la matriz en un número de operaciones conocido con anticipación para un dado tamaño de matriz. Los métodos iterativos se aproximan a la solución a partir de algún valor inicial semilla de la solución en un número de pasos no determinado a priori. La performance de estos métodos mejora notablemente con precondicionamiento.

En este trabajo se analizan los comportamientos de resolvedores directos e iterativos para matrices no simétricas con almacenamiento ralo tipo Gustaffson [1], y se muestran los métodos iterativos MCG (Modified Conjugate Gradient), BCG (BiConjugate Gradient), CGS (Conjugate Gradient Squared) y GMRES (Generalized Minimal RESidual), el precondicionador ILU (Incomplete Lower-Upper factorization) y los pre-precondicionadores Block Diagonal y Diagonal Scaling. Se realiza una comparación de la performance de cada método y se concluye que el método más económico en CPU y memoria es el CGS, siendo el más recomendable para matrices muy mal condicionadas el GMRES.

Abstract

Frequently the matrix systems resulting from the discretization equations arising from Finite Element Method are non symmetric. The direct solvers use a complete factorization of the matrix with a number of operations known in advance for a given matrix size. The iterative methods tend to the solution from an initial guess using a number of operations not known a priori. The performance of these methods is greatly improved when preconditioners are used.

In this work the behaviour of direct and iterative solvers for non symmetric matrices with Gustaffson [1] sparse storage is analysed. The results obtained by means of the iterative methods MCG, BCG, CGS and GMRES, the preconditioner ILU and the pre-preconditioners Block Diagonal and Diagonal Scaling are shown. The different methods are compared and it is concluded that the CGS method performed best, but GMRES is recommended for very ill-conditioned problems.

METODOS DIRECTOS DE RESOLUCION

Consideremos un sistema matricial del tipo:

$$A\,x = b \qquad (1)$$

donde A es no simétrica. Los métodos directos usualmente factorizan la matriz A en su forma LU:

$$A = L\,U \qquad (2)$$

donde L es una matriz triangular inferior y U es triangular superior, siendo diag(L) = I en general (I es la matriz identidad). Una vez obtenidas L y U por sustitución directa y retrosustitución obtenemos la solución.

Notemos que una vez obtenida la factorización (2), la resolución del mismo sistema con distintos términos independientes es inmediata.

Si la matriz A es rala, las matrices L y U generalmente no lo son, perdiendo las ventajas del almacenamiento ralo y obligando a un fuerte consumo de memoria. Cuando la matriz A es densa, los sistemas directos suelen ser una buena opción para resolver el problema.

Para los experimentos numéricos realizados en éste trabajo se utilizo la subrutina de resolución directa MA28 de Harwell [2].

METODOS ITERATIVOS DE RESOLUCION

En esta sección nos dedicaremos exclusivamente a los métodos del tipo de gradientes conjugados.

Si la matriz A es suficientemente rala y grande, resulta atractivo el uso de A como multiplicador unicamente. Los métodos de gradientes conjugados resuelven elegantemente el problema sin factorizar la matriz.

1) El método de gradientes conjugados modificado (MCG).

Sea una solución inicial x^0 del sistema (1) dada como semilla. El residuo inicial vale:

$$r^0 = b - A\,x^0 \qquad (3)$$

Si usamos A como multiplicador unicamente, y entonces trabajamos con polinomios en A, una forma general de expresar diferentes algoritmos será [3]:

$$x^{n+1} = x^n + \alpha_n\, p^n \qquad (4)$$

$$p^n = \theta_n(A)\, r^0 \qquad (5)$$

donde θ_n es un polinomio de grado n en A. Entonces, operando:

$$\begin{aligned} r^{n+1} &= b - A\,x^{n+1} = b - A\,x^n - \alpha\, A\,p^n \\ &= r^n - A\,\alpha_n\,\theta_n(A)\,r^0 \\ &= r^0 - A\left\{\alpha_n\theta_n(A) + \alpha_{n-1}\theta_{n-1}(A) + \ldots + \alpha_0\theta_0(A)\right\} r^0 \\ &= \phi_{n+1}(A)\,r^0 \end{aligned} \qquad (6)$$

donde se ve inmediatamente que $\phi_n(0) = 1$

Deseamos entonces hallar ϕ_n de forma que $| \phi_n(A)\, r^0 |$ sea mínimo. Para matrices simétricas y definidas positivas (MSDP) esto se logra con el método de gradientes conjugados.

Definamos el espacio Π_n^1 de la siguiente forma:

$$\Pi_n^1 = \left\{ \psi_n \mid \psi_n(0) = 1,\ \psi_n \text{ es un polinomio de grado} \leq n \right\} \quad (7)$$

entonces queremos encontrar $\phi_n \in \Pi_n^1$ tal que:

$$| \phi_n(A)\, r^0 | \leq | \psi_n(A)\, r^0 | \qquad \forall \psi_n \in \Pi_n^1 \quad (8)$$

para la norma (MSDP)

$$| r |^2 = r^T A^{-1} r \quad (9)$$

El siguiente método de gradientes conjugados obtiene ϕ_n que cumple (8) (n indica el número de iteración):

$$\begin{aligned} & p^n = r^n + \beta_n p^{n-1} \ ; \ p^{-1} = 0 \\ & \beta_n = \rho_n / \rho_{n-1} \quad ; \ \rho_n = r^{nT} r^n \\ & x^{n+1} = x^n + \alpha_n p^n \quad ; \ \alpha_n = \rho_n / \sigma_n \ ; \ \sigma_n = p^{nT} A\, p^n \\ & r^{n+1} = r^n - \alpha_n A\, p^n \end{aligned} \quad (10)$$

El método de gradientes conjugados fué propuesto por primera vez por Hestenes y Stieffel [4].

Puede verse que el método de gradientes conjugados se reduce a hallar x^n que verifique [5]:

$$\begin{cases} x^n \in K_n \\ A x^n - b \perp A^{j-1} r^0 \qquad j = 1,2,\ldots,n \end{cases} \quad (11)$$

donde $K_n = \text{Span} \left\{ r^0, Ar^0, \ldots, A^{n-1} r^0 \right\}$ es un subespacio de Krylov.

Los vectores de búsqueda son conjugados

$$p^{kT} A\, p^n = 0 \qquad k \neq n \quad (12)$$

y de ahí el nombre del método.

El método de gradientes conjugados exige que la matriz sea simétrica definida positiva. Para lograr esto ponemos :

$$M x = c \quad (13)$$

$$M = A^T A \ ; \ c = A^T b \quad (14)$$

El sistema a resolver será ahora el (13). Este método es el MCG, que tiene la desventaja de muy lenta convergencia porque la matriz M suele estar muy mal condicionada . Como el método de gradientes conjugados garantiza la convergencia (si no hay errores de redondeo) en n iteraciones para una matriz de orden n, el MCG cumple la misma propiedad.

2) El método de los gradientes biconjugados (BCG).

Generalicemos los conceptos del método de gradientes conjugados para matrices no simétricas. El algoritmo (10) puede escribirse en términos de ϕ_n y θ_n como :

$$\begin{aligned} \theta_{-1} &= 0 \\ \phi_0 &= 1 \\ \theta_n &= \phi_n + \beta_n \theta_{n-1} \\ \phi_{n+1} &= \phi_n - \alpha_n \psi \theta_n \end{aligned} \qquad (15)$$

donde $\psi(A) = A$ puede ser considerado como un polinomio. Además

$$\begin{aligned} \beta_n &= \rho_n / \rho_{n-1} \\ \alpha_n &= \rho_n / \sigma_n \\ \rho_n &= (\phi_n , \phi_n) \\ \sigma_n &= (\phi_n , \psi \phi_n) \end{aligned} \qquad (16)$$

donde (ϕ , θ) es una forma bilineal definida por :

$$(\phi , \theta) = r^{0^T} \phi(A^T) \theta(A) r^0 \qquad (17)$$

Si A es no simétrica, el algoritmo no minimiza más el residuo, ya que (9) ya no cumple con los requisitos para ser una norma.
Redefinimos entonces (ϕ , θ) de la (17) a :

$$(\phi , \theta) = \hat{r}^{0^T} \phi(A) \theta(A) r^0 \qquad (18)$$

que no es en general un producto interno. $\hat{r}^0$ es un vector a elegir. Vemos que la (18) cumple lo siguiente :

$$(\phi , \theta) = (\theta , \phi) \qquad (19)$$

$$(\phi , \xi \theta) = (\xi \phi , \theta) \qquad (20)$$

para cualquiera sean los polinomios ϕ, θ y ξ en A.
Puede probarse que [3]

$$(\phi_{n+1}, \phi_k) = 0 \quad k < n + 1 \; ; \; (\theta_n , \psi \theta_k) = 0 \quad k < n \qquad (21)$$

y por lo tanto, si A es tal que (18) es un producto interno, el residuo está en un subespacio cuya dimensión decrece en uno en cada iteración, exactamente igual que en el método de gradientes conjugados.
Definamos entonces

$$\begin{aligned} r^n &= \phi_n(A) r^0 \\ p^n &= \theta_n(A) r^0 \\ \hat{r}^n &= \phi_n(A^T) \hat{r}^0 \\ \hat{p}^n &= \theta_n(A^T) \hat{r}^0 \end{aligned} \qquad (22)$$

donde r^0 es el residuo inicial (3) y $\hat{r}^0$ es algún vector a especificar por el usuario. Koniger y Anderson [6] proponen usar $\hat{r}^0 = U^T L^{-1} r^0$, donde U y L son matrices de precondicionamiento a estudiar en más adelante. Definimos también

$$\rho_n = \hat{r}^{n^T} r^n$$

$$\sigma_n = \hat{p}^{n^T} A\, p^n \qquad (23)$$

donde todas estas definiciones están de acuerdo a la forma bilineal elegida (18). Obtenemos entonces el siguiente algoritmo, propuesto inicialmente por Fletcher (1976):

METODO DE GRADIENTES BICONJUGADOS (BCG)

Elegimos r^0 y ponemos

$$r^0 = b - A\, x^0$$

$$p^{-1} = \hat{p}^{-1} = 0$$

$$p^n = r^n + \beta_n\, p^{n-1}$$

$$\hat{p}^n = \hat{r}^n + \beta_n\, \hat{p}^{n-1}$$

$$r^{n+1} = r^n - \alpha_n\, A\, p^n$$

$$\hat{r}^{n+1} = \hat{r}^n - \alpha_n\, A^T \hat{p}^n$$

$$x^{n+1} = x^n + \alpha_n\, p^n$$

$$\alpha_n = \rho_n / \sigma_n$$

$$\beta_n = \rho_n / \rho_{n-1} \; ; \; \beta_0 = 0$$

$$\rho_n = \hat{r}^{n^T} r^n$$

$$\sigma_n = \hat{p}^{n^T} A\, p^n \qquad (24)$$

Teniendo en cuenta (21) resulta:

$$\hat{r}^{m^T} r^k = 0 \qquad k < n$$

$$\hat{p}^{m^T} A\, p^k = 0 \qquad k < n \qquad (25)$$

lo que implica que $\hat{p}^k$ y p^k son conjugados con respecto a A, que es lo que da el nombre al método.

3) El método de gradientes conjugados cuadrados (CGS).

La idea es construir un algoritmo de forma que el residuo sea $\phi_n^2(A)\, r^0$, entonces si BCG converge será $|\phi_n^2(A)\, r^0| < |\phi_n(A)\, r^0|$ [3]. De las ecuaciónes (15) se deduce:

$$\theta_n^2 = \phi_n^2 + \beta_n^2 \theta_{n-1}^2 + 2 \beta_n \phi_n \theta_{n-1}$$

$$\phi_{n+1} = \phi_n^2 + \alpha_n^2 \psi^2 \theta_n^2 - 2 \alpha_n \phi_n \psi \phi_n$$

$$\theta_n \phi_n = \phi_n^2 + \beta_n \phi_n \theta_{n-1}$$

$$\theta_n^2 = \theta_n \phi_n + \beta_n \left(\phi_n \theta_{n-1} + \beta_n \theta_{n-1}^2 \right)$$

$$\phi_{n+1} \theta_n = \phi_n \theta_n - \alpha_n \psi \theta_n^2$$

$$\phi_{n+1}^2 = \phi_n^2 - \alpha_n \psi \left(\phi_n \theta_n + \phi_{n+1} \theta_n \right) \qquad (26)$$

Teniendo en cuenta (18) y (18) podemos poner:

$$\rho_n = (\phi_n , \phi_n) = (1 , \phi_n^2)$$

$$\sigma_n = (\phi_n , \psi \phi_n) = (1 , \psi \phi_n^2) \qquad (27)$$

Por último, definamos los vectores auxiliares f, g y h de la siguiente manera:

$$f^n = \phi_n^2(A)\, r^\circ$$

$$g^n = \theta_n^2(A)\, r^\circ$$

$$h^n = \phi_n(A)\, \theta_{n-1}(A)\, r^\circ \qquad (28)$$

De lo expuesto resulta entonces:

METODO DE GRADIENTES CONJUGADOS CUADRADOS (CGS)

$$f^\circ = b - A x^\circ$$

$$g^{-1} = h^\circ = 0$$

$$\theta_n \phi_n = f^n + \beta_n h^n$$

$$g^n = \theta_n \phi_n + \beta_n \left(\beta_n g^{n-1} + h^n \right)$$

$$h^{n+1} = \theta_n \phi_n - \alpha_n A g^n$$

$$x^{n+1} = x^n + \alpha_n \left(\theta_n \phi_n + h^{n+1} \right)$$

$$f^{n+1} = f^n - \alpha_n A \left(\theta_n \phi_n + h^{n+1} \right)$$

$$\alpha_n = \rho_n / \sigma_n$$

$$\beta_n = \rho_n / \rho_{n-1} \quad ; \quad \beta_\circ = 0$$

$$\rho_n = \hat{f}^{\circ T} f^n$$

$$\sigma_n = \hat{f}^{\circ T} A g^n \qquad \text{y se adopta :}$$

$$\hat{f}^\circ = b - A x^\circ \qquad (29)$$

El método CGS fué propuesto por Sonneveld et al [3] en 1985.

4) El método del residuo mínimo generalizado (GMRES).

El GMRES fué desarrollado en 1983 por Saad y Schultz [7,8], y está claramente explicado por F. Shakib [9] en su tesis.

Sea una solución aproximada $x^0 + z$, donde z pertenece al subespacio de Krylov $K = \text{span}\left\{ r^0, Ar^0, \ldots, A^{k-1}r^0 \right\}$ y r^0 es el residuo inicial (3). El GMRES minimiza la norma L2 del residuo $| b - A(x^0 + z) |$ en K. Se genera una base ortonormal de K por el método modificado de Gramm-Schmidt:

$$
\begin{aligned}
u_1 &= \frac{r^0}{|r^0|} \\
i &= 1, \ldots, k \\
\hat{u}_{i+1} &= A\, u_i \\
j &= 1, \ldots, i \\
\beta_{i+1,j} &= \left(\hat{u}_{i+1}, u_j \right) \\
\hat{u}_{i+1} &= \hat{u}_{i+1} - \beta_{i+1,j}\, u_j \\
u_{i+1} &= \frac{\hat{u}_{i+1}}{|\hat{u}_{i+1}|}
\end{aligned}
\qquad (30)
$$

Podemos poner entonces $K = \text{span}\left\{ u_1, u_2, \ldots, u_k \right\}$. Sea U_k la matriz formada por los primeros k u_i : $U_k = \left[u_1, u_2, \ldots, u_k \right]$. Puede demostrarse que

$$A\, U_k = U_{k+1}\, H_k \qquad (31)$$

donde H_k es la matriz de Hessenberg de (k+1) x k:

$$
H_k = \begin{bmatrix}
\beta_{2,1} & \beta_{3,1} & \cdots & \beta_{k,1} & \beta_{k+1,1} \\
|\hat{u}_2| & \beta_{3,2} & \cdots & \beta_{k,2} & \beta_{k+1,2} \\
0 & |\hat{u}_3| & & \vdots & \vdots \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\vdots & 0 & \ddots & \ddots & \vdots \\
\vdots & \vdots & \ddots & |\hat{u}_k| & \vdots \\
0 & 0 & \cdots & 0 & |\hat{u}_{k+1}|
\end{bmatrix}
\qquad (32)
$$

Expresemos z como una combinación lineal de los vectores u_j ;

$$z = \sum_{j=1}^{k} y_j\, u_j \qquad (33)$$

y definamos el vector de k+1 componentes:

$$e = \left[\, | r^{\circ} |, \, 0, \, \ldots \, , \, 0 \, \right]$$

La norma del residuo la expresamos entonces como :

$$\begin{aligned} | \, b - A \left[x^{\circ} + z \right] \, | &= | \, r^{\circ} - A \left[\sum_{j=1}^{k} y_j \, u_j \right] \, | \\ &= | \, r^{\circ} - A \, U_k \, y \, | = | \, U_{k+1} \left[e - H_k \, y \right] \, | \\ &= | \, e - H_k \, y \, | \end{aligned} \tag{34}$$

entonces

$$\min_{z \in K} | \, b - A \, (x^{\circ} + z) \, | = \min_{y \in R^k} | \, e - H_k \, y \, | \tag{35}$$

La minimización de (35) se obtiene aprovechando la estructura casi diagonal de H_k mediante el algoritmo Q-R [7].

Definamos la matriz H de forma que:

$$H = R \, H_k \tag{36}$$

donde

$$R = R_k \, R_{k-1} \, \ldots \, R_2 \, R_1 \tag{37}$$

y $R_1, \, R_2, \, \ldots \, , \, R_k$ son rotaciones de la forma :

$$R_j = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & 0 & \\ & & 1 & & & & \\ & & & \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} & & & \\ & & & & 1 & & \\ & 0 & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \qquad \begin{matrix} \\ \\ \\ j \\ \\ \\ \\ j+1 \end{matrix} \tag{38}$$

R_j es una matriz de (k+1)x(k+1). c_j y s_j se eligen de tal forma que H sea triangular superior y $c_j^2 + s_j^2 = 1$ para que R_j sea ortogonal. Si ponemos

$$\bar{e} = R \, e \tag{39}$$

resulta la minimización

$$\min_{y \in R^k} | \, e - H \, y \, | = | \, \bar{e}_{k+1} \, | \tag{40}$$

La solución se obtiene de reemplazar y en (33). El vector y satisface la minimización (40) mediante la resolución por retrosustitución del sistema :

$$\begin{bmatrix} H_{1,1} & \cdots & H_{1,k} \\ 0 & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ 0 & \cdots & H_{k,k} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} \bar{e}_1 \\ \vdots \\ \bar{e}_k \end{bmatrix} \quad (41)$$

El algoritmo resulta entonces :

METODO DEL RESIDUO MINIMO GENERALIZADO (GMRES)

Sean A, b, k, c_{max} y l_{max} datos y x^0 una semilla, entonces: .

$c = c_{max}$ b

$l = 1, l_{max}$ -

$u_1 = b - A\,x^0$

$e_1 = u_1$

$u_1 = \dfrac{u_1}{u_1}$

$i = 1, k$

$u_{i+1} = A\,u_i$

$j = 1, i$

$\beta_{i+1,j} = \left(u_{i+1}, u_j \right)$

$u_{i+1} = u_{i+1} - \beta_{i+1,j}\,u_j$

$h^i = \left\{ \beta_{i+1,1}, \cdots, \beta_{i+1,i}, \quad u_{i+1} \right\}^T$

$u_{i+1} = \dfrac{u_{i+1}}{h^i_{i+1}}$

$j = 1, i-1$

$$\begin{bmatrix} h^i_j \\ h^i_{j+1} \end{bmatrix} = \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{bmatrix} h^i_j \\ h^i_{j+1} \end{bmatrix}$$

$r = \sqrt{\left(h^i_i\right)^2 + \left(h^i_{i+1}\right)^2}$

$c_i = \dfrac{h^i_i}{r}$; $s_i = \dfrac{h^i_i}{r}$; $h^i_i = r$; $h^i_{i+1} = 0$

$\bar{e}_{i+1} = - s_i\,\bar{e}_i$

$\bar{e}_i = c_i\,\bar{e}_i$

si $| \bar{e}_{i+1} | \leq \varepsilon$ ó $i = k$ resolver y de (41)

$$x = x + \sum_{j=1}^{i} y_j u_j$$

si $| \bar{e}_{i+1} | \leq \varepsilon$ terminan ciclos GMRES (fin loop i)

próximo i (42)

PRECONDICIONAMIENTO

Todos los algoritmos iterativos de la sección anterior dependen, en cuanto a la velocidad de convergencia, del número de condición de la matriz A (el número de condición nos dice cuán fácil es invertir la matriz). En general las velocidades de convergencia se mejoran notablemente utilizando un precondicionamiento adecuado, que mejora el número de condición de la matriz.

Definamos el número de condición de una matriz A como :

$$k = \frac{| \lambda_{max} |}{| \lambda_{min} |} \qquad (43)$$

donde λ_{max} y λ_{min} son los valores máximo y mínimo de los autovalores de A. Una matriz bien condicionada tendrá un k cercano a 1, y una mal condicionada tendrá un k grande. Cuando k es grande, el polinomio (8-5) será de un grado alto [9], y la convergencia será lenta.

Para mejorar la velocidad de convergencia se utilizan dos niveles para mejorar el número de condición de la matriz: el pre-precondicionamiento y el precondicionamiento. La diferencia conceptual entre el primer nivel y el segundo es que en el pre-precondicionamiento se arma una nueva matriz con la cual se opera luego (llevando entonces multiplicación de matrices), mientras que una matriz de precondicionamiento nunca se mutiplica debido al alto costo de estas operaciones con matrices grandes. El pre-recondicionador debe ser una matriz muy simple cuya multiplicación por la matriz A no sea muy costosa. La primera opción es escalear la diagonal mediante el proceso denominado diagonal scaling y que podemos poner como:

$$\begin{aligned} & D = \text{diag}\,(A) \\ & D^{-1/2} A\, D^{-1/2} D^{1/2} x = D^{-1/2} b \\ & M\, y = c \\ & M = D^{-1/2} A\, D^{-1/2} \;\; ; \;\; y = D^{1/2} x \;\; ; \;\; c = D^{-1/2} b \end{aligned} \qquad (44)$$

De esta forma el orden de magnitud de los valores de la matriz se iguala, y la diagonal queda :

$$\text{diag}\,(M) = I \qquad (45)$$

donde I es la matriz identidad.

Una forma algo más sofisticada de pre-precondicionamiento es el block diagonal scaling (BDS) que utiliza una matriz de la forma de la figura 1.

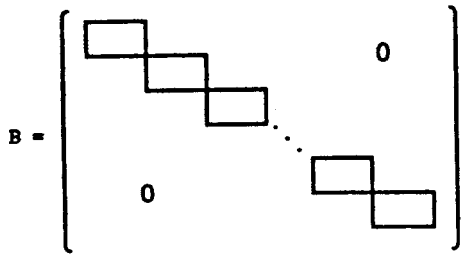

Figura 1

Definimos la matriz B como :

$$B = \text{block } (A)$$

donde el tamaño de los bloques es el número de ecuaciones nodales del sistema. La matriz debe estar ordenada de manera de colocar todas las ecuaciones de un dado nodo en filas sucesivas para que el BDS sea efectivo.

Descomponemos B en su factorización LU :

$$B = L_B U_B \tag{46}$$

y entonces llevamos el sistema a resolver a:

$$M y = c \quad ; \quad M = L_B^{-1} A U_B^{-1}$$
$$y = U_B x \; ; \; c = L_B^{-1} b \tag{47}$$

En el caso del diagonal scaling, la transformación al sistema M y = c no altera la estructura de la matriz (M tiene la misma estructura que A). En el caso del block diagonal la estructura de M no es la misma que la de A, teniendo entonces un costo adicional como pre-precondicionador cuando el almacenamiento es del tipo Gustafsson. Si el almacenamiento es del tipo skyline no existe costo adicional [9].

Dentro del grupo de precondicionadores, se busca una buena aproximación a la inversa de A. La procedencia de esta matriz de precondicionamiento puede ser cualquiera, mientras cumpla con el objetivo de mejorar el número de condición de la matriz. Los precondicionadores más estudiados y que han dado mejores resultados se basan en realizar una factorización incompleta de la matriz A en dos matrices triangulares L y U, tal como se hace en un método directo pero desechando con algún criterio algunos valores.

La factorización incompleta fue estudiada primero para matrices simétricas [10,11] y luego extendida a matrices no simétricas [12,13]. Consideremos una factorización de A del tipo

$$A = L U \tag{48}$$

Si A es una matriz rala, en general L y U perderán esa virtud, como vimos anteriormente. Intentemos entonces obtener una aproximación de L y U de tal forma que mantengan la misma estructura de A, tal como sugiere Kershaw [13]. Diremos que las matrices L y U así obtenidas son la factorización incompleta de A (factorización ILU) con llenado de orden cero. Simplemente, los términos de la factorización que caerían en un punto de A donde hay un cero no los calculamos.

Si la matriz A es bastante densa podemos asegurar que:

$$\left[L\ U \right]^{-1} \approx A^{-1} \qquad (49)$$

Si la matriz A es rala, la aproximación de llenado cero puede no ser buena. Una opción es aumentar el llenado de L y U de tal forma de mejorar la factorización. Esto puede hacerse con dos criterios : a) elegir nuevos lugares en la estructura donde permitiremos no ceros; y b) permitir a algunos valores (en algún rango de interés) sean no ceros a pesar de caer en un lugar no permitido en la estructura. A diferentes niveles de llenado de las matrices L y U los llamaremos de orden uno, dos, etc.

Una vez obtenidas las matrices L y U , el sistema a resolver queda:

$$A\ x = b\ ;\ A = L^{-1}\ M\ U^{-1}$$

$$x = U\ y\ ;\ b = L^{-1}\ c \qquad (50)$$

donde M, c e y son los sistemas pre-precondicionados de (44) o (47).

EXPERIMENTOS NUMERICOS

La comparación de prestaciones para calcular matrices no simétricas entre un resolvedor directo (MA28) en *simple precision* y los métodos iterativos antes descriptos en *doble precision* muestran que los últimos son al menos un orden de magnitud más veloces (la diferencia se hace más grande cuanto mayor es la matriz). Los requerimientos de memoria de un resolvedor directo también son mucho mayores que los de los resolvedores iterativos, siendo también la diferencia más notable en matrices muy grandes (Si la matriz no es muy rala las diferencias pueden ser despreciables o incluso comportarse mejor un resolvedor directo). Todo lo explicado aquí se refiere a matrices con muy poco llenado, típicas del método de elementos finitos). Brussino y Sonnad [14] corrieron diferentes casos con matrices de orden 13000 con 200000 no ceros y obtuvieron, para la matriz peor condicionada, que el resolvedor directo es unas 50 veces más lento y consume unas 10 veces más memoria que un iterativo. Nosotros utilizamos para comparar tres matrices de orden 2121 con 14300 no ceros de un problema de convección-difusión en una expansión súbita con Pe = 10, 1000 y 100000. En la tabla I se muestran los resultados, con los tiempos en segundos.

Tabla I : Orden de la matriz: 2121; No ceros: 14300

	MA28	CGS
CPU (Pe = 10/1000/100000)	984 / 487 / 940	40 / 24 / 33
Memoria utilizada (words)	358400	171400

Es clara la diferencia de tiempo de cálculo y de memoria utilizada y para problemas más grandes se hace cada vez mayor. Es de mayor interés el resultado de la comparación entre los diferentes métodos iterativos. Para realizar esta comparación generamos cuatro matrices de diferentes problemas de Navier-Stokes. En la tabla II vemos los resultados de la comparación entre MCG, BCG y CGS y GMRES para una matriz resultante de una cavidad cuadrada de Re = 400. Se presentan los valores obtenidos sin precondicionamiento y con precondicionamiento Diagonal Scaling. Si se muestra un solo valor es precondicionado. En la figura 2 vemos la convergencia de los diferentes métodos en función de la iteración. Se observa en esta figura que los métodos para matrices no simétricas son de convergencia irregular.

Tabla II : Orden de la matriz: 507; No ceros: 7567

	MCG	BCG	CGS	$GMRES^{50-25}$
Número de iteraciones	123-43	207-30	129-21	225-42
Tiempo de cálculo (CPU)	59.7-22.5	98.8-16.0	63.5-12.5	75.2-20.6
Memoria utilizada (words)	32000	32000	32800	42400

En la tablas III y IV vemos los resultados obtenidos para matrices generadas con cavidades cuadradas con Re = 1000, la 3 con una red regular de 21x21 nódos y la 4 de una red densificada de 634 nodos. En los casos en que se muestran resultados del GMRES aparece la dimensión del espacio de Krylov utilizada.

Tabla III : Orden de la matriz: 1323; No ceros: 20447

	BCG	CGS	$GMRES^{50}$
Número de iteraciones	415-90	335-73	
Tiempo de cálculo (CPU)	682-156	562-128	181
Memoria utilizada (words)	119000	117800	185000

Tabla IV : Orden de la matriz: 1902; No ceros: 29512

	BCG	CGS	$GMRES^{50}$
Número de iteraciones	371-79	274-67	
Tiempo de cálculo (CPU)	894-202	684-173	225
Memoria utilizada (words)	174000	176000	281000

Finalmente, corrimos un caso con una matriz de orden 6363 generada con una expansión súbita de Re = 200. En la tabla V vemos los resultados obtenidos.

En todas las corridas se utilizó pre-precondicionamiento diagonal-scaling y precondicionamiento ILU. Realizamos pruebas con diferentes numeraciones de forma tal de que las matrices de la descomposición LU sean lo más ralas posible, y para numeración tal que la matriz resulte lo más banda posible. En los tiempos de cálculo no se notaron grandes diferencias.

Tabla V : Orden de la matriz: 6363; No ceros: 100527

	BCG	CGS	$GMRES^{70}$
Número de iteraciones	173	96	
Tiempo de cálculo (CPU)	1512	854	2020
Memoria utilizada (words)	589000	595000	984000

El efecto de la programación en doble precisión es notable. Los tiempos de cálculo caen en alrededor del 20 % para matrices mal condicionadas o muy grandes y es aproximadamente igual al de simple precisión para matrices pequeñas. Esto se debe a que el número de iteraciones para convergencia se hace menor aunque cada iteración es más costosa.

Todas las corridas fueron realizadas con un paquete de rutinas

desarrollado por los autores en una computadora MicroVax. El valor aceptable para convergencia lo fijamos en 1e-06 para $| r | / | b |$.

CONCLUSIONES

De los resultados mostrados en los experimentos numéricos y de la experiencia acumulada en la utilización de las subrutinas de resolución de matrices no simétricas, concluimos lo siguiente:

- El CGS con precondicionamiento ILU es el método de mejor performance para la resolución de sistemas matriciales no-simétricos ralos. Sin embargo, debe tenerse en cuenta que el CGS no garantiza convergencia.
- Para sistemas matriciales poco ralos, la resolución directa tiene una velocidad aproximadamente igual a los métodos iterativos, con la ventaja de una rápida resolución si existen varios sistemas a resolver con la misma matriz y diferente término independiente.
- Para matrices muy mal condicionadas, el GMRES tiene la ventaja de permitir aumentar el número de vectores del espacio de Krylov hasta converger, pero al costo de mayor memoria y de mayor tiempo de cálculo. Para valores óptimos del parámetro k (dimensión del subespacio de Krylov), el comportamiento del GMRES es aproximadamente igual a la del BCG, aunque no puede saberse a priori este valor. Si el valor de k es muy pequeño, es usual que el GMRES no converja.

BIBLIOGRAFIA

[1] S. Pissanetsky, *Sparse Matrix Technology*
[2] MA28A/AD, Harwell Subroutine Library (1977)
[3] P. Sonneveld, P. Wesseling y P. de Zeeuw, *Multugrid and conjugate gradient methods as convergence acceleration techniques*, in Multigrid Methods for Integral and Differential Equations, Clarendon Press, Oxford, pp 117-167 (1985)
[4] M. Hestenes y E. Stieffel, *Methods of conjugate gradients for solving linear systems*, Nat. Bur. of Stds J. Res., vol 49, p 409 (1952)
[5] Y. Saad, *Krylov space methods for solving large unsymmetric linear systems*, Math of Comp., vol 37, p 155 (1981)
[6] A. Koniges y D. Anderson, *A preconditioned Biconjugated gradient routine for the solution of linear asymmetric matrix equations arising from 9-point discretization*, Comp. Phys. Comm. vol 43, pp 297-302 (1978)
[7] Y. Saad y M. Schultz, GMRES: *A Generalized Minimal RESidual algorithm for solving nonsymmetric linear systems*, Technical report # 254, Yale University (1983)
[8] Y. Saad y M. Schultz, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems*, Math of Comp., vol 44, pp 417-424 (1985)
[9] F. Shakib, Ph. D. Thesis, Stanford University (1987)
[10] J. A. Meijerink y H. Van der Vorst, *An iterative solution method for linear system which the coefficient matrix is symmetric M-matrix*, Math. of Comp., vol 31, pp 148-162 (1977)
[11] T. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math of Comp., vol 34, pp 473-497 (1980)
[12] D. Kershaw, *The incomplete Cholesky conjugate gradient method for the iterative solution of systems of linear equations*, J. of Comp. Phys., vol 26, pp 43-65 (1978)
[13] D. Kershaw, *On the problem of unstable pivots in the incomplete LU conjugate gradient method*, J. Comp. Phys., vol 38, pp 114-123 (1980)
[14] G. Brussino y V. Sonnad, *A comparison of direct and preconditioned iterative techniques for sparse unsymmetric systems of linear equations*, Int. J. for Num. Meth. Engng, vol 28, pp 801-815 (1989)

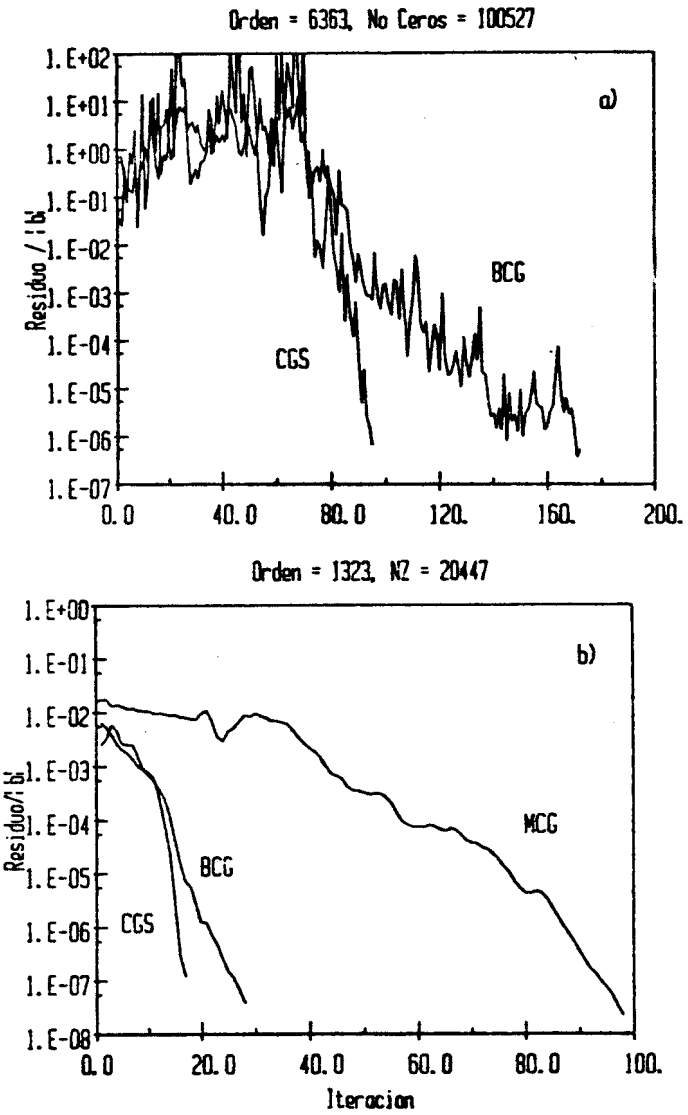

Figura 2 : a) Comportamiento de los residuos de los métodos BCG y CGS para una matriz de orden 6363.

b) Comportamiento de los residuos de los métodos MCG, BCG y CGS para una matriz de orden 1323.