Mecánica Computacional Vol. XVI
compilado por G. Etse y B. Luccioni
Tucumán, Septiembre de 1996

Cálculo Numérico en Computadoras Vectoriales

Andrea Yommi * y Alberto Cardona †
INTEC - Güemes 3450 - 3000 Santa Fe - Argentina

Resumen

El objetivo principal del trabajo es la programación de algoritmos para su utilización en computadoras vectoriales adaptando el software de base *Templates* [1]. Se trabajó con el procesador vectorial Intel i860. Los algoritmos analizados son todos del tipo iterativo y sirven para la resolución de sistemas de ecuaciones algebraicas lineales. Las experiencias numéricas realizadas permiten determinar entre los métodos estudiados cuales tienen mejor performance, teniendo como parámetros de referencia el número de iteraciones empleadas y el tiempo de ejecución de los mismos.

Abstract

The objective of this work is to program algorithms to use in vector computers apdating the software *Templates* [1]. We use the vector processor Intel i860. The algorithms analyzed are all of iterative type, to solve linear systems of equations. The numerical experiences allow us to conclude between all of the methods those which have better performance, having as reference parameters the number of iterations and the execution time carry out by them.

1 Introducción

Resolver un sistema lineal rectangular $\mathbf{Ax} = \mathbf{b}$, constituye una herramienta básica en una gran cantidad de aplicaciones. Si la matriz es pequeña, los algoritmos de resolución se basan en métodos directos, que en general son más rápidos y precisos que los iterativos. Puesto que $\mathbf{A}$ debe guardarse completamente, limitaciones de memoria restringen el orden del sistema a resolver. Por otro lado, si $\mathbf{A}$ es rala, bien condicionada y de tamaño suficientemente grande ($O(10^4)$ ecuaciones), es conveniente aplicar métodos iterativos ya que los directos se tornan ineficientes. Estos generan una secuencia $\{\mathbf{x}^{(k)}\}$ de aproximaciones a la solución $\mathbf{x}$, y sólo sirven para resolver una clase específica de problemas, debido a que el radio de convergencia de los mismos depende fuertemente de las propiedades espectrales de la matriz $\mathbf{A}$. Como consecuencia, estos métodos requieren una matriz de precondicionamiento, la cual transforma el sistema lineal dado en otro con un espectro más favorable. En este trabajo se resuelven sistemas lineales ralos de la forma

$$\mathbf{Ax} = \mathbf{b} \tag{1}$$

siendo $\mathbf{A}$ una matriz de orden $n \times n$. Se emplean métodos iterativos estacionarios y no estacionarios.

*Becaria de la Universidad Nacional del Litoral, programa Cientibeca.

†Profesor, Universidad Nacional del Litoral, e Investigador del *Consejo Nacional de Investigaciones Científicas y Técnicas de la República Argentina.*

2 Métodos iterativos para resolver sistemas lineales

2.1 Métodos iterativos estacionarios

Se entiende por métodos iterativos estacionarios para resolver el sistema lineal (1) aquéllos que pueden representarse en la forma

$$\mathbf{x}^{(k)} = \mathbf{B}\,\mathbf{x}^{(k-1)} + \mathbf{c} \tag{2}$$

siendo $\mathbf{B}$ la matriz de iteración del método [1], [2]. El siguiente teorema [3] asocia la convergencia de estos métodos con el radio espectral de la matriz $\mathbf{B}$.

Teorema 1 *Sean* $\mathbf{b} \in I\!R^n$ *y* $\mathbf{A} = \mathbf{M} - \mathbf{N} \in I\!R^{n\times n}$ *no singular. Entonces, la secuencia* $\{\mathbf{x}^{(k)}\}$ *definida por (2) converge para cualquier vector de comienzo* $\mathbf{x}^{(0)}$ *si y sólo si*

$$\rho(\mathbf{B}) = \max_{1\le i\le n} |\lambda_i(\mathbf{B})| < 1 \tag{3}$$

2.1.1 Método de Jacobi

Está definido para matrices con diagonal principal no nula. El mismo examina las ecuaciones del sistema en forma independiente. Si $\mathbf{D}$, $\mathbf{L}$ y $\mathbf{U}$ denotan la diagonal, la parte triangular inferior y la parte triangular superior de $\mathbf{A}$ respectivamente, entonces la iteración del método se puede expresar como

$$\mathbf{x}^{(k)} = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\,\mathbf{x}^{(k-1)} + \mathbf{D}^{-1}\mathbf{b} \tag{4}$$

Por lo tanto, $\mathbf{B}_J = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})$ es la matriz de iteración del método de Jacobi y la aplicación del teorema 1 a $\mathbf{B}_J$, indica que la iteración de Jacobi converge si $\mathbf{A}$ es estrictamente diagonal dominante [2].

2.1.2 Método de Gauss Seidel

A diferencia del método de Jacobi, las nuevas componentes se utilizan en el cálculo de la próxima componente, razón por la cual las actualizaciones no se pueden realizar simultáneamente. La representación matricial del método tiene la forma

$$\mathbf{x}^{(k)} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\,\mathbf{x}^{(k-1)} + (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b} \tag{5}$$

y la matriz de iteración del mismo es $\mathbf{B}_{GS} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}$. Se demuestra que si $\mathbf{A} \in I\!R^{n\times n}$ es simétrica y definida positiva, entonces la iteración de Gauss Seidel converge para cualquier aproximación inicial [3].

2.1.3 Sobrerrelajaciones Sucesivas (SOR)

Este método es una variante de Gauss Seidel que procura aumentar la velocidad de convergencia. Para tal fin se introduce un parámetro ω, y se toma el promedio ponderado entre la iteración previa y la de Gauss Seidel

$$(\mathbf{D}+\omega\mathbf{L})\,\mathbf{x}^{(k)} = [\,(1-\omega)\mathbf{D} - \omega\mathbf{U}\,]\,\mathbf{x}^{(k-1)} + \omega\mathbf{b} \tag{6}$$

La matriz de iteración de SOR [1], [3] es $\mathbf{B}_\omega = (\mathbf{D}+\omega\mathbf{L})^{-1}[\,(1-\omega)\mathbf{D}-\omega\mathbf{U}\,]$ y satisface $\rho(\mathbf{B}_\omega) \ge |\,\omega-1\,|$, de modo que sólo puede haber convergencia para valores del parámetro comprendidos entre cero y dos.

2.2 Métodos iterativos no estacionarios

En estos métodos se particiona $\mathbf{A}$ en la forma $\mathbf{A} = \mathbf{M} - \mathbf{N}$, y se define la k-ésima iteración por

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \mathbf{y}^{(k)} \tag{7}$$

siendo $\mathbf{y}^{(k)} = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k-1)})$ un vector del subespacio k-ésimo de Krylov K_k definido por:

$$K_k(\mathbf{M}^{-1}\mathbf{A}, \mathbf{M}^{-1}\mathbf{r}^{(0)}) = \text{span}\,\{\mathbf{M}^{-1}\mathbf{r}^{(0)}, (\mathbf{M}^{-1}\mathbf{A})\mathbf{M}^{-1}\mathbf{r}^{(0)}, \ldots, (\mathbf{M}^{-1}\mathbf{A})^{k-1}\mathbf{M}^{-1}\mathbf{r}^{(0)}\} \tag{8}$$

Aquí $\mathbf{M}$ es llamada la matriz de precondicionamiento. Si $\mathbf{M} = \mathbf{I}$, los métodos no precondicionados conducen a soluciones pertenecientes al subespacio $K_k(\mathbf{A}, \mathbf{r}^{(0)})$. Por lo tanto, los métodos de Krylov o de proyección, se basan en construir una base apropiada para el subespacio de Krylov y luego resolver el sistema (1) proyectado sobre este subespacio [4].

2.2.1 Gradiente Conjugado (CG)

Se utiliza para resolver sistemas lineales con matriz de coeficientes simétrica y definida positiva. Construye la aproximación k-ésima como un elemento del subespacio actual de Krylov (8) de manera tal de minimizar la norma del error $\mathbf{E}(\mathbf{x}^{(k)}) = (\mathbf{x}^{(k)} - \mathbf{x})^t\mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x})$ a lo largo de direcciones de búsqueda $\mathbf{A}$ - conjugadas. La proyección de (1) sobre este subespacio proporciona una matriz tridiagonal simétrica, de modo que la construcción de una base para éste puede hacerse mediante una relación de recurrencia de tres términos [1], [3]. Esta base está formada por los residuos de las iteraciones los cuales son mutuamente ortogonales. La velocidad de convergencia del método depende del número de condición de la matriz $\mathbf{M}^{-1}\mathbf{A}$.

2.2.2 Método del Residuo Mínimo Generalizado (GMRES)

Este método se puede aplicar cuando $\mathbf{A}$ es no simétrica e indefinida, y consiste en generar una secuencia de vectores ortogonales $\{\mathbf{v}^{(k)}\}_{k\geq 1}$ mediante el algoritmo de Arnoldi. Debido a la ausencia de simetría, la matriz que representa la proyección de (1) sobre el subespacio de Krylov $K_k(\mathbf{A}, \mathbf{v}^{(1)})$ deja de ser tridiagonal y resulta en cambio una matriz superior de Hessenberg $\mathbf{H}_k$.
La iteración de GMRES se elige como en (7). Aquí $\mathbf{y}^{(k)} = \mathbf{V}_k\mathbf{z}^{(k)}$ donde $\mathbf{z}^{(k)}$ se elige de manera tal de minimizar la norma 2 del residuo. Esto se hace mediante la resolución de un problema de mínimos cuadrados que involucra a la matriz $\mathbf{H}_k$. Debido a su estructura especial, la estrategia a seguir consiste en calcular la factorización $\mathbf{QR}$ de $\mathbf{H}_k$ mediante rotaciones de planos [5]. La desventaja del método, es que tanto el almacenamiento requerido como el trabajo computacional crecen linealmente con las iteraciones. Para sobrellevar tal dificultad, se recomienzan las iteraciones cada m pasos del algoritmo, siendo m un parámetro a determinar [1], [5].

2.2.3 Gradiente Bi-Conjugado (BiCG)

Se trata de una generalización del método CG para sistemas lineales no simétricos y no singulares. Consiste en aplicar el algoritmo de bi-ortogonalización de Lanczos, el cual genera dos secuencias de vectores mutuamente ortogonales, ligadas a través de $\mathbf{A}$ por una matriz tridiagonal. En BiCG, una de estas secuencias corresponde a los residuos $\mathbf{r}^{(k)}$ y se basa en la matriz $\mathbf{A}$; la otra, se denota por $\tilde{\mathbf{r}}^{(k)}$ y se basa en $\mathbf{A}^t$. Como en CG, se verifica una relación de recurrencia de tres términos, razón por la cual el almacenamiento y el trabajo por iteración se mantienen constantes.
BiCG busca una aproximación del tipo (7) caracterizada por una condición de Galerkin. Es decir, $\mathbf{y}^{(k)} \in K_k(\mathbf{A}, \mathbf{r}^{(0)})$, mientras que los residuos $\mathbf{r}^{(k)}$ son ortogonales al subespacio $\tilde{K}_k(\mathbf{A}^t, \tilde{\mathbf{r}}^{(0)})$.
En la práctica, BiCG suele tener un comportamiento bastante irregular y en ciertas ocasiones puede fallar o tener inestabilidad numérica [6], [7].

2.2.4 Gradiente Conjugado Cuadrado (CGS)

Se aplica en la resolución de sistemas lineales no simétricos y es una variante de BiCG, con la diferencia de que las actualizaciones de las secuencias generadas por $\mathbf{A}$ y $\mathbf{A}^t$ se aplican a los mismos vectores, no necesitando trabajar con $\mathbf{A}^t$. En la etapa k de BiCG, el residuo puede mirarse como el producto de un polinomio en $\mathbf{A}$ y el residuo inicial $\mathbf{r}^{(k)} = \mathbf{P}_k(\mathbf{A})\,\mathbf{r}^{(0)}$. El hecho de que $\mathbf{r}^{(k)} \to 0$ cuando k crece, indica que la matriz $\mathbf{P}_k(\mathbf{A})$ actúa como un operador de contracción, por lo que resulta ventajoso calcular $\mathbf{P}_k^2(\mathbf{A})$. Por lo tanto, CGS es un algoritmo que genera tal polinomio y calcula el residuo de las iteraciones como

$$\mathbf{r}^{(k)} = \mathbf{P}_k^2(\mathbf{A})\,\mathbf{r}^{(0)} \tag{9}$$

En general, CGS puede llegar a doblar la convergencia de BiCG [1], [8].

2.2.5 Gradiente Bi-Conjugado Estabilizado (BiCGSTAB)

Se ha diseñado para resolver sistemas lineales no simétricos y para evitar el comportamiento irregular de CGS. Este método construye las aproximaciones al igual que los métodos del tipo CG y calcula los residuos mediante la relación $\mathbf{r}^{(k)} = \mathbf{Q}_k(\mathbf{A})\,\mathbf{P}_k(\mathbf{A})\,\mathbf{r}^{(0)}$ siendo $\mathbf{Q}_k(\mathbf{A})$ un polinomio en $\mathbf{A}$ de grado k definido por

$$\mathbf{Q}_k(\mathbf{A}) = \prod_{i=1}^{k}(1 - \omega_i \mathbf{A}) \tag{10}$$

Las constantes $\omega_i, i = 1, \ldots, k$ se eligen de manera tal de minimizar $\mathbf{r}^{(k)}$ en la norma 2 como una función de ω_i, motivo por el cual la convergencia del método tiende a suavizarse respecto a CGS [9].

3 Precondicionamiento

Debido a que los métodos iterativos dependen de las propiedades espectrales de la matriz de coeficientes, resulta conveniente transformar (1) en un sistema equivalente que mejore las mismas. Si $\mathbf{M}$ es una aproximación de la matriz $\mathbf{A}$, el sistema $\mathbf{M}^{-1}\mathbf{A}\,\mathbf{x} = \mathbf{M}^{-1}\,\mathbf{b}$ provee la misma solución que (1), aunque sus propiedades espectrales son más favorables. Por lo general, la matriz $\mathbf{M}$ se da en forma factorizada $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$ y se aplican los algoritmos dados en la sección anterior para resolver el sistema lineal equivalente:

$$(\mathbf{M}_1^{-1}\mathbf{A}\,\mathbf{M}_2^{-1})(\mathbf{M}_2\,\mathbf{x}) = \mathbf{M}_1^{-1}\,\mathbf{b} \tag{11}$$

Usualmente, se llama a $\mathbf{M}_1$ el precondicionamiento a izquierda y a $\mathbf{M}_2$ el precondicionamiento a derecha [1].

Precondicionamiento de Jacobi: Consiste en tomar la matriz $\mathbf{M}$ como la diagonal de $\mathbf{A}$. De este modo, la matriz escalada $\mathbf{M}^{-1}\mathbf{A}$ tiene elementos unitarios en la diagonal. Esta implementación sólo necesita espacio de memoria para almacenar los elementos de la diagonal de $\mathbf{A}$ o sus recíprocos [1].

Factorización Incompleta: La descomposición $\mathbf{LU}$ incompleta de $\mathbf{A}$ consiste en generar una matriz triangular inferior $\mathbf{L}$ y una matriz triangular superior $\mathbf{U}$, de modo tal de preservar la estructura rala de $\mathbf{A}$. Es decir, se descartan los elementos no nulos generados por el proceso de factorización que corresponden a posiciones nulas de $\mathbf{A}$ [1]. En los ejemplos se empleó la forma de precondicionamiento $\mathbf{M}_1 = \mathbf{LU}$ y $\mathbf{M}_2 = \mathbf{I}$.

4 Procesador Vectorial Intel i860

Las computadoras vectoriales aumentan la velocidad de cálculo debido a la realización simultánea de operaciones en varios procesadores. El i860 es un microprocesador superescalar RISC diseñado

para obtener máximo rendimiento en las aplicaciones numéricas intensivas. Cuenta con procesadores independientes para operaciones enteras, suma en punto flotante y producto en punto flotante. La velocidad de procesamiento pico en esta placa (80 megaflops) puede llegar a ser de hasta 20 veces la velocidad pico en un 486 DX2. Para que el procesador i860 sea eficaz, deben diseñarse los algoritmos de forma tal que logren presentar al procesador las operaciones a realizar en un orden apropiado. Junto al i860, se dispone de una biblioteca de Subrutinas Básicas del Algebra lineal *BLAS* [4] agrupadas en tres niveles

- Nivel 1: Realiza operaciones de bajo nivel entre vectores. Estas involucran $O(n)$ operaciones en punto flotante y $O(n)$ movimientos de datos, siendo n la longitud de los vectores.
- Nivel 2: Corresponde a operaciones entre matrices y vectores e involucran $O(n^2)$ operaciones escalares y $O(n^2)$ movimientos de datos.
- Nivel 3: Ejecuta operaciones entre matrices. Requiere $O(n^2)$ movimientos de datos y $O(n^3)$ operaciones en punto flotante.

5 Experiencias Numéricas

Los problemas propuestos corresponden a casos particulares del operador elíptico

$$\mathbf{L}u = \mathrm{a}\frac{\partial^2 u}{\partial x^2} + \mathrm{b}\frac{\partial^2 u}{\partial y^2} + \mathrm{c}\frac{\partial^2 u}{\partial z^2} + \mathrm{d}\frac{\partial u}{\partial x} + \mathrm{e}\frac{\partial u}{\partial y} + \mathrm{f}\frac{\partial u}{\partial z} + \mathrm{g}\,u \tag{12}$$

con condiciones de borde tipo Dirichlet usando grillas rectangulares en dos y tres dimensiones. Los sistemas considerados provienen de la discretización de tales ecuaciones diferenciales empleando diferencias finitas o elementos finitos. La dimensión del sistema de ecuaciones n corresponde al número de puntos interiores de la grilla rectangular. En todos los casos las iteraciones se comenzaron con $\mathbf{x}^{(0)} = \mathbf{0}$ a excepción del segundo problema que comienza a iterar con $\mathbf{x}^{(0)} = \mathbf{1}$. En cuanto al vector lado derecho del sistema a resolver, se consideró: para el segundo y último problema, un vector de componentes todas iguales a uno, mientras que para el resto, cada componente del vector lado derecho es la suma de los elementos que componen la fila correspondiente en la matriz **A**. El número de iteraciones se restringió a $maxit = k.n$, siendo k una constante seleccionada según el tamaño de la grilla. Para todos los problemas se usaron tres alternativas: (SP) sin precondicionamiento, (PJ) con precondicionamiento de Jacobi y (FI) con Factorización incompleta. Además se resolvieron con y sin el uso del procesador vectorial i860.
En el método GMRES, se determinó el parámetro de recomienzo m en función del espacio de memoria disponible. En el método SOR, se ensayaron distintos valores para el parámetro de relajación ω.

Criterios de Parada: Para cada algoritmo se propuso un número máximo de iteraciones a realizar (*maxit*) en términos de n. Para los métodos estacionarios se consideró $maxit = 3$ n. Debido a que todos los cálculos fueron realizados en doble precisión, se pidió una tolerancia en el residuo $\mathbf{r}^{(k)}$ de 1.0E-15 (tol). Los métodos terminan de iterar al llegar a la precisión especificada, o en caso contrario al llegar al máximo de iteraciones permitidas. El criterio de parada seleccionado para los algoritmos no estacionarios es

$$\| \mathbf{r}^{(k)} \|_2 \leq tol \, \|\mathbf{b}\|_2 \tag{13}$$

y para los métodos estacionarios

$$\frac{\| \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} \|}{\| \mathbf{x}^{(k)} \|} \leq tol \tag{14}$$

Almacenamiento de la matriz de coeficientes: Todos los problemas planteados tienen una matriz rala (aproximadamente entre 90 % y 95 % de elementos nulos). Fueron almacenadas usando el esquema de almacenamiento reducido por filas [1], el cual recorre las filas de **A** y guarda sólo los elementos distintos de cero en posiciones contiguas de memoria.

5.1 Problema de Poisson en dos dimensiones F2SH:

Se considera el operador diferencial elíptico

$$\mathbf{L}u = -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} \tag{15}$$

el cual se discretiza en una malla uniforme de tamaño 70×70. El sistema lineal que resulta tiene una matriz de coeficientes **A** simétrica, rala y definida positiva de orden $n = 4900$ con 24220 elementos distintos de cero. Se permitió un máximo de 1225 iteraciones. La matriz **A** no es estrictamente diagonal dominante, lo que indica que el método de JACOBI puede no converger.
La implementación del precondicionamiento de Jacobi en los métodos no estacionarios no mejora la convergencia respecto a la versión sin precondicionamiento, necesitando en algunos casos más tiempo para converger. Entre los métodos no estacionarios probados, sin duda CG proporciona la mejor opción. Esta aproximación, usando factorización incompleta, toma $k = 91$ iteraciones en reducir el residuo inicial por el factor tol, y sin precondicionamiento toma $k = 178$ iteraciones. El método BiCG llega también a convergencia pero como se duplican las operaciones a realizar por iteración respecto de CG, lo mismo ocurre con el tiempo de ejecución. Usando factorización incompleta, CGS y BiCGSTAB convergen en 63 y 65 iteraciones respectivamente (las dos terceras partes que CG), aunque éstos necesitan realizar el doble de operaciones entre vectores y productos matriz - vector por iteración.
Para valores pequeños de n, digamos ($n \leq 100$), la aplicación de factorización incompleta no ofrece ventajas respecto a la versión no precondicionada de los métodos. Además debe tenerse en cuenta el tiempo de construcción de las matrices que forman tal descomposición.
El método SOR se probó para valores de $n = 100$, 400 y 900 y para valores del parámetro de relajación comprendidos entre cero y dos. Los resultados obtenidos son los siguientes: para $n = 100$, ω debe ser mayor o igual que 0.6, para $n = 400$ debe ser $\omega \geq 0.8$, y para $n = 900$, $\omega \geq 1.2$, (en este último caso Gauss Seidel no converge). Los mejores resultados obtenidos, teniendo en cuenta el número de iteraciones requeridas y el tiempo empleado para alcanzar convergencia son: para $n = 100$ y usando $\omega = 1.6$, SOR necesita 72 iteraciones, cuando $n = 400$, y $\omega = 1.8$, 162 iteraciones y finalmente para $n = 900$ con el mismo valor de ω, SOR requiere 259 iteraciones.

5.2 Problema de Poisson en tres dimensiones F3SH:

En este caso se resuelve el problema

$$\mathbf{L}u = -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial z^2} \tag{16}$$

utilizando una malla de $17 \times 17 \times 17$. El sistema lineal que resulta de su discretización tiene una matriz de coeficientes simétrica, rala y definida positiva de orden $n = 4913$ con 32657 elementos no nulos. Se consideró un máximo de 1228 iteraciones. En este caso, el método de JACOBI converge, aunque lo hace muy lentamente: realiza 1978 pasos en 99 segundos con el procesador vectorial i860, mientras que sin éste tarda 138 segundos.
Para este ejemplo se graficó el logaritmo del residuo de las iteraciones correspondientes a los métodos no estacionarios. Al igual que el problema anterior, el precondicionamiento de Jacobi no acelera la convergencia respecto a la versión sin precondicionamiento, como puede apreciarse en la similitud de las curvas de la figura 1. No ocurre lo mismo con la factorización incompleta, donde se registra casi un 50% de disminución en el número de iteraciones empleadas por los métodos no estacionarios. En la figura 2 se muestran las curvas de convergencia para estos métodos. Como la gráfica lo indica, CGS y BiCGSTAB convergen más rápido que los otros métodos y en menor tiempo. La excepción es GMRES, el cual podría converger en menos iteraciones. Lo que ocurre es que el parámetro de recomienzo debió tomarse muy pequeño ($m = 6$), por limitaciones de espacio de memoria.

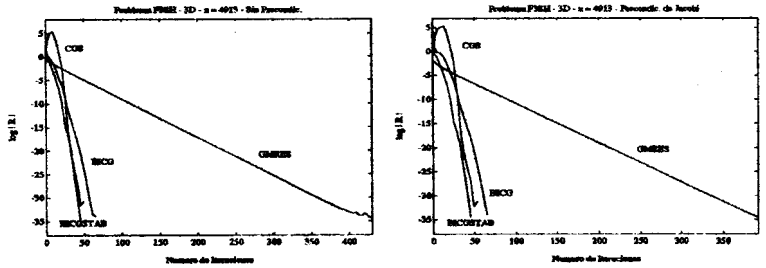

Figura 1: Evolución de la norma del residuo con las iteraciones. Izquierda: sin precondicionamiento; derecha: con precondicionamiento de Jacobi.

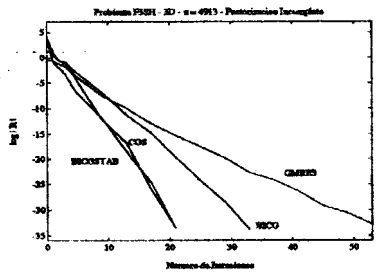

Figura 2: Evolución de la norma del residuo con las iteraciones utilizando factorización incompleta.

En cuanto al método SOR, se corrió para valores de $n = 125$, 512 y 1000 y para valores del parámetro de relajación comprendidos entre cero y dos. Se obtuvieron los siguientes resultados: para $n = 125$, ω debe ser mayor o igual que 0.2 y para $n = 512$ y $n = 1000$ debe ser $\omega \geq 0.4$. Los mejores resultados obtenidos son: para $n = 125$ con $\omega = 1.4$, SOR necesita 40 iteraciones para converger, cuando $n = 512$, y $\omega = 1.6$, 70 iteraciones y por último, si $n = 1000$ y se utiliza el mismo valor de ω, SOR requiere 72 iteraciones.

5.3 Problema PDE1:

Consiste en resolver

$$Lu = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + 20\,\exp(3.5\,(x^2+y^2))\frac{\partial u}{\partial x} + 70\,x\,\exp(3.5\,(x^2+y^2))\,u \qquad (17)$$

en la misma malla que el problema F2SH. La matriz del sistema que resulta es no simétrica, rala e indefinida de orden $n = 4900$ con 24220 elementos distintos de cero. Se permiten hasta 1225 iteraciones. Si no se utiliza precondicionamiento el problema es muy difícil de resolver. Por cierto, sólo se obtuvo convergencia con el empleo de factorización incompleta. Los métodos BiCGSTAB y CGS realizan 17 iteraciones en aproximadamente 4 segundos para obtener una reducción del residuo inicial de 10^{-15}. Por otro lado BiCG y GMRES necesitan 29 y 60 iteraciones respectivamente para alcanzar la misma precisión en la solución.

Para valores pequeños de n ($n \leq 125$), la aplicación de la factorización incompleta ofrece importantes resultados, con casi un 90 % de disminución de la cantidad de iteraciones empleadas por los métodos para alcanzar convergencia respecto a la versión sin precondicionamiento, y en algunos casos, logra

Tabla 1: *t1:* tiempo [sg] empleado con el procesador vectorial i860; *t2:* tiempo [sg] empleado sin el procesador vectorial i860

Problema PDE2

Métodos	I			II			III		
	k	$t1$	$t2$	k	$t1$	$t2$	k	$t1$	$t2$
BiCG	428	38	53	428	46	60	30	7	7
CGS	305	29	41	305	33	40	15	3	3
BiCGSTAB	-	-	-	-	-	-	16	4	4
GMRES	1072	70	99	1010	68	102	26	4	5

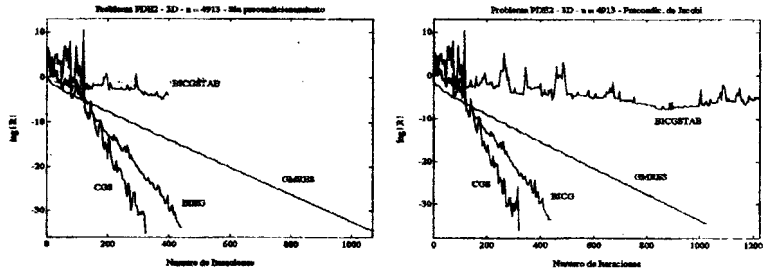

Figura 3: Evolución de la norma del residuo con las iteraciones. Izquierda: sin preconicionamiento; derecha: con precondicionamiento de Jacobi.

converger cuando antes no lo hacía. Sin embargo, se debe tener en cuenta que los tiempos mencionados no incluyen el tiempo que insume la construcción de la descomposición **LU** incompleta de **A**.

5.4 Problema PDE2:

Se resuelve el operador elíptico

$$\mathbf{L}u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + 1000\,\frac{\partial u}{\partial x} \tag{18}$$

cuya discretización lleva a solucionar un sistema con matriz no simétrica, rala e indefinida de orden $n = 4913$ y 32657 elementos diferentes de cero. La tabla 1 indica el número de iteraciones k y el tiempo empleado por cada algoritmo con y sin el uso de la placa vectorial i860. Se observa que la aplicación de la factorización incompleta ofrece excelentes resultados. A pesar de que se obtiene una disminución en el tiempo empleado al utilizar el procesador i860, en las versiones sin precondicionamiento y con precondicionamiento de Jacobi, esto se podría apreciar mejor para un problema de mayor tamaño.

Como se puede ver en los ejemplos anteriores, se confirma lo mismo con la implementación del precondicionamiento de Jacobi. Esto puede visualizarse en la figura 3. Se observa también que en general los métodos se comportan en una forma bastante irregular. Por otro lado, la aplicación de la factorización incompleta suaviza el comportamiento de los residuos (figura 4). Los métodos que obtienen resultados razonables en lo que respecta al número de iteraciones empleadas y el tiempo de ejecución de los mismos son CGS y BiCGSTAB.

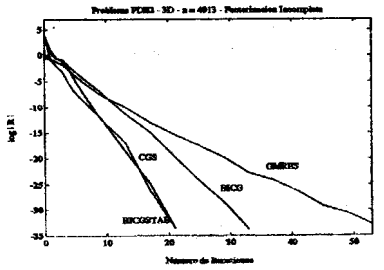

Figura 4: Evolución de la norma del residuo con las iteraciones utilizando factorización incompleta

Tabla 2: *t1:* tiempo [sg] empleado con el procesador vectorial i860; *t2:* tiempo [sg] empleado sin el procesador vectorial i860

Problema PDE3

Métodos	I			II		
	k	$t1$	$t2$	k	$t1$	$t2$
CGS	364	30	44	390	37	54
GMRES	-	-	-	382	23	35

5.5 Problema PDE3:

Se considera el operador

$$Lu = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + 1000 \exp(x\,y)\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) \tag{19}$$

cuya discretización en una malla de tamaño 70×70 lleva a resolver un sistema cuya matriz de coeficientes **A** es no simétrica, rala e indefinida de orden $n = 4900$ y con 24220 elementos no nulos. La generación de tal matriz es la que insume mayor cantidad de tiempo. Se consideró un máximo de 1225 iteraciones.
Este problema particular no pudo ser resuelto empleando factorización incompleta. Utilizando el precondicionamiento de Jacobi, convergen los métodos CGS y GMRES, éste último con recomienzo de las iteraciones cada 6 pasos del mismo. En la versión sin precondicionamiento, sólo converge CGS y con un comportamiento bastante irregular. La tabla 2 muestra la cantidad de iteraciones empleadas y el tiempo de ejecución de los mismos con y sin el uso de la placa vectorial i860.

6 Conclusiones

La resolución de los problemas propuestos, indica que los métodos iterativos no estacionarios prevalecen por sobre los estacionarios. Para los problemas con matriz simétrica y definida positiva, el método CG proporciona resultados razonables empleando menor espacio en memoria y realizando la mitad de operaciones que las que realizan los otros métodos. En el caso de sistemas con matriz no simétrica e indefinida, se destacan BiCGSTAB y CGS. En este último, el residuo suele tener un comportamiento bastante irregular. Si no se tienen limitaciones de memoria, otra opción es GMRES. Se ha probado para otros tamaños de problema que funciona tan bien como los otros métodos, aunque depende fuertemente del parámetro de recomienzo de las iteraciones.

Para ciertos sistemas lineales de gran dimensión, es imprescindible el empleo de precondicionamientos para poder llegar a una solución. Según las experiencias realizadas, factorización incompleta es una buena alternativa.

Agradecimientos

Este trabajo recibió apoyo económico de *Conicet* dentro del proyecto PID-BID 238 y de la Universidad Nacional del Litoral a través del proyecto CAI+D.

Referencias

[1] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine ans H. van der Vorst, "Templates for the solution of Linear Systems: Building Blocks for Iterative Methods", SIAM, Philadelphia, 1993.

[2] Kendall E. Atkinson, "An Introduction to numerical analysis (2nd edition)", John Wiley & Sons, 1988.

[3] G. H. Golub y C. Van Loan, "Matrix Computations (2nd edition)", The John Hopkins Univ. Press, Baltimore, MD (1993).

[4] 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, Philadelphia, 1991.

[5] Y. Saad and M. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems", SIAM J. Sci. Statist. Comput., **7** (1986), pp. 856-869.

[6] Y. Saad, "The Lanczos Biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems", SIAM J. Numer. Anal., **19** (1982), pp. 485-506.

[7] R. Freund, M. Gutknecht, and N. Nachtigal, "An implementation of the look - ahead Lanczos algorithm for non - Hermitian matrices", SIAM J. Sci. Comput., **14** (1993), pp. 137-158.

[8] P. Sonneveld, "CGS, a fast Lanczos - type solver for nonsymmetric linear systems", SIAM J. Sci. Statist. Comput., **10** (1989), pp. 36-52.

[9] H. van der Vorst, "Bi - CGSTAB: A fast and smothly converging variant of Bi - CG for the solution of nonsymmetric linear systems", SIAM J. Sci. Statist. Comput., **13** (1992), pp. 706-708.