FACTORIZACIÓN INCOMPLETA $IC(\ell, \tau, M)$ POR BLOQUES PARA MATRICES GENERADAS POR MÉTODOS "LOCAL DISCONTINUOUS GALERKIN"

Por

Carlos Andres Theran Suarez

Tesis sometida en cumplimiento parcial de los requerimientos para el grado de

MAESTRÍA EN CIENCIAS

en

COMPUTACIÓN CIENTIFICA

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ

2014

Aprobada por:

Erwin Suazo, Ph.D Miembro, Comité Graduado	Fecha
Marko Schutz, Dr Miembro, Comité Graduado	Fecha
Paul Castillo, Ph.D Presidente, Comité Graduado	Fecha
Silvina Cancelos, Ph.D Representante de Estudios Graduados	Fecha
Omar Colon, Ph.D Director del Departamento	Fecha

Resumen

En este trabajo se presenta una factorización incompleta de Cholesky por bloques LL^T. Esta factorización es discutida como una técnica de precondicionamiento, la cual mejora el condicionamiento espectral de la matriz, lo que implica una disminución en el número de iteraciones requeridas para la convergencia de métodos iterativos como lo es el método del Gradiente Conjugado Precondicionado (**PCG**). Para la construcción del precondicionador se considera una técnica de niveles propuesta en [18] la cual calcula la estructura de elementos no cero del precondicionador, este proceso es conocido como fase simbólica. Uno de los problemas que genera la construcción de un precondicionador para grandes sistemas lineales es la gran cantidad de memoria que se requiere para el almacenamiento de L. Para controlar el consumo de memoria, se usa dos paramétros: el parámetro umbral τ y el parámetro de memoria m, los cuales permiten retener elementos no ceros del factor L. Una versión por bloque de estas técnicas es propuesta para matrices simétricas definidas positivas generadas por el método Local Discontinuous Galerkin.

Abstract

This work presents a incomplete Cholesky factorization by blocks LL^T. This factorization is discussed as a preconditioning technique which improves the spectral condition of the matrix. This provides a reduction of the number of iterations required for the convergence of iterative methods such as the preconditioned conjugated gradient method. For the construction of the preconditioner a proposed technique of levels is considered in [18] which the structure of non zero elements of the preconditioner are calculated, this process is known as the symbolic phase. One of the issues created by the construction of a preconditioner for large lineal systems is the amount of memory required for the storage of L. To be able to control the memory uptake two parameters are used: the threshold parameter τ and the memory parameter m, which allow the elimination of none zero elements of the L factor. A block version of these techniques is proposed for symmetric positive definite matrices generated by the Local Discontinuous Galerkin method.

AGRADECIMIENTOS

Este trabajo esta dedicado a todas las personas que fueron parte de mi vida durante estos tres últimos años.

Índice general

página

Índice de tablas

Tabla página

Índice de figuras

Figura página

LISTA DE ABREVIATURAS

DG	Discontinuous Galerkin.
LDG	Local Discontinuous Galerkin.
CSR	Compressed Sparse Row.
CG	Conjugate Gradient.
PCG	Preconditioned Conjugate Gradient.

Capítulo 1
INTRODUCCIÓN

Uno de los problemas numéricos que surge con más frecuencia en el cálculo científico, es la solución de grandes sistemas lineales cuya matriz asociada es hueca (matriz cuya mayoría de entradas es cero). Este tipo de sistemas son generados por la discretización de ecuaciones diferenciales parciales por diferentes métodos numéricos como lo son Diferencias Finitas, Volumen Finito y Elemento Finito.

Estos métodos culminan resolviendo un sistema lineal $Ax = b$ donde A es una matriz de grandes dimensiones. El solucionar este tipo de sistemas de manera eficiente es uno de los objetivos más importantes. Cuando A es simétrica y definida positiva los métodos más conocidos y efectivos para solucionar este tipo de sistemas son el método del Gradiente Conjugado **GC** [24], el método del Gradiente Conjugado Precondicionado **PCG** [26] y se podría considerar el método de Descenso. El número de iteraciones requeridas para alcanzar una tolerancia fija por estos métodos dependen del condicionamiento espectral de la matriz, es decir, entre mayor sea el condicionamiento numérico de la matriz mayor será el número de iteraciones necesarias para la convergencia fijada una tolerancia. Algunos de los métodos de discretización proveen una matriz con un mal condicionamiento espectral, de aquí la necesidad de mejorar el condicionamiento del sistema.

La idea usual para reducir el condicionamiento espectral es el uso de un precondicionador (obtener una matriz M la cual es una aproximación de A, tal que

la matriz del sistema $M^{-1}Ax = M^{-1}b$ tenga un mejor condicionamiento espectral). Una de las técnicas más simples para precondicionar es utilizar la factorización incompleta de tipo LU, donde L es una matriz triangular inferior unitaria y U es una matriz triangular superior. La idea básica de esta factorización es realizar una descomposición LU de A sin alterar los elementos ceros de A durante la factorización.

Para matrices simétricas y definidas positiva, un precondicionador que se usa frecuentemente es la factorización incompleta de Cholesky **IC** (factorización incompleta que permite mantener los elementos ceros de la matriz original realizando una descomposición de Cholesky LL^T de la matriz A, donde L es una matriz triangular inferior).

Una de las preocupaciones es la esparcidad de A, debido a que se desea mantener dicha esparcidad pero a su vez mejorar el condicionamiento numérico de esta. ¿Que patrón de esparcidad debería tener el factor L para obtener un eficiente precondicionador el cual mejore el condicionamiento numérico y el consumo de memoria?. Para alcanzar estos objetivos se hace uso de una factorización incompleta de Cholesky. Meijerink y Vorst presentan en [11] dos estrategias, la primera es mantener el patrón de esparcidad de la matriz original A, esta técnica de precondicionamiento permite mantener los elementos ceros de A durante un paso de la factorización completa de Cholesky por medio de estrategias predeterminadas, esta factorización es conocida como *IC(0)*, la segunda permite algunas entradas de llenado (remplazar aquellos elementos ceros de A por un elemento no cero), algunos autores definen el patrón de esparcidad tal que L sea una matriz de banda, con esta estrategia es posible predecir la cantidad de memoria requerida pero esto implica que los elementos eliminados

sólo dependan de la estructura de A y no de las entradas de A, lo cual podría eliminar elementos grandes y dejar elementos casi ceros.

Jones y Plassmann en [29] proponen aproximar un número fijo de elementos no ceros por columnas ignorando el patrón de esparcimiento de A, reteniendo sólo las entradas de mayor magnitud, de esta manera el espacio en memoria es controlada.

A pesar de las técnicas propuestas para la construcción de un precondicionador Kershaw en [15] muestra que la factorización incompleta de Cholesky no siempre existe. En general la interrogante es saber cuando una factorización de Cholesky existe dado el caso que al actualizar los elementos de la parte triangular inferior de la matriz original de A se están omitiendo posibles entradas no ceros al factor L, con esto no hay garantía que las submatrices resultantes sean definidas positivas la cual es una de las condiciones para la existencia de dicha factorización.

Algunas estrategias para garantizar la estabilidad de la factorización incompleta de Cholesky han sido propuestas por Meijerink y Vorst [11], Mateuffel [31, 32], Lin y Moré [17] y Fan [30]. Meijerink y Vorst garantizan la estabilidad de una factorización incompleta de Cholesky para M-matrices, Lin y Moré extienden estos resultados mostrando la existencia de una factorización de Cholesky para H-matrices. Manteuffel [32] demuestra que la factorización incompleta de Cholesky es posible para matrices que son diagonal dominantes y definida positivas y también es estable para matrices Stieltjes y además que sean definidas positivas.

Una de las razones por la cual la factorización incompleta de Cholesky falla es que el pivote $l_{ii} \leq 0$, esta condición inhabilita la posibilidad de calcular $\sqrt{l_{ii}}$.

La técnica utilizada en este trabajo es alterar aquellos elementos $l_{ii} \in \vartheta = \{l_{ii} | l_{ii} \leq 0, i = 1, 2, \ldots, n\}$ de tal manera que $\mid l_{ii} \mid > \sum_{\substack{j=0 \\ j \neq i}}^{n} \mid l_{ij} \mid^2$, esto podría convertir la matriz A en una matriz estrictamente diagonal dominante, pero solamente se altera para aquellas filas donde $l_{ii} \in \vartheta$. Esta técnica es conocida con el nombre de *desplazamiento* la cual fue propuesta inicialmente por Manteuffel, el cual realiza una descomposición de la matriz original A a la descomposición de Jacobbi.

$$A = I - B$$

y se considera una descomposición.

$$A(\alpha) = I - \frac{1}{1+\alpha} B$$

Para un valor de α suficientemente grande $A(\alpha)$ será estrictamente diagonal dominante de esta manera la factorización incompleta de Cholesky de $A(\alpha)$ será estable.

Otra técnica para garantizar la existencia de una factorización incompleta de Cholesky realizando un desplazamiento fue propuesta por Jennings y Malik [33] asignando.

$$a_{ii}^k = a_{ii}^k + \sigma \mid a_{ij}^k \mid, \quad a_{jj}^k = a_{jj}^k + \frac{1}{\sigma} \mid a_{ji}^k \mid$$

Si a_{ij}^k es descartado durante el k-esimo paso bajo la hipótesis que A es definida positiva. Por otro lado Hladíd, Reed y Swoboda en [34] sugieren el uso de un parámetro $\omega \in [0, 1]$ donde

$$a_{ii}^k = a_{ii}^k + \omega \mid a_{ij}^k \mid, \quad a_{jj}^k = a_{jj}^k + \omega \mid a_{ji}^k \mid$$

Gustafsoon en [23] introduce un concepto de niveles para establecer el patrón de esparcimiento S_k del factor L_k. El patrón de esparcidad lo define de la siguiente manera, S_0 es el patrón de esparcimiento de A, luego se toma

$$S_{k+1} = S_k \cup R_k$$

donde R_k es el patrón de esparcimiento de $L_k L_k^T$. Por ejemplo el nivel-1 permite que cualquier elemento no cero de la matriz original realice un aporte (ingreso de un elemento no cero al factor L) al factor L_1 e introducir directamente cualquier elemento no cero durante la eliminación, el nivel-2 podría incluir el nivel-1 de llenado y cualquier elemento no cero introducido directamente por la eliminación de los elementos no ceros del nivel-1. Este método necesita determinar el patrón de esparcimiento y la cantidad de memoria requerida. La factorización simbólica puede ser usada para determinar la posición de los elementos no ceros y la cantidad de memoria requerida. El uso de esta técnica puede generar un rápido incremento en el número de elementos no ceros de L_k a L_{k+1}.

D'Azevedo, Forsyth y Tang proponen en [16] una estrategia para calcular el patrón de esparcimiento por medio de *reachable set* y *Shortest path length*, otra técnica para calcular el patrón de esparcimiento es propuesta en [19] por Hyson y Photen la cual hace uso de *incomplete path theorem*. Una estrategia alterna para establecer las entradas de llenado es el parámetro de umbral conocido como tolerancia-umbral (*drop-tolerance*), esta estrategia tiene la ventaja que depende las entradas de A. Una de las estrategias con este parámetro fue propuesta en [14] por Munksgaard, una desventaja de esta estrategia es que la memoria que se requiere es impredecible. Si el umbral es grande el factor L tendrá menos entradas cero pero esto generaría un pobre precondicionador.

Una de las estrategia la cual controla la cantidad de memoria utilizada es la propuesta por Lin y Moré, el cual retiene los $l_k + m$ elementos más grandes en el

factor L, donde m es un parámetro de memoria.

Con la idea de construir un buen precondicionador el cual reune las técnicas antes mencionadas, niveles, parámetro umbral y parámetro de memoria. En [18] Scott y Tüman construyen un precondicionador donde los niveles son asignados a cada elemento según su magnitud, el parámetro de memoria retiene los m elementos mas grandes por columnas y el parámetro umbral retiene aquellos elementos cuya magnitud no exceda el umbral, esta factorización incompleta fue realizada a nivel puntual.

En este trabajo A será generada por el método Local Discontinuous Galerkin (**LDG**) aplicado a problemas elípticos [6, 7]. Este método genera matrices por bloque, además tienen la característica de ser simétricas, definida positivas y huecas. Cuando las matrices son obtenidas por el método **LDG**, Castillo en [9] mostró de manera teórica y computacional que el condicionamiento espectral de la matriz de rigidez tienen un comportamiento asintótico de $\mathrm{O}(h^{-2})$, de aquí la necesidad de mejorar el condicionamiento espectral de A.

Se propone generar una factorización incompleta de Cholesky por bloque como técnica de precondicionamiento para mejorar el condicionamiento espectral de la matriz generada por **LDG**, permitiendo un menor número de iteraciones al momento de solucionar el sistema lineal. Esta factorización incompleta de Cholesky permitirá la entrada de elementos no ceros durante la factorización en el factor L por medio de tres parámetros; niveles l, umbral τ y memoria m. Para observar el rendimiento de la factorización incompleta por bloque se compara con su versión puntual y con la versión puntual $IC(0)$ de Matlab.

Capítulo 2
MÉTODOS ITERATIVOS Y PRECONDICIONAMIENTO

El método directo de Eliminación Gaussiana [35, 36] es una técnica bastante utilizada, sin embargo su complejidad aritmética es $O(n^3)$. Por otra parte el utilizar eliminación Gaussiana modifica los elementos cero de la matriz en elementos no cero generando un alto consumo de memoria para su almacenamiento. Para evitar este problema se hace uso de métodos iterativos como por ejemplo, el método del Gradiente Conjugado y el método del Gradiente Conjugado Precondicionado.

La convergencia de estos métodos iterativos depende en gran manera del condicionamiento espectral de la matriz. Es necesario hacer las siguientes definiciones para mostrar algunos resultados de convergencia.

Definición 2.0.1. *Dado una norma* $\| \cdot \|$ *sobre un espacio* $\mathbb{R}^n$ *de vectores n-dimensional con entradas reales, la norma matricial subordinada sobre el espacio* $\mathbb{R}^{n\times n}$ *de* $n \times n$ *matrices con entradas reales esta definida por.*

$$\| A \| = \max_{\boldsymbol{v}\in\mathbb{R}^n_*} \frac{\| A\boldsymbol{v} \|}{\| \boldsymbol{v} \|}$$

Algunos ejemplos de normas matriciales:

- $\| A \|_\infty = \max_{i=1}^n \sum_{j=1}^n | a_{ij} |$
- $\| A \|_1 = \max_{j=1}^n \sum_{i=1}^n | a_{ij} |$
- $\| A \|_2 = \max_{i=1}^n \sqrt{|\sigma_i|}$

donde σ_i para $i = 1, \ldots, n$ son autovalores de $A^T A$, denotando a $\sigma_{max} = \max_{i=1}^{n}\{\sigma_i\}$ [10].

Definición 2.0.2. *Sea $A \in \mathbb{R}^{n\times n}$ una matriz no singular, la condición espectral de A relativa a $\|\cdot\|_2$ está definida por*

$$\kappa_2(A) = \| A \|_2 \ \| A^{-1} \|_2$$

Proposición 2.0.1. *Para una matriz no singular $A \in \mathbb{R}^{n\times n}$*

$$\kappa_2(A) = \left(\frac{\sigma_{max}}{\sigma_{min}}\right)^{1/2}$$

donde σ_{max} y σ_{min} denota el máximo y mínimo autovalor de $A^T A$ respectivamente.

En el caso que A sea simétrica y definida positiva se tiene que $\kappa_2(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$.

2.1. Método Descenso Rápido

El método de Descenso Rápido se interpreta como un proceso de proyección unidimensional, el cual aproxima la solución de $Ax = b$ por medio de la iteración $x = x + \alpha r$, donde $r = b - Ax$.

Cada paso obtenido por el método de Descenso Rápido minimiza

$$f(x) = \| x - x^* \|_A^2 = (A(x - x^*), (x - x^*))$$

Sobre todo los vectores de la forma $x + \alpha d$, donde d es la dirección negativa de $-\nabla f$. La dirección negativa del gradiente permite acelerar el decrecimiento de f, la garantía de la convergencia esta sujeta a que A sea una matriz simétrica y definida positiva, permitiendo obtener.

$$\begin{aligned}(r - \alpha Ar, r) &= 0\\ (r,r) - \alpha(Ar,r) &= 0\\ \alpha &= \frac{(r,r)}{(Ar,r)}\end{aligned}$$

Algoritmo 1 Rapido descenso

Entrada: $A, b, x, tol, maxiter, r = b - Ax, p = Ar$

1: **para** $i = 1$ to $maxiter$ **hacer**
2: $\alpha \leftarrow \frac{(r,r)}{(Ar,r)}$
3: $x \leftarrow x + \alpha r$
4: $r \leftarrow r - \alpha p$
5: $p := Ar$
6: $e = \frac{\|r\|}{\|b\|}$
7: **si** $e \leq tol$ **entonces**
8: break
9: **fin si**
10: **fin para**

2.2. Método Gradiente Conjugado

El método del Gradiente Conjugado [24] es un método para minimizar la siguiente función cuadrática

$$\hat{x} = \min_{x \in \mathbb{R}^n} \psi(x), \quad \psi(x) = \frac{1}{2} x^T Ax - b^T x. \tag{2.1}$$

Donde $A \in \mathbb{R}^{n \times n}$ es simétrica, definida positiva y $b \in \mathbb{R}^n$. Sin pérdida de generalidad se puede ver a x como una variable unidimensional obteniendo

$$\nabla\psi(x) = Ax - b, \quad \nabla^2\psi = A \ \text{ matriz Hessiana.} \tag{2.2}$$

Dado que la matriz Hessiana es simétrica y definida positiva, existe un único $\hat{x}$ que minimiza la función $\psi(\cdot)$, lo que implica que es solución del sistema $Ax = b$.

El método del Gradiente Conjugado aproxima a $\hat{x}$ con un vector x_j y una dirección p_j

$$x_{j+1} = x_j + \alpha_j p_j, \;\; \alpha_j = \min \psi(x_j + \alpha p_j) \;\; \text{con} \;\; \alpha \in \mathbb{R}. \tag{2.3}$$

El siguiente teorema es considerado la base para la construcción del método del Gradiente Conjugado.

Teorema 2.2.1. *Si la dirección definida en (2.3) satisfacen que* $p_j^T A p_i = 0$ *para* $i \neq j$ *y los* $\{x_j\}_{j=0}^{i}$ *son obtenidos por (2.3) entonces*

$$x_j = \min_{x=y+\alpha p_j} \psi(x), \;\; \textit{para todo} \;\; 1 \leq j \leq k$$

Algoritmo 2 Gradiente Conjugado (CG)

Entrada: $r_0 = Ax_0 - b, p_0 = -r_0, x, tol, maxiter$

para $j = 1$ to $maxiter$ **hacer**
  $\alpha_j \leftarrow \frac{(r_j, r_j)}{(Ap_j, p_j)}$
  $x_{j+1} \leftarrow x_j + \alpha_j p_j$
  $r_{j+1} \leftarrow r_j - \alpha_j A p_j$
  $\beta_j \leftarrow \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}$
  $p_{j+1} = -r_{j+1} + \beta_j p_j$
  $e = \frac{\|r\|}{\|b\|}$
  si $e \leq tol$ **entonces**
    break
  fin si
fin para

A continuación se presenta una estimación de la convergencia del método del Gradiente Conjugado.

Teorema 2.2.2. *El error despues de n iteraciones del algoritmo del método del gradiente conjugado puede ser acotado como sigue:*

$$\| x_* - x_n \|_A \leq \frac{2}{\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^n + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n} \| x_* - x_0 \|_A \leq 2 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n \| x_* - x_0 \|_A$$

donde $\kappa = \kappa(A) = \frac{\sigma_m}{\sigma_1}$ *es el condicionamiento espectral de* A.

Demostración. Ver [37, 38]. □

2.3. Método Gradiente Conjugado Precondicionado

Precondicionar es transformar el sistema lineal original en uno equivalente que tenga un condicionamiento numérico agrupado en 1; con el propósito de reducir el número de iteraciones. Sea M una matriz no singular y un precondicionador de A, la matriz M debe satisfacer algunas condiciones pero lo más relevante es que el $\sigma(M^{-1}A) \approx 1$.

El precondicionador puede ser aplicado por la izquierda del sistema original obteniendo así

$$M^{-1}Ax = M^{-1}b. \tag{2.4}$$

Una manera alternativa es aplicar el precondicionador M por la derecha

$$AM^{-1}u = b, \;\; x = M^{-1}u. \tag{2.5}$$

En ocasiones el precodicionador es factorizado de la siguiente forma

$$M = M_I M_D$$

Donde usualmente M_I y M_D son matrices triangulares, en este caso se puede precondicionar como

$$M_I^{-1}AM_D^{-1}u = M_I^{-1}b, \;\; x = M_D^{-1}u. \tag{2.6}$$

Considerese una matriz A simétrica y definida positiva, además se asume que M es un precodicionador de A, dado que A es simétrica es imperativo preservar la simetría de A para la convergencia de **CG**, más aún, si se asume que M es simétrica y definida positiva el sistema puede ser precondicionado como (2.4), (2.5), (2.6).

Una manera de preservar simetría es trabajar con respecto a otro producto interio. Por ejemplo, sea M una matriz simétrica y definida positiva, se puede considerar el M-producto interior definido como sigue

$$(x, y)_M \equiv (Mx, y) \equiv (x, My), \quad \forall x, y \in \mathbb{R}^n.$$

Con esto $M^{-1}A$ es autoadjunto o simétrica con respecto al producto interior en $(\cdot)_M$

$$(M^{-1}Ax, y)_M = (Ax, y) = (x, Ay) = (x, M(M^{-1}A)y) = (x, M^{-1}Ay)_M.$$

Así el método de Gradiente Conjugado puede ser escrito en este producto interior, definiendo $r_j = b - Ax_j$ el residuo original y $z_j = M^{-1}r_j$ el residuo del sistema precondicionado, se tiene $(z_j, z_j)_M = (r_j, z_j)$ y $(M^{-1}Ap_j, p_j)_M = (Ap_j, p_j)$. Con esto se obtiene el algoritmo 3.

Algoritmo 3 Gradiente Conjugado Precondicionado (PCG)

Entrada: $r_0 = b - A, z_0 = M^{-1}r_0, p_0 = z_0, x, tol, maxiter$

1: **para** $j = 1$ to $maxiter$ o $e \leq tol$ **hacer**
2: $\quad \alpha_j \leftarrow \frac{(r_j, z_j)}{(Ap_j, p_j)}$
3: $\quad x_{j+1} \leftarrow x_j + \alpha p_j$
4: $\quad r_{j+1} \leftarrow r_j - \alpha_j Ap_j$
5: $\quad z_{j+1} \leftarrow M^{-1}r_{j+1}$
6: $\quad \beta_j \leftarrow \frac{(r_{j+1}, z_{j+1})}{(r_j, z_j)}$
7: $\quad p_{j+1} \leftarrow z_{j+1} + \beta_j p_j$
8: $\quad e = \frac{\|r\|_2}{\|b\|_2}$
9: **fin para**

Uno de los métodos clásicos para precondicionar es la factorización incompleta de Cholesky estudiada por Meijerink y Van der Vorst [11].

Capítulo 3
FACTORIZACIÓN INCOMPLETA DE CHOLESKY

Una manera de solucionar sistemas lineales $Ax = b$, con A simétrica y definida positiva, es descomponer la matriz A en un producto de dos matrices LL^T, donde L es una matriz triangular inferior. El problema se reduce a solucionar un sistema triangular superior y uno triangular inferior, optimizando el número de operaciones necesarias para solucionar el sistema.

3.1. Factorización completa de Cholesky

Teorema 3.1.1. *Sea $A \in \mathbb{R}^{(n+1)\times(n+1)}$ simétrica y definida positiva, luego su factorización completa de Cholesky existe y es de la forma.*

$$A = \begin{pmatrix} a_{11} & \hat{a}_1^T \\ \hat{a}_1 & B_1 \end{pmatrix} = \begin{pmatrix} l_1 & \hat{0}^T \\ \hat{l}_1 & L_1 \end{pmatrix} \cdot \begin{pmatrix} l_1 & \hat{l}_1^T \\ \hat{0} & L_1^T \end{pmatrix}$$

Se define $l_1, a_{11} \in \mathbb{R}$ donde $\hat{0}, \hat{l_1}, \hat{a_1} \in \mathbb{R}^n$ con $L_1 \in \mathbb{R}^{n\times n}$, tales que

$$\begin{aligned} a_{11} &= l_1 \cdot l_1 \Rightarrow l_1 = \sqrt{a_{11}} \\ \hat{a}_1 &= l_1 \cdot \hat{l}_1 \Rightarrow \hat{l}_1 = \frac{\hat{a_1}}{l_1} \\ \hat{a}_1^T &= l_1 \cdot \hat{l}_1^T \\ B_1 &= \hat{l}_1 \cdot \hat{l}_1^T + L_1 L_1^T \Rightarrow L_1 \cdot L_1^T = (B_1 - \hat{l}_1 \hat{l}_1^T)\cdot \end{aligned}$$

Ahora si A es definida positiva se tiene que B_1 es también definida positiva, si se toma $\hat{y} \in \mathbb{R}^n_*$ tal que $x = (0 \ \ \hat{y})^T \in \mathbb{R}^{n+1}$ se tiene

$$\begin{aligned}\begin{pmatrix} 0 & \hat{y}^T \end{pmatrix} \begin{pmatrix} a_{11} & \hat{a}_1^T \\ \hat{a}_1 & B_1 \end{pmatrix} \begin{pmatrix} 0 \\ \hat{y}^T \end{pmatrix} &= \begin{pmatrix} \hat{y}^T \cdot \hat{a}_1 & \hat{y} \cdot B_1 \end{pmatrix} \begin{pmatrix} 0 \\ \hat{y}^T \end{pmatrix} \\ &= \hat{y}^T B \hat{y} \\ &> 0\end{aligned}$$

de esta manera B_1 es definida positiva. Veamos que $L_1 L_1^T$ es definida positiva, sea $\hat{x} \in \mathbb{R}^n_*$ luego

$$\hat{x}^T L_1 \cdot L_1^T \hat{x} = (\hat{x}^T L_1)(\hat{x}^T L_1)^T = \| \hat{x}^T L_1 \|^2 > 0$$

esto implica que $L_1 L_1^T$ es una matriz definida positiva y además simétrica lo que garantiza la existencia de la factorización de Cholesky. Una de las formas que aparece en la literatura para la factorización de Cholesksy es LDL^T, donde D es una matriz diagonal.

Sea $A_1 = L_1 L_1^T$ de manera que

$$A = \left(\begin{array}{c|c|c} 1 & 0 & \hat{0}^T \\ \hline & 1 & \hat{0}^T \\ \hline \hat{l}_1 & & \\ & \hat{l_2} & I \end{array}\right) \cdot \left(\begin{array}{cc|c} a_{11} & & \\ & a_{22} & \hat{0}^T \\ \hline & & \\ \hat{0} & & A_1 \end{array}\right) \cdot \left(\begin{array}{c|c|c} 1 & 0 & \hat{l_1}^T \\ \hline & 1 & \hat{l_2}^T \\ \hline \hat{0} & & \\ & \hat{0} & I \end{array}\right)$$

dado que A_1 es simétrica y definida positiva existe la factorización de Cholesky. Repitiendo el proceso se obtiene $A_2 = L_2 L_2^T$, la cual es simétrica y definida positiva, de esta manera se obtiene $A = LL^T$, donde L es una matriz triangular inferior. El

algoritmo 4 presenta una factorización completa de Cholesky.

Algoritmo 4 Factorización completa de Cholesky

```
1: para j = 1 : n hacer
2:     l_jj = √a_jj
3:     para k = 1 : j − 1 hacer
4:         para i = j + 1 : n hacer
5:             l_ij = l_ij − l_ik l_jk
6:         fin para
7:     fin para
8:     para i = j + 1 : n hacer
9:         l_ij = l_ij / l_jj
10:        a_ii = a_ii − l_ij^2
11:    fin para
12: fin para
```

Cuando la matriz A es hueca, durante el proceso $L_1 L_1^T = B_1 - \hat{l_1} \cdot \hat{l_1}^T$ el patrón de esparcimiento de $L_1 L_1^T$ varia de B_1, esto debido al aporte proporcionado por la matriz $\hat{l_1} \cdot \hat{l_1}^T$ la cual es una matriz de rango 1. Esto indica que la matriz es no cero pues $\hat{l}_1 \in \mathbb{R}^n_*$. De esta manera la matriz triangular inferior L tendrá más elementos no ceros que la parte triangular inferior de A, generando un alto costo en memoria e incrementando el costo al resolver un sistema triangular.

Como se mostró en la sección (2.3) una manera de mejorar el condicionamiento espectral de la matriz y a su vez disminuir el número de iteraciones requerida por **PCG**, es obtener un buen precondicionador M. Esta matriz M es generada por diferentes técnicas conocidas como *factorizaciones incompletas*. Dado que el método **LDG** genera matrices simétricas y definidas positivas, se presenta una factorización incompleta de Cholesky como técnica de precondicionamiento.

3.2. Factorización incompleta de Cholesky

Dado que una factorización incompleta de Cholesky no siempre existe; ver [15], se presentan las propiedades necesarias que requiere una matriz A para garantizar

la estabilidad y existencia de una factorización incompleta de Cholesky.

La interrogante es saber cuando una factorización incompleta de Cholesky existe, dado que durante el proceso de actualizar los elementos de la parte triangular inferior de A se omiten posibles entradas no cero en el factor L, causando que la matriz generada en cada paso de la factorización no sean definidas positivas.

A continuación se presenta un teorema que garantiza la existencia de una factorización incompleta de Cholesky.

Definición 3.2.1. *Una matriz A es una M-matriz si $a_{ij} \leq 0$ para $i \neq j$, A es no singular y su inversa es no negativa ($A^{-1} \geq 0$).*

Definición 3.2.2. *Dadas las matrices $A, K, R \in \mathbb{R}^{n\times n}$ tal que $A = K - R$ es una descomposición regular de A si K es no singular, $K^{-1} \geq 0$ y $R \geq 0$.*

Teorema 3.2.1. *Si A es una M-matriz y además simétrica, existe una única matriz triangular inferior L y una matriz simétrica no negativa R, donde $l_{ij} = 0$ si $a_{ij} = 0$ y $r_{ij} = 0$ si $a_{ij} \neq 0$, tal que la descomposición $A = LL^T - R$ es una descomposición regular.*

Demostración. Ver prueba en [11] □

El resultado teorema 3.2.1 fue extendido para H-matrices por Lin y Móre [17]. Se presentan algunas condiciones suficientes para garantizar que la factorización incompleta de Cholesky no falle, es decir, sea estable.

Definición 3.2.3. *Se dice que una factorización incompleta de Cholesky es estable si*

$$l_{ii} > 0, \quad i = 1, 2, 3, \ldots, n.$$

Los elementos l_{ii} no pueden ser muy pequeños porque esto permite que M sea casi singular generando una matriz $M^{-1}A$ con un mal condicionamiento espectral.

Teorema 3.2.2. *Si A es diagonal dominante y definida positiva, entonces la factorización incompleta de Cholesky es estable.*

La estabilidad de una factorización incompleta depende en gran manera en el patrón de esparcimiento del factor L, si se permite mayor número de entradas no ceros, la factorización incompleta se aproxima a una factorización completa la cual es estable para cualquier matriz definida positiva.

Una de las primeras técnicas para producir un precondicionador por medio de una factorización incompleta de Cholesky fue preservar el patrón de esparcimiento de A, la cual es conocida como factorización incompleta de Cholesky **IC(0)** [11].

3.2.1. Factorización incompleta $IC(0)$

Esta técnica permite conservar la misma cantidad de elementos ceros de A durante el proceso de una factorización completa de Cholesky la cual fue presentada en [11]. Se define

$$S = \{a_{ij} | a_{ij} \neq 0 \quad \text{donde} \quad i, j \in \mathbb{N}\}$$

Como el conjunto de todos los elementos no ceros de la matriz A. La modificación que se efectúa en el algoritmo 4 para generar $IC(0)$ consiste en actualizar aquellos elementos l_{ij} del factor L si el elemento $a_{ij} \in S$, el algoritmo 5 produce $IC(0)$.
La estrucura de las matrices generadas por la discretización de ecuaciones diferenciales parciales por el método Local Discontinuous Galerkin para mayas conformes, se muestran en la figura 3-1.

Con la idea de generar más entrada de elementos no cero de las que aporta el patrón de esparcimiento de A en el factor L, se produce una técnica que utiliza un parámetro el cual permite obtener más elementos no ceros en el factor L del que

puede aportar $IC(0)$, esta técnica es conocida como $IC(\tau)$.

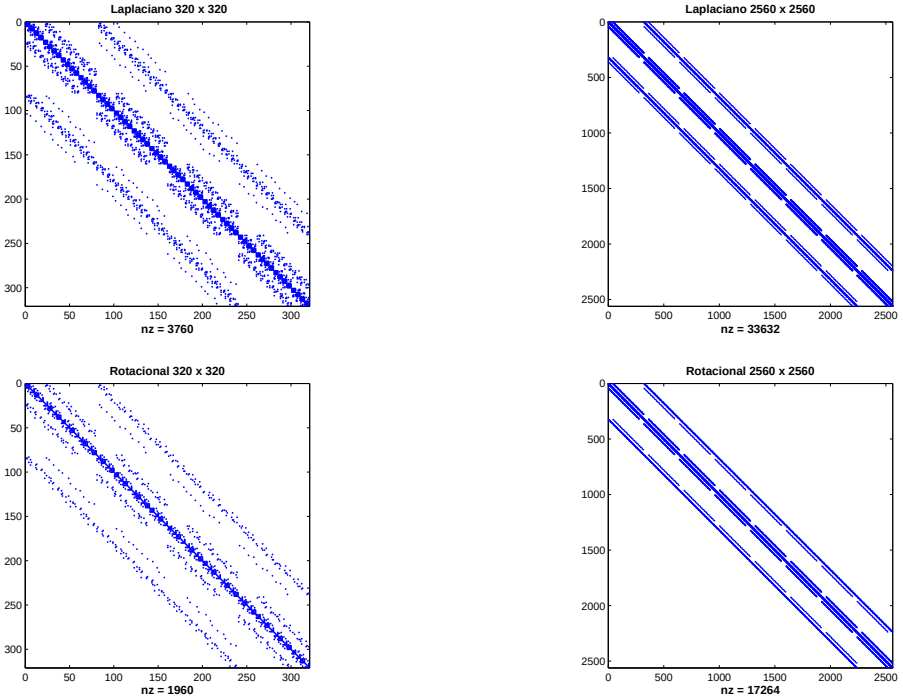


Figura 3–1: Matrices Generadas por el método **LDG**, para cierto tipo de mallas.

Algoritmo 5 Factorización incompleta de Cholesky $IC(0)$

para $j = 1 : n$ **hacer**
  $l_{jj} = \sqrt{a_{jj}}$
  para $k = 1 : j - 1$ **hacer**
    para $i = j + 1 : n$ **and** $a_{ij} \in S$ **hacer**
      $l_{ij} = l_{ij} - l_{ik} l_{jk}$
    fin para
  fin para
  para $i = j + 1 : n$ **hacer**
    $l_{ij} = l_{ij} / l_{jj}$
    $a_{ii} = a_{ii} - l_{ij}^2$
  fin para
fin para

3.2.2. Factorización incompleta $IC(\tau)$

Se propuso una técnica en [14] que permite introducir elementos no ceros en el factor L si estos elementos exceden un parámetro umbral τ. Se eliminan los elementos a_{ij}^k durante el k-esimo paso, si

$$| a_{ij}^k | \leq \tau \sqrt{a_{ii}^k a_{jj}^k}.$$

Donde τ es una tolerancia preestablecida. Otras estrategias para eliminar elementos con el parámetro umbral τ pueden ser vistas en [10], donde τ_k se ha definido como el producto de τ por la norma 2 de la k-esima fila de A.

Esta factorización incompleta es conocida como $IC(\tau)$. Una desventaja de eliminar entradas por medio de la estrategia del parámetro umbral es que la memoria requerida es no predecible. No existe una manera para determinar el valor de τ tal que la memoria requerida tenga una cota. En este trabajo si el valor de τ aumenta se produce menos elementos no cero en el factor L, por el contrario si el valor de τ disminuye se permite más entradas no ceros en el factor L.

El algoritmo 6 genera una factorización incompleta de Cholesky permitiendo entradas por medio del parámetro umbral τ.

Algoritmo 6 Factorización incompleta de Cholesky $IC(\tau)$

para $j = 1 : n$ **hacer**
  $l_{jj} = \sqrt{a_{jj}}$
  $w = a_{j+1:n,j}$
  para $k = 1 : j - 1$ **hacer**
    para $i = j + 1 : n$ **and** $l_{jk} \in S$ **hacer**
      $w_i = w_i - l_{ik} l_{jk}$
    fin para
  fin para
  para $i = j + 1 : n$ **hacer**
    $w_i = w_i / l_{jj}$
  fin para
  $\tau_j = \tau \parallel w \parallel$
  para $i = j + 1 : n$ **hacer**
    $w_i = 0$ cuando $\mid w_i \mid < \tau_j$
  fin para
  para $i = j + 1 : n$ **hacer**
    $l_{i,j+1} = w_i$
  fin para
  para $i = j + 1 : n$ **hacer**
    $a_{ii} = a_{ii} - l_{ij}^2$
  fin para
fin para

Una de las estrategias para controlar la cantidad de memoria utilizada fue propuesta por Lin y Móre en [17] la cual es conocida como $IC(m)$.

3.2.3. Factorización incompleta $IC(m)$

Esta factorización incompleta de Cholesky hace uso de un parámetro de memoria m el cual controla la cantidad de elementos permitidos por columnas. Solo se guardarán los m elementos más grandes en valor absoluto, permitiendo eliminar elementos muy pequeños. El algoritmo 7 desarrolla una factorización incompleta de Cholesky haciendo uso del parámetro de memoria.

Algoritmo 7 Factorización incompleta de Cholesky $IC(m)$

para $j = 1 : n$ **hacer**
  $l_{jj} = \sqrt{a_{jj}}$
  $w = a_{j+1:n,j}$
  para $k = 1 : j - 1$ **hacer**
    para $i = j + 1 : n$ **and** $l_{jk} \in S$ **hacer**
      $w_i = w_i - l_{ik} l_{jk}$
    fin para
  fin para
  para $i = j + 1 : n$ **hacer**
    $w_i = w_i / l_{jj}$
  fin para
  para $i = j + 1 : n$ **hacer**
    $l_{i,j+1} = w_i$
  fin para
  $I = (i_k)_{k=1,\ldots,p.}$ contiene los índices de los primeros m entradas $\mid w_i \mid$ para $i = j + 1 : n$ mas grande.
  para $k = 1 : m$ **hacer**
    $l_{i_k,j} = w_{i_k}$
  fin para
  para $i = j + 1 : n$ **hacer**
    $a_{ii} = a_{ii} - l_{ij}^2$
  fin para
fin para

Con la necesidad de establecer un patrón de esparcimiento para el factor L en [23] se introduce el concepto de niveles, esta factorización es conocida como $IC(\ell)$ donde ℓ es el parámetro de niveles.

3.2.4. Factorización incompleta $IC(\ell)$

Esta factorización permite ingresar más elementos no cero en el factor L por medio de los elementos no cero de la matriz A, esta estrategia trabaja de la siguiente manera.

Si se asigna $\ell = 1$, se permite que cualquier elemento no cero de la matriz A realice un aporte al factor L_1, introduciendo cualquier elemento no cero durante un paso de la factorización de Cholesky. Para un valor de $\ell = 2$ podría incluir los

elementos no ceros ingresados para $\ell = 1$ y cualquier elemento no cero introducido directamente por la factorización de Cholesky de los elementos no ceros obtenidos de $\ell = 1$.

Este método determina el patrón de elementos no ceros y la cantidad de memoria requerida antes de realizar la factorización incompleta de Cholesky. Esta factorización se caracteriza por tener dos procesos o fases. La primera fase es conocida como factorización numérica, esta fase produce el posible patrón de elementos no cero del factor L. La segunda fase presenta una factorización incompleta de Cholesky producida por el algoritmo 5.

Varias estrategias se han implementado para localizar los posibles elementos no ceros durante una factorización incompleta de Cholesky. A continuación se presentarán tres estrategias las cuales permiten conocer un posible patrón de esparcimiento por medio de la factorización simbólica.

Se define un patrón de esparcimiento en [23] de elementos no cero para una matriz B como el siguiente conjunto

$$NZ[B] = \{(i,j)|b_{ij} \neq 0\}.$$

Dado un patrón de esparcimiento P, se denota $\hat{B} := B[P]$ como la matriz extraída desde B con el patrón de esparcimiento definido por P.

$$\hat{b}_{ij} = \begin{cases} b_{ij} & \text{si } (i,j) \in P, \\ 0 & \text{otro caso,} \end{cases}$$

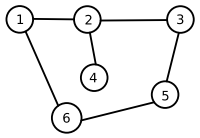
Sea $A = A_0$ y $G_k = (V_k, E_k)$ para $k = 0, 1, \ldots, n-1$ el grafo de A_k donde

$$V_k = \{v_{k+1}, \ldots, v_n\}, \quad E_k = \{(v_i, v_j) | a_{ij}^k \neq 0, \ \ i, j > k\}.$$

La noción entrada de llenado por niveles lo definen mediante un conjunto alcanzable (*reachable sets*) y longitud más corta de un camino (*shortest path length*) en el grafo G_0. Se define $S_k = \{v_1, \ldots, v_n\}$ como el conjunto de nodos eliminados en el paso k, es de notar que $S_k \subset V_0$. Un nodo u se dice que es alcanzable (*reachable*) de un vértice v a través de S_k, si existe un camino $(v, u_{i_1}, \ldots, u_{i_m}, u)$ en el grafo G_0, tal que cada $u_{i_j} \in S_k$ para $1 \leq j \leq m$. Para $m = 0$ con $u, v \notin S_k$ los vértices es alcanzable (*reachable*) a través de S_k. El conjunto de *reachable* de v a través de S_k es denotado por

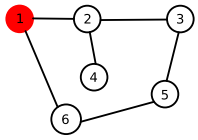
$$Reach(v, S_k) := \{u | u \text{ es reachable de } v \text{ a través de } S_k\}$$

Sea $v_i, v_j \notin S_k$ y $v_j \in Reach(v_i, S_k)$ con *shortest path length* $(v_i, u_{i_1}, \ldots, u_{i_m}, v_j)$. El *shortest path length* de v_i hasta v_j a través de S_k es definido como m. Si $v_j \notin Reach(v_i, S_k)$ entonces el camino entre v_i y v_j será ∞. Para una mejor comprensión se presenta el siguiente ejemplo. En el lado derecho se muestran las matrices actualizadas después de una iteración y al lado izquierdo el respectivo grafo de adyacencia.



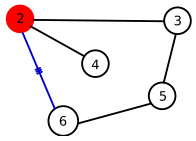
$$\begin{bmatrix} x & x & & & & x \\ x & x & x & x & & \\ & x & x & & x & \\ & x & & x & & \\ & & x & & x & x \\ x & & & & x & x \end{bmatrix}$$

Para la primera iteración se genera.



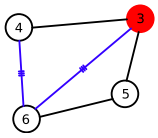
$$\begin{bmatrix} x & x & & & & x \\ x & x & x & x & & \\ & x & x & & x & \\ & x & & x & & \\ & & x & & x & x \\ x & & & & x & x \end{bmatrix}$$

Para la segunda iteración.



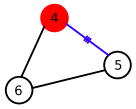
$$\begin{bmatrix} x & x & & & & x \\ x & x & x & x & & X \\ & x & x & & x & \\ & x & & x & & \\ & & x & & x & x \\ x & X & & & x & x \end{bmatrix}$$

Para la tercera iteración.



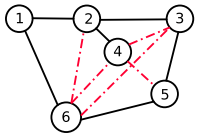
$$\begin{bmatrix} x & x & & & & x \\ x & x & x & x & & X \\ & x & x & & x & X \\ & x & & x & & X \\ & & x & & x & x \\ x & X & X & X & x & x \end{bmatrix}$$

Para la cuarta iteración.



$$\begin{bmatrix} x & x & & & & x \\ x & x & x & x & & X \\ & x & x & & x & X \\ & x & & x & X & X \\ & & x & X & x & x \\ x & X & X & X & x & x \end{bmatrix}$$

Ahora se presenta la matriz con el posible patrón de esparcimiento.



$$\begin{bmatrix} x & x & & & & x \\ x & x & x & x & & X \\ & x & x & & x & X \\ & x & & x & X & X \\ & & x & X & x & x \\ x & X & X & X & x & x \end{bmatrix}$$

Entonces el nivel de llenado para un par de nodos ordenados (v_i, v_j) para un grafo G_k durante la eliminación será

$$level_{ij}^{k} = \begin{cases} m & \text{si } a_{ij} \text{ es una posible entrada,} \\ \infty & \text{otro caso.} \end{cases}$$

Donde m es *shortest path length* entre v_i y v_j a través de S_k y $i, j > k$. Por otra parte se tiene que

$$level_{ij}^{k} = \begin{cases} 0 & \text{Si } a_{ij} \neq 0, \\ \infty & \text{otro caso.} \end{cases}$$

Por cada paso del proceso de eliminación se generan nuevos *shortest path* entre v_i y v_j a través de nuevos conjuntos de nodos eliminados, así los niveles pueden ser actualizados como

$$level_{ij}^{k} = \text{mín}(Level_{ik}^{k-1} + Level_{kj}^{k-1} + 1, Level_{ij}^{k-1})$$

una vez completado la factorización simbólica asignando a cada elemento un nivel, se define como D_k el patrón de esparcimiento de aquellos posibles elementos no ceros cuyo nivel excede el parámetro ℓ

$$D_k = \{(i,j) | level_{ij}^{k} > \ell, \;\; i, j > k\}.$$

Los elementos que pertenecen a D_k serán descartados permitiendo que aquellos elementos que pertenecen a $NZ[A_k] - D_k$ sean un aporte como entrada de llenado

$$NZ[A_k] - D_k = \{(i,j) | level_{ij}^{k} \leq \ell, \quad \text{donde } i, j > k\}$$

Se explican dos estrategias en [19] para asignar niveles de llenado, refiriéndose a estas estrategias como *Sum* (Estategia I) y *max* (Estrategia II).

Suponiendo que a_{ij} es una entrada de llenado aportada por las entradas a_{ik} y a_{kj} la

estrategia I establece

$$level_{ij} = \min_{1 \leq h < \min(i,j)} \{level_{i,k} + level_{k,j} + 1\}.$$

La estrategia II esta dada por

$$level_{ij} = \min_{1 \leq k < \min(i,j)} \max\{level_{i,k}, level_{k,j}\} + 1.$$

Por medio de la estrategia I, el nivel asignado a una entrada permite conocer el número mínimo de veces para el cual su actualización es dividida por un pivote, sin mostrar el momento en el cual se realiza el aporte. Por otro lado la estrategia II permite saber en que iteración de la factorización númerica se realiza un aporte, como otra de su característica, esta puede ser desarrollada de manera recursiva por una factorización $LL(1)$, ℓ veces. La entrada sería la estructura de niveles calculado durante la iteración previa.

El algoritmo 8 genera una factorización simbólica presentado por Hysom y Photen.

Algoritmo 8 Factorización simbólica

```
1: para j = 1 : n hacer
2:     Inicia fase: admite entrada en A y asigna los niveles cero
3:     adj'(j) = 0
4:     para t ∈ adj(j) hacer
5:         level_jt = 0
6:         inserta t en adj'(j)
7:         por cada elemento i ∈ adj'(j) no marcado con i < j en orden ascendente
8:         para t ∈ adj'(i) con t > i hacer
9:             wt = Estategia(level_j,i, level_i,t)
10:            si wt ≤ l entonces
11:                si t ∈ adj'(i) entonces
12:                    inserta t en adj'(j)
13:                    level_j,t = wt
14:                si no
15:                    level_j,t = mín{level_i,t, wt}
16:                fin si
17:            fin si
18:        fin para
19:    fin para
20: fin para
```

$$Estrategia(level_{i,k}, level_{k,j})\{return\ \ level_{i,k} + level_{k,j} + 1\}$$

$$Estrategia(level_{i,k}, level_{k,j})\{return\ \ \text{máx}\{level_{i,k}, level_{k,j}\} + 1\}$$

Estas estrategias tienen la desventaja que los elementos cercano a ceros podrían aportar la misma cantidad de elementos no ceros que aquellas entradas con un valor lejano a cero. En [18] se propone una estrategia que asigna niveles que depende al tamaño de los elementos de A.

3.2.5. Factorización incompleta $IC(\ell, \tau, m)$

Para asegurar que las pequeñas entradas contribuyen a menores niveles de llenado en la factorización incompleta, Scott y Tüman proponen preasignar niveles

($level_{i,j}$) para cada entra de A, dicho nivel depende de $\mid a_{ij} \mid$.

Una vez calculado el valor absoluto de todas las entradas de A, se selecciona el mayor valor absoluto de las entrada ($mbig$) y el menor valor absoluto de las entrada ($msmall$), luego se distribuye el logaritmo de cada $\mid a_{ij} \mid$ entre $mgrp = [log(mbig) - log(msmall)] + 1$ grupos. A cada grupo se le asigna un índice de 1 a $mgrup$ de tal manera que las entradas con valor absoluto más pequeños se almacenan en el grupo con índice 1 y aquellas cuyo valor absoluto son más grande en el grupo con índice $ngrp$. Las estrategias para preasignar niveles depende de las siguientes condiciónes si $\ell < ngrp$ o $\ell \geq ngrp$.

Para el caso $\ell < ngrp$ se tiene.

$$level_{ij} = \begin{cases} [k_{ij}/q] & \text{Si } mod(k_{ij}, q) = 0, \\ \text{mín}(\ell, [k_{ij}/q] + 1) & \text{otro caso,} \end{cases} \tag{3.1}$$

donde $q = [ngrp/\ell]$ y k_{ij} $(1 \leq k_{ij} \leq ngrp)$ son los índices de los grupos donde el elemento $\mid a_{ij} \mid$ pertenece. De esta manera las entradas más pequeñas pueden contribuir a un solo nivel de llenado, y para valores más grandes a ℓ niveles.

Para el caso $\ell \geq ngrp$ se tiene

$$level_{i,j} = \ell - (ngrp - k_{ij}) \tag{3.2}$$

de igual manera con esto se obtiene que las pequeñas entradas de A contribuyen a menores entradas de niveles en el patrón de esparcimiento de L. Por otro lado aquellas entradas que son más pequeñas en valor absoluto que la raíz cuadrada de precisión de maquina multiplicado por la entrada de valor absoluto más grande del grupo con índice 1, se asigna $level_{ij} = -(n + 1)$. Esto permite eliminar entradas casi ceros durante la factorización simbólica. El algoritmo 9 es una modificación

del algoritmo 8, el cual presenta una factorización simbólica que calcula el posible patrón de esparcimiento de la columna k de L. Los índices de las entradas en la columna k son regresada en $adj'(k)$, y nz_k es el número de dichas entradas y $length$ es un arreglo de tamaño n.

Algoritmo 9 Factorización simbólica $IC(\ell)$

```
1: Asignar k como visitado
2: sea length = 0 y nz_k = 0
3: mientras El queue no este vacio hacer
4:     Tomar i del queue
5:     para j ∈ adj(i) hacer
6:         si j no ha sido visitado and level_ij ≠ −(n + 1) entonces
7:             Asignar j como visitado
8:             si j < k and length(i) < level_ij entonces
9:                 Agregar j al queue
10:                length(j) = length(i) + 1
11:                si j > i entonces
12:                    nz_k = nz_k + 1
13:                    Agregar j a adj'(k)
14:                fin si
15:            fin si
16:        fin si
17:    fin para
18: fin mientras
```

En la linea 6 del algoritmo 9 permite descartar aquellos elementos cuyo nivel es $n+1$, descartando aportes de aquellas entradas con valores muy pequeños. En la linea 8 se restringe los posibles aportes de una entrada según su nivel asignado. De esta manera se contribuye a menos entradas de llenado de las que podría aportar el algoritmo 8 dado que $level_{ij} < \ell$.

Las diferentes técnicas para la factorización incompleta de Cholesky como $IC(\tau)$, $IC(m)$, $IC(\ell)$ son combinadas, generando un buen precondicionador que permita reducir el número de iteraciones haciendo uso del método **PCG**, y a su vez disminuir el consumo en memoria. El algoritmo 10 presenta una combinación entre $IC(\tau)$,$IC(m)$ y $IC(\ell)$.

Algoritmo 10 Factorización incompleta $IC(\ell, \tau, m)$

1: Asignar los niveles a las entradas de A con las estrategias (3.1) y (3.2)
2: Calculo del patrón de esparcidad de L con Algoritmo 9.
3: **para** $j = 1 : n$ **hacer**
4: $l_{jj} = \sqrt{a_{jj}}$
5: $w = a_{j+1:n,j}$
6: **para** $k = 1 : j - 1$ **hacer**
7: **para** $i = j + 1 : n$ **and** cuando $l_{jk} \neq 0$ **hacer**
8: $w_i = w_i - l_{ik} l_{jk}$
9: **fin para**
10: **fin para**
11: **para** $i = j + 1 : n$ **hacer**
12: $w_i = w_i / l_{jj}$
13: **fin para**
14: $\tau_j = \tau \parallel w \parallel$
15: **para** $i = j + 1 : n$ **hacer**
16: $w_i = 0$ cuando $\mid w_i \mid < \tau_j$
17: **fin para**
18: Sea $I = (i_k)_{k=1,\ldots,p}$ un arreglo que contiene los índices de las m entradas más grande de $\mid w_i \mid, i = j + 1 : n$
19: **para** $k = 1 : m$ **hacer**
20: $l_{i_k j} = w_{i_k j}$
21: **fin para**
22: **para** $i = j + 1 : n$ **hacer**
23: $a_{ii} = a_{ii} - l_{ij}^2$
24: **fin para**
25: **fin para**

En el algoritmo 10 se denota a $a_{j+1:n,j}$ como las entradas de la $j-esima$ columna desde la fila $j + 1$ hasta la fila n.

3.2.6. Factorización incompleta de Cholesky Desplazada

La existencia de una factorización incompleta de Cholesky no esta garantizada a menos que se satisfagan las condiciones de los teoremas (3.2.1) y (3.2.2). Cuando las condiciones de los teoremas antes mencionados no se cumplen, Manteuffel en [32] propone realizar una descomposición de Jacobbi sobre la matriz original A

$$A = I - B$$

y se considera una descomposición

$$A(\alpha) = I - \frac{1}{1+\alpha}B.$$

Para un valor de α suficientemente grande $A(\alpha)$ será estrictamente diagonal dominante de esta manera la factorización incompleta de Cholesky de $A(\alpha)$ será estable.

A medida que α aumenta $A(\alpha)$ se aproxima a la identidad, así que el gran interés es para pequeños valores de α. Se han obtenido algunos resultados para estimar el valor de α bajo algunas condiciones de la matriz A.

Supongamos que α_a es tal que la factorización incompleta es estable para todo $\alpha > \alpha_a$, en general α_a no es conocida. Manteuffel obtuvo algunos resultados.

Otra estrategia para el desplazamiento fue propuesta en [33]. Si el elemento a_{ij}^j con $(i > j)$ es rechazado durante la factorización numérica, los elementos de las diagonales a_{jj}^j y a_{ii}^j son modificados de la siguiente manera

$$\bar{a}_{jj} = a_{jj} + s \mid a_{ij} \mid \quad \text{y} \quad \bar{a}_{ii} = a_{ii} + \frac{1}{s} \mid a_{ij} \mid .$$

Para asegurar la estabilidad durante la factorización se escoge

$$s = k \left(\frac{a_{jj}}{a_{ii}} \right)^{1/2}.$$

Luego

$$\bar{a}_{jj} = a_{jj} \left(1 + \frac{k \mid a_{ij} \mid}{(a_{ii}a_{jj})^{1/2}} \right) \quad \text{y} \quad \bar{a}_{ii} = a_{ii} + \left(1 + \frac{\mid a_{ij} \mid}{k(a_{ii}a_{jj})^{1/2}} \right)$$

las pruebas experimentales realizadas por Jennings y Malik indican que para $k = 1$ se obtuvieron mejores resultados.

Hladíd, Reed y Swodoba en [34] proponen una estrategia la cual permite que la matriz LL^T generada por una factorización incompleta de Cholesky sea definida positiva, permitiendo modificar los elementos de las diagonales a_{jj}, a_{ii} de la matriz original con aquellos elementos rechazado l^*_{ij} durante la factorización numérica.

$$a_{ii} = a_{ii} + \gamma_i, \;\; a_{jj} = a_{jj} + \gamma_j, \quad \gamma_i, \gamma_j > 0$$

las constantes positivas γ_i, γ_j deben satisfacer la siguiente condición

$$\gamma_i \gamma_j = (l^*_{ij})^2.$$

Una manera de escogerla seria

$$\begin{aligned} \gamma_i &= \left(\frac{\hat{a}_{ii}}{\hat{a}_{jj}}\right)^{1/2} \mid l^*_{ij} \mid \\ \gamma_j &= \left(\frac{\hat{a}_{ii}}{\hat{a}_{jj}}\right)^{1/2} \mid l^*_{ij} \mid \end{aligned}$$

donde los elementos $\hat{a}_{jj}$, $\hat{a}_{jj}$ están dado por

$$\hat{a}_{jj} = \left(a_{jj} - \sum_{k=1}^{j+1} l^2_{jk}\right), \quad \hat{a}_{ii} = a_{ii}.$$

Otra alternativa es tomar γ_i, γ_j simplemente como

$$\gamma_i = \gamma_j = \mid l^*_{ij} \mid .$$

En los experimentos numéricos realizados en [34] se encuentra resultados similares con las dos estrategias.

Todas las técnicas presentadas en esta sección (3.2) fueron analizadas para matrices puntuales, es decir, matrices donde los elementos $a_{ij} \in \mathbb{R}$. En la siguiente sección se trabaja las mismas técnicas aplicadas a matrices por bloques, es decir,

matrices donde los elementos $A_{ij} \in \mathbb{R}^{m \times m}$.

3.3. Factorización incompleta de Cholesky por bloques

Para una factorización completa por bloques se hace uso de la técnica presentada en la sección (3.1). En este caso las submatrices de L se evalúan mediante las expresiones.

$$L_{ij}L_{jj}^T = A_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{kj}^T \tag{3.3}$$

$$L_{ii}L_{ii}^T = A_{ii} - \sum_{k=1}^{i-1} L_{ik}L_{ki}^T \tag{3.4}$$

La ecuación (3.3) implica resolver una serie de sistemas triangulares en los que las incógnitas son los elementos L_{ij}, los vectores de los sistemas son las columnas de la matriz de la parte derecha. La ecuación (3.4) realiza una factorización completa de Cholesky puntual a la matriz $A_{ii} - \sum_{k=1}^{i-1} L_{ik}L_{ki}^T$ obteniendo L_{ii}. Estas son la operaciones más importantes durante una factorización de Cholesky por bloques.

A continuación se presentan las modificaciones de las técnicas presentadas en las secciones (3.2.1-3.2.5).

La fatorización $IC(0)$ por bloque es obtenida haciendo uso de la técnica presentada en la sección 3.2.1 y el algoritmo 5, sustituyendo la linea 5 por (3.3) y la linea 2 por (3.4). Para aplicar la técnica presentada en las sección (3.2.2) $IC(\tau)$ por bloque, en la linea 14 del algoritmo 6 se sustituye el cálculo del valor absoluto de un elemento $a_{ij} \in \mathbb{R}$ por el cálculo de una norma matricial [10] para un elemento $A_{ij} \in \mathbb{R}^{m \times m}$.

La técnica que fue descrita en la sección (3.2.2) a nivel puntual, para el caso de $IC(\tau)$ por bloque, los elementos del vector w en el algoritmo 6 son submatices de la matriz original forzando a cambiar la estrategia utilizada en la linea 12. Para el caso por bloques se toma el máximo valor de las normas matriciales de los elementos A_{ij} que componen al vector w.

Para la factorización simbólica presentada en la sección (3.2.5) de manera puntual no presenta alteración para una factorización simbólica por bloques, dado que con esta factorización se busca calcular el posible patrón de esparcimiento del factor L el cual no depende de la estructura de A. Por otro lado, para preasignar niveles como se propone en [18] a nivel puntual, sufre algunas modificaciones para su adaptación por bloques.

En este trabajo se propone realizar las siguientes modificaciones para adaptar el trabajo de Scott y Tüma a una versión por bloques. El preasignar los niveles dependerán de $\| A_{ij} \|_1$. Se calcula la norma de todas las entras y se denota como $(mbig)$ a la mayor norma entre las entradas y $(msmall)$ a la menor norma entre las entradas. Se distribuye a cada $\| A_{ij} \|_1$ en $mgrp = \lceil mbig - msmall \rceil + 1$ grupos. A cada grupo se le asignado un índice y una amplitud, el índice inicia de 1 hasta $mbig$ guardando aquellas entradas de menor norma en el grupo con índice 1 y las mayores en el grupo con índice $mbig$. Para calcular la amplitud de los grupos se toma $\lceil (mbig - msmall)/mgrp \rceil$, Así el primer grupo con índice 1, admite elementos cuya norma esta entre $msmall$ y $\lceil (mbig - msmall)/mgrp \rceil$. Las estrategias utilizadas para asignar los niveles son las descritas en la ecuación (3.1) y (3.2). Cuando se preasigna todo los niveles a cada una de las entradas se hace uso del algoritmo 9, dando paso a la factorización numérica para el cual se implementa el algoritmo 5 a

nivel de bloques.

Cuando se trabaja con el parámetro de memoria se guardan los m primeros elementos de mayor norma. El algoritmo 11 realiza una factorización $IC(\ell, \tau, m)$ por bloques el cual es una modificación del algoritmo 10.

Algoritmo 11 Factorización incompleta $IC(\ell, \tau, m)$ por bloques

1: Asignar los niveles a las entradas de A con las estrategias (3.1) y (3.2)
2: Calculo del patrón de esparcidad de L con Algoritmo 9.
3: **para** $j = 1 : n$ **hacer**
4: $L_{jj}L_{jj}^T$ Factorización de Cholesky puntual $< dpotf2 >$
5: $W = A_{j+1:n,j}$
6: **para** $k = 1 : j - 1$ **hacer**
7: **para** $i = j + 1 : n$ **and** cuando $L_{jk} \neq 0$ **hacer**
8: $W_i = W_i - L_{ik}L_{jk}$ Multiplicación matriz por matriz $< dgemm >$
9: **fin para**
10: **fin para**
11: **para** $i = j + 1 : n$ **hacer**
12: $INV = L_{jj}^{-1}$ Calculo inversa de $L_{jj} < dtrtri >$
13: $W_i = W_i INV^T$ Multiplicación matriz por matriz $< dgemm >$
14: **fin para**
15: $\tau_j = \tau \parallel W \parallel$
16: **para** $i = j + 1 : n$ **hacer**
17: $W_i = [0]$ cuando $\parallel W_i \parallel < \tau_j$
18: **fin para**
19: Sea $I = (i_k)_{k=1,\ldots,p}$ un arreglo que contiene los índices de las m entradas más grande de $\mid w_i \mid, i = j + 1 : n$
20: **para** $k = 1 : m$ **hacer**
21: $L_{i_k j} = W_{i_k j}$
22: **fin para**
23: **para** $i = j + 1 : n$ **hacer**
24: $A_{ii} = A_{ii} - L_{ij}L_{ji}^T$ Multiplicación matriz por matriz $< dsyrk >$
25: **fin para**
26: **fin para**

De igual manera como ocurre la falta de estabilidad en una factorización incompleta a nivel puntual, se presenta fallos durante una factorización incompleta

por bloques. Para compensar este problema se diseña un desplazamiento que permitirá el cálculo de una factorización incompleta de Cholesky por bloques.

Una factorización incompleta de Cholesky a nivel de bloques puede fallar si los bloques L_{ii} no son definidas positivas. Por medio de la siguiente proposición se intenta remediar este problema.

Proposición 3.3.1. *Si $A \in \mathbb{R}^{n\times n}$ es simétrica, de diagonales positiva y estrictamente diagonal dominante, entonces A es definida positiva.*

Demostración. Dado que $A \in \mathbb{R}^{n\times n}$, existe una matriz diagonal Λ y una matriz ortonormal Σ talque $A = \Sigma\Lambda\Sigma^T$, donde los elementos de Λ son los autovalores σ_i, $i = 1\ldots, n$ y las columnas de Σ son sus correspondientes vector propio. Por el teorema de Gerschgorin [10], los autovalores de A están ubicados en la unión de discos

$$d_i = \{z \in \mathbb{C} :\mid z - a_{ii} \mid \leq \sum_{j\neq i} \mid a_{ij} \mid\}, \qquad i = 1, \ldots, n.$$

Por hipótesis los elementos diagonales de A son positivos y además es estrictamente diagonal dominante permitiendo

$$d_i = \{z \in \mathbb{C} : 0 < a_{ii} - \sum_{j\neq i} \mid a_{ij} \mid \leq z \leq \sum_{j} \mid a_{ij} \mid\}, \qquad i = 1, \ldots, n.$$

Luego todos los autovalores son positivos, así $\forall x \in \mathbb{R}^n$, $x \neq 0$ se tiene,

$$x^T Ax = x^T\Sigma\Lambda\Sigma^T x = y^T\Lambda y > 0, \text{ donde } \quad y = \Sigma^T x \neq 0$$

□

La técnica presentada en este trabajo consiste en convertir aquellos elementos L_{ii} donde la factorización de Cholesky falla, en una matriz definida positiva, haciendo uso de la proposición (3.3.1).

La estrategia es transformar la matriz $L_{ii} = (l_{ij})$ en estrictamente diagonal dominante con elementos de la diagonal positivos, para esto se aplica lo siguiente

$$l_{ii} =\mid l_{ii} \mid + \sum_{\substack{j=1 \\ j\neq i}}^{n} (l_{ij})^2.$$

La ventaja de esta técnica como desplazamiento, es que sólo se está alterando los bloques L_{ii} donde la factorización completa de Cholesky falla.

Capítulo 4
RESULTADOS NUMÉRICOS

Los siguientes resultados numéricos se han generado por la implementación de una factorización incompleta $IC(\ell, \tau, m)$, la cual genera una factorización de la forma $A = LL^t$, donde L es una matriz triangular inferior. Esta factorización se implementa junto al método del Gradiente Conjugado Precondicionado. Las matrices tomadas para los experimentos son matrices generadas por el método Local Discontinuous Galerkin para la discretización del operador Laplaciano y Helmholtz. En los experimentos se ha tomado un vector inicial $x_0 = 0$, los criterios de parada estan basados en el error del sistema precondicionado, si este satisface $\| r \| < 10^{-13}$ y en un máximo de 1000 iteraciones.

Para restringir el número de entradas no ceros que se produce al aumentar los niveles durante la factorización simbólica, se proporciona un parámetro umbral τ, el cual restringe aquellas entradas no cero durante la factorización numérica si estas satisfacen la siguiente condición: Si $\| A_{ij} \|_1 < \tau \, \text{máx}\{\| A_{ij} \|_1 : (i,j) \in \wp\}$, donde A_{ij} es una entrada de A, entonces dicha entrada A_{ij} se eliminará como un factor de L, es decir, que con el parámetro τ se espera mayor elementos ceros en la matriz L a medida que este aumente.

Otra manera de restringir entradas en el fator L es haciendo uso del parámetro de memoria m, Con este paŕametro solo se tomará los m primeros elementos de mayor norma $\| \cdot \|_1$, evitando entradas de llenado cuyos elementos de las matrices

tengan menor norma.

4.1. Experimentos para el Operador Laplaciano

Las matrices utilizadas para generar los datos fueron de la discretización del Laplaciano por medio el método *LDG* [39], dichas matrices cumplen con la característica de ser simétricas y definida positivas (Tabla 4-1).

Nombre	dim. Global	dim. Bloques	BNz
Matriz 1	320×320	4×4	3760
Matriz 2	2560×2560	4×4	33632
Matriz 3	20480×20480	4×4	283840

Tabla 4–1: Información de las matrices.

4.1.1. Experimentos para $IC(\ell, 0, 0)$

Para ver el efecto producido en el número de iteraciones requeridas por **PCG**, la cantidad de memoria requerida por L y el tiempo de ejecución por el parámetro de niveles, se han discriminado los parámetros de memoria m y el parámetro umbral τ asignando un valor de cero a cada uno.

Con el aumento del parámetro de niveles ℓ, se espera una disminución en el número de iteraciones para la convergencia de método **PCG**. En contraste se obtendría mayor elementos no cero en el factor L aumentando el consumo de memoria y un aumento en el tiempo de ejecución para la factorización numérica.

Con los experimentos realizados se detalla el número de iteraciones para la convergencia del método **PCG**, un indicador del consumo de memoria $\frac{Nz(L)}{Nz(A)}$, el cual se espera que aumente al incrementar los elementos no ceros de L (Nz(L)), y el tiempo de ejecución en las diferentes fases de la implementación de la factorización

simbólica, factorización numérica y el método **PCG**.

Las tablas 4-2,4-3,4-4 muestran el número de niveles, la cantidad de bloques no ceros Nz(L) del factor L, el número de iteraciones (Iter) al aplicar el Gradiente Conjugado Precondicionado. El parámetro de niveles toma valores entre 1 y 15.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	4080	1.08	15
3	4080	1.08	15
5	8716	2.32	7
8	13236	3.52	4
11	13430	3.58	2
15	13460	3.58	2

Tabla 4–2: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$, $\tau = 0$ y variado ℓ. Para la matriz 1 descrita en la tabla 4-1.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	39362	1.17	26
3	65493	1.94	16
5	148835	4.42	8
8	288809	8.58	7
11	397327	11.81	5
15	438773	13.04	3

Tabla 4–3: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$, $\tau = 0$ y variado ℓ. Para la matriz 2 descrita en la tabla 4-1.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	342918	1.209	49
3	600455	2.341	32
5	1501143	5.291	18
8	3415645	12.048	11
11	5725489	20.408	9

Tabla 4–4: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$, $\tau = 0$ y variado ℓ. Para la matriz 3 descrita en la tabla 4-1.

En las tablas 4-2,4-3,4-4 se puede observar que al aumentar el parámetro de niveles ℓ, se presenta una disminución en el número de iteraciones. Cabe resaltar, que a medida el número de nivel incrementa se tiene más entradas no ceros durante la factorización numérica, esto podría permitir una mejor aproximación de A.

4.1.2. Experimentos para $IC(\ell, \tau, 0)$

Con el objetivo de restringir entradas al factor L, se hace uso del parámetro umbral τ, combinado con los aportes que proporciona el parámetro ℓ e ignorando el parámetro de memoria m, el cual permite observar la influencia de τ en la factorización, resaltando como el parámetro umbral retiene elementos del fator L

Con esta combinación se espera una disminución en el número de iteraciones utilizadas para el **PCG** al disminuir el parámetro τ. Cuando el parámetro τ tiende a cero se genera un aumento en número de elementos no ceros en L, generando un mayor consumo en memoria. Este parámetro podría aumentar el tiempo de ejecución de la factorización numérica.

La tabla 4-5 muestra los resultados de los experimentos realizados variando τ con valores entre 10^{-3} y 10^{-8}, dejando el parámetro de memoria fijo en cero. y variando el nivel ℓ

Nivel	$\tau = 10^{-3}$		$\tau = 10^{-5}$		$\tau = 10^{-8}$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	3787	15	3787	15	3787	15
3	3787	15	3787	15	3787	15
5	8158	7	8199	7	8199	7
8	11296	5	12509	4	12531	4
11	11345	4	12669	3	12721	2
15	11345	4	12669	3	12721	2

Tabla 4–5: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$ y variado ℓ y τ. Para la matriz 1 descrita en la tabla 4-1.

En la tabla 4-5 se puede notar que al disminuir el parámetro umbral τ se produce un aumento en el número de bloques no ceros en el factor L, obteniendo consigo una disminución en el número de iteraciones para alcanzar convergencia.

En la tabla 4-6 se muestra la cantidad de memoria que necesita el factor L, variando los parámetros de niveles y umbral, a su vez se muestra un indicador del costo de memoria $\frac{Nz(L)}{Nz(A)}$.

Nivel	$\tau = 10^{-3}$		$\tau = 10^{-5}$		$\tau = 10^{-8}$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	1.01	0.55	1.01	0.55	1.01	0.55
3	1.01	0.55	1.01	0.55	1.01	0.55
5	2.17	1.13	2.22	1.13	2.22	1.13
8	3.03	1.55	3.33	1.71	3.33	1.71
11	3.03	1.55	3.44	1.73	3.44	1.74
15	3.03	1.55	3.44	1.73	3.44	1.74

Tabla 4–6: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 1 de la tabla 4-1.

En la tabla 4-6 se puede notar que al aumentar el valor de $\frac{Nz(L)}{Nz(A)}$ aumenta el consumo de memoria para el almacenamiento de L, mostrando el incremento de llenado permitido en la factorización simbólica y numérica al incrementar el parámetro de nivel y disminuir el umbral.

Al solucionar un sistema precondicionado se disminuye el número de iteraciones para la convergencia del método **PCG**, comparado con el número de iteraciones requeridas para solucionar el sistema no precondicionado. Pero al variar los valores de los parámetros se genera un cambio de elementos no ceros en el factor L, afectando el tiempo para resolver el sistema. La tabla 4-7 muestra los tiempos de ejecución de las diferentes fases de la factorización; factorización simbólica (F.S) y factorización numérica (F.N), así como también el tiempo de ejecución del método **PCG** (Solver).

Nivel	$\tau = 10^{-3}$			$\tau = 10^{-5}$			$\tau = 10^{-8}$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	0.00	0.03	0.03	0.00	0.03	0.03	0.00	0.03	0.03
3	0.00	0.03	0.03	0.00	0.03	0.03	0.00	0.03	0.03
5	0.00	0.13	0.03	0.01	0.13	0.03	0.00	0.13	0.03
8	0.01	0.27	0.03	0.01	0.31	0.02	0.01	0.31	0.02
11	0.01	0.27	0.02	0.01	0.32	0.02	0.01	0.32	0.01
15	0.01	0.28	0.02	0.01	0.31	0.02	0.01	0.33	0.01

Tabla 4–7: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, \tau, 0)$, haciendo uso de la matriz 1 de la tabla 4-1.

En la tabla 4-7 se puede notar que si se incrementa el valor del parámetro de nivel ℓ y se disminuye el parámetro umbral τ se aumenta el tiempo de la factorización numérica. Este efecto es debido a que el número de bloques no ceros incrementa generando así mayor número de operaciones. Por otra parte, se presenta una disminución en el tiempo de ejecución al utilizar el solver.

Nivel	$\tau = 10^{-3}$		$\tau = 10^{-5}$		$\tau = 10^{-8}$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	38738	26	38738	26	38738	26
5	140165	10	142521	10	142521	10
10	299365	6	347806	5	357663	5
15	334940	6	407445	4	423769	3
20	335152	6	407723	4	424150	2

Tabla 4–8: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$ y variado ℓ y τ. Para la matriz 2 descrita en la tabla 4-1.

En la tabla 4-8 se puede observar que al disminuir τ el número de entradas no ceros aumenta disminuyendo el número de iteraciones del Gradiente Conjugado Precondicionado.

La tabla 4-9 muestra los tiempos de ejecución al implementar $IC(\ell, \tau, m)$ para los valores presentados en la tabla 4-8. Se observa que el aumento del parámetro de nivel ℓ y dejando τ fijo, genera un incremento en el tiempo de ejecución durante la factorización numérica y durante el solver. Este aumento en el tiempo de ejecución se debe al incremento en el número de operaciones que se debe realizar con respecto a la cantidad de elementos no ceros que aporta el parámetro ℓ. Por otro lado, si se aumenta el valor de τ y con $\ell \geq 10$, se observa una disminución en el tiempo de ejecución para el solver, esto por el número de iteraciones requeridas para la convergencia del método **PCG**. En la tabla 4-10 se muestra el uso de memoria utilizado para el almacenamiento de la matriz L con los diferentes valores para los parámetros τ y m.

Nivel	$\tau = 10^{-3}$			$\tau = 10^{-5}$			$\tau = 10^{-8}$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	0.07	1.62	0.56	0.07	1.63	0.60	0.07	1.62	0.60
5	0.16	19.74	0.68	0.16	19.90	0.65	0.16	19.95	0.65
10	0.40	91.90	0.83	0.40	106.94	0.82	0.40	111.19	0.84
15	0.51	122.54	0.92	0.51	149.12	0.82	0.51	158.08	0.66
20	0.52	122.64	0.93	0.52	149.60	0.84	0.52	158.05	0.53

Tabla 4–9: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, \tau, 0)$, haciendo uso de la matriz 2 de la tabla 4-1.

Nivel	$\tau = 10^{-3}$		$\tau = 10^{-5}$		$\tau = 10^{-8}$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	1.162	5.53	1.162	5.53	1.16	5.53
5	4.347	19.07	4.347	19.38	4.34	19.38
10	9.090	40.33	11.111	46.79	11.11	48.11
15	10	45.08	12.5	54.76	12.50	56.94
20	10	45.10	12.5	54.79	12.50	56.99

Tabla 4–10: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 2 de la tabla 4-1.

Los resultados presentados en las tablas 4-10,4-11,4-12 y 4-13 son realizados con la matriz 3 descrita en la tabla 4-1. Se inicia un experimento permitiendo variar el parámetro de nivel ℓ y el parámetro umbral τ, dejando $m = 0$. De esta manera se muestra el rendimiento al hacer uso de $IC(\ell, \tau, 0)$.

Nivel	$\tau = 10^{-3}$		$\tau = 10^{-5}$		$\tau = 10^{-8}$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	337390	49	337390	49	337390	49
5	1414883	18	1432660	18	1432660	18
10	4047524	10	4713468	10	4803016	10
15	5655037	10	8278151	7	8603313	7

Tabla 4–11: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$ y variado ℓ y τ. Para la matriz 3 descrita en la tabla 4-1.

En la tabla 4-10 se muestra el consumo de memoria requerido para el almacenamiento de la matriz L.

Nivel	$\tau = 10^{-3}$		$\tau = 10^{-5}$		$\tau = 10^{-8}$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	0.84	47.93	1.19	47.93	1.19	47.93
5	5	191.79	5.26	194.17	5.26	194.17
10	14.28	543.29	16.66	632.20	20	644.16
15	20	757.92	33.33	1108.14	33.33	1151.56

Tabla 4–12: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 3 de la tabla 4-1.

Los tiempos de ejecución de los experimentos de la tabla 4-12 se presentan en la tabla 4-13.

Nivel	$\tau = 10^{-3}$			$\tau = 10^{-5}$			$\tau = 10^{-8}$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	1.19	109.68	8.97	1.18	108.66	9.08	1.23	113.06	9.89
5	2.16	2532.38	11.50	2.16	2466.59	11.63	2.16	2498.95	11.30
10	6.34	15668.70	17.07	6.30	18864.40	20.37	7.38	19134.80	20.54
15	13.06	43482.00	24.04	12.91	55361.30	24.98	13.43	58853.40	28.03

Tabla 4–13: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, \tau, 0)$, haciendo uso de la matriz 3 de la tabla 4-1.

A continuación se mostrarán algunas gráficas las cuales representan la norma del residuo obtenido haciendo uso de método del Gradiente Conjugado Precondicionado, mostrando una reducción en el número de iteraciones al alterar los valores de los parámetros ℓ, τ, $m = 0$.

En la figura 4-1 y 4-2 se han comparado los residuos obtenidos haciendo uso del precondicionador $IC(\ell, \tau, m)$, en el cual se presenta una disminución en el número de iteraciones a medida que el parámetro ℓ aumenta y el parámetro de umbral τ disminuye.

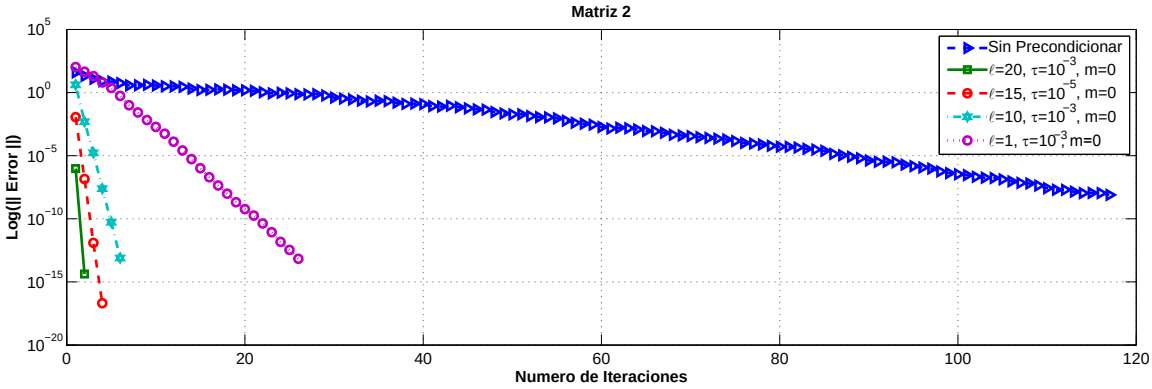


Figura 4–1: Historial de los residuos del sistema precondicionado y sin precondicionar, para la matriz 2 definido en la tabla 4-1.

Se muestra en la figura 4-2 el efecto producido sobre el número de iteraciones al utilizar **PCG**, cuando se incrementa el parámetro umbral τ y se deja fijo el parámetro de niveles.

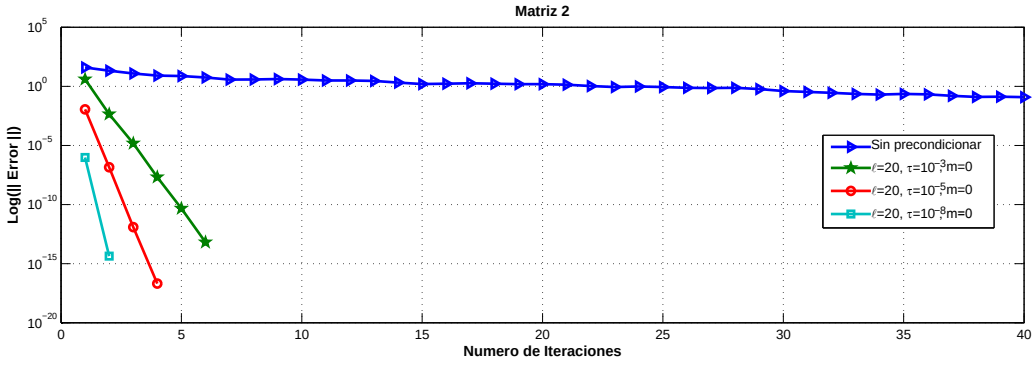


Figura 4–2: Historial de los residuos del sistema precondicionado y sin precondicionar, mostrando solo 40 iteraciones para la matriz 2 definido en la tabla 4-1.

En la figura 4-1 se presenta una disminución en el número de iteraciones al disminuir el valor del parámetro τ. Este efecto es debido a que el precondicionador utilizado para el **PCG** se aproxima más a A cuando τ tiende a cero permitiendo disminuir el número de iteraciones.

4.1.3. Experimentos para $IC(\ell, 0, m)$

Con esta factorización incompleta se desea explorar el uso del parámetro de nivel ℓ con el parámetro de memoria m, con el objetivo de establecer la cantidad de elementos no ceros por columnas administrando el consumo de memoria para el almacenamiento de L.

Se espera que al aumentar el parámetro de memoria se requiera menor número de iteraciones para **PCG** y un aumento en el consumo de memoria.

La tabla 4-14 muestra los resultados al aplicar $IC(\ell, \tau, m)$ donde el parámetro de nivel ℓ cambiará y también el parámetro de memoria m, asignando un valor de cero al parámetro umbral τ. Con esta tabla se pretende mostrar como trabaja el precondicionador usando solamente el parámetro de memoria y parámetro de niveles.

Nivel	$m = 20$		$m = 40$		$m = 60$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	4020	15	4080	15	4080	15
3	4020	15	4080	15	4080	15
5	5598	11	8368	7	8713	7
8	5751	11	10375	7	13216	4
11	5751	11	10386	7	13403	2
15	5751	11	10386	7	13430	2

Tabla 4–14: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 1 descrita en la tabla 4-1.

En la tabla 4-14 se puede observar que si se toma valores pequeños para m, se obtiene menor número de elementos no ceros en el factor L pero se genera mayor número de iteraciones. Si se compara la tabla 4-14 y 4-5, se puede notar que para valores $NZ(L)$ semejantes se obtiene un mismo número de iteraciones para alcanzar la convergencia, como por ejemplo en la tabla 4-5 para valores $\ell = 5$, $\tau = 10^{-5}$ se tiene igual número de iteraciones al aplicar el grandiente conjugado precondicionado que al compararlo con la tabla 4-8 para valores $\ell = 5$, $m = 40$.

En la tabla 4-15 se muestra el consumo de memoria en el factor L.

Nivel	$m = 20$		$m = 40$		$m = 60$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	1.07	0.58	1.08	0.58	1.08	0.58
3	1.08	0.58	1.08	0.58	1.08	0.58
5	1.49	0.79	2.27	1.16	2.32	1.20
8	1.53	0.81	2.77	1.43	3.57	1.80
11	1.53	0.81	2.77	1.43	3.57	1.83
15	1.53	0.81	2.77	1.43	3.57	1.83

Tabla 4–15: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 1 de la tabla 4-1.

Al ver la información de la tabla 4-15 se puede notar que al aumentar el número de $Nz(L)$, aumenta el valor de $\frac{Nz(L)}{Nz(A)}$ y a su vez aumenta la cantidad de memoria requerida para almacenar la matriz L.

En la tabla 4-16 se presentan los tiempos de ejecución de las diferentes fases haciendo uso de los parámetros de nivel y de memoria.

Nivel	$m = 20$			$m = 40$			$m = 60$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	0.00	0.06	0.05	0.00	0.06	0.03	0.00	0.04	0.03
3	0.00	0.06	0.05	0.00	0.06	0.03	0.00	0.04	0.03
5	0.01	0.07	0.05	0.01	0.08	0.03	0.01	0.09	0.03
8	0.01	0.08	0.05	0.01	0.09	0.03	0.01	0.12	0.03
11	0.01	0.08	0.05	0.01	0.10	0.03	0.01	0.12	0.03
15	0.01	0.08	0.05	0.01	0.12	0.03	0.01	0.12	0.03

Tabla 4–16: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, \tau, 0)$, haciendo uso de la matriz 1 de la tabla 4-1.

En la tabla 4-16 se presentan los tiempos de corrida de las diferentes fases de la factorización $IC(\ell, \tau, m)$. Al incrementar el valor del parámetro de memoria se tiene un aumento en el tiempo de corrida en la fase numérica, esto debido a el número de elementos no ceros permitido en el factor L, y se nota una disminución en el tiempo al ejecutar el solver. En la tabla 4-13 se presenta algunos datos que difieren con lo antes mencionado pero esto se debe que cuando se toma $\ell = 1, 3$ y $m = 60$ el número de elementos por filas proporcionados durante la factorización simbólica son menores que m, de manera que no se requiere actualizar el número de entradas por filas ahorrando así operaciones y consigo la disminución del tiempo.

Nivel	$m = 3$		$m = 8$		$m = 15$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	7677	89	20219	38	33824	27
5	7677	89	20407	38	37402	25
10	7677	89	20407	38	37462	25
15	7677	89	20407	38	37462	25
20	7677	89	20407	38	37462	25

Tabla 4–17: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 2 descrita en la tabla 4-1.

Nivel	$m = 20$		$m = 40$		$m = 60$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	38187	26	39362	26	39362	26
5	48856	20	92271	12	124871	10
10	49022	20	94776	12	139635	10
15	49022	20	94831	12	139847	10
20	49022	20	94831	12	139847	10

Tabla 4–18: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 2 descrita en la tabla 4-1.

Nivel	$m = 80$		$m = 100$		$m = 120$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	39362	26	39362	26	39362	26
5	142760	10	148655	10	148835	10
10	183332	9	225504	8	265622	7
15	183859	9	226680	8	268269	7
20	183859	9	226680	8	268269	7

Tabla 4–19: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 2 descrita en la tabla 4-1.

En la tabla 4-20 se presentan los datos de la memoria utilizada para los diferentes valores del parámetro m utilizados en las tablas 4-17, 4-18 y 4-19, en las cuales se podrá observar que al aumentar $\frac{Nz(L)}{Nz(A)}$ se requiere mayor memoria para el almacenamiento de la matriz L.

Nivel	$m = 3$		$m = 8$		$m = 15$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	0.22	1.38	0.60	3.06	1.00	4.87
5	0.22	1.38	0.60	3.08	1.11	5.35
10	0.22	1.38	0.60	3.08	1.11	5.35
15	0.22	1.38	0.60	3.08	1.11	5.35
20	0.22	1.38	0.60	3.08	1.11	5.35

Tabla 4–20: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 1 de la tabla 4-1.

Nivel	$m = 20$		$m = 40$		$m = 60$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	1.13	5.45	1.17	5.61	1.17	5.61
5	1.45	6.88	2.74	12.68	3.71	17.03
10	1.45	6.90	2.81	13.01	4.15	19.00
15	1.45	6.90	2.81	13.02	4.15	19.00
20	1.45	6.90	2.81	13.02	4.15	19.00

Tabla 4–21: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 2 de la tabla 4-1.

Nivel	$m = 80$		$m = 100$		$m = 120$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	1.17	5.61	1.17	5.61	1.17	5.61
5	4.34	19.42	4.54	20.20	4.54	20.23
10	5.55	24.83	7.14	30.46	8.33	35.82
15	5.55	24.90	7.14	30.62	8.33	36.17
20	5.55	24.90	7.14	30.62	8.33	36.17

Tabla 4–22: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 2 de la tabla 4-1.

Las tablas 4-23, 4-24 y 4-25 muestran los tiempos de ejecución en segundos de los resultados que se obtuvo en las tablas 4-20, 4-21 y 4-22 respectivamente.

Nivel	$m = 3$			$m = 8$			$m = 15$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	0.07	25.11	0.84	0.07	23.55	0.55	0.07	12.06	0.54
5	0.16	26.79	0.86	0.16	28.00	0.58	0.16	27.63	0.55
10	0.40	29.58	0.84	0.40	34.07	0.59	0.41	37.32	0.55
15	0.66	30.32	0.83	0.52	36.13	0.56	0.52	40.51	0.55
20	0.53	30.37	0.83	0.52	36.06	0.58	0.52	140.50	0.54

Tabla 4–23: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, 0, m)$, haciendo uso de la matriz 2 de la tabla 4-1.

Nivel	$m = 20$			$m = 40$			$m = 60$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	0.07	5.28	0.56	0.07	1.59	0.57	0.08	1.62	0.57
5	0.16	29.18	0.55	0.16	30.99	0.57	0.16	29.29	0.59
10	0.41	40.41	0.53	0.41	50.69	0.55	0.40	68.50	0.65
15	0.53	44.09	0.52	0.52	56.29	0.57	0.52	78.71	0.67
20	0.53	44.01	0.53	0.53	57.53	0.55	0.52	78.05	0.64

Tabla 4–24: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, 0, m)$, haciendo uso de la matriz 2 de la tabla 4-1.

Nivel	$m = 80$			$m = 100$			$m = 120$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	0.07	1.60	0.59	0.07	1.59	0.58	0.08	1.59	0.61
5	0.16	24.15	0.69	0.16	19.90	0.73	0.16	19.37	0.68
10	0.39	80.68	0.74	0.40	93.31	0.83	0.41	105.69	0.85
15	0.53	92.54	0.75	0.52	107.66	0.82	0.52	123.58	0.86
20	0.53	92.00	0.75	0.53	108.23	0.83	0.52	124.08	0.86

Tabla 4–25: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, 0, m)$, haciendo uso de la matriz 2 de la tabla 4-1.

De las tablas 4-23, 4-24 y 4-25, es posible inferir que al incrementar los elementos no ceros en el factor L, se presenta un aumento en el tiempo de ejecución en cada una de las diferentes fases.

En la tabla 4-26 se asigna un valor fijo para el parámetro umbral $\tau = 0$ y variando los valores de los parámetros de niveles y memoria.

Nivel	$m = 20$		$m = 40$		$m = 60$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	329505	50	342918	49	342918	49
5	142760	10	148655	10	148835	10
10	398812	38	778696	23	1157440	19
15	398812	38	779242	19	1537199	17

Tabla 4–26: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 3 descrita en la tabla 4-1.

Se presenta en la tabla 4-26 la cantidad de memoria requerida para el almacenamiento de la matriz L, en el cual se espera que al aumentar el valor de $\frac{Nz(L)}{Nz(A)}$, aumente la cantidad memoria utilizada para almacenar L.

Nivel	$m = 20$		$m = 40$		$m = 60$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	0.86	46.88	0.82	48.67	0.82	48.67
5	0.71	56.07	0.36	106.02	0.25	150.63
10	0.71	56.13	0.36	106.85	0.24	157.42
15	0.71	56.13	0.36	106.93	0.24	157.57

Tabla 4–27: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 3 de la tabla 4-1.

Nivel	$m = 20$			$m = 40$			$m = 60$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	1.23	2834.810	9.41	1.15	105.82	9.24	1.19	107.25	9.19
5	2.16	15802.5	2.11	13381.80	8.84	8.84	2.16	11309.40	9.95
10	6.25	16833.2	8.09	6.25	18864.40	8.49	6.54	19047.70	9.81
15	14.65	22532.2	8.23	14.25	24418.90	8.89	13.68	42035.70	9.70

Tabla 4–28: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, 0, m)$, haciendo uso de la matriz 3 de la tabla 4-1.

En la figura 4-3 se compara la norma de los residuos haciendo uso de los parámetros $\ell = 15$, $\tau = 0$ y variando el nivel de memoria m.

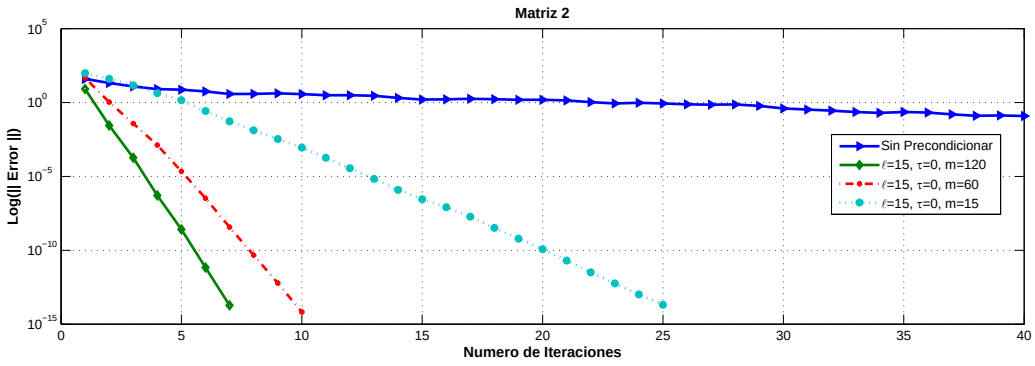


Figura 4–3: Historial de la norma de los residuos para valores $\ell = 15$, $\tau = 0$, $m = 15, 60, 120$, para la matriz 2 definida en la tabla 4-1.

En la figura 4-3 se puede ver como el aumentar el parámetro de memoria produce una disminución en el número de iteraciones requeridas para la convergencia hacia la solución del sistema. Por otro lado se genera un aumento en el tiempo de ejecución en la factorización numérica.

Ahora se presentará una comparación entre el número de iteraciones requeridas haciendo uso de **PCG** para una matriz precondicionada por $IC(\ell, \tau, 0)$ y $IC(\ell, 0, m)$.

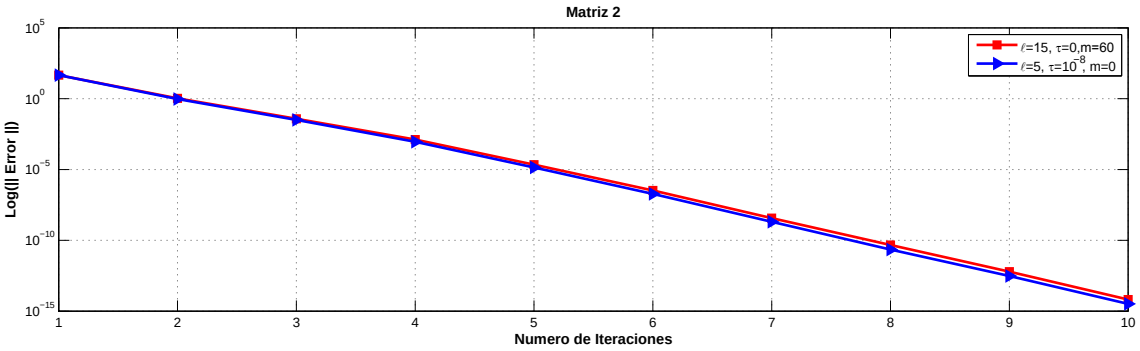


Figura 4–4: Historial de los residuos de $IC(5, 10^{-8}, 0)$ y $IC(15, 0, 60)$, para la matriz 2 de la tabla 4-1.

La figura 4-4 fue generada tomando valores para $\ell = 5$, $\tau = 10^{-8}$ y $m = 0$, y comparando estos con $\ell = 15$, $\tau = 0$, $m = 60$. En la figura 4-4 se observa una ventaja al utilizar $IC(5, 10^{-8}, 0)$ dado que la norma del error para este es menor que la generada por $IC(15, 0, 60)$.

La figura 4-5 muestra como el refinamiento de la malla influye en el número de iteraciones para la convergencia del método **PCG**.

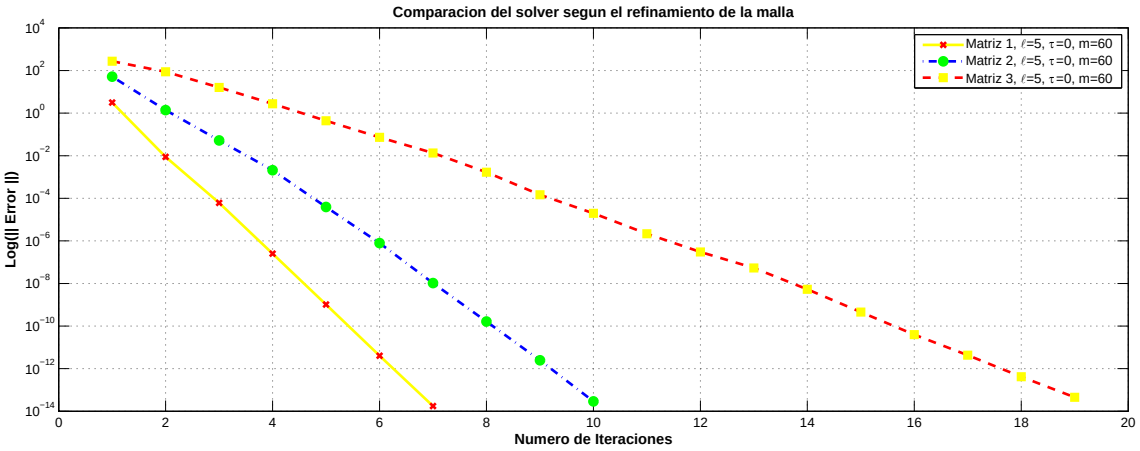


Figura 4–5: Historial de los residuo con los diferentes refinamiento haciendo uso de $ICl(5, 0, 60)$, para la matriz 2 de la tabla 4-1.

En la figura 4-5 se presenta la convergencia del método de Gradiente Conjugado Precondicionado con los diferentes refinamiento para un sistema. Se puede observar que para mallas más refinadas se genera mayor número de iteración para la convergencia de la solución del sistema.

4.1.4. Condicionamiento espectral

Los experimentos anteriores han demostrado una disminución en el número de iteraciones requeridas para la convergencia del método **PCG**, cuando se aumenta los elementos no cero en L. Esta disminución en el número de iteraciones no depende necesariamente de la cantidad de elementos no ceros en L, si no, que depende de la propiedad espectral de la matriz de coeficientes.

Para validar esta disminución de iteraciones se presentan resultados numéricos, los cuales muestran que el condicionamiento espectral de la matriz asociada al sistema precondicionado se aproxima a 1, esto es, $\sigma(M^{-1}A) \approx 1$. Las siguientes figuras 4-6, 4-7 y 4-8 muestran el condicionamientos espectral del sistema precondicionado con la matriz 2 de la tabla 4-1 para $\ell = 1$. En las figura 4-6 y 4-7, se ha definido

a la matriz A como la matriz 2 y la matriz M como la matriz precondicionada por $IC(1,0,0)$.

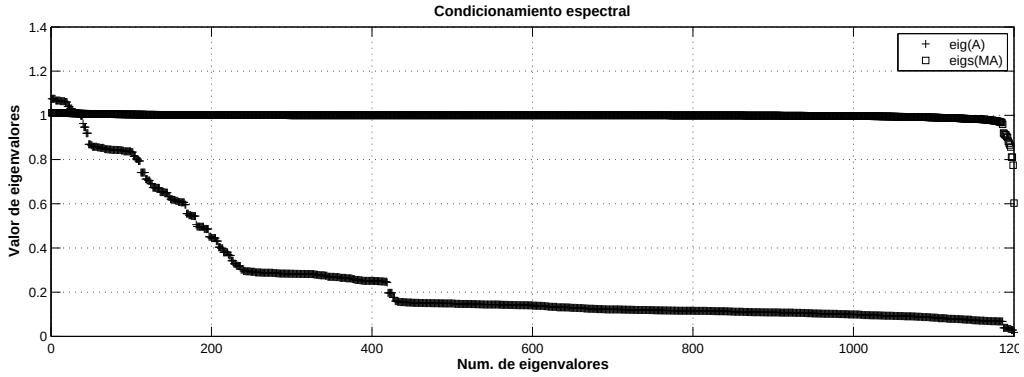


Figura 4–6: Condicionamiento espectral con la matriz precondicionada con $IC(1,0,0)$, para la matriz 1 de la tabla 4-1.

En la figura 4-6 se puede observar como la matriz precondicionada provee un mejor condicionamiento espectral. La figura 4-7 muestra los 120 primeros autovalores de la matriz precondicionada y sin precondicionar.

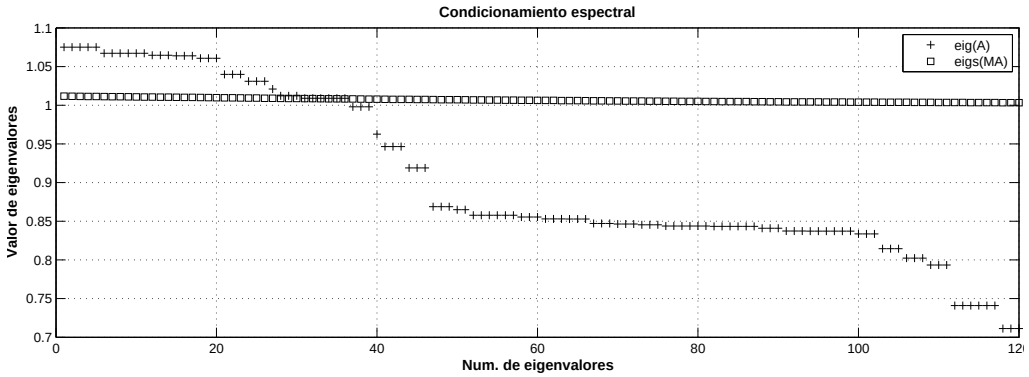


Figura 4–7: Condicionamiento espectral con la matriz precondicionada con $IC(1,0,0)$, para la matriz 1 de la tabla 4-1.

Se compara en la figura 4-8 el condicionamiento espectral de las matrices precondicionadas por los precondicionadores generados por $IC(1,0,0)$, $IC(5,0,0)$, $IC(15,0,0)$.

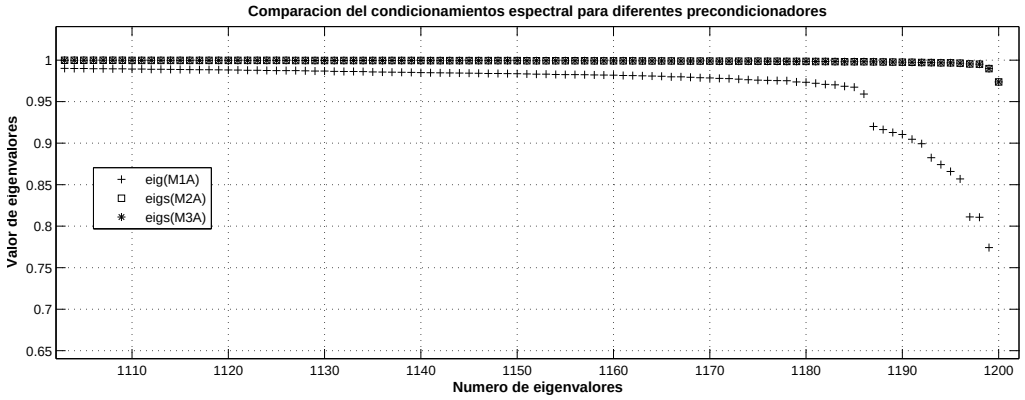


Figura 4–8: Comparación de los condicionamientos espectrales de las matrices precondicionadas por $IC(1,0,0)$, $IC(5,0,0)$, $IC(15,0,0)$, para la matriz 1 de la tabla 4-1.

En la figura 4-8 los precondicionadores $M1, M2, M3$ son generados por $IC(1,0,0)$, $IC(5,0,0)$, $IC(15,0,0)$ respectivamente. Se puede observar como el incrementar los niveles proporciona un mejor condicionamiento espectral y a su vez se logró observar que se disminuye el número de iteraciones.

Se analizará el condicionamiento espectral de la matriz 2 y de la matriz 2 precondicionada para diferentes valores del parámetro ℓ. Primero se compara el condicionamiento espectral de la matriz 2 con la matriz 2 precondicionada por medio de $IC(1,0,0)$.

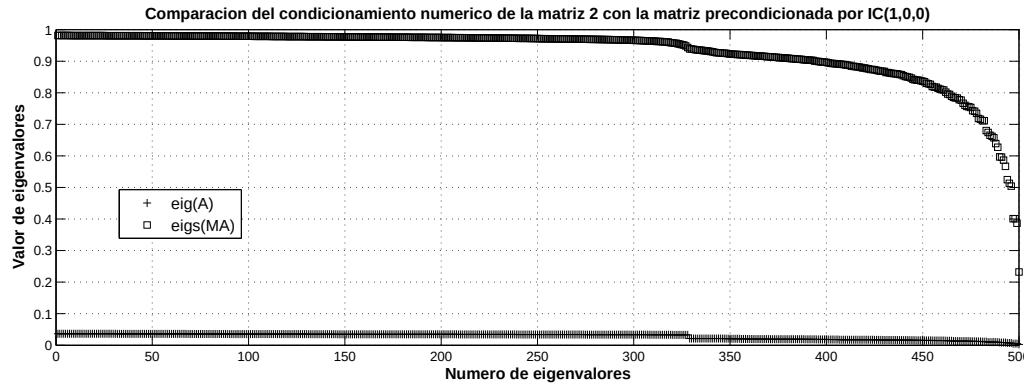


Figura 4–9: Comparación de los condicionamientos espectrales de las matrices precondicionadas por $IC(1,0,0)$ y la matriz 2 de la tabla 4-1.

En la figura 4-9 se observar que $\sigma(M^{-1}A) \approx 1$ para la matriz 2 precondicionada por $IC(1,0,0)$. Ahora se compara el condicionamiento espectral para la matriz 2 precondicionada por $IC(1,0,0)$, $IC(3,0,0)$ y $IC(15,0,0)$.

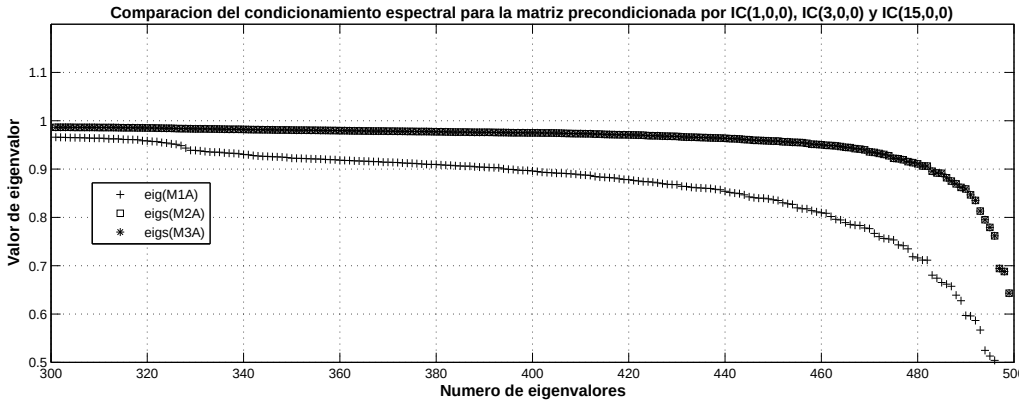


Figura 4–10: Condicionamientos espectrales de las matrices precondicionadas por $IC(1,0,0)$, $IC(3,0,0)$, $IC(15,0,0)$, para la matriz 2 de la tabla 4-1.

En la tabla 4-3 se observa una disminución en el número de iteraciones, tal efecto es causado por el condicionamiento espectral de la matriz precondicionada. En la figura 4-10 se observa como se aproxima el condicionamiento espectral de la matriz a 1 cuando se aumenta el parámetro de niveles y este permite mayor entradas de elementos no ceros.

Con estos resultados queda mostrado la relación que existe entre el condicionamiento espectral de una matriz con el número de iteraciones requeridas para la convergencia de un método iterativo.

Para las pruebas realizadas hasta este punto se hizo uso de la norma matricial $\|\cdot\|_1$. Ahora se presentará algunas pruebas tomando la norma matricial de Frobenius $\|\cdot\|_F$, con el propósito de comparar los resultados con las dos normas antes

mencionadas.

Precondicionador $IC(\ell, 0, 0)$

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	39362	1.17	26
3	65493	1.94	16
5	148835	4.42	10
8	288809	8.58	7
11	397327	11.81	5
15	397327	13.04	3

Tabla 4–29: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$, $m = 0$ y variado ℓ. Para la matriz 2 descrita en la tabla 4-1

Al comparar la tabla 4-29 con la tabla 4-3, se observa la misma cantidad de elementos no cero en L y el mismo número de iteraciones por cada valor del parámetro ℓ. Luego el cambiar la norma matricial no altera la selección del posible patrón de esparcimiento.

Precondicionador $IC(\ell, \tau, 0)$

Nivel	$\tau = 10^{-3}$		$\tau = 10^{-5}$		$\tau = 10^{-8}$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	38708	26	38708	26	38708	26
5	140040	10	142491	10	142500	10
10	297723	7	347764	5	357639	5
15	332695	7	406976	4	423747	3
20	332908	7	407253	4	424129	2

Tabla 4–30: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$ y variado ℓ y $\tau = 0$. Para la matriz 2 descrita en la tabla 4-1

Con la tabla 4-30 se puede observar como el cambio de la norma matricial para calcular el valor del umbral τ logra influir en la cantidad de elementos permitidos en el factor L. Para la norma matricial de Frobenius se obtiene menos elementos no ceros en el factor L al compararlo con la norma matricial $\|\cdot\|_1$, generando el mismo número de iteraciones obtenidos con $\|\cdot\|_1$. Con esto se puede inferir que el uso de la norma matricial de Frobenius puede ser más útil cuando se requiere tener mayor esparcimiento en el factor L, necesitando menos espacio en memoria para el almacenamiento de la matriz L. Pero cuando se desea garantizar el mínimo número de iteraciones posibles para la convergencia del método **PCG** se recomienda hacer uso de la norma matricial $\|\cdot\|_1$.

Ahora se analiza como podría influir el cambio de la norma matricial haciendo uso del parámetro de memoria m.

Precondicionador $IC(\ell, 0, m)$

Nivel	$m = 80$		$m = 100$		$m = 120$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	39362	26	39362	26	39362	26
5	142760	10	148655	10	148835	10
10	183332	9	225504	8	265622	7
15	183859	9	226680	8	268269	7
20	183859	9	226680	8	268269	7

Tabla 4–31: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 2 descrita en la tabla 4-1.

Al comparar los datos del cuadro 4-16 con 4-31, se observa que no se produjo cambio alguno al utilizar la norma matricial de Frobenius al momento de escoger los bloques más grandes. Con esto se puede recomendar cualquiera de las dos normas matriciales, cuando se trabaja con el parámetro de memoria m.

Basado en la necesidad de reducir al máximo el número de iteraciones para la convergencia del método **PCG**, se sigue los experimentos numéricos con la norma matricial $\| \cdot \|_1$.

4.2. Experimentos para el operador Helmholtz

Los experimentos que se realizaron a continuación están basados en la discretización de la ecuación

$$\nabla \times \nabla \times E + (1 - \omega^2)E = J \tag{4.1}$$

por medio del método LDG [39]. Se obtendrán algunos resultados satisfactorios al implementar el precondicionador $IC(\ell, \tau, m)$. Para los experimentos numéricos se emplea un vector inicial $x_0 = 0$, y un vector fuente b. El criterio de parada

$$\| r \|_2 \leq 10^{-8}$$

donde r es un vector residual generado por el método del Gradiente Conjugado Precondicionado, en adición se a impuesto un limite de 2000 iteraciones. El símbolo"-"significa que no alcanzo la convergencia en las 2000 iteraciones.

Las característica de la matriz A son las siguientes:

Nombre	dim. Global	dim. Bloques	NzB
Matriz 4	320×320	12×12	1960
Matriz 5	2560×2560	12×12	17264

Tabla 4–32: Información de las matrices.

Las matrices generadas por la ecuación (4.1) con $\omega = 0{,}25$ son

Nombre	dim. Global	dim. Bloques	NzB
Matriz 4.1	320×320	12×12	1960
Matriz 5.1	2560×2560	12×12	17264

Tabla 4–33: Información de las matrices.

- Son matrices simétricas y definidas positivas.
- El flujo numérico es seleccionado en la dirección del vector $(1{,}0, 1{,}0, 1{,}0)^t$.
- La secuencia de mallas son: 320, 2560 celdas (ref=refinamiento).
- Se utiliza una base polinómica de Lagrange de grado $p = 1$, número de grado de libertad por componente son 4, por celdas son 12.

4.2.1. Experimentos para $IC(\ell, 0, 0)$

Inicialmente el parámetro de memoria y el parámetro umbral toman un valor de cero y varia el parámetro de niveles. Lo que se desea con este experimento es ver como el parámetro de niveles influye en la construcción del precondicionador sin descartar elementos con el parámetro umbral y el parámetro de memoria.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	1870	0.95	–
5	1920	0.98	428
10	7436	3.80	–
15	9940	5.07	10
20	10014	5.12	2

Tabla 4–34: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y $m = 0$ variado ℓ. Para la matriz 4 descrita en la tabla 4-32

En la tabla 4-34 se presenta una disminución en el número de iteraciones al aumentar el número de elementos no ceros en el factor L, con los valores $\ell = 10$, $m = 0$ y $\tau = 0$ se presenta una divergencia al usar el método de Gradiente Conjugado Precondicionado. Este efecto puede ser causado por los desplazamientos realizados durante la factorización númerica. Este caso particular produjo 106 desplazamientos obteniendo sin éxito 13 factorizaciones completas a los elementos de la diagonal de L. Por otro lado, para el valor de $\ell = 1$ no se alcanzo la convergencia en las 2000 iteraciones. Para $\ell = 1$ se produjo 66 desplazamientos no obteniendo una factorización completa para 3 elementos.

Para observar el historial del residuo y el comportamiento de la convergencia se presentan las figuras 4-11,4-12 y 4-13, para los datos suministrados en la tabla 4-34.

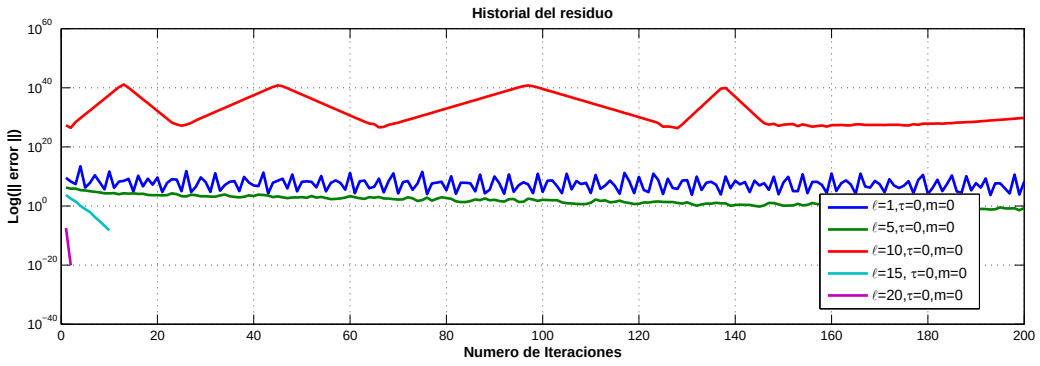


Figura 4–11: Historial de los residuo con diferentes valores para l y tomando $\tau = 0$, $m = 0$, Matriz 4 de la tabla 4-32.

La figura 4-12 compara el historial de la convergencia de $IC(5, 0, 0)$, $IC(15, 0, 0)$, $IC(20, 0, 0)$.

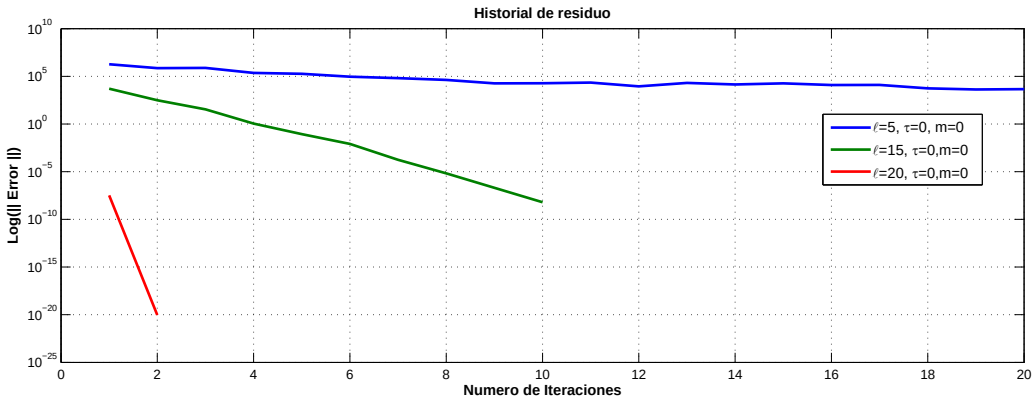


Figura 4–12: Historial de los residuo con diferentes valores para l y tomando $\tau = 0, m = 0$, para la matriz 4 de la tabla 4-12.

La figura 4-11 muestra el comportamiento de la convergencia al utilizar el método del Gradiente Conjugado Precondicionado. Se puede observar que para $IC(10, 0, 0)$ el residuo se mantuvo constante con orden de 10^{30}, mientras que para $IC(1, 0, 0)$ el residuo no disminuyó oscilando entre 10^5 y 10^6. por otro lado, para $IC(15, 0, 0)$ y $IC(20, 0, 0)$ se presenta una convergencia hacia la solución.

En la tabla 4-35 se realizan experimentos con la matriz 5 variando los niveles y la memoria, obteniendo consigo la cantidad de bloques no ceros del factor L, Nz(L), el indicador de memoria $\frac{Nz(A)}{Nz(L)}$, la cantidad de iteraciones (Iter).

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	17353	1.01	–
5	34821	2.04	–
15	251201	16.66	–
20	317597	18.39	–
30	338881	19.62	8
60	338974	19.63	2

Tabla 4–35: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$, $m = 0$ y variado ℓ. Para la matriz 5 descrita en la tabla 4-31

La figura 4-13 muestra el comportamiento de la convergencia del método **PCG**.

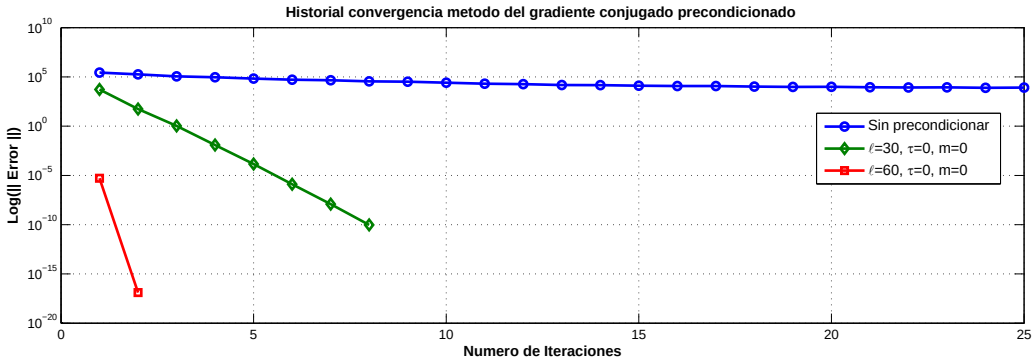


Figura 4–13: Historial de los residuo con diferentes valores para ℓ y tomando $\tau = 0$, $m = 0$, Matriz 5.

En la figura 4-13 se presenta una disminución en el número de iteraciones haciendo uso del método **PCG** para la matriz precondicionada por $IC(\ell, \tau, m)$ al compararlo con la matriz no precondicionada utilizando el método del Gradiente Conjugado. Por otra parte se observa que al aumentar el parámetro de nivel ℓ se obtiene una disminución en el número de iteraciones requeridas para alcanzar la convergencia hacia la solución con una exactitud deseada. Una mejor observación en el comportamiento de la convergencia haciendo uso del método **PCG** puede ser apreciada en la figura 4-14.

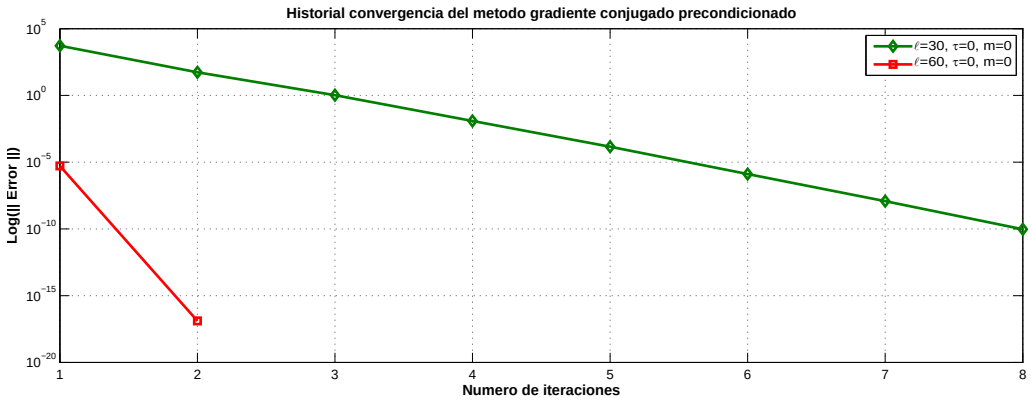


Figura 4–14: Historial de los residuo con diferentes valores para ℓ y tomando $\tau = 0$, $m = 0$, para la matriz 5 de la tabla 4-32.

Se generan nuevos experimentos haciendo uso de la matriz 4.1 de la tabla 4-33. En la tabla 4-36 se puede observar el número de iteraciones requerida para la convergencia del método **PCG** y además el índice de memoria.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	1870	0.95	–
5	1920	0.98	464
10	7436	3.80	–
15	9940	5.07	11
20	10014	5.12	2

Tabla 4–36: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$, $m = 0$ y variado ℓ. Para la matriz 4.1 descrita en la tabla 4-33

Para los niveles $\ell = 1$, $\ell = 10$, no se logró obtener una convergencia hacia la solución, debido que durante la factorización numérica para $\ell = 1$ en los bloques $A_{221,211}$, $A_{281,281}$, $A_{284,284}$ se obtuvo una factorización completa de Cholesky resaltando que algunos de los menores principales del bloque no son definidos positivo, con $\ell = 10$ se obtuvo que para 11 elementos de las diagonales no se logro obtener una factorización completa de Cholesky. Esto aun realizando la estrategia para el

desplazamiento.

Para $\ell = 5$ se obtuvieron 56 desplazamientos logrando de manera satisfactoria las factorizaciones completas de Cholesky y para $\ell = 15$, $\ell = 20$ no se obtuvo desplazamientos.

La tabla 4-37 presenta resultados de precondicionar la matriz 5.1 por medio del precondicionador $IC(\ell, 0, 0)$ donde L tomará valores desde 1 hasta 60.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	17353	1.00	–
5	34821	2.01	–
10	139220	8.06	–
15	251201	14.55	–
30	338881	19.62	8
60	338974	19.63	2

Tabla 4–37: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$, $m = 0$ y variado ℓ. Para la matriz 5.1 descrita en la tabla 4-33

En la tabla 4-37 se puede observar como al incrementar los niveles se obtuvo una convergencia hacia la solución haciendo uso del método **PCG**. Para valores $\ell = 1, \ldots, 15$ no se obtuvo convergencia debido a fallidos cálculos durante una factotización completa de Cholesky para los elementos de las diagonales, almenos para $\ell = 1$ se obtuvo 1020 desplazamientos, para los cuales en 141 bloques no se produjo una factorización completa de Cholesky, esto debido a que uno de los menores principales no eran definida positiva. Para $\ell = 5, 10, 15$ se produjo alrededor de 350

desplazamientos para los cuales 100 de los bloques sus menores principales resultaron ser no definida positiva.

4.2.2. Experimentos para $IC(\ell, \tau, 0)$

Con esta factorización incompleta se varia los niveles y el parámetro umbral, con el objetivo de eliminar posibles entradas de llenados generadas por la fase simbólica y esperando una disminución en el número de iteraciones para la convergencia del **PCG**. Obteniendo consigo la cantidad de bloques no ceros del factor L, Nz(L), la cantidad de iteraciones (Iter).

Nivel	$\tau = 10^{-1}$		$\tau = 10^{-3}$		$\tau = 10^{-7}$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	1870	–	1870	–	1870	–
5	1920	428	1920	428	1920	428
15	8093	–	9921	11	9940	10
20	8169	–	9992	8	10014	2

Tabla 4–38: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 4 descrita en la tabla 4-32

En la tabla 4-38 algunos resultados obtenidos no fueron satisfactorios dado que no se alcanzó la convergencia hacia la solución. Esto puede ser a que se ha producido muchos desplazamientos durante la factorización numérica, generando así un precondicionador no definido positivo imposibilitando la convergencia del gradiente conjugado precondicionado.

En la tabla 4-39 se realizan experimentos con la matriz 5 de la tabla 4-32 variando los niveles y el parámetro umbral, obteniendo consigo la cantidad de bloques

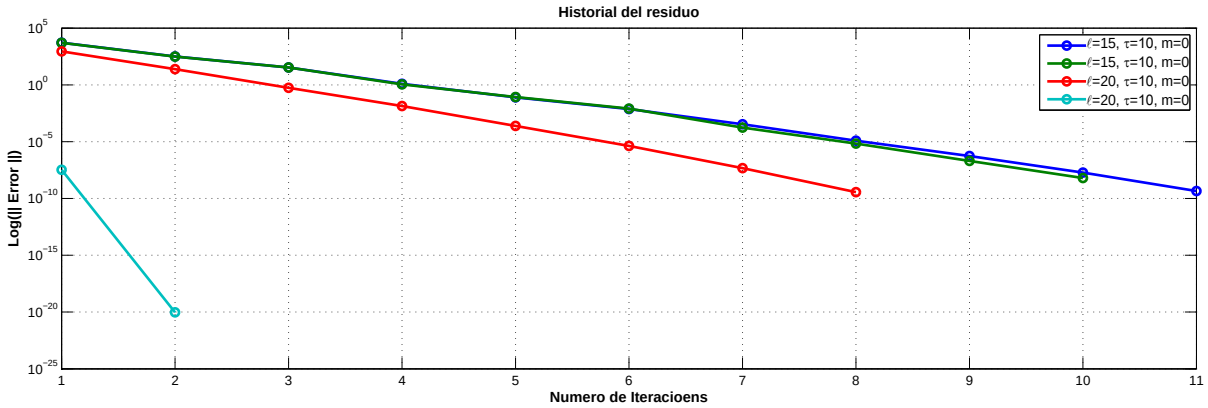


Figura 4–15: Historial de los residuo con diferentes valores para ℓ y tomando $\tau = 10^{-3}$, $\tau = 10^{-7}$, $m = 0$, Matriz 4.

no ceros del factor L, Nz(L) y la cantidad de iteraciones *Iter*.

Nivel	$\tau = 10^{-1}$		$\tau = 10^{-3}$		$\tau = 10^{-7}$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
30	223593	–	326778	23	338871	8
60	229909	–	326836	23	338964	3

Tabla 4–39: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$ y variado ℓ, $\tau = 0$. Para la matriz 5 descrita en la tabla 4-32.

La tabla 4-40 presenta los tiempos de ejecución para la tabla 4-39.

Nivel	$\tau = 10^{-1}$			$\tau = 10^{-3}$			$\tau = 10^{-7}$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
30	0.36	93.91	391.68	0.36	129.88	7.00	0.36	134.02	2.76
60	0.37	95.90	410.90	0.37	132.24	6.90	0.37	136.15	1.23

Tabla 4–40: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, \tau, 0)$, haciendo uso de la matriz 5 de la tabla 4-32.

Para comparar los resultados de la tabla 4-39, se presenta la figura 4-16 la cual compara la norma de los residuos obtenidos.

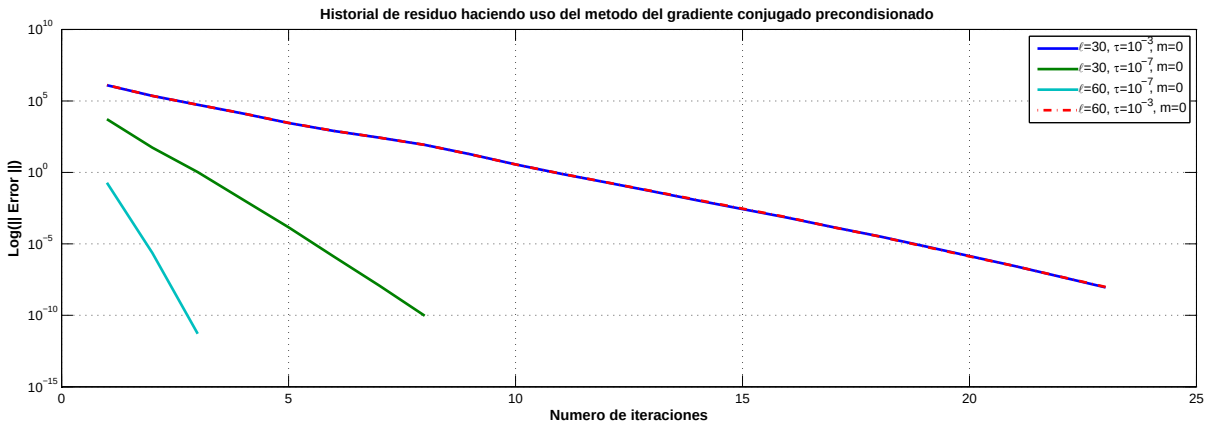


Figura 4–16: Historial de los residuo con diferentes valores para $\ell = 30$, $\ell = 60$ y tomando $\tau = 10^{-3}, 10^{-7}$, $m = 0$, para la matriz 5 descrita en la tabla 4-32.

Se puede notar que el aumento del parámetro de nivel ℓ y el parámetro de umbral τ proporciona un menor número de iteraciones para alcanzar la convergencia deseada.

Los experimentos de la tabla 4-41 y 4-42 serán para las matrices la tabla 4-33.

Nivel	$\tau = 10^{-1}$		$\tau = 10^{-3}$		$\tau = 10^{-7}$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	1870	–	1870	–	1870	–
5	1920	457	1920	457	1920	457
15	8093	–	9921	11	9940	11
20	8169	–	9992	8	10014	2

Tabla 4–41: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$ y variado ℓ y τ. Para la matriz 4.1 descrita en la tabla 4-33

En la tabla 4-41 se puede observar como el aumento de niveles y una disminución en el valor del parámetro τ se produce un aumento en las entradas no ceros del factor L, y consigo una disminución en el número de iteraciones.

Nivel	$\tau = 10^{-1}$		$\tau = 10^{-3}$		$\tau = 10^{-7}$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
30	234736	–	326946	24	338871	8
60	221274	–	327030	24	338964	3

Tabla 4–42: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 5.1 descrita en la tabla 4-33

El intentar disminuir el número de entradas no ceros en el factor L, se genera un aumento en el número de iteraciones necesarias para la convergencia de la solución, inclusive se podría no obtener la convergencia hacia la solución.

4.2.3. Experimentos para $IC(\ell, 0, m)$

En la tabla 4-43 se realizan experimentos con la matriz 4 descrita en la tabla 4-32, variando los niveles y la memoria, dejando el parámetro τ fijo en cero. Con esta factorización se desea ver como influye el parámetro de memoria en el llenado de elementos no ceros del factor L.

Nivel	$m = 8$		$m = 15$		$m = 20$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	1757	–	1870	–	1870	–
5	1772	–	1920	428	1920	428
15	2365	–	4098	–	5278	–
20	2365	–	4098	–	5278	–

Tabla 4–43: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 4 descrita en la tabla 4-32

Nivel	$m = 40$		$m = 80$		$m = 100$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
5	1920	428	1920	428	1920	428
15	9225	–	9940	10	9940	10
20	9243	–	10014	2	10014	2

Tabla 4–44: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 4 descrita en la tabla 4-32.

Nivel	$m = 150$		$m = 170$		$m = 200$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
30	305409	–	337637	13	338881	8
60	305412	–	337718	13	338974	2

Tabla 4–45: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 5 descrita en la tabla 4-32

En la tabla 4-46 se presenta el tiempo de ejecución para $m = 150$, $m = 170$ y $m = 200$.

Nivel	$m = 150$			$m = 170$			$m = 200$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
30	0.36	136.00	583.72	0.36	141.53	4.28	0.36	135.75	2.78
60	0.37	137.00	542.37	0.37	153.59	4.39	0.37	137.13	0.89

Tabla 4–46: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, 0, m)$, haciendo uso de la matriz 4 de la tabla 4-32.

La figura 4-17 compara el historial de la norma del residuo generado por los sistemas precondicionados mediante $IC(30, 0, 170)$, $IC(30, 0, 200)$, $IC(60, 0, 200)$,

$IC(60, 0, 170)$.

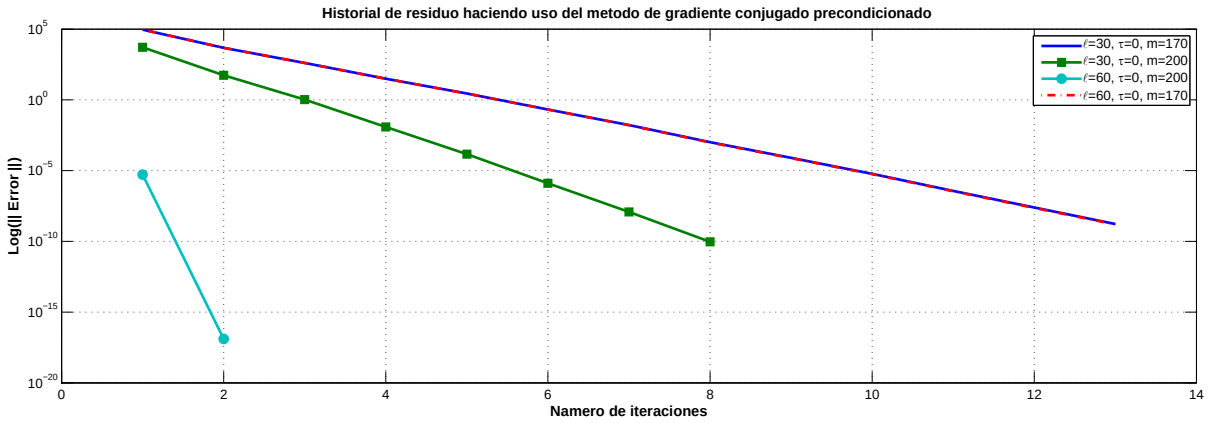


Figura 4–17: Historial de los residuo con diferentes valores para $\ell = 30$, $\ell = 60$ y tomando $\tau = 0$, $m = 170$, $m = 200$, para la matriz 5 de la tabla 4-32.

Se puede notar que el aumento del parámetro de nivel ℓ y el parámetro de memoria m proporciona un menor número de iteraciones para alcanzar la convergencia deseada.

La figura 4-18 muestra una comparación entre los resultados de la tabla 4-39 y 4-42. En la figura 4-18 es posible inferir que el utilizar un valor adecuado para el parámetro de memoria m se puede alcanzar la convergencia hacia la solución del sistema en menos iteraciones.

En la tabla 4-40 y 4-46 se puede observar que durante la fase numérica al utilizar $IC(\ell, 0, m)$ el tiempo de ejecución es mayor que el requerido en $IC(\ell, \tau, 0)$, mientras que el tiempo de ejecución del solver es menor al utilizar $IC(\ell, 0, m)$.

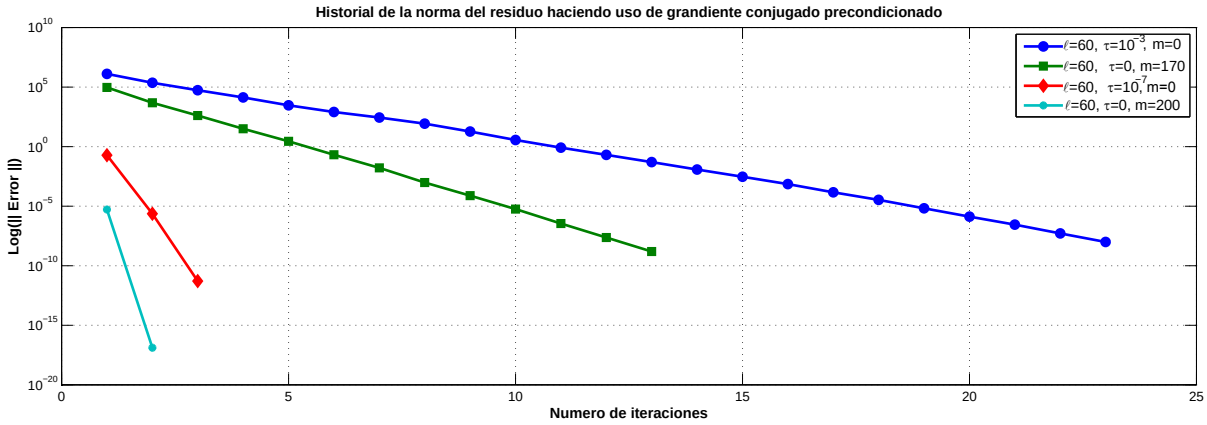


Figura 4–18: Historial de los residuo con diferentes valores para $\ell = 60$ y tomando $\tau = 10^{-3}, 10^{-7}$, $m = 170, 200$, para la Matriz 5 de la tabla 4-32.

Los resultados de la tabla 4-47 y 4-48 son para matrices generadas por la discretización de la ecuación (4.1) con $\omega = 0{,}25$

Nivel	$m = 40$		$m = 80$		$m = 100$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
5	1920	457	1920	457	1920	457
15	9225	–	9940	11	9940	11
20	9243	–	10014	2	10014	2

Tabla 4–47: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 4.1 descrita en la tabla 4-33

En la tabla 4-47 al incrementar el parámetro de memoria se obtiene una disminución en el número de iteraciones al aplicar el método del Gradiente Conjugado Precondicionado.

Ahora se muestra los resultados cuando se hace uso del parámetro de memoria para la matriz 5.1.

Nivel	$m = 150$		$m = 170$		$m = 200$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
30	305409	–	337637	13	338881	8
60	305412	–	337718	13	338974	2

Tabla 4–48: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 4.1 descrita en la tabla 4-33.

En la tabla 4-48 se observa que el incremento del parámetro de memoria genera una disminución en el número de iteraciones para obtener una solución aproximada al utilizar el método **PCG**. Como también se presenta un aumento en el número de elementos no ceros en el factor L.

4.2.4. Condicionamiento espectral

A continuación se presentan algunos resultados numéricos que relacionan el condicionamiento espectral de la matriz con el número de iteraciones requeridas para la convergencia del método **PCG**.

En las figuras 4-19 y 4-20 comparan el condicionamiento espectral de la matriz 5 descrita en la tabla 4-32 con el condicionamiento espectral de la matriz 5 precondicionada por $IC(1, 0, 0)$.

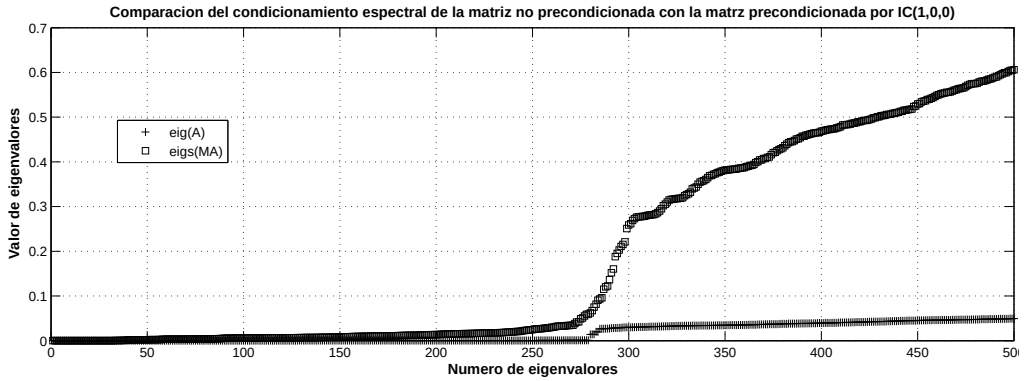


Figura 4–19: Comparación del condicionamiento espectral de la matriz 5 y la matriz 5 precondicionada por $IC(1, 0, 0)$, para la matriz 5 de la tabla 4-32.

Ahora se realiza una comparación con el condicionamiento espectral de la matriz 5 precondicionada por $IC(1,0,0)$, $IC(5,0,0)$, $IC(20,0,0)$.

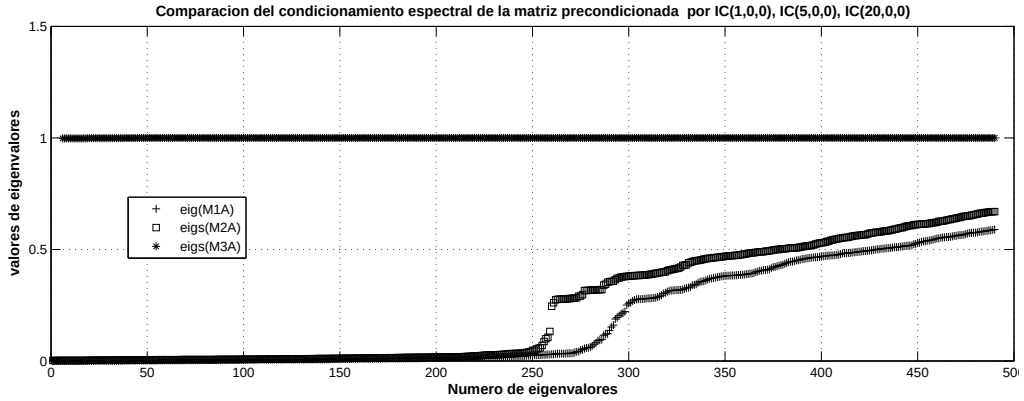


Figura 4–20: Comparación del condicionamiento espectral de la matriz precondicionada por $IC(1,0,0)$, $IC(5,0,0)$,$IC(20,0,0)$, para la matriz 4 de la tabla 4-32.

En la figura 4-20 se puede observar como el incremento del parámetro ℓ, permite mejorar el condicionamiento espectral de la matriz asociada al sistema permitiendo obtener la convergencia del método en un menor número de iteraciones.

En la siguiente sección se presenta algunos resultados numéricos haciendo uso de la factorización incompleta de Cholesky $IC(\ell,\tau,m)$ de manera puntual, con las matrices utilizadas para las pruebas de $IC(\ell,\tau,m)$ por bloques. Se inicia haciendo pruebas con las matrices generadas por el operador Laplaciano y luego se sigue con las matrices generadas por el operador Helmholtz.

4.3. Factorización $IC(\ell,\tau,m)$ puntual para el operador Laplaciano

Para los resultados numéricos se ha tomado un vector inicial $x_0 = 0$, los criterios de parada están basados en el error del sistema precondicionado, si este satisface $\| r \| < 10^{-13}$ y en un máximo de 1000 iteraciones.

Nombre	dim. Global	dim. Bloques	Nz
Matriz 7	1280×1280	1×1	60160
Matriz 8	10240×10240	1×1	538112
Matriz 9	81920×81920	1×1	4541440

Tabla 4–49: Información de las matrices.

4.3.1. Experimentos para $IC(\ell, 0, 0)$ puntual

A continuación se presentarán experimentos numéricos donde el parámetro de nivel ℓ toma valores que varia entre 1 a 15, mientras los parámetros de memoria m y umbral τ se han dejado en cero. Con esto se muestra la influencia que tiene el parámetro ℓ en la cantidad de elementos no ceros del factor L, afectando a su vez el número de iteraciones requeridas al usar el método **PCG**.

La tabla 4-50 muestra el número de iteraciones (Iter) requerida al variar el parámetro de niveles.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	200176	3.33	8
5	212816	3.57	4
15	212960	3.57	2

Tabla 4–50: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y $m = 0$ variado ℓ. Para la matriz 7 descrita en la tabla 4-49

Para la tabla 4-50 se presenta una disminución en el número de iteraciones al incrementar la cantidad de elementos no ceros en el factor L.

Ahora se realizará una prueba numérica asignando a los parámetros ℓ, τ y m los mismo valores utilizados en la prueba anterior para la matriz 8 descrita en la

tabla 4-49.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
1	6491360	12.19	9
5	6995360	13.15	5

Tabla 4–51: Valores del parámetro ℓ varían con $m = 0$ y $\tau = 0$, para la matriz 8.

Con los datos suministrados en el cuadro 4-48 se aprecia una disminución en el número de iteraciones al incrementar el valor del nivel.

4.3.2. Experimentos para $IC(\ell, \tau, 0)$ puntual

Los resultados numéricos de la tabla 4-52, 4-52 y 4-53 se realizan tomando un valor entre 10^{-3} y 10^{-7} para el parámetro τ, con la idea de resaltar como el parámetro umbral τ retiene elementos se ha dejado el parámetro de memoria con un valor fijo en cero.

Nivel	$\tau = 10^{-1}$		$\tau = 10^{-5}$		$\tau = 10^{-7}$	
	Nz(L)	Ite	Nz(L)	Iter	Nz(L)	Iter
1	980557	36	4393669	9	5665416	9
5	1065040	36	4764053	6	6126870	5

Tabla 4–52: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $m = 0$ y variado ℓ y τ. Para la matriz 8 descrita en la tabla 4-49

Nivel	$\tau = 10^{-1}$		$\tau = 10^{-5}$		$\tau = 10^{-7}$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	1.85	11.41	8.33	50.47	10.63	65.03
5	2	12.38	9.09	54.71	11.49	70.31

Tabla 4–53: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 8 de la tabla 4-49.

Nivel	$\tau = 10^{-1}$			$\tau = 10^{-5}$			$\tau = 10^{-7}$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	18.58	8862.96	0.48	18.79	31345.10	1.85	19.70	41526.40	0.69
5	19.20	10294.80	0.34	18.74	36465.90	0.26	18.68	46841.50	0.27

Tabla 4–54: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, \tau, 0)$, haciendo uso de la matriz 8 de la tabla 4-49.

La figura 4-21 compara el historial de la norma del residuo para el sistema precondicionado por $IC(1, 10^{-7}, 0)$ y $IC(1, 10^{-1}, 0)$.

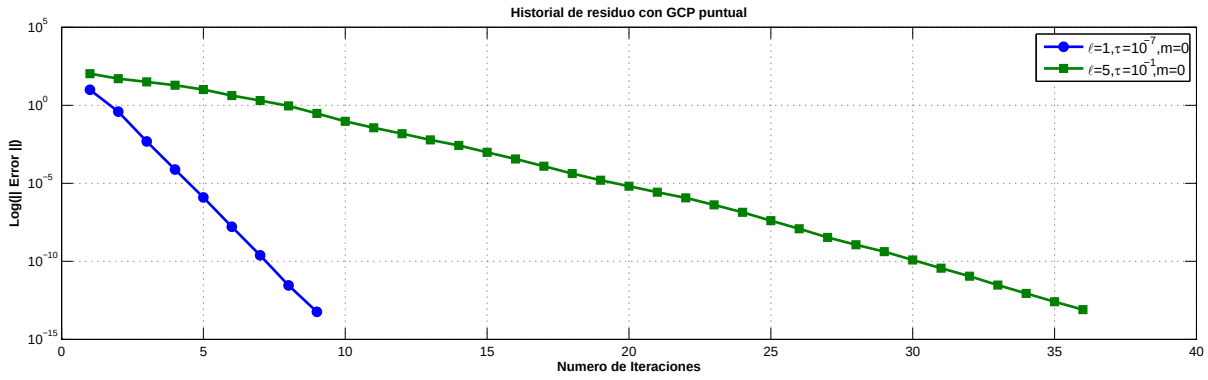


Figura 4–21: Historial de los tiempos para $IC(\ell, \tau, 0)$ con $l = 1, 5$. $\tau = 10^{-1}, 10^{-7}$, Matriz 8.

Estos resultados serán comparados en la figura 4-22 con los resultados obtenidos por bloques con la matriz 2 descrita en la tabla 4-1. La comparación esta realizada según el índice $\frac{Nz(L)}{Nz(A)}$ de elementos no ceros en el factor L, se esta comparando $IC(1, 10^{-5}, 0)$ versión puntual con $IC(10, 10^{-5}, 0)$ versión por bloques y se compara $IC(5, 10^{-5}, 0)$ versión puntual con $IC(20, 10^{-5}, 0)$ version por bloques. La infomación respecto a la cantidad de memoria utilizada en la versión puntual y por bloques

pueden ser vista en los cuadros 4-50 y 4-10 respectivamente.

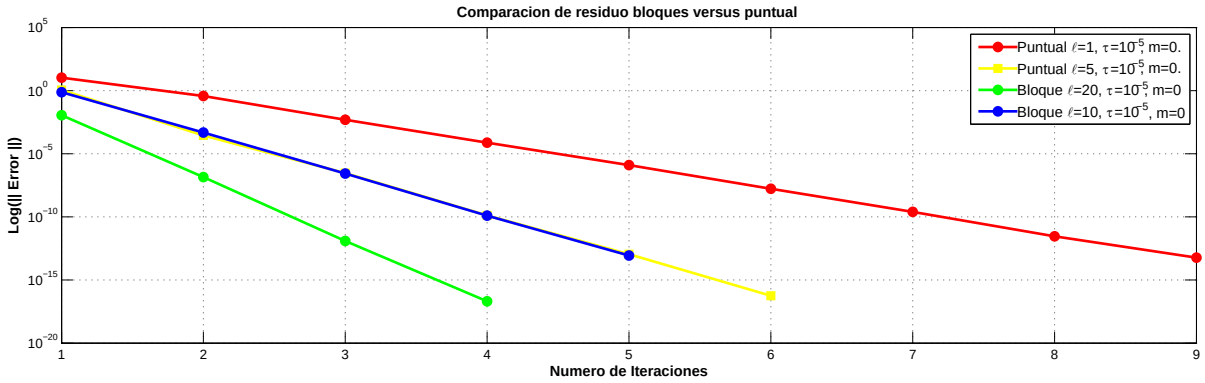


Figura 4–22: Comparación de los residuos generados por la versión puntual y la versión por bloque con la matriz 2 y matriz 8.

Las matrices comparadas cumples con las siguientes características:

- La matriz 2 precondicionada con $\ell = 10$, $\tau = 10^{-5}$, $m = 0$, genera un factor L asociado a un índice de 46.79. Esta fue comparada con la matriz 8 precondicionada con $\ell = 1$, $\tau = 10^{-5}$, $m = 0$, que genera un factor L asociado a un índice de 50.47.
- La matriz 2 precondicionada con $\ell = 20$, $\tau = 10^{-5}$, $m = 0$, genera un factor L asociado a un índice de 54.79. Esta fue comparada con la matriz 8 precondicionada con $\ell = 1$, $\tau = 10^{-5}$, $m = 0$, que genera un factor L asociado a un índice de 54.71.

Estas comparaciones muestran el beneficio de utilizar un precondicionador por bloques a un precondicionador puntual. Dado que al utilizar el precondicionador por bloque se requiere de menos iteraciones para alcanzar la convergencia hacia la solución haciendo uso del método **PCG**.

4.3.3. Experimentos para $IC(\ell, 0, m)$ puntual

Ahora se mostrará algunos resultados numéricos donde el parámetro principal es el parámetro de memoria combinado con el parámetro de niveles.

Nivel	$m = 80$		$m = 100$		$m = 120$	
	Nz(L)	Iter	Nz(L)	Iter	Nz(L)	Iter
1	782710	494	966806	545	1150180	486
5	782840	483	967042	371	1150576	224

Tabla 4–55: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 8 descrita en la tabla 4-49

Ahora se analizará la memoria ocupada por la prueba realizada en la figura 4-55.

Nivel	$m = 80$		$m = 100$		$m = 120$	
	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb	$\frac{Nz(L)}{Nz(A)}$	Memoria Mb
1	1.47	9.15	1.81	11.25	2.17	13.35
5	1.47	9.15	1.81	11.26	2.17	13.36

Tabla 4–56: Se describe la cantidad de memoria requerida para el almacenamiento del factor L y la proporcionalidad entre la cantidad de elementos no ceros de A y L, para la matriz 8 de la tabla 4-49.

La figura 4-57 muestra los tiempo de ejecución de las diferente fases.

Nivel	$m = 80$			$m = 100$			$m = 120$		
	F.S	F.N	Solver	F.S	F.N	Solver	F.S	F.N	Solver
1	20.04	9290.85	3.72	20.04	11338.40	4.85	19.45	11866.80	5.14
5	19.53	10183.30	4.30	19.06	11084.40	3.74	19.74	12607.70	2.28

Tabla 4–57: Se presenta los tiempos de ejecución para las diferentes fases de una factorización $IC(\ell, 0, m)$, haciendo uso de la matriz 8 de la tabla 4-49.

En la figura 4-23 se ha presentado una comparación con la versión puntual y la versión por bloques de la factorización $IC(\ell, \tau, m)$ donde el parámetro de relevancia es el parámetro de memoria m y el parámetro de nivel ℓ. Haciendo uso del precondicionador por bloques se presenta una disminución en el número de iteraciones para

alcanzar la convergencia hacia la solución con una tolerancia fija.

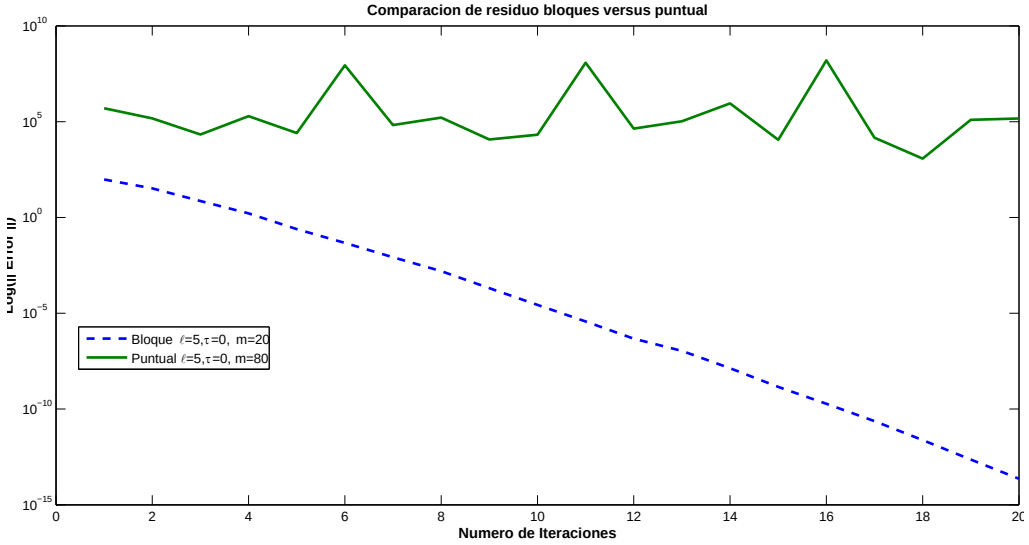


Figura 4–23: Comparación de los residuos generados por la versión puntual y la versión por bloque con la matriz 2 descrita en la tabla 4-1 y matriz 8 descrita en la tabla 4-49.

El criterio de comparación fue la cantidad $\frac{Nz(L)}{Nz(A)}$ la cual es el índice del número de elementos agregados no ceros al factor L, con este criterio se tomo las siguientes matrices para la comparación.

- La matriz 2 precondicionada con $\ell = 5$, $\tau = 0$, $m = 20$, en la cual el factor L tiene 1.45 veces más elementos no cero que la matriz 2.
- La matriz 8 precondicionada con $\ell = 5$, $\tau = 0$, $m = 80$, en la cual el factor L tiene 1.47 veces más elementos no cero que la matriz 8.

Esto muestra la ventaja de utilizar un precondicionador por bloques, dado que este disminuye el número de iteraciones requeridas para la convergencia de la solución cuando se utiliza el mismo índice de elementos no ceros en le factor L.

A continuación se mostrarán experimentos puntuales con el operador Helmholtz.

4.4. Factorización $IC(\ell, \tau, m)$ puntual para el operador Helmholtz

Para los resultados numéricos se ha tomado un vector inicial $x_0 = 0$, los criterios de parada están basados en el error del sistema precondicionado, si este satisface $\| r \| < 10^{-8}$ y en un máximo de 1000 iteraciones. Los datos proporcionados en la figura 4-58 son las características de las matrices que se han utilizado.

Nombre	dim. Global	dim. Bloques	Nz
Matriz 10	3840×3840	1×1	282240

Tabla 4–58: Información de las matrices.

4.4.1. Experimentos para $IC(\ell, 0, 0)$ puntual

A continuación se presentan experimentos numéricos donde el parámetro principal que determina los aportes al factor L será el parámetro de niveles ℓ. Los parámetros de memoria m y umbral τ tomarán un valor de cero. Con esto se desea mostrar la influencia que tiene el parámetro ℓ en el patrón de esparcimiento del factor L afectando a su vez el número de iteraciones requeridas al usar el método **PCG**.

En la tabla 4-59 muestra el número de iteraciones (Iter) requerida al variar el parámetro de niveles.

Nivel	Nz(L)	$\frac{Nz(L)}{Nz(A)}$	Iter
10	1362000	4.82	19
15	1420320	5.03	5
20	1420896	5.03	2

Tabla 4–59: Se describe la cantidad de elementos no ceros para el precondicionador y el número de iteraciones, para $\tau = 0$ y variado ℓ y m. Para la matriz 10 descrita en la tabla 4-58.

En la tabla 4-59 se puede observar como el aumento del parámetro ℓ genera una disminución en el número de iteraciones proporcionando a su vez un incremento en el consumo de memoria.

Al comparar la tabla 4-38 con la tabla 4-59 se tiene lo siguiente:

- $\ell = 5$ a nivel por bloques se consigue la convergencia del método **PCG** en 428 iteraciones y de manera puntual no se alcanzo una convergencia, dado que la factorización incompleta de Cholesky falló.
- Comparando $\ell = 15, 20$ en ambas tablas se podría decir que el precondicionador puntual es más eficiente que a nivel de bloques. Pero el tiempo que tomó construir el precondicionador a nivel puntual fue de 30 minutos, mientas que a nivel de bloques tomo en construir el precondicionador 0.4 segundos.

Una vez más es notorio la ventaja que tiene precondicional a nivel de bloques que a nivel puntual. Para $IC(\ell, \tau, 0)$ y $IC(\ell, 0, m)$ no se logro convergencia debido a que la factorización incompleta de Cholesky no existe a pesar de las técnicas de desplazamiento utilizada.

4.4.2. $IC(0)$ Matlab versus $IC(\ell, \tau, m)$ puntual

Con el propósito de mostrar la eficiencia con respecto al número de iteraciones necesarias para alcanzar la convergencia con una tolerancia establecida haciendo uso de la implementación de $IC(\ell, \tau, m)$ se compara dicha implementación con la factorización incompleta de Cholesky $IC(0)$ de Matlab.

	$IC(0)$	$IC(1, 10^{-1}, 0)$	$IC(1, 10^{-3}, 0)$	$IC(5, 0, 0)$	$IC(10, 0, 0)$
Iter	24	22	8	4	2
$\| r \|$	$1{,}82 \times 10^{-12}$	$1{,}94 \times 10^{-14}$	$5{,}31 \times 10^{-15}$	$4{,}16 \times 10^{-15}$	$1{,}82 \times 10^{-27}$

Tabla 4–60: Comparación entre el número de iteraciones requerida por $IC(l, \tau, m)$ versus $IC(0)$.

Para una mejor visualización de las comparaciones presentadas en la tabla 4-60, se presenta la figura 4-24 que compara el historial de la norma del residuo para los

anteriores precondicionadores.

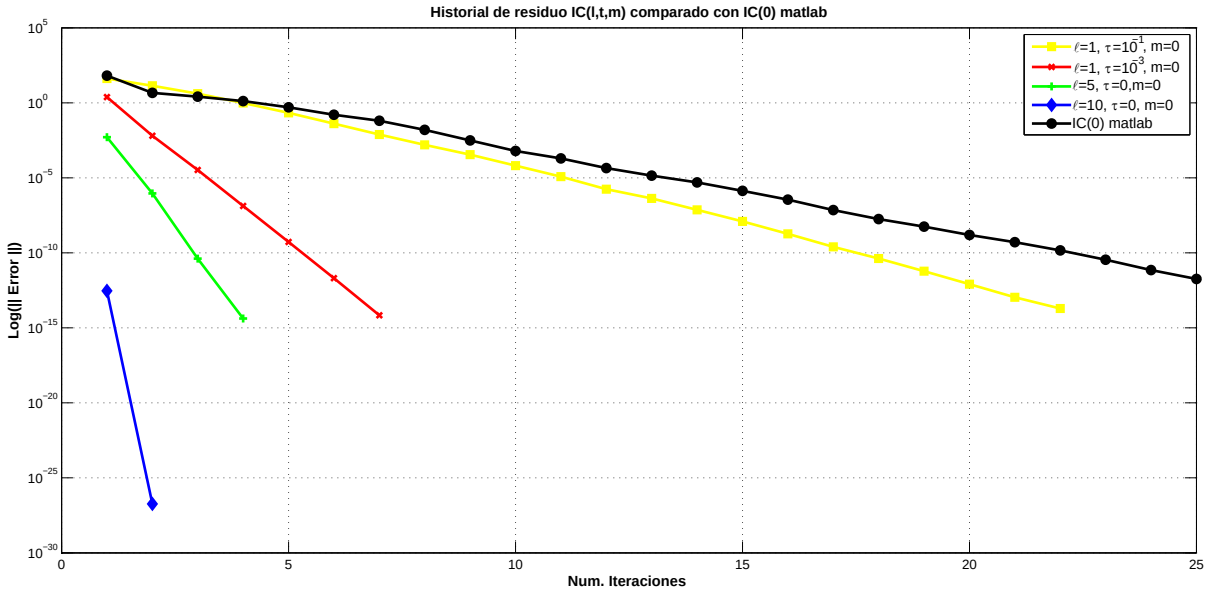


Figura 4–24: Comparación de los residuos generados por la versión $IC(l, \tau, m)$ versus $IC(0)$ de matlab para la matriz 7.

En la figura 4-24 se puede observar como se presenta un aumento en el número de iteraciones a medida que la cantidad de elementos no ceros disminuye en el factor L, permitiendo acercarse a la factorización $IC(0)$ de Matlab.

Capítulo 5
CONCLUSIÓN

Se ha presentado un nuevo precondicionador por bloques que es calculado por una factorización incompleta de Cholesky, en la cual se hace uso de tres parámetros que permitió el llenado de elementos no ceros en el factor L. Cuando se precondiciona el sistema lineal generado por la discretización del operador Laplaciano y del operador Helmholtz para baja frecuencia por medio del método Local Discontinuous Galerkin, se disminuye el número de iteraciones para la convergencia del método **PCG** en un 97 % y 99 % respectivamente, en comparación al número de iteraciones requeridas haciendo uso del sistema no precondicinado. Por otro lado se mostró como el condicionamiento espectral del sistema precondicionado se aproximó a 1, el cual se mejoró al incrementar el valor del parámetro de niveles, esto permitió disminuir el condicionamiento numérico de la matriz.

Al comparar el precondicionador por bloque y el puntual en base al índice de elementos no cero en el factor L, se mostró que a nivel de bloques se obtiene la convergencia del método **PCG** en un menor número de iteraciones, además que la construcción de un precondicionador a nivel puntual tarda más tiempo que por bloque. Por otro lado se ha comparado con la versión $IC(0)$ de matlab y se obtuvo un menor número de iteraciones al utilizar $IC(\ell, \tau, m)$.

Para generar un buen precondicionador para los sistemas lineales generados por la discretización del operador Laplaciano, se recomienda asignar un valor para el parámetro de niveles entre 5 y 10. Ahora para los sistemas generados por la discretización del operador Helmholtz de baja frecuancia, con un refinamiento de alrederdor de 500 celdas se recomienda tomar el parámetro $\ell \geq 15$. Cuando se toma un refinamiento de aproximadamente 5000 celdas se requiere un valor del parámetro ℓ mayor que 20. Por otro lado si el refinamiento de la maya es mayor que 30000, se requiere aumentar el parámetro de nivel ℓ, proporcionando un mayor consumo de memoria, lo cual imposibilita el almacenamiento del factor L. En los casos antreriores el valor del parámetro τ es recomendable tomarlo lo suficientemente pequeño, para eliminar aquellas entradas que son casi cero. Con respecto al parámetro de memoria es recomendable usarlo cuando se desea una cantidad fija de elementos no ceros en el factor L.

Con estas observaciones se puede recomendar el uso del precondicionador propuesto para matrices que son generadas por la discretización del operador Laplaciano por métodos Discontinuos aún con un refinamiento de alrededor de 30000 celdas. En comparación a la técnica de multiniveles la ventaja que se ofrece es una implementación más sencilla. Por otro lado para el operador Helmholtz de baja frecuencia, $IC(\ell, \tau, m)$ resulta ser un buen precondicionador para mallas no muy grandes. En particular este precondicionador puede resultar provechoso para aproximaciones de alto orden, en las cuales no se requier de mallas muy finas.

Cuando se trabaja con matrices de orden 10^3 es necesario el incremento del parámetro de nivel para generar una disminución en el número de iteraciones para la convergencia del método **PCG**. El incremento del parámetro de nivel genera una

mayor cantidad de operaciones aritméticas para la construcción de un precondicionador, generando un aumento en el tiempo de ejecución durante la fase numérica. Para abordar este problema se propone la paralelización del algoritmo bajo el paradigma de memoria compartida. Por otra parte se propone realizar un estudio teórico para la estabilidad de $IC(\ell, \tau, m)$ por bloque.

APÉNDICES

Appendix

Apéndice A
IMPLEMENTACIÓN DE $IC(\ell, \tau, M)$

En esta sección se presenta la implementación en el lenguaje $C++$ de una factorización incompleta $IC(\ell, \tau, m)$. El código esta creado orientado a objetos, permitiendo un código modular y un mejor mantenimiento del mismo. El código esta dividido en 6 clases, para las cuales se presentará sus miembros públicos.

Para el almacenamiento de las matrices, se hace uso de una estructura conocida como *CSR*. El código A.1 es la clase de $CSR - BlOCK$.

Código A.1: Clase CSR-BLOCK

```
#ifndef _CSR_BLOCK_H
 #define _CSR_BLOCK_H

  #include <fstream>
  #include <iostream>
  #include "MatrixD.h"
  #include "lapack2.h"
  #include "lapack3.h"
  #include "lapack.h"
    class CSR_BLOCK
    {
        friend class Operation;
        public:
                CSR_BLOCK();
                ~CSR_BLOCK();
                CSR_BLOCK(const char *namefile);
                void subMatrix (int & inf, const char *namefile);
                void get(MatrixD **&ll, unsigned int **jl,unsigned int *lnzi,
                unsigned int *&D_Row, unsigned int *&D_Colum);
                bool pertenece1(unsigned int k,unsigned int i
                                ,unsigned int &index);
                void Solver(MatrixD &C);
```

```
                void Bubblesort_increase(unsigned int k);
                void MV(double B[], double S[]);
                unsigned int Get_numRow();
                unsigned int Get_numColum();
                void Pattern();
                void Merge(MatrixD *LL,MatrixD *AA,unsigned int index,
                        unsigned int Lja[],const unsigned int limit);
                int getLnz();
                double getAllocatingMemory();
                int getMaxNzi();

        private:
        unsigned int numRow_;
        unsigned int numColum_;
        unsigned int Dim_Row_;
        unsigned int Dim_Colum_;
        unsigned int *nzi_;
        unsigned int **ja_;
        MatrixD **AA_;
        unsigned int Dimglobal_;

    };

#endif
```

La clase presentada en el código A.1 muestra las funciones pertinentes para el almacenamiento de la matriz, así como la función que resuelve un sistema de un bloque denso.

Recordando que los elementos de la matriz son bloque, se ha creado una clase para el tratamiento de estos bloques. El código A.2 muestra la clase MATRIXD con sus funciones.

Código A.2: Clase MATRIXD

```
#ifndef _MATRIXD_H
 #define _MATRIXD_H

  #include <fstream>
  #include <iostream>
  #include "lapack2.h"
  #include "lapack3.h"

using namespace std;
    class MatrixD
    {
        public:
```

```
        MatrixD();
        MatrixD(unsigned int Row,unsigned int Column);
        void SetUp(unsigned int Row,unsigned int Column);
        ~MatrixD();
        void readstream(int &inf, const char *namefile);
        void MV(double x[], double y[], char Trasnp);
        void MM(const MatrixD &B, MatrixD &C,char Transp_A,
                char Transp_B, double beta);
        void MMT(MatrixD &C, double B);
        int max(unsigned int a,unsigned int m);
        int min(unsigned int a,unsigned int m);
        MatrixD& operator=(const MatrixD &right);
        inline void deepCopy( const MatrixD & original );
        MatrixD& operator-(const MatrixD &Ob);
        void Inverse();
        MatrixD& Traspose();
        MatrixD& Copy(MatrixD &C);
        void Cholesky_fact(unsigned int i, unsigned int j, ofstream &escribe);
        void System_Solver(double B[],char Trans);
        void TiangularSystem_Solver(MatrixD &B);
        void TiangularSystem_Ax(double B[], char Trans);
        double Norm(char I);
        void Triangular_inverse();
        double getAllocatingMemory();

    private:
        double *a_ij;
        unsigned int row_;
        unsigned int column_;
};

#endif
```

El código A.2 muestras las funciones que permite realizar operaciones entre matrices y matriz vector, entre las operaciones se encuentra multiplicaciones entre matrices, calculo de inversa, solucionar un sistema triangular superior y operaciones como matriz vector, cabe notar que cada bloque es una matriz densa.

Para construir la estrategia de los niveles se ha construido una clase que permite asignar a cada bloque un nivel, esta clase se puede ver en el código A.3.

Código A.3: Clase OPERATION

```
#ifndef _Operation_H
 #define _Operation_H

  #include <fstream>
  #include <iostream>
```

```
  #include "CSR_BLOCK.h"


  struct Nodo{
  unsigned int i;
  unsigned int j;
  double norm;
  void operator =(Nodo element)
  {
    i=element.i;
    j=element.j;
    norm=element.norm;
  }
 };

    class Operation
    {
        friend class SPVectors;
        public:
                Operation();
                ~Operation();
                void Setup(CSR_BLOCK &A);
                void StrategyI(CSR_BLOCK &A,unsigned int l);
                void MaxMin(double &max, double &min, int size ,const Nodo N[]);
                void QuickSort(Nodo V[],int izq, unsigned int der );
                void Level(CSR_BLOCK &A);


        private:
        int **level_;
        unsigned int  *nli_;
        unsigned int colum_;

    };

#endif
```

Para calcular el patrón de elementos no ceros del factor L teniendo en cuenta los niveles de los elementos, se construye una clase que permita calcular el patrón de esparcimiento. El código A.4 muestra la clase de la que se menciona.

Código A.4: Clase SPVectors

```
#ifndef _SPVectors_H
 #define _SPVectors_H

  #include <fstream>
  #include <iostream>
  #include "CSR_BLOCK.h"
  #include "Operation.h"
  #include "MatrixD.h"
```

```
    class SPVectors
    {
        public:
                SPVectors();
                ~SPVectors();
                void SETUP(CSR_BLOCK &A);
                void Bubblesort_increase(unsigned int imax,
                                         unsigned int array[]);
                bool binary_search(unsigned int A[], unsigned int key,
                          int imin,  int imax, ofstream & escribe, const int k);
                void Simbolic_Sparsity(Operation &P);
                void Merge(unsigned int F[],unsigned int V[],
                       unsigned int &sf,unsigned int sv,
                       unsigned int size,unsigned int Aporte[]);
                void Pattern();
                int pertenece(unsigned int key,unsigned int cja[],
                             int cont);
                void QuickSort(unsigned int V[],int izq, unsigned int der );
                void Copy_Structure(unsigned int *&nzk, unsigned int **&ju);
                void Pint_Nzj();

        private:
        unsigned int **cja_;
        unsigned int *nzj_;
        unsigned int size;
        unsigned int **adj;
        unsigned int *Pnzk;

    };

#endif
```

Para la factorización numérica se ha creado una clase, que permite obtener el factor $\tilde{L}$ y sus entradas. El código A.5 muestra la clase para esta fase numérica.

Código A.5: Clase BIC0-H

```
#ifndef _BIC0_H
 #define _BIC0_H

  #include <fstream>
  #include <iostream>
  #include "CSR_BLOCK.h"
  #include "SPVectors.h"
  #include "lapack2.h"
  #include "lapack3.h"
  #include "lapack.h"
    class BIC0
    {
        public:
                BIC0();
                ~BIC0();
                BIC0(CSR_BLOCK &A,SPVectors &V);
                void BMV(unsigned int i, unsigned int j,
```

```
                double X[], double Y[]);
        void MM(unsigned int i, unsigned int j,
                unsigned int m, unsigned int n, MatrixD &C);
        void getInverse(unsigned int i, unsigned int j,MatrixD &C);
        void TransposeMatrix(unsigned int i, unsigned j, MatrixD &C);
        void Cholesky(unsigned int i, unsigned int j,MatrixD &C);
        void Solver(unsigned int i,unsigned int j, double B[]);
        void Triangular_Solver(unsigned int i,unsigned int j, MatrixD &B);
        void MMT(unsigned i, unsigned j, MatrixD &C);
        void System_Solver(double b[],double R[]);
        void fact_BIC0(const double tau, const int p);
        bool pertenece1(unsigned int k,unsigned int i,unsigned int &index);
        void Ux(double x[],const double b[],unsigned int BlockColum);
        void Lx(double x[],const double b[],unsigned int BlockColum);
        void Bubblesort_increase(unsigned int imax,
                                 unsigned int array[]);


        void QuickSort(double a[],int index[],int p, int r);
        int partition(double a[],int index[], int p, int r);
        void quicksort(double ptr_j[],int index[],int izq, int der );
        void quicksort_decrease(double ptr_j[],int index[],int izq, int der );

        inline bool binary_search(unsigned int A[], unsigned int key, int imin,
                          int imax, unsigned int & k);
        void MV(double x0[], double S[]);
        void MTV(double x0[], double S[]);
        double getAllocatingMemory();


    private:
    unsigned int nRow_;
    unsigned int nColum_;
    unsigned int *D_Row_;
    unsigned int *D_Colum_;
    unsigned int *Lnzi_;
    unsigned int **jL_;

    MatrixD **LL_;
    MatrixD *invLL_;

};
```

Una vez obtenida la matriz L se procede a resolver el sistema precondicionado con el método del Gradiente Conjugado Precondicionado. Se ha diseñado una clase, la cual se muestra en el código A.6 que se encarga de este proceso.

Código A.6: Clase BIC0-H

```
#ifndef _PGM_H
 #define _PGM_H

  #include <fstream>
```

```
#include <iostream>
#include "CSR_BLOCK.h"
#include "BIC0.h"
#include "IDEN.h"
#include "SPD.h"
#include "MatrixD.h"
#include "lapack2.h"
#include "lapack3.h"
#include "lapack.h"

using namespace std;
  // template<class Datatype>
   class PGM
   {
       public:

              PGM();
              PGM(CSR_BLOCK &Ob, const char *namefile);
              ~PGM();
              void applay_PGM(CSR_BLOCK &Ob,BIC0 &LL,unsigned int max_iter,
                   double x0[],double e, double y[]);

              void solver(CSR_BLOCK &Ob,BIC0 &LL,unsigned int &max_iter,
                          double x_j[],double &e);

              void solverI(CSR_BLOCK &Ob,IDEN &I,unsigned int &max_iter,
                          double x_j[],double e);
              void solverII(CSR_BLOCK &Ob,unsigned int &max_iter,
                          double x_j[],double &e);
              unsigned int NumIter();
              double DotProduct(int size, double a[],double b[]);
              double norm_2(unsigned int size,double R[]);
              double norm_inf(unsigned int size, double R[]);

       private:
       double *b_;
       //MatrixD *r_;
       unsigned int dim_Row;
       unsigned int dim_Col;
       unsigned int Iter;
   };

#endif
```

Bibliografía

[1] W. Reed and T. Hill, Triangular mesh methods for the neutron transport equation, Proceedings of the American Nuclear Society (1973).

[2] J. Douglas and T. Dupont, Interior penalty procedures for elliptic and parabolic Galerkin methods, Springer-Verlag 141 (1976) Lecture Notes in Phys. 58.

[3] G. Barker, Finite element methods for elliptic equations using nonconforming elements, Math. of Comp. 31 (1977) 45–59.

[4] D. Arnold, An interior penalty finite element method with Discontinuous elements, SIAM J. Numer. Anal. 19 (1982) 742–760.

[5] B. Riviére, M. Wheeler and V. Girault, A prior error estimates for finite element methods based on Discontinuous approximation space for elliptic problems, SIAM J. Numer. Anal. 39 (2001) 902–931.

[6] B. Cockburn and C. Shu, The local Discontinuous Galerkin method for time-dependent convection-diffusion system, SIAM J. Numer. Anal. 35 (1998) 2440–2463.

[7] P. Castillo, B. Cockburn, I. Perugia and D. Schotzau, An a priori error analysis of the local Discontinuous Galerkin method for elliptic problems, SIAM J. Numer. Anal 38 (2000) 1676–1706.

[8] D. Arnold, D. Brezzi, B. Cockburn and D. Marini, Unified analysis of Discontinuous Galerkin methos for elliptic problem., SIAM 39 (2002) 1749–1779.

[9] P. Castillo, Performance of Discontinuous Galerkin method for elliptic PDEs, SIAM J. Sci. Comptu. 24 (2002) 524–547.

[10] Y. Saad, Iterative Methods for Sparse Linear System, second ed., SIAM, 2003.

[11] J. Meijerink and H. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Mathematics of Computation 31 (1977) 148–162.

[12] E. D'Azevedo, F. Forsyth and W. Tang, Towards a cost effective ILU preconditioner with high level fill, BIT 31 (1992) 442–463.

[13] N. Li and Y. Saad, Crout version of the ILU factorization with pivoting for sparse symmetric matrices, ETNA, Electronic Transactions on Numerical Analysis 20 (2005) 75–85.

[14] N. Munksgaard, Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients, ACM Trans. Math. Software 6 (1980) 206–219.

[15] D. Kershaw, The incomplete Cholesky conjugate gradient method for the iterative solution of system of linear equation, J. Comp. Phys. 26 (1978) 43–65.

[16] E. D'Azevedo, Forsyth and W. Tang, Drop tolerance preconditioning for incompressible viscous flow, J. Comp. Phys. 44 (1992) 301–312.

[17] C. Lin and J. Moré, Incompleted Cholesky factorizations with limited memory, SIAM J. Sci. Compt. 21 (1999) 24–45.

[18] J. Scott and M. Tůma, The importance of structure in algebraic preconditioners, BIT. Numer. Math 51 (2011) 385–404.

[19] D. Hysom and A. Pothen, Level-based incomplete LU factorization: Graph model and algorithms, Technical Report, Lawrence Livermore National Laboratory UCRL-JC-150789, 2002.

[20] J. Zhang and T. Xiao, A multilevel block incomplete cholesky preconditioner for solving normal equations in linear least squares problems, Jornal of Appplied Matematics and Computing 11 (2003) 59–80.

[21] P. Castillo and E. Velazquez, A numerical study of a semi-algebraic multilevel preconditioner for the local Discontinous Galerkin method, Int. J. Numer. Math 74 (2008) 255–268.

[22] P. Castillo and F. Sequeira, Computational aspects of the local Discontinuous galerkin method on unstructured grids in three dimensions, Journal of Mathematical and Computer Modelling 57 (2013) 2279–2288.

[23] I. Gustafsoon, An class of firts order factorization methods, BIT 18 (1978) 142–156.

[24] M. Hestenes and E. Stiefel, Method of Conjugate Gradients for solving Linear Systems, Research of the National Bureau of Standards, 49 (1952) .

[25] S. Petrova and A. Solov'ev, The Origin of the Method of Steepest Descent, Historia Mathematica, 24 (1997) 361-375.

[26] O. Axelsson, L. Yun and Kolotilina, Preconditioned Conjugate Gradient Methods, Springer-Verlag, 24 (1989) 2-3.

[27] V. Eijkhout and R. Pozo, Data Structures and Algorithms for Distributed Sparse Matrix Operation Technical Report, University of Tennessee KNoxville TN 37996-1301 , 1994.

[28] V. Eijkhout, Distributed Sparse Data Structures for Linear Algebra Operations, Technical Report, University of Tennessee KNoxville CS 92-169,TN 37996-1301, 1994.

[29] M. Jones and P. Plassmann, An improved incomplete Cholesky Factorization, ACM Transactions on Mathematical Software, 21 (1995) 5-17.

[30] K. Fan, Note on M-matrices, Quart. J. Math. Oxford, 11 (1960) 43-49.

[31] T. Manteuffel, An Incomplete Factorization Technique for Positive Definite Linear System, Mathematics of Computation, 34 (1980) 473-497.

[32] T. Manteuffel, Shifted incomplete Cholesky factorization, in Sparse Matrix Proceeding SIAM, Philadelphia, 34 (1979) 307-327.

[33] A. Jenning and G. Malik, Partial elimination, J. Inst. Maths. Appl., 20 (1977) 307-316.

[34] I. Hladi, M. Reed and G. Swoboda Robust preconditioner for linear elasticity FEM analysis, Int. J Num. Meths. Eng., 40 (1997) 2109-2117.

[35] F. Dopico, Alan Turing and the Origins of modern Gaussian elimination Technical Report, University Carlos III de Marid CSIC-UAM-UC3M-UCM, 2013.

[36] A. Turing, Rounding-off errors in matrix processes, Quart. J. Mech. Appl. Math 1 (1948) 287-308.

[37] Magnus, N. Arnold and E. Stiefel, Method of conjugate gradients for solving linear systems., J. Research Nat. Bur. Standards 49 (1953) 409-436.

[38] David G. Luenberger, Linear and nonlinear programming., second ed., Kluwer Academic Publishers, Boston, MA, 2003.

[39] A. Alvarado, Método LDG para la ecuación vectorial de Helmholtz, Universidad de Puerto Rico Recinto Mayaguez, 2014.

FACTORIZACIÓN INCOMPLETA $IC(\ell, \tau, M)$ POR BLOQUES PARA MATRICES GENERADAS POR MÉTODOS "LOCAL DISCONTINUOUS GALERKIN"

Carlos Andres Theran Suarez
carlos.theran@upr.edu
Departamento de Ciencias Matemáticas
Consejero: Paul Castillo
Grado: Maestría en Ciencias
Fecha de Graduación: Noviembre 2014

En este trabajo se presenta una factorización incompleta de Cholesky por bloques LL^T. Esta factorización es discutida como una técnica de precondicionamiento, la cual mejora el condicionamiento espectral de la matriz, lo que implica una disminución en el número de iteraciones requeridas para la convergencia de métodos iterativos como lo es el método del Gradiente Conjugado Precondicionado (**PCG**). Para la construcción del precondicionador se considera una técnica de niveles propuesta en [18] la cual calcula la estructura de elementos no cero del precondicionador, este proceso es conocido como fase simbólica. Uno de los problemas que genera la construcción de un precondicionador para grandes sistemas lineales es la gran cantidad de memoria que se requiere para el almacenamiento de L. Para controlar el consumo de memoria, se usa dos paramétros: el parámetro umbral τ y el parámetro de memoria m, los cuales permiten retener elementos no ceros del factor L. Una versión por bloque de estas técnicas es propuesta para matrices simétricas definidas positivas generadas por el método Local Discontinuous Galerkin.