# Técnicas Algebraicas de Precondicionamiento para la resolución de Sistemas Lineales

## Tesis Doctoral

**Mayo, 2002**


**Presentada por:**
Germán Larrazábal Serrano

**Dirigida por:**
José M. Cela Espín

UPC

Universidad Politécnica de Cataluña
Departamento de Arquitectura de Computadores

Tesis doctoral presentada por Germán A. Larrazábal Serrano
para obtener el grado de Doctor en Ciencias especialidad Informática
por la Universidad Politécnica de Cataluña,
Barcelona, España

# Índice General

# Índice de Figuras

# Índice de Tablas

# Prefacio

Esta tesis se centra en el estudio de técnicas de precondicionamiento para la resolución de sistemas lineales de ecuaciones, provenientes de la resolución de ecuaciones diferenciales en derivadas parciales. La característica común de los sistemas lineales objeto de interés es su enorme tamaño, y el hecho de que la matriz de coeficientes asociada a estos sistemas es dispersa. Se utilizan los modelos de programación de paso de mensajes y memoria compartida, y se orienta este trabajo a problemas de un tamaño medio, hasta $10^5$ ecuaciones.

La resolución de sistemas lineales de ecuaciones es, comúnmente, el núcleo computacional más costoso, en cuanto a tiempo de ejecución, de muchas simulaciones numéricas industriales, aunque otros problemas tales como cálculo de autovalores también suelen ocurrir. Típicamente, estos problemas consumen una significante porción del tiempo computacional requerido por una simulación completa. Una reciente revisión sobre el actual uso de supercomputadores de alto rendimiento indica que más del 70% del tiempo computacional es usado para resolver grandes sistemas lineales de ecuaciones. Un impacto industrial muy importante debe ocurrir si el rendimiento de los métodos usados para resolver estos sistemas pudiera ser mejorado.

El problema de resolver sistemas lineales ha sido ampliamente analizado para arquitecturas de memoria compartida, donde el paralelismo que se suele explotar es de grano fino. Sin embargo, el interés en explotar el paralelismo grano grueso, junto con la aparición de librerías que garantizan la portabilidad de los programas basados en paso de mensaje y junto al avance de la tecnología, ha hecho posible que en estos momentos se este comenzado a desarrollar aplicaciones industriales sobre multicomputadores de memoria compartida-distribuida. Las técnicas de descomposición en dominios son una forma de distribución de datos que permiten la explotación del paralelismo de paso de mensaje, y todo el paralelismo interno de cada dominio permite la explotación del paralelismo de memoria compartida.

En esta tesis se intenta cubrir todos los aspectos teóricos y algorítmicos de los métodos más usados para resolver sistemas lineales dispersos de ecuaciones. Por

ello en primer lugar, se estudia los métodos iterativos más importantes, las técnicas más comunes de precondicionamiento y los métodos multinivel. En segundo lugar, se analiza la formulación numérica del problema y su paralelismo, y se propone el uso de un precondicionador paralelo que disminuye significativamente el tiempo de ejecución. Finalmente, se muestran los resultados obtenidos para un conjunto de problemas provenientes de la resolución de ecuaciones diferenciales en derivadas parciales.

Para cubrir todos los aspectos mencionados, se ha organizado esta tesis en 4 capítulos. En el capítulo 1, se introduce la problemática de la resolución de sistemas lineales dispersos. Se describen los métodos iterativos más importantes y se discute acerca del criterio de parada de estos métodos. Además, se describen los precondicionadores de carácter general que estos métodos usan.

En el capítulo 2, se estudia otra técnica para resolver sistemas lineales dispersos de ecuaciones: los métodos multinivel. Se describe ampliamente los aspectos teóricos y algorítmicos de estos métodos. Seguidamente, se describen brevemente los métodos multinivel geométricos. Finalmente, se realiza un estudio detallado sobre los diferentes métodos multinivel algebraicos, de los operadores de complejidad, y del uso de éstos como precondicionadores de carácter general.

En el capítulo 3, se analiza el enfoque numérico del problema que permite una descomposición en dominios, sistema complemento o sistema de Schur. Además, se describen algunas técnicas algebraicas, las más conocidas, para precondicionar este sistema.

En el capítulo 4, se proponen varios precondicionadores algebraicos paralelos para el sistema complemento de Schur. La construcción de estos precondicionadores carece por completo de comunicaciones y logra una reducción muy notable del número de iteraciones del método iterativo usado. Además, en la construcción de estos precondicionadores se puede explotar dos niveles de paralelismo: paso de mensaje y memoria compartida. El desarrollo de estos precondicionadores se hace en el contexto de la resolución de ecuaciones diferenciales en derivadas parciales.

Por último, se resume el conjunto de aportaciones y conclusiones extraidas de esta tesis.

# Capítulo 1

# Métodos Iterativos

En este capítulo se discute sobre los diferentes métodos iterativos para la resolución de sistemas lineales. Se presenta una revisión unificada de los métodos de proyección y se analizan los métodos más importantes. Se realiza un breve estudio sobre los criterios de parada para los métodos iterativos. Finalmente, se describen los precondicionadores clásicos más importantes.

## 1.1 Introducción

La resolución de sistemas lineales es posiblemente el núcleo computacional más importante de la mayoría de las aplicaciones de ingeniería. En el pasado las técnicas numéricas de resolución de sistemas lineales más usadas eran los métodos directos. Sin embargo, estos métodos presentan serias limitaciones de aplicación hacia matrices dispersas. Por ejemplo, una malla de 50x50x50 nodos con 5 grados de libertad por nodo, caso típico de la ecuación de Euler en dinámica de fluidos, genera un sistema lineal de 625.000 filas, con un promedio de unos 150 elementos no nulos por fila, lo que genera un llenado de la matriz de únicamente el 0,024%, y un volumen de datos de unos 715 Mbytes. Aunque se aplique técnicas de reordenación que minimicen el llenado introducido por un método directo, este llenado no bajará del 0,5%, lo que genera un volumen de datos de aproximadamente 14 Gbytes. Si se tiene en cuenta que este ejemplo está muy por debajo de los requerimientos de un caso real, se ve claramente la imposibilidad de usar métodos directos al menos en gran parte de las aplicaciones. Además, los métodos directos presentan un grado de paralelismo bajo lo que limita su escalabilidad en sistemas paralelos. Por último, los métodos directos pueden presentar problemas graves de exactitud numérica tanto por exceso como por defecto. Un sistema de coma flotante en doble precisión permite tener hasta 15 cifras decimales exactas. La propagación del error al usar un método directo multiplica el error relativo en los datos por el número de condición

de la matriz, que suele estar en el rango $10^3$-$10^9$ en la mayoría de las aplicaciones industriales. Esto da un margen de 12 a 4 cifras exactas en la solución. Este margen es adecuado para muchas de las aplicaciones, pero en algunos casos se puede tener problemas de exactitud por defecto. Esta falta de exactitud es irresoluble usando un método directo. En otros casos, donde únicamente se requieren unas pocas cifras exactas (3 ó 4) se puede estar haciendo trabajo en vano.

La solución a todos estos inconvenientes la dan los métodos iterativos. Estos métodos no modifican la estructura de la matriz de coeficientes, ya que su principal operación es el producto matriz por vector. Por ello, ni requieren tanta memoria como los métodos directos, ni presentan tantas dificultades de paralelización, ya que la operación matriz por vector es altamente paralela. Además, aunque la convergencia de un método iterativo puede ser lenta, siempre existe la posibilidad de ajustarse a la exactitud deseada.

Los métodos iterativos se dividen en los llamados métodos clásicos y los llamados métodos de proyección. Existe una extensa bibliografía que analiza el comportamiento de los métodos iterativos clásicos: Jacobi, Gauss-Seidel, SOR, SSOR, Chebyshev [Axe85] [Var62].

Esta tesis se centra exclusivamente en los llamados métodos de proyección, que son los más eficaces. En concreto, nos interesan los métodos GMRES($m$), Gradiente Biconjugado y sus variantes BiCGstab, CGS, y el Gradiente Conjugado (CG). Todos ellos han sido ampliamente estudiados en cuanto a convergencia y propiedades numéricas [CIKM94] [NRT92] [SS86] [Fle76] [Son89] [SF93] [dV92] [Wei94] [PSS92].

Todos los métodos iterativos presentan en general una convergencia demasiado lenta, así que es preciso introducir mejoras en el esquema numérico que aceleren la convergencia. La principal mejora se logra mediante la introducción de un precondicionador. Existen tres opciones básicas: usar los llamados precondicionadores clásicos, usar los precondicionadores multinivel, o usar los precondicionadores basados en descomposición en dominios.

Los precondicionadores clásicos se basan en manipulaciones puramente algebraicas de la matriz para obtener algún tipo de aproximación de la inversa. Los precondicionadores clásicos más conocidos son las factorizaciones incompletas y los precondicionadores polinomiales [AMS89] [AMO92] [BMT94] [BCMM95] [EOV90] [MdV77] [Saa85] [dV82] [Yea89] [Zla82]. En los últimos años se están investigando un nuevo tipo de precondicionadores clásicos llamados Aproximación Dispersa de la

Inversa (SParse Approximate Inverse, SPAI) [GS93b] [GS94] [GS95] [KY93].

En la década de los 60, los matemáticos rusos Fedorenko y Bakhavalov [Fed62] [Bak66] desarrollaron una nueva técnica iterativa de resolución de sistemas lineales provenientes de EDPs. Dicha técnica se basaba en métodos iterativos clásicos y en explotar información adicional proveniente de discretizaciones más gruesas del mismo problema. A principios de la década de los 70 esta técnica comenzó a popularizarse en occidente gracias a los trabajos de A. Brandt [Bra77] que la denominó método Multinivel o Multimalla.

Los métodos multinivel presentan una complejidad en operaciones lineal con el tamaño del sistema, frente a la complejidad cuadrática de los métodos clásicos. Como contrapartida consumen mucha más memoria y son menos versátiles en su aplicación. Para una recopilación de todos los métodos multinivel ver [Bri87] [Hac85] [McC89] [AV89a] [Rud93] [Tum89] [Ton90] [Yse82]. Hoy en día, existen métodos multinivel que destacan por la sencillez de su aplicación a cualquier tipo de discretización, estos son los Algebraic Multigrid Method (AMG) [AV89b] [AV90] [Stu99] [VMB96].

Una última opción que se tiene es usar técnicas de descomposición en dominios que reducen el problema original al llamado sistema de complemento de Schur, y aplicar a dicho sistema algún método iterativo con algún precondicionador específico, por ejemplo el Modified Schur Complement (MSC(k)) [Chan85]. Para una descripción exhaustiva de los precondicionadores específicos para descomposición en dominios ver las actas de la conferencia anual sobre métodos de descomposición en dominios [SIA88] [SIA89] [SIA90] [SIA91] [SIA92].

La pregunta que se plantea es qué combinación entre método y precondicionador es más conveniente usar. La respuesta no es única, depende del tamaño del sistema, la arquitectura usada y la naturaleza del problema físico que se pretende resolver. En líneas generales se puede decir que la combinación método de proyección más factorización incompleta es la más robusta de todas, pero es la que presenta un menor grado de paralelismo. Los métodos multinivel aunque presentan una complejidad lineal sólo son realmente rentables para sistemas muy grandes ($> 10^5$ ecuaciones), y además su generalización a otros campos como el de Cadenas de Markov aún no está resuelta. Por su parte, los métodos de descomposición en dominios que trabajan con el sistema de complemento de Schur son de aplicabilidad limitada debido a la falta de precondicionadores generales para este esquema numérico. En [Meie88] y [Cela93] puede encontrarse una comparación entre diferentes opciones sobre mul-

tiprocesadores de memoria compartida.

Los métodos multinivel presentan una complejidad lineal [AV91] [CN92], pero el sobrecoste que implican sólo se amortiza en sistemas muy grandes. Por otro lado, la respuesta en tiempo de los métodos que usan factorizaciones incompletas se degrada rápidamente al ser de complejidad cuadrática. Los métodos que usan descomposición en dominios también suelen presentar complejidad cuadrática, pero atenuada por el hecho de usar precondicionadores más eficientes, aunque menos generales, que las factorizaciones incompletas.

El resto del capítulo está organizado de la siguiente forma. En primer lugar se describen los métodos de proyección más importantes. Seguidamente, se presenta un breve estudio sobre los criterios de parada de los métodos iterativos. Luego, se discute sobre la problemática de los precondicionadores, y se describen los precondicionadores polinomiales, las diferentes familias de factorizaciones incompletas y los precondicionadores SPAI.

## 1.2 Metodología de los Métodos de Proyección

Dado un sistema lineal $n \times n$

$$Ax = b \tag{1.1}$$

Sea el subespacio vectortial $K$ de dimensión $m < n$, generado por la base $V \equiv [v_1, \ldots, v_m]$. Se toma como aproximación de $x$ el vector $\tilde{x} = Vy$, donde $y$ es un vector de dimensión $m$. Existen diferentes criterios para seleccionar $y$, el criterio más habitual es forzar que el vector residuo, $r = b - A\tilde{x}$, sea ortogonal a otro subespacio $\Lambda$ de dimensión $m$ generado por la base $W \equiv [w_1, \ldots, w_m]$, es decir, se impone que:

$$W^t \cdot (b - AVy) = 0 \tag{1.2}$$

Se tiene entonces que $y = (W^tAV)^{-1} \cdot W^tb$, suponiendo que la matriz $W^tAV$ sea no singular. Es decir, se ha reducido el problema original a resolver un sistema lineal de $m \times m$.

Los diferentes métodos de proyección se limitan a elegir diferentes subespacios $K$ y $\Lambda$ para hacer la aproximación. Además, habitualmente los métodos se formulan sobre el sistema del error en vez de sobre el sistema original. Es decir, dada una primera aproximación a la solución $x_0$, buscamos un vector de corrección $e$, que

aproxime la solución exacta del sistema $A\tilde{e} = r_0$, donde $\tilde{e} = \tilde{x} - x_0$ es el vector error. Entonces la aproximación a la solución se dará como $x = x_0 + e$.

Así pues, el algoritmo genérico de todo método de proyección es el siguiente:

MIENTRAS NO CONVERGENCIA
1. ELEGIR $V = [v_1 \dots v_m]$ Y $W = [w_1 \dots w_m]$
2. CALCULAR $r = b - Ax$
3. CALCULAR $y = (W^t AV)^{-1} \cdot W^t r$
4. CALCULAR $x = x + Vy$

FINMIENTRAS

La metodología general de los métodos de proyección también se aplica a los problemas de autovalores, de hecho ambos problemas, resolución de sistemas lineales y cálculo de autovalores, están estrechamente relacionados. Para problemas de autovalores el proceso de proyección es idéntico, pero ahora $b = \lambda x$, con lo cual queda sustituyendo en (1.2):

$$W^t \cdot (AVy - \lambda Vy) = 0 \Rightarrow W^t \cdot AVy = \lambda W^t Vy \tag{1.3}$$

Este es un problema generalizado de autovalores de dimensión $mxm$. Usualmente en problemas de autovalores se eligen $K$ y $\Lambda$ de forma que $W^t V = I$. Así el problema queda reducido a un problema de autovalores de dimensión $m \times m$.

Aunque existen muchas opciones para elegir los subespacios $K$ y $\Lambda$ la elección más habitual es que $K$ y $\Lambda$ sean subespacios de Krylov.

**Definición 1.2.1** *Se define el subespacio de Krylov de dimensión m asociado a la matriz A y al vector v como el subespacio generado por la base* $\{v, Av, A^2v, \dots, A^{m-1}v\}$, *y se denota por* $K_m(A, v)$.

Uno de los principales motivos por el cual se usan subespacios de Krylov es la sencillez con la que se puede invertir la matriz $W^t AV$, y la sencillez con la que se puede generar una base ortonormal de $K_m(A, v)$. La generación de bases ortonormales juega un papel fundamental en los métodos de proyección. Al proceso de generación de una base ortonormal se le denomina método de Arnoldi.

### 1.2.1 Método de Arnoldi

El sistema clásico de generación de bases ortonormales es el método de Gram-Schmidt, que partiendo de un conjunto de vectores linealmente independientes genera un conjunto de vectores ortonormales. Cuando el proceso de ortonormalización y

la generación de los vectores de partida se entremezcla, se denomina a este proceso método de Arnoldi [Arn51] [SS86].

El algoritmo de Arnoldi genera una base ortonormal $W_m = \{w_1, \ldots, w_m\}$ del subespacio $K_m(A, v)$ con $\|v_1\| = 1$. Dicho algoritmo puede verse en la figura 1.1. Por razones de estabilidad numérica en una realización práctica del algoritmo se debe usar la forma modificada del método de Gram-Schmidt.

$w_1 = v_1$
Do $j = 1, m-1$
  Do $i = 1, j$
    $h_{ij} =< Aw_j, w_i >$
  EndDo
  $\hat{w}_{j+1} = Aw_j - \sum_{i=1}^{j} h_{ij} w_i$
  $h_{j+i,j} = \|\hat{w}_{j+1}\|$
  $w_{j+1} = \hat{w}_{j+1}/h_{j+1,j}$
EndDo

Figura 1.1: Método de Arnoldi.

Se debe señalar que después de $m$ iteraciones del método de Arnoldi se tiene un conjunto de $(m+1)$ vectores columna ortonormales que forman la matriz $W_{m+1}$, y una matriz de dimensión $(m+1) \times m$, que se denomina $\hat{H}_m$, y cuyos elementos no nulos son los coeficientes $h_{ij}$ generados por el algoritmo. También se tiene una matriz de dimensión $m \times m$, que se denomina $H_m$, que no es más que la matriz $\hat{H}_m$ a la que se le ha eliminado la fila $(m+1)$. La matriz $H_m$ es simplemente la proyección sobre el subespacio $K_m(A, v_1)$ de la matriz $A$. Es decir,

$$H_m = W_m^t A W_m \tag{1.4}$$

donde $W_m = [w_1, \ldots, w_m]$. Por su parte la matriz $\hat{H}_m$ cumple la relación:

$$AW_m = W_{m+1} \hat{H}_m \tag{1.5}$$

La comprobación de lo anterior es inmediato por simple sustitución.

## 1.3 Métodos de Proyección

Seguidamente se describen los métodos de proyección más importantes. Al final de cada descripción del método se cita la referencia original donde está descrito, o bien

una referencia actual que lo describe ampliamente. Para un resumen general de todos los métodos de proyección ver [AMS90] [Saa89] [FM84].

### 1.3.1 Ortogonalización Completa (FOM)

Sea $x_0$ una aproximación inicial a la solución, sea $r_0 = b - Ax_0$ el residuo inicial, y sea $K_m(A, v_1)$ el subespacio de Krylov generado por $\{v_1, Av_1, A^2v_1, \ldots, A^{m-1}v_1\}$ con $v_1 = r_0/\|r_0\|$.

El método FOM propone calcular la aproximación a la solución como $x_m = x_0 + z_m$ con $z_m \in K_m(A, v_1)$, imponiendo la condición de Galerkin:

$$r_m \perp K_m(A, v_1)$$

Este es un método de proyección donde $K = \Lambda = K_m(A, v_1)$.

En primer lugar se genera una base ortonormal de $K_m(A, v_1)$ mediante el método de Arnoldi. Se denomina a esta base como $W_m = [w_1, \ldots, w_m]$. Entonces, se puede expresar la aproximación a la solución como $x_m = x_0 + W_m y$, donde $y$ es un vector de $m$ componentes. Transformando esta expresión se obtiene que:

$$r_m = r_0 - AW_m y = \|r_0\|v_1 - AW_m y$$

Imponiendo ahora la condición de Galerkin y dado que la base $W_m$ es ortonormal se tiene que:

$$0 = W_m^t r_m = \|r_0\|W_m^t v_1 - W_m^t AW_m y = \|r_0\|e_1 - H_m y \tag{1.6}$$

donde $e_1$ es el primer vector de la base canónica de dimensión $m$. Finalmente, se debe calcular $y = H_m^{-1}\|r_0\|e_1$, lo que será trivial dado que $H_m$ es Hessemberg superior. El algoritmo del método de ortogonalización completa se puede ver en la figura 1.2.

Notar que para calcular el punto de parada no es necesario calcular el vector $x$ explícitamente en cada iteración, ya que la norma del residuo en la iteración $k$-ésima se puede calcular como:

$$\|r_k\| = \|b - Ax_k\| = h_{j+1,j}|e_k^t y_k| \tag{1.7}$$

En [Saa81] se describe ampliamente este método. Para una revisión más detallada consultar dicha referencia.

Elegir $x_0$; Calcular $r_0 = b - Ax_0$ y $w_1 = r_0/\|r_0\|$
While not convergence
    Do $j = 1, m-1$
        Do $i = 1, j$
            $h_{ij} =< Aw_j, w_i >$
        EndDo
        $\hat{w}_{j+1} = Aw_j - \sum_{i=1}^{j} h_{ij} w_i$
        $w_{j+1} = \hat{w}_{j+1}/h_{j+1,j}$
    EndDo
    Calcular $y_m = H_m^{-1} \cdot \|r_0\| e_1$
EndWhile
Calcular $x = x_0 + Wy$

Figura 1.2: Método de Ortogonalización Completa.

### 1.3.2 Generalizado del Residuo Mínimo (GMRES)

Sea $x_0$, una aproximación inicial a la solución, sea $r_0 = b - Ax_0$ el residuo inicial, y sea $K_m(A, v_1)$ el subespacio de Krylov generado por $\{v_1, Av_1, A^2v_1, \ldots, A^{m-1}v_1\}$ con $v_1 = r_0/\|r_0\|$.

El método GMRES propone calcular la aproximación a la solución como $x_m = x_0 + z_m$ con $z_m \in K_m(A, v_1)$, imponiendo la condición de que la norma del residuo $r_m$ sea mínima para todos los vectores $z_m$. Este es un método de proyección donde de nuevo $K = \Lambda = K_m(A, v_1)$.

En primer lugar se genera una base ortonormal de mediante el método de Arnoldi, de nuevo se denominará $W_m = [w_1, \ldots, w_m]$. El problema que ahora se debe resolver es:

$$min\|b - A(x_0 + z_m)\| = min\|r_0 - Az_m\| = min\|\beta v_1 - AW_m y\| \tag{1.8}$$

donde $\beta = \|r_0\|$. Usando la ecuación (1.5) se tiene que:

$$min\|\beta v_1 - AW_m y\| = min\|W_{m+1}(\beta e_1 - \hat{H}_m y)\| \tag{1.9}$$

donde $e_1$ es el primer vector de la base canónica. Dado que $W_{m+1}$ es ortonormal se tiene que el vector $y$ es la solución de:

$$min\|\beta e_1 - \hat{H}_m y)\| \tag{1.10}$$

La forma más sencilla de calcular $y$ es mediante la factorización $QR$ de la matriz $\hat{H}_m$ usando rotaciones de Givens, esto es $Q_m\hat{H}_m = R_m$. La matriz $\hat{H}_m$ es Hessemberg superior, la matriz $Q_m$ de dimensión $(m+1)\times(m+1)$ es el producto acumulado de todas las matrices de rotación, y la matriz $R_m$ de dimensión $(m+1)\times m$ es triangular superior con su ultima fila igual a cero. De esta forma el algoritmo de factorización $QR$ se puede aplicar sucesivamente en cada iteración del método, manteniendo así la factorización actualizada. El problema finalmente queda reducido a resolver el siguiente sistema triangular superior:

$$R_m y = g_m = Q_m \beta e_1 \tag{1.11}$$

Ya que la matriz $Q_m$ sólo se usa para calcular el lado derecho de este sistema no es necesario almacenarla sino que basta con aplicar las sucesivas rotaciones que forman $Q_m$ al vector $\beta e_1$.

De nuevo, igual que en el método FOM, no es preciso construir en cada iteración la solución ya que el residuo en el paso $m$-ésimo se puede evaluar como la componente $(m+1)$ del vector $\beta e_1$.

Elegir $x_0$; Calcular $r_0 = b - Ax_0$ y $w_1 = r_0/\|r_0\|$
While (not convergence)
    Do $j = 1, m-1$
        Do $i = 1, j$
            $h_{ij} =< Aw_j, w_i >$
        EndDo
        $\hat{w}_{j+1} = Aw_j - \sum_{i=1}^{j} h_{ij} w_i$
        $h_{j+1,j} = \|\hat{w}_{j+1}\|$
        $w_{j+1} = \hat{w}_{j+1}/h_{j+1,j}$
        Calcular la rotación de Givens
        Actualizar $g$ y la matriz $R$
        IF (convergencia) GOTO 10
    EndDo
10: Calcular $y_m = R^{-1} \cdot g$
    Calcular $x = x_0 + Wy$
EndWhile

Figura 1.3: Algoritmo del GMRES(m).

El método tiene el problema de que se deben ir almacenando los vectores de

la base ortonormal para al final poder formar el vector $z_m$. Lógicamente esto implica un fuerte gasto de memoria. Para salvar esta dificultad, la táctica habitual es marcar un número máximo de iteraciones, al cabo de las cuales el método se reinicializa tomando como aproximación inicial la última aproximación propuesta. El algoritmo del GMRES($m$) con parámetro de reinicio $m$ se muestra en la figura 1.3.

En [SS86] se describe ampliamente este método. Para una revisión más detallada consultar dicha referencia.

### 1.3.3 Gradiente Conjugado (CG, Lanczos Simétrico)

Cuando la matriz $A$ es simétrica positiva definida la matriz $H_m$ es tridiagonal en lugar de Hessenberg superior. Esto permite una serie de simplificaciones importantes de cálculo en el método de Arnoldi, al algoritmo resultante se le llama método simétrico de Lanczos. El método simétrico de Lanczos calcula una matriz $n \times n$ simétrica tridiagonal tal que:

$$T = W^t A W \tag{1.12}$$

donde $W = [w_1, \ldots, w_m]$ es una base ortonormal y

$$T = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & \cdots & \cdots & \cdots & \\ & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & & \beta_n & \alpha_n \end{bmatrix}$$

Reescribiendo la ecuación (1.12) como $AW = WT$ se tiene que:

$$A w_j = \beta_j w_{j-1} + \alpha_j w_j + \beta_{j+1} w_{j+1} \tag{1.13}$$

con $j = 1, 2, \ldots, n$, entendiendo que $\beta_1 = \beta_{n+1} = 0$. Multiplicando la ecuación (1.13) por $w_j^t$ se tiene que:

$$\alpha_j = w_j^t A w_j \tag{1.14}$$

También de (1.13) se tiene la relación:

$$\beta_{j+1} w_{j+1} = (A - \alpha_j I) w_j - \beta_j w_{j-1} \equiv \hat{w}_{j+1} \tag{1.15}$$

Si $\hat{w}_{j+1} \neq 0$ y como $\|w_{j+1}\|_2 = 1$, se tiene que:

$$\beta_{j+1} = \|\hat{w}_{j+1}\|_2 \tag{1.16}$$

Las relaciones (1.14) y (1.15) inducen un método iterativo para calcular los coeficientes $\alpha_j$ y $\beta_j$. El método simétrico de Lanczos fue formulado originalmente como un algoritmo para el cálculo de los autovalores de $A$, es decir, se calculan los coeficientes $\alpha_j$ y $\beta_j$ hasta $m < n$, y se aproximan los autovalores de $A$ mediante los autovalores de $T_m$, que son triviales de calcular dado que $T_m$ es simétrica tridiagonal [Lan50] [Lan52] [GL89].

De forma independiente Hesteness y Stifield formularon el algoritmo del Gradiente Conjugado que no es más que la versión del algoritmo simétrico de Lanczos, pero aplicado a la resolución de un sistema de ecuaciones [HS52] [GL89] [Ste91]. Ellos derivaron el método como un algoritmo de optimización para encontrar el mínimo de la forma cuadrática:

$$\phi(x) = \frac{1}{2}x^t Ax - b^t x \tag{1.17}$$

Se demuestra que el algoritmo del Gradiente Conjugado es un método de optimización que reduce la norma del error respecto a la matriz $A$ en cada paso. Para un análisis profundo de las propiedades de minimización del Gradiente Conjugado ver [Hest80]. En concreto se cumple que:

$$\|\hat{x} - x_k\|_A \leq [\frac{1 - \kappa(A)^{1/2}}{1 + \kappa(A)^{1/2}}]^{2k} \cdot \|\hat{x} - x_0\|_A \tag{1.18}$$

donde $\kappa(A)$ es el número de condición de $A$. El algoritmo del Gradiente Conjugado puede verse en la figura 1.4.

Elegir $x_0$; Calcular $r_0 = b - Ax_0$
While (not convergence)
$\beta_k = \frac{<r_{k-1}, r_{k-1}>}{<r_{k-2}, r_{k-2}>}$
$p_k = r_{k-1} + \beta_k \cdot p_{k-1}$
$\alpha_k = \frac{<r_{k-1}, r_{k-1}>}{<p_{k-1}, Ap_{k-1}>}$
$x_k = x_{k-1} + \alpha_k \cdot p_k$
$r_k = r_{k-1} + \alpha_k \cdot Ap_k$
EndWhile

Figura 1.4: Algoritmo del Gradiente Conjugado.

En [GL89] se describe ampliamente este método. Para una revisión más detallada consultar dicha referencia.

### 1.3.4 Gradiente BiConjugado (BiCG, Lanczos No Simétrico)

Se ha presentado el método simétrico de Lanczos como un caso particular de método de Arnoldi cuando la matriz $A$ es simétrica positiva definida, ya que las propiedades de minimización de los algoritmos derivados del método de Arnoldi tiene su equivalente en el método simétrico de Lanczos [Ashb90]. Sin embargo, prescindiendo del objetivo de minimización, existe otra forma de generalizar a matrices no simétricas el método de Lanczos.

Sean $W = [w_1, \ldots, w_n]$ y $V = [v_1, \ldots, v_n]$ dos bases ortonormales tales que $V^{-t} = W$. El método no simétrico de Lanczos pretende obtener una matriz tridiagonal no simétrica que cumpla:

$$T = V^{-1}AV \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \gamma_2 & \alpha_2 & \beta_3 & & & \\ & \gamma_3 & \alpha_3 & \beta_4 & & \\ & & \cdots & \cdots & \cdots & \\ & & & \gamma_{n-1} & \alpha_{n-1} & \beta_n \\ & & & & \gamma_n & \alpha_n \end{bmatrix} \tag{1.19}$$

Reescribiendo la ecuación (1.19) como $AV = VT$ y $A^tW = WT^t$ se tienen dos ecuaciones equivalentes a la ecuación (1.13) del caso simétrico:

Elegir $x_0$; Calcular $r_0 = b - Ax_0$
Elegir $\hat{r}_0 = r_0$; $p_0 = \hat{p}_0$; $\rho_0 = 0$
While (not convergence)
$\rho_k =< \hat{r}_{k-1}, r_{k-1} >$
$\beta_k = \frac{\rho_k}{\rho_{k-1}}$
$p_k = r_{k-1} + \beta_k \cdot p_{k-1}$
$\hat{p}_k = \hat{r}_{k-1} + \beta_k \cdot \hat{p}_{k-1}$
$\alpha_k = \frac{\rho_k}{<\hat{p}_k, Ap_k>}$
$x_k = x_{k-1} + \alpha_k \cdot p_k$
$r_k = r_{k-1} + \alpha_k \cdot Ap_k$
$\hat{r}_k = \hat{r}_{k-1} + \alpha_k \cdot A^t\hat{p}_k$
EndWhile

Figura 1.5: Algoritmo del Gradiente BiConjugado.

$$Av_j = \beta_j v_{j-1} + \alpha_j v_j + \gamma_{j+1} v_{j+1} \tag{1.20}$$

$$A^t w_j = \gamma_j w_{j-1} + \alpha_j w_j + \beta_{j+1} w_{j+1} \tag{1.21}$$

con $j = 1, 2, \ldots, n$ y entendiendo que $\beta_1 = \beta_{n+1} = \gamma_1 = \gamma_{n+1} = 0$. De forma equivalente al caso simétrico se obtienen ahora las siguientes relaciones:

$$\alpha_j = w_j^t A w_j \tag{1.22}$$

$$\gamma_{j+1} v_{j+1} = (A - \alpha_j I) v_j - \beta_j v_{j-1} \equiv \hat{v}_{j+1} \tag{1.23}$$

$$\beta_{j+1} w_{j+1} = (A - \alpha_j I) w_j - \gamma_j w_{j-1} \equiv \hat{w}_{j+1} \tag{1.24}$$

En [Fle76] se propone el método equivalente al Gradiente Conjugado para sistemas no simétricos basado en las relaciones (1.22), (1.23) y (1.24). El algoritmo del Gradiente Biconjugado puede verse en la figura 1.5.

Dado que ahora $A$ ya no define una norma, no existe ninguna propiedad de minimización del error equivalente a la del Gradiente Conjugado. Notar que el algoritmo del BiCG genera dos bases mutuamente ortogonales de los espacios $K = K_m(A, r_0)$ y $\Lambda = K_m(A^t, \hat{r}_o)$, respectivamente.

En [Fle76] se describe ampliamente este método. Para una revisión más detallada consultar dicha referencia.

### 1.3.5 Gradiente Conjugado al Cuadrado (CGS)

Sonneveld observó que cuando el BiCG converge las dos secuencias de vectores $r_j$ y $\hat{r}_j$ convergen hacia cero. Dado que los vectores $r_j$ y $\hat{r}_j$ pueden escribirse como:

$$r_j = P_j(A) r_0 \tag{1.25}$$

$$\hat{r}_j = P_j(A^t) \hat{r}_0 \tag{1.26}$$

Se puede escribir entonces:

$$< \hat{r}_j, r_j > = < P_i(A^t) \hat{r}_0, P_j(A) r_0 > = < \hat{r}_0, P_i(A) P_j(A) r_0 > \tag{1.27}$$

Es decir, los parámetros escalares del BiCG se pueden calcular sin necesidad de usar $A^t$, usando únicamente productos por la matriz $A$ como indica la ecuación (1.27).

Elegir $x_0$; Calcular $r_0 = b - Ax_0$
Elegir $\hat{r}_0 = r_0$; $p_0 = q_0$; $\rho_0 = 0$
While (not convergence)
$\rho_k = < \hat{r}_0, r_{k-1} >$
$\beta_k = \frac{\rho_k}{\rho_{k-1}}$
$u = r_{k-1} + \beta_k \cdot q_{k-1}$
$p_k = u + \beta_k \cdot (q_{k-1} + \beta_k \cdot p_{k-1})$
$\alpha_k = \frac{\rho_k}{<\hat{p}_0, Ap_k>}$
$q_k = u - \alpha_k \cdot p_k$
$x_k = x_{k-1} + \alpha_k \cdot (u + q_k)$
$r_k = r_{k-1} + \alpha_k \cdot (u + q_k)$
EndWhile

Figura 1.6: Algoritmo del CGS.

De esta forma se demuestra que se puede calcular $\hat{r}_k$ como:

$$\hat{r}_k = P_k^2(A) r_0 \tag{1.28}$$

El método resultante se denominó Gradiente Conjugado Cuadrado (CGS). El algoritmo del CGS puede verse en la figura 1.6.

En [Son89] se describe ampliamente este método. Para una revisión más detallada consultar dicha referencia.

### 1.3.6 Gradiente BiConjugado Estabilizado (BiCGstab)

El principal inconveniente del CGS es que presenta una convergencia a saltos. Esto sucede en situaciones como la siguiente, supongamos que $r_0$ tiene pequeñas componentes en alguna de las direcciones de los autovectores de $A$, y que $P_j(\lambda)$ toma valores grandes para dichas componentes, pero de forma que el producto no contribuye significativamente a la norma de $r_j$. Sin embargo, el valor que tome $P_j^2(\lambda)$ en esas mismas componentes puede ser significativamente más grande de forma que dichas componentes dominen el residuo $r_j$. Esto provoca residuos sucesivos que crecen, y que súbitamente decrecen cuando la contribución en esas componentes se cancela por la evolución del método.

Van der Vost propuso, para suavizar la convergencia a saltos del CGS, modificar el polinomio que se usaba para el cálculo de $r_k$. En concreto, propuso usar:

Elegir $x_0$; Calcular $r_0 = b - Ax_0$
Elegir $\hat{r}_0 = r_0$; $p_0 = q_0 = 0$; $\rho_0 = \alpha_1 = \omega_0 = 1$
While (not convergence)
$\rho_k =< \hat{r}_0, r_{k-1} >$
$\beta_k = \frac{\rho_k}{\rho_{k-1}} \frac{\alpha_k}{\omega_{k-1}}$
$p_k = r_{k-1} + \beta_k \cdot (p_{k-1} - \omega_{k-1} q_{k-1})$
$q_k = Ap_k$
$\alpha_k = \frac{\rho_k}{<\hat{r}_0, q_k>}$
$s_k = r_{k-1} - \alpha_k \cdot q_k$
$t_k = As_k$
$\omega_k = \frac{<t_k, s_k>}{<t_k, t_k>}$
$x_k = x_{k-1} + \alpha_k \cdot p_k + \omega_k \cdot s_k$
$r_k = s_k + \omega_k \cdot t_k$
EndWhile

Figura 1.7: Algoritmo del BiCGstab.

$$r_k = Q_k(A) P_k(A) r_0 \tag{1.29}$$

$$Q_k(\lambda) = (1 - \omega_1 \lambda)(1 - \omega_2 \lambda) \ldots (1 - \omega_k \lambda) \tag{1.30}$$

donde las constantes $\omega_i$ son elegidas en cada iteración de tal forma que la norma de $r_k$ es minimizada respecto a $\omega_k$. En la figura 1.7 se puede ver el algoritmo del BiCGstab.

En [dV92] se describe ampliamente este método. Para una revisión más detallada consultar dicha referencia.

## 1.4 Criterio de Parada

La pregunta que se plantea es: Qué condición de convergencia se debe usar para establecer un criterio de parada de un método iterativo? En esta sección se describen algunos detalles para lograr conseguir una respuesta.

Un criterio de parada natural para un método iterativo es que alguna medida del error por defecto o por exceso no supere una tolerancia $\epsilon$. Se asume que el residuo,

$r = b - Ay$ está disponible para cada $y$, y que las normas de $y$, $r$ y $A$ pueden ser calculadas o estimadas.

**Teorema 1.4.1** *El error por defecto*

$$\eta_{E,f}(y) = min\{\epsilon : (A + \triangle A)y = b + \triangle b, \quad \|\triangle A\| \leq \epsilon\|E\|, \quad \|\triangle b\| \leq \epsilon\|f\|\}$$

*está dado por*

$$\eta_{E,f}(y) = \frac{\|r\|}{\|E\|\|y\| + \|f\|},$$

*donde $r = b - Ay$, $E$ es una matriz de perturbación y $f$ es un vector de perturbación.*

Para la demostración del teorema ver [Hig96].

Del Teorema anterior se tienen las siguientes equivalencias, para cualquier norma matricial, que pueden ser usadas como criterio de parada:

$$\|r\| \leq \epsilon\|b\| \iff Ay = b + \triangle b, \quad \|\triangle b\| \leq \epsilon\|b\| \tag{1.31}$$

$$\|r\| \leq \epsilon\|A\|\|y\| \iff (A + \triangle A)y = b, \quad \|\triangle A\| \leq \epsilon\|A\| \tag{1.32}$$

$$\|r\| \leq \epsilon(\|A\|\|y\| + \|b\|) \iff (A + \triangle A)y = b + \triangle b, \quad \|\triangle A\| \leq \epsilon\|A\|, \quad \|\triangle b\| \leq \epsilon\|b\| \tag{1.33}$$

Estas desigualdades permanecen ciertas al reemplazar las normas por valor absoluto, pero para evaluar (1.32) y (1.33) el producto $|A||y|$ deberá ser calculado, lo cual es costoso en un método iterativo.

De estas condiciones, generalmente (1.33) es la mejor, asumiendo que es aceptable perturbar $A$ y $b$. Hay que hacer notar la importancia de incluir $\|A\|$ y $\|b\|$ en la condición sobre $\|r\|$; una condición $\|r\| \leq \epsilon\|A\|$, aunque la escala es independiente, no es una cota para el error relativo por defecto. La condición (1.31) es la comúnmente usada, pero puede ser muy estricta, y algunas veces posiblemente no satisfecha. Para ver por qué, notese que el residuo de la solución exacta redondeada $fl(x) = x + \triangle x$, $|\triangle x| \leq u|x|$, satisface, para cualquier norma,

$$\|r\| = \|b - A(x + \triangle x)\| \leq u\|A\|\|x\|,$$

y

$$\frac{\|A\|\|x\|}{\kappa(A)} \leq \|b\| \leq \|A\|\|x\|.$$

si $A$ es mal condicionada y $x$ es una solución con norma grande, es decir, $\|x\| \approx \|A^{-1}\|\|b\|$, entonces $\|b\|$ está cerca de su cota inferior. Por lo tanto, (1.31) es mucho más difícil de satisfacer que (1.33).

Para el error por exceso las condiciones se derivan del residuo y de $A^{-1}$. La igualdad $x - y = A^{-1}r$ guía las cotas de las normas y las componentes del error por exceso, tal como $\frac{\|x-y\|}{\|y\|} \leq \frac{A^{-1}\|r\|}{\|y\|}$. Ya que estas cotas dependen de $A^{-1}$, no son triviales de calcular. Algunos métodos iterativos automáticamente producen estimaciones de los autovalores extremos, y por lo tanto $\kappa_2(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$. Para matrices dispersas simétricas definidas positivas la $\|A^{-1}\|_2$ puede ser estimada a un costo computacional bajo usando el método de Lanczos.

## 1.5 Precondicionamiento

Todos los métodos iterativos precisan de algún tipo de acelerador (precondicionador) que reduzca el número de iteraciones hasta la convergencia. La idea básica de un precondicionador es buscar una matriz $M$ que sea una buena aproximación de $A$, y que sea mucho más fácil de invertir. Entonces, se usa $M^{-1}$ para corregir las sucesivas aproximaciones de un método iterativo. Es decir, si $r^m$ es el residuo en la iteración $m$-ésima, se calcula $\varepsilon^m = M^{-1}r^m$ que es una aproximación del error en el paso $m$-ésimo, y se corrige la solución como $x^m = x^m + \varepsilon^m$.

Para precondicionar el sistema (1.1) se debe buscar una matriz $M^{-1}$ tal que el número de condición de la matriz precondicionada sea mucho menor que el de la matriz $A$. El sistema lineal (1.1) puede ser precondicionado de varias formas: por la derecha, por la izquierda y de forma simétrica.

$$AM^{-1}Mx = b \tag{1.34}$$

$$M^{-1}Ax = M^{-1}b \tag{1.35}$$

$$M^{-1/2}AM^{-1/2}M^{1/2}x = M^{-1/2}b \tag{1.36}$$

Aunque en principio todas estas formas de precondicionamiento tienen efectos equivalentes, algunas tienen ciertas ventajas prácticas frente a otras. La principal

ventaja viene dada por la relación que existe entre el residuo del sistema precondicionado ($\tilde{A}\tilde{x} = \tilde{b}$) y el residuo del sistema original.

Para la forma (1.34):

$$\tilde{A} = AM^{-1}, \quad \tilde{b} = b, \quad \tilde{x} = Mx$$

$$\tilde{r} = \tilde{b} - \tilde{A}\tilde{x} = b - AM^{-1}Mx = b - Ax = r$$

Para la forma (1.35):

$$\tilde{A} = M^{-1}A, \quad \tilde{b} = M^{-1}b, \quad \tilde{x} = x$$

$$\tilde{r} = \tilde{b} - \tilde{A}\tilde{x} = M^{-1}b - M^{-1}Ax = M^{-1}(b - Ax) = M^{-1}r$$

Para la forma (1.36):

$$\tilde{A} = M^{-1/2}AM^{-1/2}, \quad \tilde{b} = M^{-1/2}b, \quad \tilde{x} = M^{1/2}x$$

$$\tilde{r} = \tilde{b} - \tilde{A}\tilde{x} = M^{-1/2}b - M^{-1/2}AM^{-1/2}M^{1/2}x = M^{-1/2}(b - Ax) = M^{-1/2}r$$

En esta tesis, se usa un precondicionamiento por la derecha dado que en este caso el residuo del sistema precondicionado coincide con el residuo del sistema original.

El precondicionador ideal es $M^{-1} = A^{-1}$. Lo que evidentemente es imposible de calcular en los problemas de la vida real. Por lo tanto, lo que se pretende es encontrar una matriz $M^{-1}$ que aproxime lo mejor posible a la matriz $A^{-1}$ con un coste computacional mínimo.

Existen tres formas clásicas para conseguir precondicionadores de carácter general, es decir, que estén basados únicamente en la información algebraica de la matriz $A$. Dichas formas son los precondicionadores polinomiales, las factorizaciones incompletas y los precondicionadores SPAI.

### 1.5.1 Precondicionadores Polinomiales

Los precondicionadores polinomiales se basan en aproximar la matriz $M^{-1}$ mediante un polinomio de grado $m$ de la matriz $A$ [AMS89] [AMO92] [EOV90] [Saa85]. Esto es:

$$M^{-1} = A^{-1} = P_m(A) = q_0 I + q_1 A + \cdots + q_m A^m \tag{1.37}$$

De esta forma, cada vez que se debe multiplicar la matriz $M^{-1}$ por un vector lo que realmente se hace es multiplicar $m$ veces la matriz $A$ por un vector. Los coeficientes del polinomio se eligen de forma que se minimice:

$$min_E \|1 - P_m(\lambda)\| \tag{1.38}$$

donde $E$ es un intervalo que incluye el espectro de $A$. El problema de minimización (1.38) tiene diferentes soluciones en función de que norma consideremos. Si se toma la norma infinito para (1.38) ($\|f\|_\infty = max_{\lambda \in E} \|f(\lambda)\|$) la solución es:

$$\lambda P_m(\lambda) = \frac{T_m(\frac{d+c-2\lambda}{d-c})}{T_m(\frac{d+c}{d-c})} \tag{1.39}$$

donde $T_m(\lambda)$ es el polinomio de Chebyshev de primera clase de grado $m$, y $E = [c, d]$. El principal problema que presentan los precondicionadores polinomiales es la necesidad de disponer de buenas cotas del espectro de $A$, ya que si no en (1.39) se requieren grados muy grandes del polinomio para que la aproximación sea buena. Esto limita mucho la aplicabilidad práctica de estos precondicionadores.

### 1.5.2 Factorizaciones Incompletas

Los únicos precondicionadores que son lo suficientemente generales, y que no requieren la estimación de ningún parámetro son las factorizaciones incompletas. La idea inicial de una factorización incompleta es factorizar la matriz $A$ (factorizaciones $LU$, $Cholesky$ o $LDL^t$) pero sin introducir todo el llenado que se produce en el proceso de factorización. En consecuencia, el precondicionador puede aplicarse resolviendo dos sistemas triangulares cuya dispersidad, y por lo tanto complejidad de cálculo, dependerá del tipo de factorización incompleta que hayamos aplicado.

Es bien conocido que los tres bucles que constituyen un algoritmo de factorización pueden ser intercambiados dando lugar a las diferentes formas $ijk$ de la factorización. La elección de la forma $ijk$ más eficiente responde en el caso de matrices densas a criterios de explotación de la localidad de datos [DGK84] [NJL94], pero en el caso de matrices dispersas estamos obligados a elegir aquella forma $ijk$ que se adapte al formato de almacenamiento usado.

Existen tres formas fundamentales de factorizaciones incompletas:

- Factorizaciones sin llenado: $ILU(0)$, $ICH(0)$ [MdV77].

- Factorizaciones con llenado, usando como criterio para la introducción del llenado la posición dentro de la matriz: $ILU(k)$, $ICH(k)$ [MdV77] [Wat81].

- Factorizaciones con llenado, usando como criterio para la introducción del llenado umbrales numéricos: $ILU(\tau)$, $ICH(\tau)$ [Zla82].

Las factorizaciones del tipo $ILU(0)$ han sido principalmente usadas en el contexto de EDPs. Son las más sencillas y no introducen llenado alguno, es decir, la factorización incompleta tiene la misma cantidad de elementos no nulos y en las mismas posiciones que en la matriz $A$. Esto permite reutilizar, para el precondicionador, todos los vectores de índices usados para la matriz, con un considerable ahorro de memoria. Sin embargo, salvo en problemas relativamente sencillos (EDPs elípticas con coeficientes constantes) este tipo de precondicionadores no son lo suficientemente potentes.

El parámetro $k$ de una factorización $ILU(k)$, en el contexto de EDPs para problemas discretizados mediante diferencias finitas, indica el número de columnas alrededor de la diagonal en las que se permite llenado. Para problemas discretizados mediante elementos finitos el criterio es diferente. Al factorizar la fila $i$-ésima asociada al nodo $i$-ésimo de la malla de elementos finitos, sólo se permite llenado en las columnas asociadas a los nodos que son $k$-vecinos del nodo $i$-ésimo. Se dice que un nodo $N$ es $k$-vecino de un nodo $M$ si el camino más corto que une $N$ con $M$ atraviesa $k$ nodos.

En general, las factorizaciones del tipo $ILU(k)$ adolecen del defecto de considerar que la importancia numérica de un llenado $l_{ij}$, depende únicamente de la proximidad topológica entre los nodos $i$ y $j$, sin tener en cuenta el fenómeno físico que la matriz $A$ representa. Habitualmente, el uso de las factorizaciones del tipo $ILU(k)$ se restringen a problemas de EDPs que modelan fenómenos de difusión y que son discretizados mediante diferencias finitas.

Las factorizaciones del tipo $ILU(\tau)$ deciden introducir o no un llenado $l_{ij}$ en función de si es superior o inferior a un umbral determinado. Dicho umbral se calcula relativo al valor de los elementos de la fila $i$-ésima de $A$ usando el parámetro $\tau$. Este cálculo se puede realizar usando cualquier medida, por ejemplo, el valor medio de los elementos de la fila $i$-ésima, o cualquier norma de la fila $i$-ésima. Estas factorizaciones son de aplicación general, pero adolecen de una falta de control fino sobre la cantidad total de llenado que se permite. Esta falta de control genera problemas de dimensionamiento del espacio de memoria reservado al precondicionador, y sobre

todo, impide buscar un buen compromiso entre complejidad de cálculo del precondicionador y la aceleración que el precondicionador introduce en el método iterativo.

En [Saa94b] y [Saa94a] se define un tipo híbrido de factorización incompleta, llamado $ILUt(fil, \tau)$, que es de aplicación totalmente genérica y que supera los inconvenientes de las factorizaciones del tipo $ILU(\tau)$. Dicha factorización sigue una doble estrategia en la introducción del llenado:

- Al factorizar la fila $i$-ésima se introducen todos los llenados $l_{ij}$ que superen un umbral numérico relativo a la fila $i$-ésima (parámetro $\tau$).

- Una vez finalizada la factorización de la fila $i$-ésima, sólo se almacenan en la estructura de datos de salida tantos elementos como tuviese la matriz $A$ en la fila $i$-ésima más 2 veces $fil$ ($fil$ más en la parte $L$, y $fil$ más en la parte $U$). Se eligen para su almacenamiento aquellos elementos con un valor absoluto mayor.

Así pues, el parámetro $\tau$ sirve para controlar el umbral numérico de cálculo, y el parámetro $fil$ sirve para controlar la cantidad efectiva de llenado.

Una modificación habitual en las factorizaciones incompletas es añadir los coeficientes no incluidos al termino de la diagonal de $U$ de tal forma que la fila $i$-ésima de $A$ y de la factorización sumen lo mismo. Son las llamadas factorizaciones $MILU$ (Modified Incomplete $LU$). Sin embargo, este tipo de modificación puede presentar algunos problemas de estabilidad numérica [Eij90b] [Eij90a].

### 1.5.3 Precondicionadores SPAI

La idea básica de los precondicionadores SPAI es buscar una matriz $M$ tal que $AM$ sea tan cercana a la identidad como sea posible. En los últimos años, ha habido un gran interés por desarrollar precondicionadores SPAI, ya que la aplicación de estos precondicionadores, en cada iteración del método iterativo, sólo requiere la multiplicación matriz por vector; que como se sabe es una operación altamente paralela. Hoy en día, hay planteadas dos metodologías principales para calcular tales precondicionadores. La primera, se basa en minimizar $\|AM - I\|$ en la norma de Frobenius [GS93b] [GS94] [GS95] [GH97] [KY93]. Dado que:

$$\|AM - I\|_F^2 = \sum_{i=1}^{N} \|Am_i - e_i\|_2^2 \tag{1.40}$$

donde $e_i$ es el vector $i$-ésimo de la base canónica. Cada columna de $M$ se puede calcular en paralelo resolviendo el problema de mínimos cuadrados:

$$min\|Am_i - e_i\|_2 \tag{1.41}$$

donde $i = 1, \ldots N$.

Para resolver (1.41) de forma eficiente es crucial la observación de que la inversa de $A$ usualmente es mucho más densa que $A$, pero los coeficientes significativos de $A^{-1}$ son únicamente unos pocos. Es decir, se espera que $M$ sea una matriz dispersa. Por lo tanto, el problema (1.41) es en realidad de una dimensión muy pequeña y puede ser resuelto de forma eficiente. Dado que la estructura de $M$ no es conocida a priori, se comienza con una estructura diagonal, y se aumenta el llenado hasta que $\|Am_i - e_i\|_2 \leq \varepsilon$ para $i = 1, \ldots N$ con $0 < \varepsilon < 1$, o bien se llega a un máximo nivel de llenado permitido. A los precondicionadores SPAI derivados de esta técnica los llamaremos LSQ-SPAI (*Least SQuare SPAI*).

La segunda técnica, se basa en descomponer $A$ como un producto de matrices ortogonales y diagonales, de manera que $A^{-1}$ quede descompuesta de forma trivial [SC98] [BMT96]. Es decir,

$$A = WDZ^t \tag{1.42}$$

donde $W$ y $Z$ son matrices ortogonales y $D$ es una matriz diagonal. Entonces, se tiene que:

$$A^{-1} = ZD^{-1}W^t \tag{1.43}$$

Si se aplica algún criterio de diezmado en el proceso de construcción de las matrices ortogonales se tiene una aproximación dispersa de $A^{-1}$. Las formas de generar la descomposición de la ecuación (1.42) pueden ser diversas. En [SC98] se usa la base generada por el GMRES; en [BMT96] se usa un algoritmo de ortogonalización basado en el método de Lanczos, tanto para el caso simétrico como no simétrico. A los precondicionadores SPAI derivados de esta técnica los llamaremos ORT-SPAI (*ORThogonal SPAI*).

La principal ventaja de estos precondicionadores SPAI es que presentan un grado de paralelismo alto, tanto en su cálculo como en su aplicación. Su principal es que no existe garantía de que $A^{-1}$ sea lo suficientemente dispersa como para que esta

aproximación sea práctica.

Aunque se han presentado algunos resultados prometedores de ambas técnicas, no está claro que su aplicabilidad sea general. Por tal motivo, a continuación se presenta un estudio experimental sobre la viabilidad de estos precondicionadores para problemas convectivos [LC99]. En este estudio se construyen ambos precondicionadores (LSQ-SPAI y ORT-SPAI) de forma de que éstos sean una cota optimista de las verdaderas implementaciones.

### Problemas de Prueba

Para los experimentos se ha usado un operador 3D escalar elíptico de segundo orden. El dominio computacional es un hexaedro de longitud igual a 1. La discretización fue hecha usando diferencias finitas con un 7-*stencil*. Es claro que este problema de prueba no es un problema industrial pero el operador elegido caracteriza una gran variedad de problemas industriales.

Sea,

$$L(\vec{u}) = \frac{\partial}{\partial x}(\alpha\frac{\partial}{\partial x}\vec{u}) + \frac{\partial}{\partial y}(\beta\frac{\partial}{\partial y}\vec{u}) + \frac{\partial}{\partial z}(\gamma\frac{\partial}{\partial z}\vec{u}) + \frac{\partial}{\partial x}(\delta\vec{u}) + \frac{\partial}{\partial y}(\zeta\vec{u}) + \frac{\partial}{\partial z}(\eta\vec{u})$$

donde

$$\vec{u}(x,y,z) = \begin{pmatrix} u_1(x,y,z) \\ \vdots \\ u_m(x,y,z) \end{pmatrix} \ , \ \alpha(x,y,z) = \begin{pmatrix} \alpha_{1,1}(x,y,z) & \cdots & \alpha_{1,m}(x,y,z) \\ \vdots & \ddots & \vdots \\ \alpha_{m,1}(x,y,z) & \cdots & \alpha_{m,m}(x,y,z) \end{pmatrix}$$

y $\beta, \gamma, \delta, \zeta$ y $\eta$ son matrices de dimension $m \times m$, donde $m$ es el número de grados de libertad asociado a cada nodo de la malla de discretización. El coeficiente de convección, $Cc$, está definido por:

$$Cc = \frac{\|\vec{w}\|_\infty}{\|\vec{v}\|_\infty}$$

donde

$$\vec{v} = \begin{pmatrix} \|\alpha\|_\infty \\ \|\beta\|_\infty \\ \|\gamma\|_\infty \end{pmatrix} \ , \ \vec{w} = \begin{pmatrix} \|\delta\|_\infty \\ \|\zeta\|_\infty \\ \|\eta\|_\infty \end{pmatrix}$$

Cuando $Cc$ tiende a 0, se dice que la ecuación es difusiva, y cuando $Cc$ tiende a infinito se dice que la ecuación es convectiva. Para $Cc = 0$, el operador es simétrico positivo definido. Cuanto más grande es $Cc$ más no simétrica es la matriz resultante

de la discretización. En los experimentos se ha usado $\vec{u} = 0$ como condición de contorno. El lado derecho del sistema ha sido generado artificialmente de tal forma de que la solución sea un vector con valores aleatorios en el intervalo [0,1].

### La matriz $A$

La matriz $A$ tiene la misma estructura en todos los casos de prueba y sólo varian sus coeficientes. La estructura de esta matriz puede ser observada en la figura 1.22, que está al final de este Capítulo. Para este estudio la matriz $A$ es de 1000×1000. Aunque esta matriz es pequeña, nos permite hacer un estudio detallado del rendimiento de estos precondicionadores.

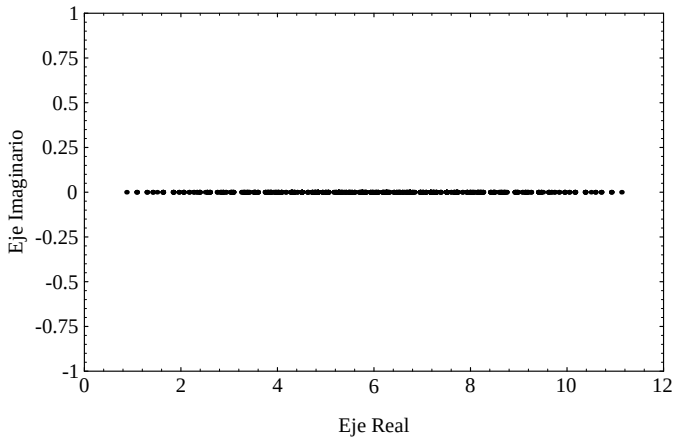



Figura 1.8: Autovalores de $A$ para $Cc = 10$.

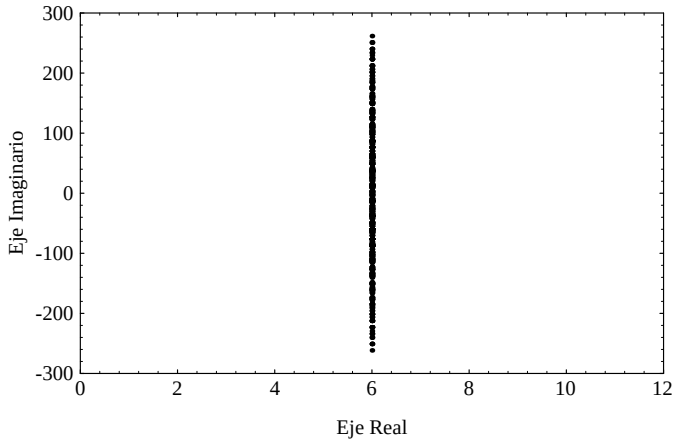



Figura 1.9: Autovalores de $A$ para $Cc = 1000$.

Hay que señalar, que la estructura de $A$ no afecta el cálculo de los precondicionadores SPAI que se calculan en esta sección. Esto no sucede con las factorizaciones

incompletas, ya que el nivel de llenado, y en consecuencia la cantidad de elementos que se desprecian en el proceso de cálculo del precondicionador depende de la estructura de la matriz. Igualmente los precondicionadores ORT-SPAI reales también se ven afectados por la estructura de $A$, aunque en menor grado que las factorizaciones incompletas. Esto se debe a que la propagación de los errores en el proceso de ortogonalización depende de la estructura de la matriz $A$.

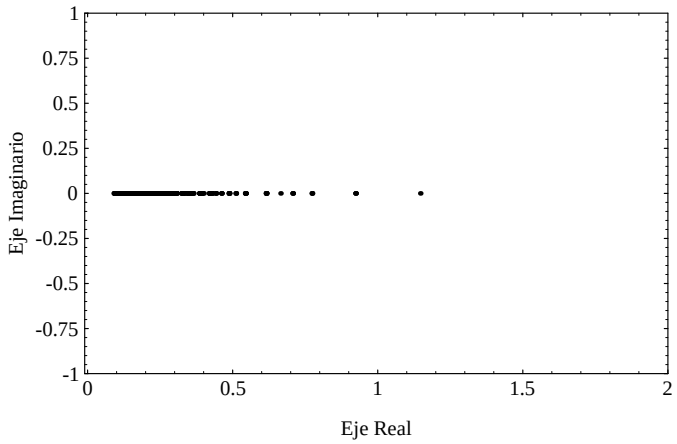



Figura 1.10: Autovalores de $A^{-1}$ para $Cc = 10$.

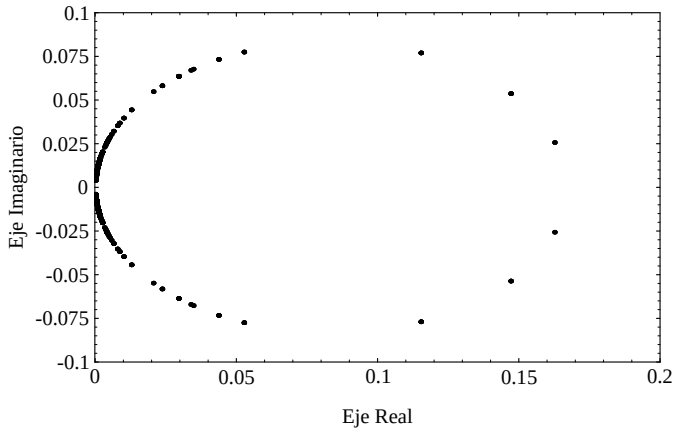



Figura 1.11: Autovalores de $A^{-1}$ para $Cc = 1000$.

En la figura 1.8 puede verse el espectro de la matriz $A$ para $Cc = 10$. En este caso la matriz es débilmente no simétrica, y en consecuencia tiene autovalores complejos con parte imaginaria despreciable. En la figura 1.9 se muestra el espectro para $Cc = 1000$, en este caso la matriz es fuertemente no simétrica, y en consecuencia la parte imaginaria de los autovalores es dominante.

En las figuras 1.10 y 1.11 se muestra los autovalores de la matriz $A^{-1}$ para los diferentes valores de $Cc$.

### La matriz $M^{-1}$

En este apartado se analiza la estructura y el espectro de los precondicionadores LSQ-SPAI ($M^{-1}$) y ORT-SPAI($M^{-1}$).

**LSQ-SPAI:** Estos precondicionadores se han calculado diezmando la matriz $A^{-1}$ en los diferentes problemas de prueba. El diezmado de la matriz $A^{-1}$ se realizó tomando los coeficientes más significativos en valor absoluto. Se hicieron dos filtrados, el primero tomando todos los coeficientes que estaban a 1 orden de magnitud del máximo elemento de $A$ y el segundo tomando los que estaban a 2 ordenes de magnitud. En las figuras 1.23, 1.25 y 1.24, que están al final de este capítulo, se puede observar la estructura de las matrices diezmadas. Para el caso $Cc = 1000$ con un diezmado a 2 ordenes de magnitud la matriz resultante es completamente densa. Por tal motivo, no se muestra su estructura. Se puede observar que para problemas dominados fuertemente por la convección ($Cc = 1000$), la matriz tiene una alta densidad independientemente del diezmado usado.

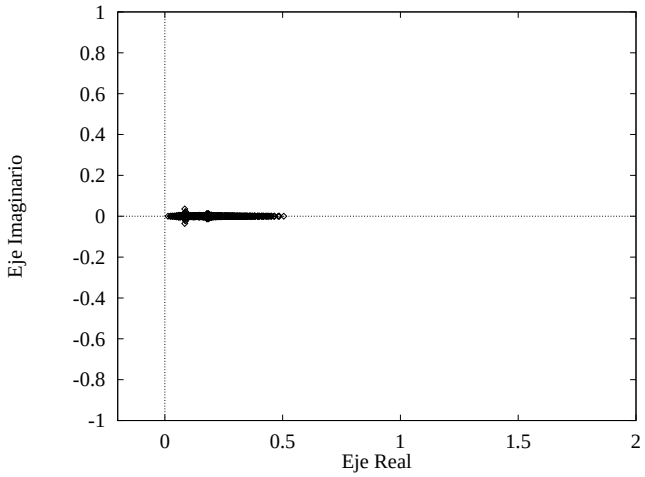



Figura 1.12: Autovalores de LSQ-SPAI($M^{-1}$) para $Cc = 10$ diezmando a 1 orden de magnitud.

El elevado número de elementos que aparecen en las matrices diezmadas indica que la cantidad de cálculo necesario para obtener un precondicionador LSQ-SPAI en un problema dominados fuertemente por la convección es muy elevada. De hecho el cálculo de un precondicionador LSQ-SPAI se basa en la suposición de que la cantidad de elementos significativos de $A^{-1}$ es pequeña. Situación que no se da en

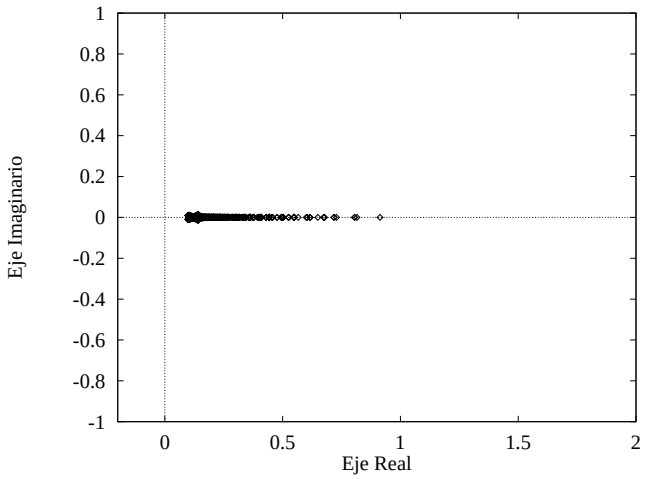



Figura 1.13: Autovalores de LSQ-SPAI($M^{-1}$) para $Cc = 10$ diezmando a 2 ordenes de magnitud.

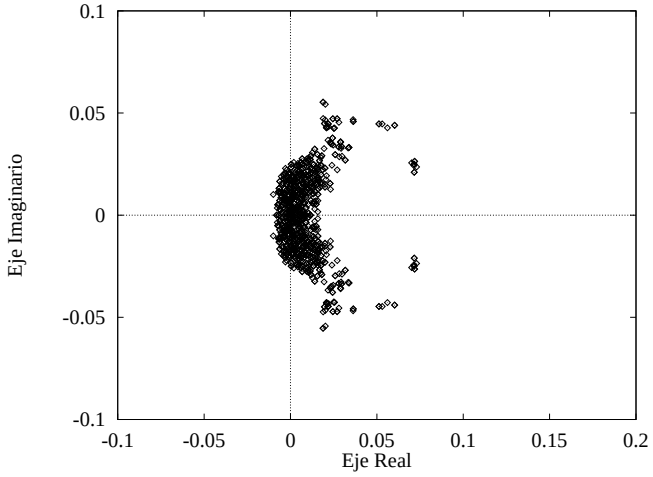



Figura 1.14: Autovalores de LSQ-SPAI($M^{-1}$) para $Cc = 1000$ diezmando a 1 orden de magnitud.

los problemas convectivos.

Prescindiendo de la cantidad de cálculo que requiere construir $M^{-1}$, se desea ahora analizar en cuanto puede dicha matriz mejorar la convergencia de un método iterativo. Para que la convergencia de un método iterativo sea notablemente acelerada el espectro de $M^{-1}$ debe "parecerse" al espectro de $A^{-1}$. En las figuras 1.12, 1.13, 1.14 y 1.15 se muestra el espectro de $M^{-1}$ para los diferentes valores de $Cc$ con los diezmados a 1 y 2 ordenes de magnitud.

Se puede observar que si se elige como precondicionador la matriz resultante de

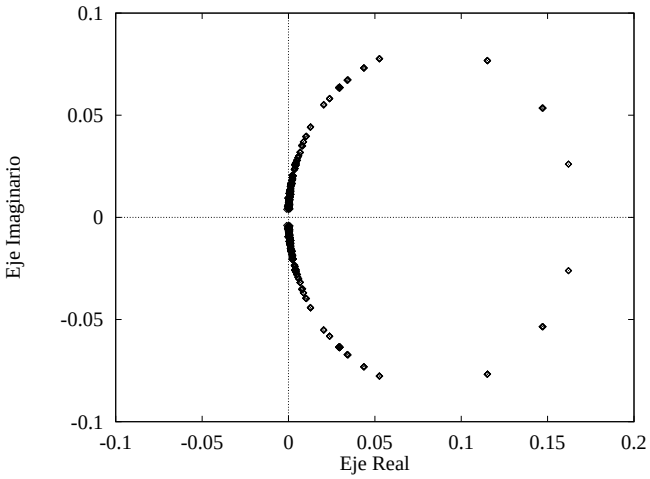



Figura 1.15: Autovalores de LSQ-SPAI($M^{-1}$) para $Cc = 1000$ diezmando a 2 ordenes de magnitud.

diezmar a 1 orden de magnitud la matriz $A^{-1}$, el espectro de $M^{-1}$ resulta bastante distorsionado respecto al de $A^{-1}$. Por lo tanto, se deduce que usar como precondicionador la matriz resultante de un diezmado a 1 orden de magnitud de $A^{-1}$, puede no ser sufiente para hacer converger al método iterativo. Sin embargo, se observa que un diezmado a 2 ordenes de magnitud resulta suficiente para que los espectros de $A^{-1}$ y $M^{-1}$ sean similares; aunque esta similitud empeora al crecer el valor de $Cc$.

**ORT-SPAI:** En esta sección se describe el espectro de los precondicionadores ORT-SPAI. Hay que recordar que dichos precondicionadores se basan en descomponer la matriz $A$ en un producto de matrices ortogonales y diagonales. En concreto este estudio se centra únicamente en el algoritmo de biortogonalización de Lanczos. Este es el método usado en [BMT96]. En concreto el algoritmo calcula dos conjuntos de vectores $\{z_i\}_{i=1}^n$, $\{w_i\}_{i=1}^n$, los cuales son $A$-biconjugados. Es decir, $w_i^T A z_j = 0$ si y sólo si $i \neq j$. Dada una matriz no singular $A \in R^{nxn}$, siempre existen

$$Z = [z_1, z_2, ...., z_n]$$

y

$$W = [w_1, w_2, ...., w_n]$$

ortogonales, tales que

$$W^t AZ = D = \begin{pmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_n \end{pmatrix}$$

con $p_i = w_i^t A z_i \neq 0$. Entonces,

$$A^{-1} = Z D^{-1} W^t \tag{1.44}$$

Esta factorización de $A$ es totalmente densa, y por lo tanto su aplicación directa para resolver el sistema lineal (1.1) está descartada.

La idea es diezmar las matrices $Z$ y $W^t$ en el proceso de cálculo de forma que resulten matrices dispersas. Para una descripción más detallada del método ver [BMT96].

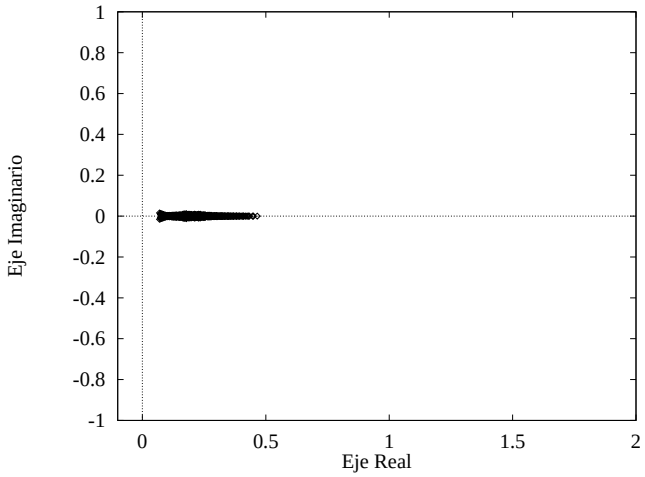



Figura 1.16: Autovalores de ORT-SPAI para $Cc = 10$ diezmando a 1 orden de magnitud.

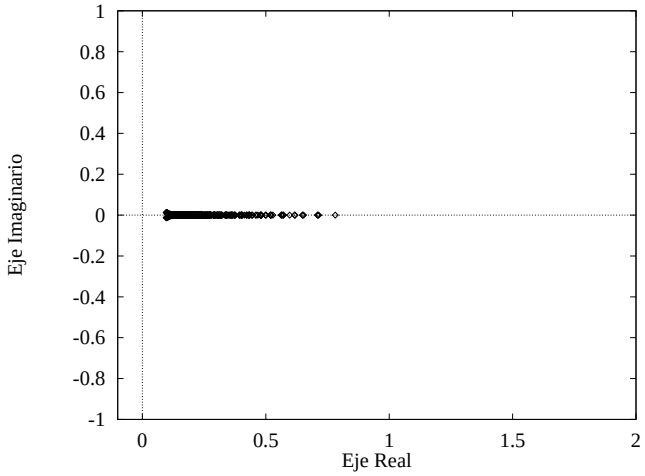



Figura 1.17: Autovalores de ORT-SPAI para $Cc = 10$ diezmando a 2 ordenes de magnitud.

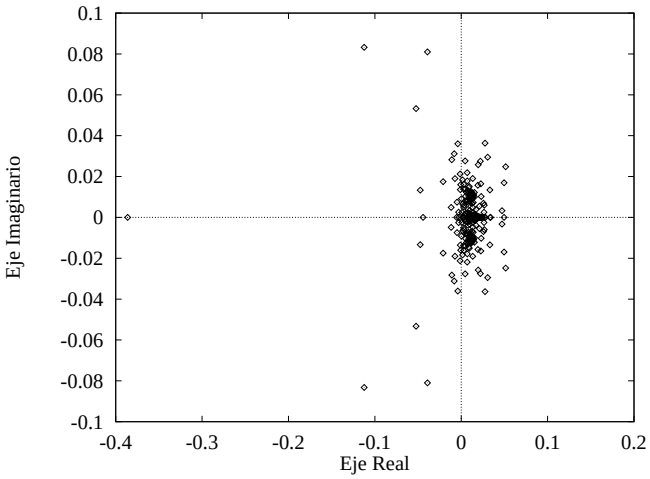



Figura 1.18: Autovalores de ORT-SPAI para $Cc = 1000$ diezmando a 1 orden de magnitud.

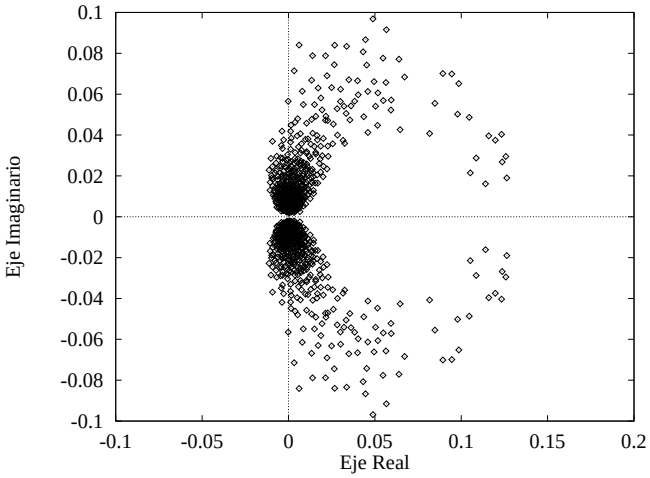



Figura 1.19: Autovalores de ORT-SPAI para $Cc = 1000$ diezmando a 2 ordenes de magnitud.

En este trabajo se realiza el cálculo exacto de $Z$ y $W^t$, y posteriormente se diezman estas matrices a 1 y 2 órdenes de magnitud de su valor máximo. Las matrices resultantes de este diezmado se denotan como $\overline{Z}$ y $\overline{W}^t$, respectivamente. En las figuras 1.26, 1.27, 1.28, 1.29, 1.30, 1.31, 1.32 y 1.33, que están al final de este Capítulo, se muestra la estructura de las matrices $\overline{Z}$ y $\overline{W}^t$. Se puede observar que para obtener matrices con alta dispersidad es necesario diezmar a 1 orden de magnitud. Sin embargo, no está claro que este diezmado sea lo suficientemente bueno para obtener un buen precondicionador para sistemas dominados fuertemente por la convección.

El precondicionador ORT-SPAI queda definido como:

$$M^{-1} = \overline{Z} D^{-1} \overline{W}^t$$

En las figuras 1.16, 1.17, 1.18 y 1.19 se puede observar el espectro de $M^{-1}$ para los diferentes valores de $Cc$ y los diferentes diezmados.

En principio cabe esperar que el cálculo de los ORT-SPAI tenga más problemas de progragación de errores que en el caso de los LSQ-SPAI, ya que los ORT-SPAI requiren un proceso de cálculo iterativo en el cual se realiza una ortogonalización. Incluso calculando de forma exacta las matrices $Z$ y $W^t$ se pierde ortogonalidad a medida que la matriz $A$ está peor condicionada, es decir, cuando la dimensión de $A$ crece o cuando $Cc$ crece. Como contrapartida aproximar las matrices $Z$ y $W^t$ con un determinado umbral, supone tener muchos más números de $A^{-1}$ que si se usa un LSQ-SPAI con el mismo umbral.

Es importante mencionar que estas dos formas artificiales de construir los precondicionadores SPAI da como resultado una cota optimista de una verdadero precondicionador SPAI, ya que el proceso de diezmado se realiza despues de obtener la matriz inversa (LSQ-SPAI) o las matrices ortogonales (ORT-SPAI). En una implementación verdadera de cualquiera de los dos SPAI el proceso de diezmado es parte del proceso de cálculo.

### Comparación de los precondicionadores

Se compara la convergencia de un método iterativo (GMRES(32)) usando los precondicionadores LSQ-SPAI, ORT-SPAI y una factorización incompleta $ILUt$ con $fil = 70$ y $\tau = 10^{-3}$.

Habitualmente se comparan los precondicionadores SPAI con la factorización ILU(0). Sin embargo, esta no es una comparación realista, dado que la factorización ILU(0) no tiene aplicabilidad en casi ningún problema industrial. Basado en la literatura, se decidio usar un precondicionador $ILUt(70, 10^{-3})$ por considerar que sus parametros $fil = 70$ y $\tau = 10^{-3}$ no introducen un excesivo llenado en la factorización.

Las figuras 1.20 y 1.21 muestran la convergencia del GMRES(32) para los precondicionadores descritos y diferentes valores de $Cc$. El valor mostrado como medida de convergencia es el residuo normalizado, es decir

$$\log \frac{\|r\|_2}{\|b\|_2}$$

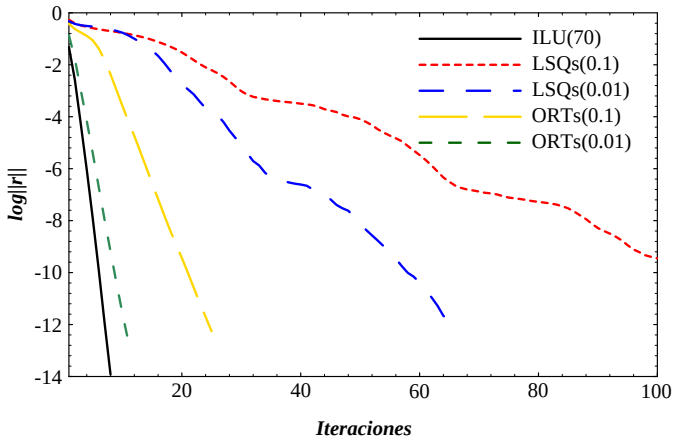



Figura 1.20: Convergencia del GMRES para $Cc = 10$.

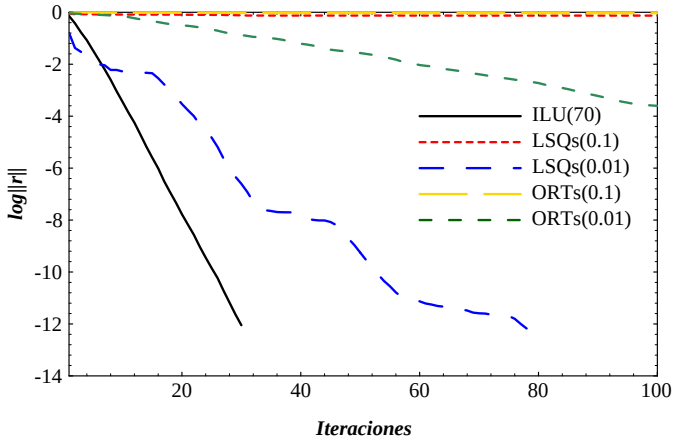



Figura 1.21: Convergencia del GMRES para $Cc = 1000$.

Se pueden observar tres fenómenos fundamentales:

- La factorización $ILUt$ supera en todos los casos a los precondicionadores SPAI. Si se centra el estudio en problemas fuertemente convectivos, el margen de ganancia de la factorización $ILUt$ es muy grande.

- Los precondicionadores ORT-SPAI con un umbral de diezmado lo suficientemente pequeño, superan a los LSQ-SPAI, salvo cuando el valor de $Cc$ es lo suficientemente grande como para que el proceso de ortogonalización sea numéricamente poco estable.

- Para tener precondicionadores SPAI que se acerquen al comportamiento de un precondicionador $ILUt$ es preciso un diezmado a 2 ordenes de magnitud. Lo

que implica una cantidad de cálculo muy superior a la que require la factorización $ILUt$.

Luego de este estudio, y tomando en cuenta que la matriz $A$ es de dimensión pequeña, se puede concluir que los precondicionadores SPAI no son prácticos para problemas industriales dominados fuertemente por la convección.

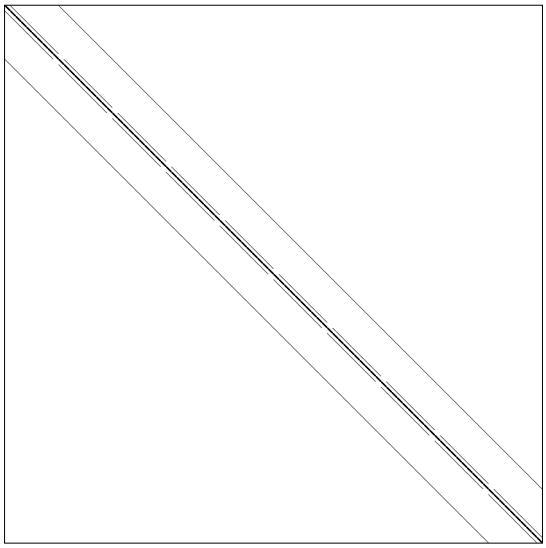


Figura 1.22: Estructura de la matriz $A$.

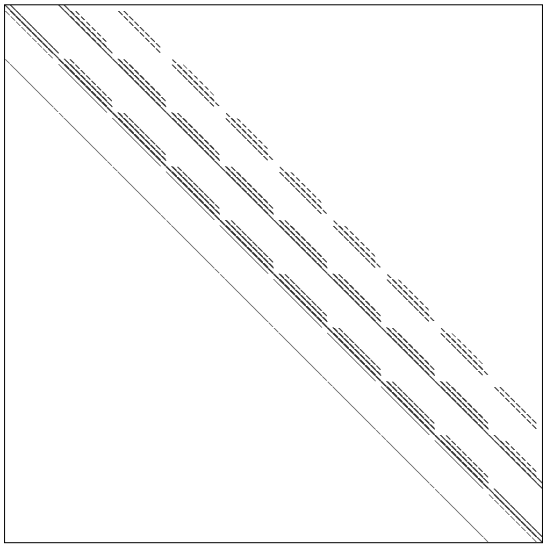


Figura 1.23: Matriz LSQ-SPAI para $Cc = 10$ y diezmado a 1 orden de magnitud.

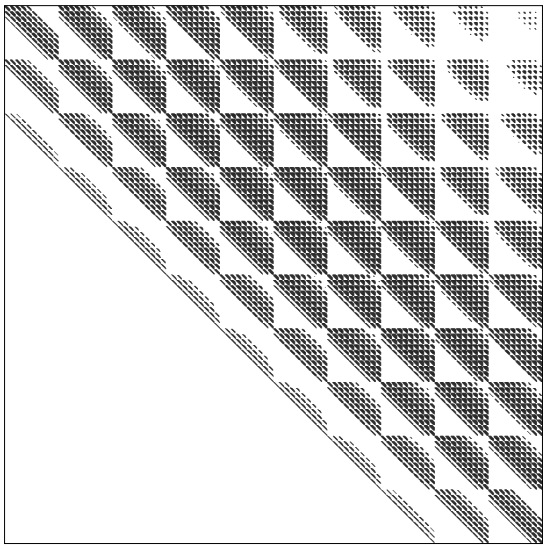


Figura 1.24: Matriz LSQ-SPAI para $Cc = 10$ y diezmado a 2 orden de magnitud.

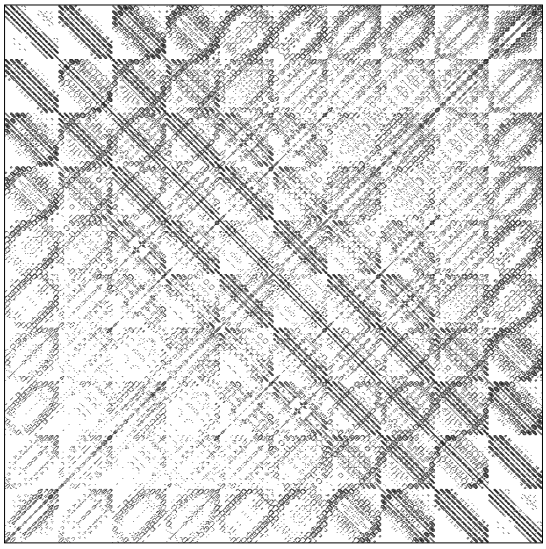


Figura 1.25: Matriz LSQ-SPAI para $Cc = 1000$ y diezmado a 1 orden de magnitud.

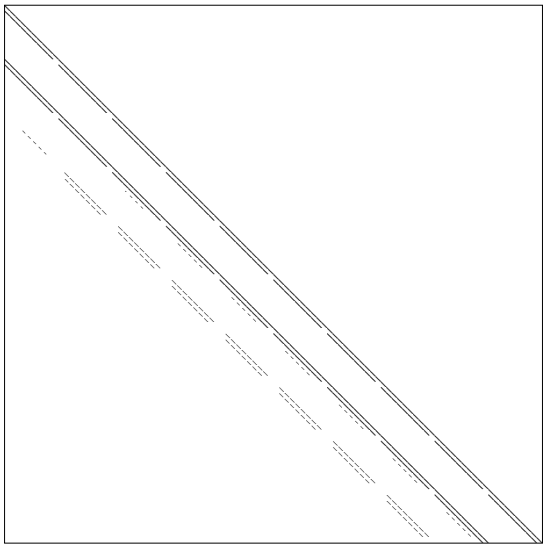


Figura 1.26: Matriz $\overline{Z}$ para $Cc = 10$ y diezmado a 1 orden de magnitud.

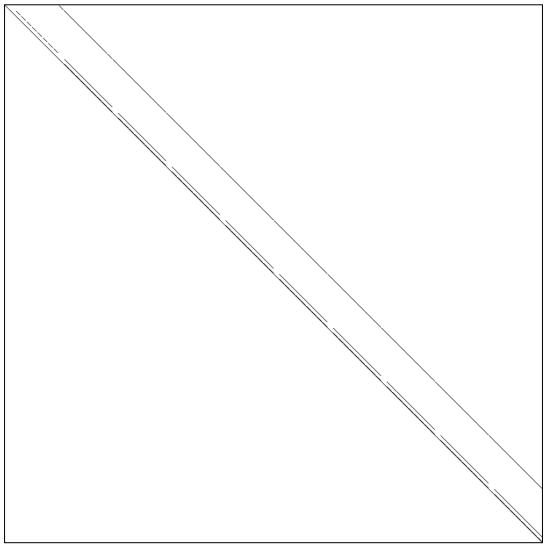


Figura 1.27: Matriz $\overline{W}^t$ para $Cc = 10$ y diezmado a 1 orden de magnitud.

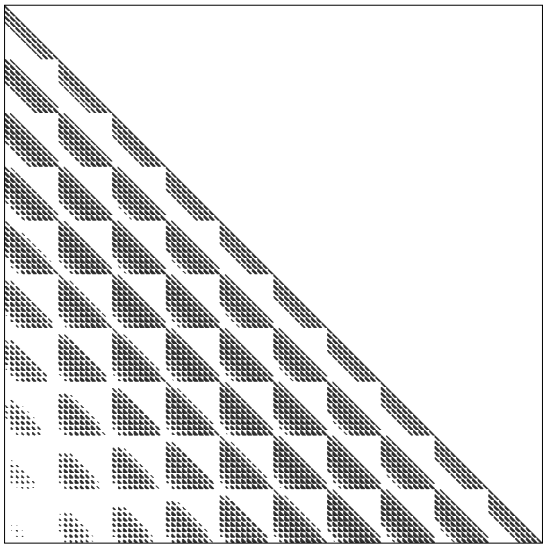


Figura 1.28: Matriz $\overline{Z}$ para $Cc = 10$ y diezmado a 2 orden de magnitud.

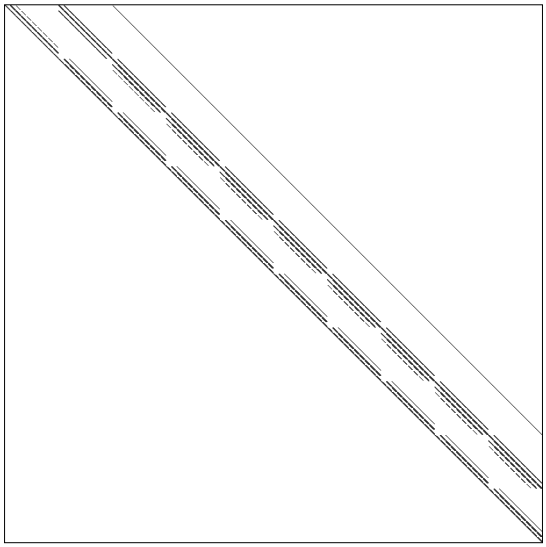


Figura 1.29: Matriz $\overline{W}^t$ para $Cc = 10$ y diezmado a 2 orden de magnitud.

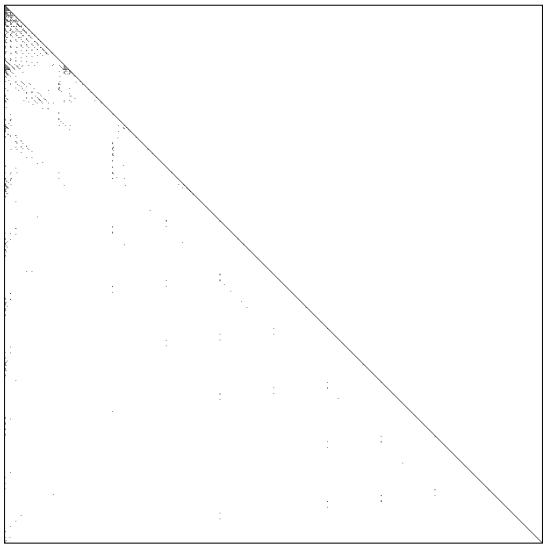


Figura 1.30: Matriz $\overline{Z}$ para $Cc = 1000$ y diezmado a 1 orden de magnitud.

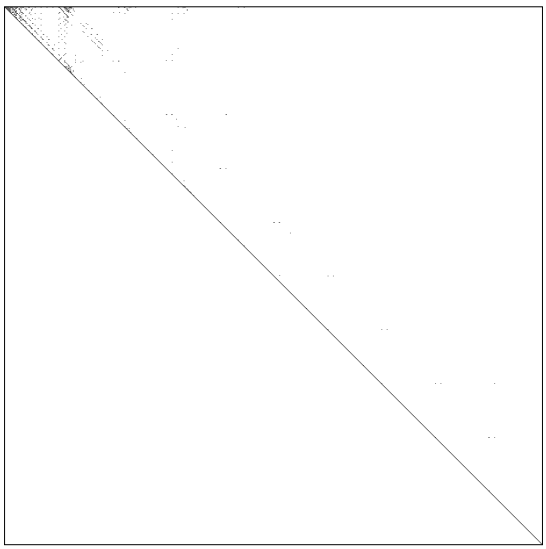


Figura 1.31: Matriz $\overline{W}^t$ para $Cc = 1000$ y diezmado a 1 orden de magnitud.

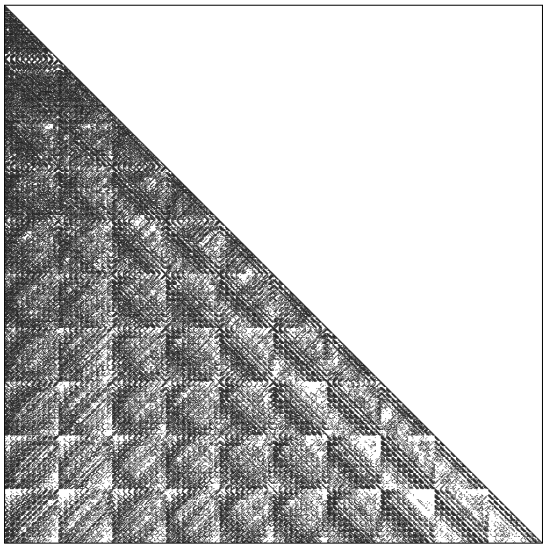


Figura 1.32: Matriz $\overline{Z}$ para $Cc = 1000$ y diezmado a 2 orden de magnitud.

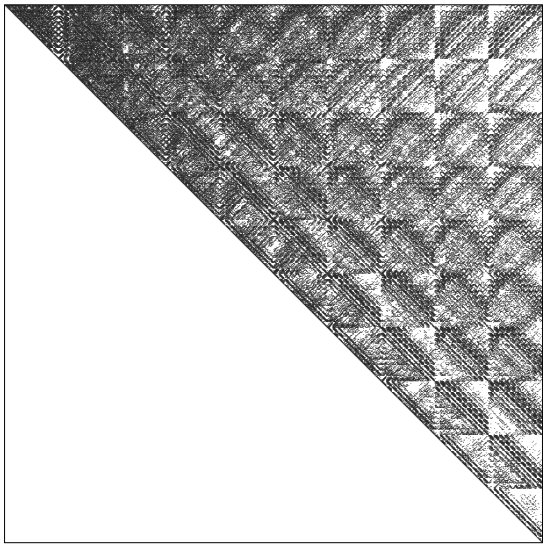


Figura 1.33: Matriz $\overline{W}^t$ para $Cc = 1000$ y diezmado a 2 orden de magnitud.

# Capítulo 2

# Métodos Multinivel

En este capítulo se estudia otra técnica de resolución de sistemas lineales dispersos: los métodos multinivel o multimalla. En primer lugar, se presenta una introducción acerca de los métodos multinivel en general. Seguidamente, se presenta una breve descripción de los métodos multinivel geométricos. Luego, se estudia detalladamente los métodos multinivel algebraicos más importantes, los basados en técnicas de interpolación y los basados en técnicas de agregación. Finalmente, se presenta un estudio experimental de convergencia para el método multinivel algebraico más robusto.

## 2.1 Introducción

En la década de los 60, los matemáticos rusos Fedorenko y Bakhavalov [Fed62] [Bak66] desarrollaron una nueva técnica iterativa de resolución de sistemas lineales provenientes de EDPs. Dicha técnica se basaba en los métodos iterativos clásicos y en explotar información adicional proveniente de discretizaciones más gruesas del mismo problema. A principios de la década de los 70 esta técnica comenzó a popularizarse en occidente gracias a los trabajos de A. Brandt [Bra77], que la denominó método Multinivel o Multimalla. Los métodos multinivel presentan una complejidad en operaciones lineal con el tamaño del sistema, frente a la complejidad cuadrática de los métodos clásicos. Como contrapartida son muy específicos y menos versátiles en su aplicación. Para una recopilación de todos los métodos multinivel ver [Bri87] [Hac85] [McC89] [AV89a] [Rud93] [Tum89] [Yse82].

Un método multinivel consta de los siguientes elementos:

1. Una secuencia de mallas con una matriz asociada a cada malla.
2. Operadores de transferencia entre mallas (Interpolador y Diezmador).

3. Un método iterativo clásico (Gauss-Seidel, Jacobi, SSOR, etc), al que se denomina relajador.

Para dar un explicación detallada del funcionamiento de un método multinivel supondremos que únicamente se tienen dos mallas, denominadas $M^h$ (la malla más fina) y $M^H$ (la malla más gruesa). Se denomina $A_h$ y $A_H$ respectivamente a las matrices provenientes de una discretización en $M^h$ y $M^H$. Las dimensiones de las matrices $A_h$ y $A_H$ son respectivamente $n \times n$ y $N \times N$. El problema es resolver un sistema lineal en la malla fina $M^h$, como el siguiente:

$$A_h u^h = f^h \tag{2.1}$$

donde $u^h, f^h \in \Re^n$, denominaremos $V^h = \Re^n$, y análogamente $V^H = \Re^N$.

Se tienen que construir dos operadores de transferencia entre $V^h$ y $V^H$. El operador de interpolación $I : V^H \to V^h$, y el operador de diezmado $R : V^h \to V^H$. Los métodos multinivel suelen imponer como condición adicional que $R = I^t$. En los métodos multinivel geométricos $R$ e $I$ se construyen de forma que:

$$A_H = R A_h I \tag{2.2}$$

En los métodos multinivel algebraicos (AMG), $A_H$ se define justamente como indica (2.2), una vez que se tengan definidos $R$ e $I$. Otra posibilidad de generar $A_H$ es discretizar la ecuación diferencial parcial sobre la malla gruesa con el mismo método que fue aplicado sobre la malla fina. Para algunas discretizaciones, estas dos opciones son equivalentes.

El método multinivel se basa en el siguiente algoritmo:

1. Relajar $\nu$ veces $A_h u^h = f^h$ en $V^h$, con solución inicial $u_0^h$.
2. Calcular $r^H = R(f^h - A_h u_\nu^h)$.
3. Resolver: $A_H e^H = r^H$ en $V^H$.
4. Corregir la aproximación en la malla fina: $u_\nu^h = u_\nu^h + I e^H$.
5. Relajar $\mu$ veces $A_h u^h = f^h$ con solución inicial $u_\nu^h$.

Si se tienen más de dos mallas anidadas, la idea se puede aplicar de forma recursiva en el paso 3 del algoritmo hasta reducir el problema a una malla suficientemente gruesa. El algoritmo anterior es conocido como V-ciclo. Existen diferentes algoritmos multinivel, dependiendo de la forma de visitar cada uno de los niveles. En la

figura 2.1 y 2.2 se puede observar los tres esquemas existentes, el V-ciclo, el W-ciclo y el F-ciclo. Los puntos negros representan operaciones de relajación o suavizado. Normalmente, el esquema F-ciclo es el más optimo debido a las correcciones frecuentes en cada nivel, pero este esquema es el más costos desde un punto de vista computacional.

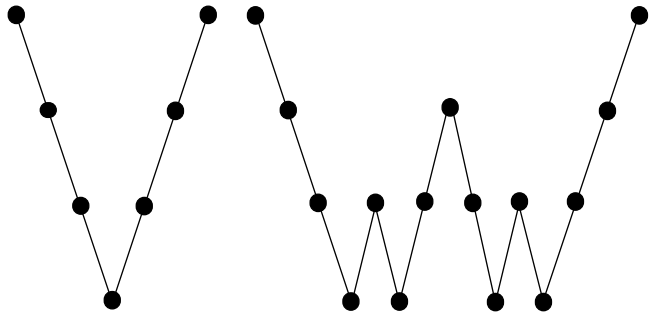


Figura 2.1: V-ciclo y W-ciclo para cuatro niveles.

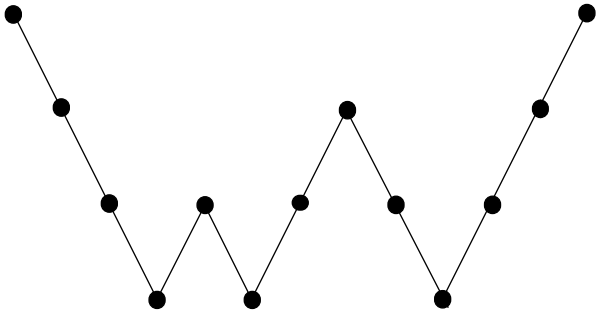


Figura 2.2: F-ciclo para cuatro niveles.

Los relajadores tipo Jacobi o Gauss-Seidel sólo reducen de forma eficaz ciertas componentes del error, en concreto, aquellas componentes asociadas a los autovalores del relajador cercanos a cero. A estas componentes se las denomina frecuencias altas del error porque en el caso de la ecuación de Poisson en un cuadrado, que fue el primer caso estudiado, los autovectores tienen una expresión analítica en forma de función senoidal, y los autovalores cercanos a cero están asociados con los autovectores cuya frecuencia es la más alta. En la figura 2.3 se puede observar el efecto del suavizado después de un paso de relajación.

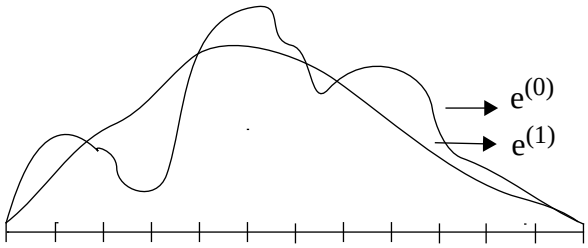



Figura 2.3: Efecto del suavizado.

La clave del éxito de un método multinivel es encontrar los operadores $R$ e $I$ adecuados, de forma que el operador de corrección malla gruesa sea capaz de corregir los errores que el relajador no es capaz de atenuar. De esta forma, únicamente con unas pocas relajaciones se obtiene la solución exacta del sistema. Si se consigue esta propiedad, la complejidad del método en su conjunto será una expresión del tipo:

$$Complejidad = O(n + N^p)$$

con $p = 2$ ó $3$ dependiendo del método usado para resolver el sistema en la malla gruesa. Si se tiene que $N^p \ll n$, se ha obtenido un método de complejidad lineal.

Para entender mejor como converge un método multinivel veamos la relación entre los espacios vectoriales asociados a cada nivel y los diferentes operadores.

Por definición $Im(R) = V^H$. Por la relación de ortogonalidad entre los subespacios de un operador lineal, se tiene que $Ker(R) \perp Im(R^t)$, y por lo tanto $Ker(R) \perp Im(I)$. Es decir, se puede descomponer $V^h$ como $V^h = Im(I) \oplus Ker(R)$, ver figura 2.4. Haciendo unas simples manipulaciones algebraicas, se tiene que, $Ker(RA_h) \perp_{A_h} Im(I)$. Entonces, tenemos que $V^h = Im(I) \oplus Ker(RA_h)$.

Por otro lado, $V^h$ se puede descomponer usando como base los autovectores del relajador. Si se denomina $L$ al subespacio cuya base está formada por los autovectores asociados a autovalores cercanos a 1, y $H$ al subespacio cuya base está formada por los autovectores asociados con autovalores cercanos a 0, se tiene que $V^h = L \oplus H$. Las letras $L$ y $H$ provienen de *Low/High frecuencies*.

En la figura 2.5 se puede observar como la energía del error ($\|e\|$) se atenúa mediante un algoritmo multinivel. En primer lugar las componentes de $e$ asociadas al subespacio $H$ son atenuadas por el relajador. A continuación, las componentes de $e$

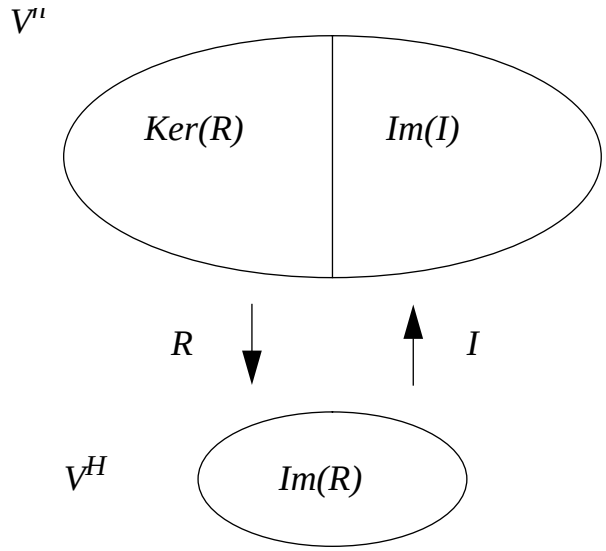



Figura 2.4: Esquema de los operadores de transferencia.

asociadas al subespacio $Im(I)$ son eliminadas por la corrección de la malla gruesa, y así sucesivamente hasta la convergencia.

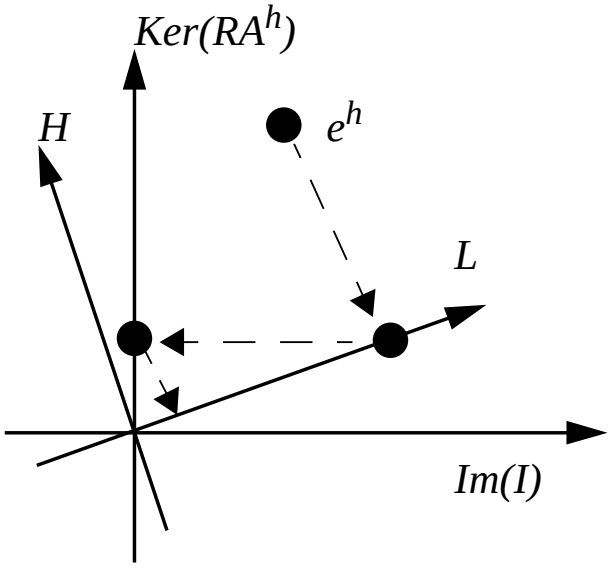
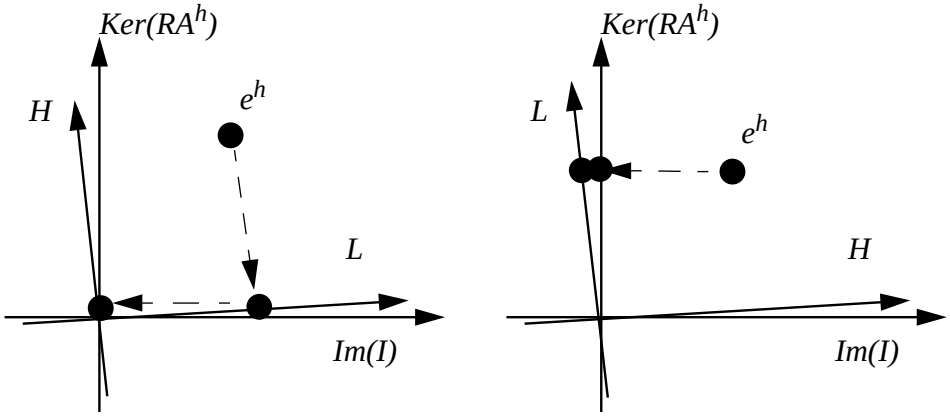



Figura 2.5: Efecto del método multinivel sobre $e^h$.

La convergencia del método multinivel es óptima si los pares ($Ker(RA^h)$, $Im(I)$) y ($H$, $L$) están dispuestos formando un ángulo de $0^o$. Sin embargo, si $L$ es "muy




(a) (b)

Figura 2.6: Caso limite. (a) Bueno. (b) Malo.

paralelo" a $Im(I)$ la convergencia será mala. En la figura 2.6 se puede observar el caso limite para las dos descomposiciones del espacio $V^h$.

Se explica así que la clave de un método multinivel sea encontrar unos operadores $R$ e $I$ que se ajusten adecuadamente al relajador.

En 1989, O. Axelsson [AV89a] [AV89b] desarrolló un nuevo enfoque al estudio de las técnicas multinivel. Su formulación conecta los métodos multinivel con los métodos de descomposición en dominios, y demuestra que los métodos multinivel son un caso particular de descomposición en dominios. En concreto, los métodos multinivel se basan en ordenar los nodos de $M^h$ en dos grupos: nodos nuevos (los que sólo pertenecen a $M^h$) y nodos viejos (los que provienen de $M^H$). Si en primer lugar se enumeran los nodos nuevos y en segundo lugar los nodos viejos, se obtiene una matriz de coeficientes $A_h$ que tiene una estructura 2x2 a bloques como la siguiente:

$$A_h = \begin{bmatrix} A_{nn} & A_{nv} \\ A_{vn} & A_{vv} \end{bmatrix} = \begin{bmatrix} A_{nn} & 0 \\ A_{vn} & S \end{bmatrix} \cdot \begin{bmatrix} I & A_{nn}^{-1} \cdot A_{nv} \\ 0 & I \end{bmatrix}$$

Se puede demostrar que la matriz $S$ es espectralmente equivalente a la matriz $A_H$. Es decir, si se usa $A_H$ como precondicionador de $S$ en cualquier método iterativo, éste convergerá en un número de iteraciones $O(1)$.

Existen dos tipos de métodos multinivel, los geométricos y los algebraicos. A continuación estudiaremos ambos métodos.

## 2.2 Multinivel Geométricos

Los métodos multinivel geométricos trabajan sobre una jerarquia de mallas. Este conjunto de mallas anidadas puede ser obtenido por refinamientos sucesivos a partir de la malla gruesa, o bien por alguna estrategia de diezmado a partir de la malla fina.

Para que un método multinivel sea programable de forma sistematica es preciso que el proceso de engrosamiento o refinamiento sea lo más simple posible. Desafortunadamente, si los coeficientes de la EDP no son constantes e isotrópicos, los errores no son atenuados por el relajador de igual forma en todas direcciones. Esto hace que el proceso de engrosamiento/refinamiento sea particular prácticamente para cada problema.

Ante la dificultad de programar un proceso de engrosamiento/refinamiento especifico para cada caso, se suele optar un mecanismo de engrosamiento/refinamiento uniforme. Entonces, es preciso buscar relajadores adecuados a este esquema uniforme. Los relajadores $ILU$ suelen presentar un comportamiento mejor que los clásicos Gauss-Seidel o Jacobi.

La implementación de eficientes y robustos relajadores para casos de modelos 2D no es difícil. Pero para aplicaciones 3D sobre mallas complejas su realización es muy complicada. Los relajadores $ILU$ pierden muchas propiedas de suavizado en situaciones 3D.

Otra estrategia para mejorar el rendimiento de un método multinivel es usar operadores de interpolación y diezmado no lineales. Sin embargo, esta estrategia sólo logra pequeñas mejoras.

Resumiendo, podriamos decir que los métodos multinivel geometricos tratan de usar esquemas simples para generar la jerarquia de mallas, y buscan operadores de relajación, interpolación y diezmado complejos. Además, en general, las dos descomposiciones del espacio $V^h$ no están orientadas de forma óptima, no es sencillo lograr una complejidad lineal y la implementación de estos métodos requiere modificar todo el software de una simulación completa.

## 2.3 Multinivel Algebraicos (AMG)

El desarrollo de estos métodos comienza en los años ochenta. Una de las motivaciones del AMG fue la observación de que el operador dependiente de la interpolación

y el operador de Galerkin pueden ser derivados directamente desde operaciones matriciales, sin ninguna referencia acerca de las mallas.

Los métodos multinivel algebraicos han sido desarrollados para resolver un sistemas lineal de ecuaciones como una caja negra. En estos métodos el operador de relajación es fijo. El proceso de engrosamiento es ejecutado automáticamente de tal manera de que en el rango de la interpolación se aproximan aquellas componentes del error que no son eficientemente reducidas por el proceso de relajación. Desde un punto de vista teórico, el método ha sido bien entendido y definido en el contexto de $M$-matrices simétricas positivo definidas, aunque, en la práctica el uso no esta restringido a tales casos.

Los métodos multinivel algebraicos sólo necesitan la información que está contenida en la matriz del sistema. Por lo tanto, los AMG son más generales que los geométricos e incluso aplicables a sistemas que tengan mallas muy irregulares o que no provengan de ninguna malla. Sin embargo, en los AMG es necesario desarrollar un fase de inicialización, llamada fase de *setup*, para construir los niveles gruesos y los operadores de transferencia. Este trabajo extra es una razón de que los AMG sean menos eficientes que los métodos multinivel geométricos, si al problema a resolver se le puede aplicar el método geométrico. Lo interesante de los AMG es su relativa robustez y su aplicabilidad en cualquier sistema lineal. Es decir, los AMG proveen una atractiva variante multinivel cuando los métodos multinivel geométricos son complicados de aplicar o cuando estos no puedan ser usados. Además de un método de solución en si mismo, los AMG pueden ser usados como precondicionadores muy eficientes.

Formalmente, un ciclo AMG puede ser descrito de la misma manera que un ciclo multinivel geométrico, excepto que los términos malla, submalla, nodos, etc. pueden ser reemplazados por conjunto de variables, subconjunto de variables, variables, etc. En esta tesis se utilizarán ambos conjuntos de términos de igual manera. Para describir los componentes formales de un AMG se re-escribe (2.1) como:

$$\sum_{j \in V^h} a_{ij}^h u_j^h = f_i^h \quad (i \in V^h) \tag{2.3}$$

donde $V^h$ denota el conjunto de índices $\{1, 2, \ldots, n\}$. Se asume que $A_h$ es una matriz dispersa. Para generar un sistema grueso a partir de (2.3) es necesario hacer una partición del conjunto $V^h$ en dos subconjuntos disjuntos, $V^h = C^h \cup F^h$, donde el subconjunto $C^h$ contiene las variables que están en el nivel grueso (*coarse nodes*) y el subconjuntos $F^h$ (*fine nodes*) es el complemento de $C^h$. Asumiendo que tal

partición está dada y definiendo $V^H = C^h$, el sistema grueso será,

$$A_H u^H = f^H \quad \text{o} \quad \sum_{l \in \Omega^H} a_{kl}^H u_l^H = f_k^H \quad (k \in V^H), \tag{2.4}$$

donde $A_H = RA_hI$, $R : V^h \longrightarrow V^H$ es el operador de diezmado e $I : V^H \longrightarrow V^h$ es el operador de interpolación. Se impone $R = I^t$. Finalmente, como en cualquier método multinivel, se necesita un proceso de suavizado $S_h$. Un paso de este proceso de suavizado es de la forma

$$u^h \longrightarrow \bar{u}^h \quad \text{donde} \quad \bar{u}^h = S_h u^h + (I_h - S_h) A_h^{-1} f^h \tag{2.5}$$

($I_h$ denota el operador identidad). Consecuentemente, el error $e^h = u_*^h - u^h$ ($u_*^h$ denota la solución exacta de (2.3)) es transformado acorde a

$$e^h \longrightarrow \bar{e}^h \quad \text{donde} \quad \bar{e}^h = S_h e^h. \tag{2.6}$$

Como se ha mencionado con anterioridad, el AMG emplea un proceso de suavizado sencillo, por ejemplo, relajaciones Gauss-Seidel ($S_h = I_h - Q_h^{-1} A_h$, donde $Q_h$ es la parte triangular inferior de $A_h$, incluyendo la diagonal) o relajación $w$-Jacobi ($S_h = I_h - wD_h^{-1} A_h$, donde $D_h = diag(A_h)$). Es claro que, a menos que $A$ sea diagonal dominante, el uso de tales métodos requiere implícitamente hipótesis adicionales sobre $A_h$.

El sistema grueso (2.4) formalmente es equivalente a las ecuaciones de corrección malla gruesa en el multinivel geométrico. En particular, $f^H$ y $u^H$ corresponden a residuos y correcciones, respectivamente. Para ser más precisos, un paso de corrección de dos niveles está definido como

$$u_n^h = u_v^h + Ie^H \quad \text{donde} \quad A_H e^H = Rr_v^h = R(f^h - A_h u_v^h), \tag{2.7}$$

donde los subíndices $n$ y $v$ representan vectores *nuevos* y *viejos*, respectivamente. Para los correspondientes errores, esto significa

$$e_n^h = K_{h,H} e_v^h \quad \text{con} \quad K_{h,H} = I_h - IA_H^{-1} RA_h, \tag{2.8}$$

donde $K_{h,H}$ es el llamado operador de corrección malla gruesa. Consecuentemente, la reducción del error, aplicando un paso de iteración completa de dos niveles, está dado por:

$$e_n^h = M_{h,H} e_v^h \quad \text{con} \quad M_{h,H}(v_1, v_2) = S_h^{v_2} K_{h,H} S_h^{v_1} \tag{2.9}$$

donde $v_1$ y $v_2$ representan pasos de pre y postsuavizado, respectivamente.

La partición $C/F$ y los operadores de transferencia $I$ y $R$ deben ser explícitamente construidos. La construcción de estas componentes es la tarea más importante de un AMG. Estas componentes deben ser construidas tal que exista una eficiente relación entre el proceso de suavizado y el operador de corrección malla gruesa. Esta relación depende del sentido algebraico de los errores suaves bajo cierto proceso de suavizado. Sin embargo, hasta los momentos, la interpretación algebraica de los errores de baja frecuencia para los relajadores como el Gauss-Seidel o el Jacobi está sólo bien estudiada para $M$-matrices simétricas positivas definidas con diagonal dominancia débil, ver [Ruge87]. Para estos casos la construcción de los espacios gruesos se basa en caracterizar sobre el grafo de la matriz como se aproximan estos errores, los cuales están caracterizados por la siguiente expresión:

$$\sum_{j \neq i} \frac{|a_{ij}|}{a_{ii}} \frac{(e_i - e_j)^2}{{e_i}^2} << 1 \tag{2.10}$$

donde $a_{ij}$ son los coeficientes de la matriz y $e_i$, $e_j$ son componentes del error. Entonces, para que se cumpla (2.10) la diagonal dominancia de la matriz debe ser significativa.

De igual manera, es importante que la partición y los operadores de transferencia sean tales que $A_H$ sea razonablemente dispersa y mucho más pequeña que $A_h$.

Los AMG están divididos en dos grupos: los métodos basados en técnicas de interpolación y los métodos basados en técnicas de agregación. La diferencia entre estos métodos está en la forma como se construyen la partición $C/F$ y los operadores de transferencia. A continuación se describirá los métodos más relevantes de ambos grupos.

### 2.3.1 Métodos de Interpolación

En 1987, J. Ruge y K. Stuben [Ruge87] proponen un AMG para $M$-matrices simétricas positivas definidas con diagonal dominante débil, usando el método de Jacobi o Gauss-Seidel como relajador. Aquí, la interpolación se basa en el concepto de conexión fuerte entre nodos. En ese trabajo se demuestra la convergencia y la complejidad lineal del método. A continuación se describirá este método con algunas variantes que fueron propuestas en posteriores trabajos [KS97] [Stu99].

#### Engrosamiento

En este método el objetivo es construir los subconjntos $C$ y $F$ de manera que los $F$-nodos sean obtenidos interpolando los $C$-nodos. Si este objetivo es tomado en

cuenta, la interpolación tiende a ser mucho mejor y como resultado la convergencia es mucho más rápida. Sin embargo, no existe una prueba algebraica de esto.

En general, la estrategia de engrosamiento es aplicada entre nodos vecinos. Sin embargo, existen estrategias de engrosamiento en donde esto no siempre sucede. Tales estrategias permiten reducir drásticamente el costo computacional de la fase *setup*, ciclo, la complejidad del operador de nivel grueso y el requerimiento de memoria. Claramente, estos beneficios tienen la desventaja de que la velocidad de convergencia se reduce significativamente, ya que la relación entre los operadores de transferancia y el relajador desmejora.

**Engrosamiento estandar:** En este método se definen los siguientes conjuntos:

$$N_i = \{j \in V : j \neq i, a_{ij} \neq 0\} \tag{2.11}$$

$$S_i = \{j \in N_i : -a_{ij} \geq \eta \; max_{l \neq i}\{-a_{il}\}\} \tag{2.12}$$

El conjunto $S_i$ es llamado conjunto de conexiones fuertes del nodo $i$. Comúnmente, $\eta$ es 0.25.

Para M-matrices, la mayoría de los acoplos fuertes son negativos. Pero esto no se puede asumir de forma general, ya que pueden existir matrices que contengan elementos con conexiones fuertes positivas. Esto se discutirá posteriormente.

La subconjuntos $C$ y $F$ resultantes deberán ser tal que todos los $F$-nodos tengan una cantidad sustancial de conectividad fuerte negativa hacia los $C$-nodos vecinos. En otras palabras, el engrosamiento es hecho en la dirección en la cual el error algebraicamente suave cambia lentamente. Notese que todas las conexiones positivas son observadas como débiles.

Los nodos de interpolación $C_i$ se definen como,

$$C_i = C \subset S_i$$

Los nodos no interpolantes $D_i$ son divididos en dos subconjuntos: uno de conexiones fuertes, $(s)$, y otro de conexiones débiles, $(w)$. Estos se definen de la siguiente manera:

$$D_i = N_i - C_i \quad D_i^s = D_i \cap S_i \quad D_i^w = D_i - S_i$$

El algoritmo de engrosamiento tiene dos etapas. En la primera, se realiza una elección rápida de los $C$-nodos. Los $C$-nodos son seleccionados según el siguiente

criterio: $C$ deberá ser un subconjunto maximal de todos los nodos con la propiedad de que dos $C$-nodos no están fuertemente conectados entre sí. En la segunda etapa, los $F$-nodos tentativos son revisados a fin de que cumplan un segundo criterio: para cada nodo $i$ en $F$, cada nodo $j$ en $S_i$ deberá estar en $C$ o deberá estar fuertemente conectado hacia al menos un nodo en $C_i$. Aquellos $F$-nodos que no cumplan este criterio son etiquetados como $C$-nodos.

**Engrosamiento agresivo:** Este tipo de engrosamiento se utiliza cuando el engrosamiento estandar pueda causar una complejidad relativamente alta. Es decir, que el requerimiento de memoria sea excesivo debido a los operadores Galerkin de los niveles gruesos. En estos casos, la matriz de Galerkin es mucho más densa que la matriz original.

Para permitir un engrosamiento agresivo, se extiende la definición de conectividad fuerte hacia nodos que no están acoplados directamente. En este caso, se introduce el concepto de $n$-conexiones fuertes de largo rango: un nodo $i$ se dice que está fuertemente $n$-conectado a un nodo $j$ a través de un camino de longitud $l$ si existe una secuencia de nodos $i_0, i_1, \ldots, i_l$ con $i = i_0$ y $j = i_l$ tal que $i_{k+1} \in S_{i_k}$. Dados los valores $p \geq 1$ y $l \geq 1$, entonces una nodo $i$ está fuertemente $n$-conectado a un nodo $j$ si existe un camino $p$ de longitud $\leq l$, se escribe $(p, l)$ tal que $i$ está fuertemente $n$-conectado a $j$ a través de cada uno de estos caminos. El conjunto $S_i$ es modificado como sigue:

$$S_i^{p,l} = \{j \in \Omega : \ i \ esta \ fuertemente \ n - conectado \ a \ j\}$$

Desde un punto de vista práctico, generalmente no es costoso explotar la fuerte $n$-conectividad. De hecho, los caso $p = 2$, $l = 2$ y $p = 1$, $l = 2$ son los mas útiles. Sin embargo, es muy costoso usar el engrosamiento agresivo en mas de un nivel. Usualmente el engrosamiento estandar es suficientemente eficiente.

**Conexiones fuertes positivas:** En los dos tipos de engrosamiento ya mencionados la construcción de los subconjuntos $C$ y $F$ está basada en acoplos negativos. Se está suponiendo que existen acoplos positivos relativamente pequeños, y esto puede ser ignorado en el proceso de engrosamiento y en la interpolación. Sin embargo, esto no puede ser asumido en todos los casos. Un proceso de construcción de los subconjuntos $C$ y $F$ deberá asegurar que para todos los $F$-nodos, los cuales tienen acoplos fuertes negativos y positivos, un número mínimo de ambos acoplos deberá estar representado en $C$. Construir tales subconjuntos dentro de un sólo paso es relativamente complicado, ya que para una gran variedad de situaciones la mayoría de las conexiones fuertes son negativas. En [Stu99] se propone una alternativa sencilla:

Después de aplicar alguno de los procesos de engrosamiento descritos anteriormente, se revisan todos los $F$-nodos o aquellos nodos en donde exista un acoplo positivo de $F$ hacia $F$. Se puede verificar esto con la condición

$$a_{ij} \geq \rho \; max_{k \neq i} |a_{ik}|, \quad j \neq i \tag{2.13}$$

donde, $\rho$ es una tolerancia que se debe ajustar. Usualmente, $\rho = 0.5$.

Si tales $j$ existen, todos los nodos $j$ que satisfagan (2.13) deberán adicionarse al conjunto $S_i$ (esto afecta la interpolación) y el nodo que tiene el acoplo más fuerte deberá ser redefinido como un $C$-nodo. Claramente, este proceso de actualización de los subconjuntos $C$ y $F$ es eficaz sólo si no existen muchas conexiones fuertes positivas.

### Interpolación

Se asume que los subconjuntos $C$ y $F$ han sido construidos mediante un engrosamiento estandar o agresivo. En el primer caso, la interpolación usada es la directa o la estandar. En el segundo caso, la interpolación usada es la multipaso. En ambos casos, la interpolación puede ser mejorada mediante pasos adicionales de relajación.

**Interpolación directa:** Para cada $i \in F$, se define el conjunto de nodos de interpolación $P_i = C_i^s (C_i^s = C \cap S_i)$ y se aproxima

$$a_{ii} e_i + \sum_{j \in N_i} a_{ij} ej = 0 \Longrightarrow a_{ii} e_i + \alpha_i \sum_{k \in P_i} a_{ik}^- e_k + \beta_i \sum_{k \in P_i} a_{ik}^+ e_k = 0 \tag{2.14}$$

con

$$\alpha_i = \frac{\sum_{j \in N_i} a_{ij}^-}{\sum_{k \in P_i} a_{ik}^-} \quad \text{y} \quad \beta_i = \frac{\sum_{j \in N_i} a_{ij}^+}{\sum_{k \in P_i} a_{ik}^+} \tag{2.15}$$

Esto inmediatamente conduce a la siguiente fórmula de interpolación:

$$e_i = \sum_{k \in P_i} w_{ik} e_k \quad \text{con} \quad w_{ik} = \begin{cases} -\alpha_i a_{ik}/a_{ii} & (k \in P_i^-) \\ -\beta_i a_{ik}/a_{ii} & (k \in P_i^+) \end{cases} \tag{2.16}$$

**Interpolación estandar:** Se puede modificar la interpolación directa para que las $F$-conexiones fuertes sean (indirectamente) incluidas en la interplación de cada $i \in F$. Esto se obtiene inmediatamente aproximando la $i$-ésima ecuación (lado derecho de la ecuación (2.14)). Primero, aproximadamente, se eliminan todos los $e_j$ ($j \in F_i^s = F \cap S_i$) por medio de la correspondiente $j$-ésima ecuación. Específicamente, para cada $j \in F_i^s$, se sustituye

$$e_j \longrightarrow -\sum_{k \in N_j} a_{jk} e_k / a_{jj} \tag{2.17}$$

en la nueva ecuación para $e_i$,

$$\hat{a}_{ii} e_i + \sum_{j \in \hat{N}_i} \hat{a}_{ij} e_j = 0 \quad \text{con} \quad \hat{N}_i = \{j \neq i : \hat{a}_{ij} \neq 0\} \tag{2.18}$$

Definiendo $P_i$ como la unión entre $C_i$ y todos los $C_j$ ($j \in F_i^s$), se define la interpolación exactamente como en (2.14)-(2.15) con todas las $a$ sustituidas por $\hat{a}$ y $N_i$ sustituido por $\hat{N}_i$.

Esta modificación usualmente mejora la calidad de la interpolación. La principal razón es que el tipo de aproximación (2.14), si se aplica a la ecuación (2.18) introduce menos error. Sin embargo, el objetivo principal es tener $F$-nodos como nodos de interpolación.

**Interpolación multipaso:** Este método de interpolación se aplica cuando los subconjuntos $C$ y $F$ son construidos mediante un engrosamiento agresivo. Esta interpolación se realiza en varios pasos, usando interpolación directa cuando sea posible y, para el resto de los nodos, usar las fórmulas de interpolación en las vecindades de los F-nodos. Cada paso individual es como sigue:

1. Usar interpolación directa para derivar las fórmulas para cada todo $i \in F$ para los cuales $C_i \neq \emptyset$ y definir el conjunto $F^*$ que contiene todas esas variables. Si $F^* = F$ parar, en otro caso seguir.

2. Para toda $i \in F \backslash F^*$ para el cual $S_i \cap F^* \neq \emptyset$ hacer lo siguiente: Tomar la $i$-ésima ecuación (lado derecho en (2.14)) y, para toda $j \in S_i \cap F^*$, sustituir

$$e_j \longrightarrow \sum_{k \in P_j} w_{jk} e_k$$

   siendo la nueva ecuación (2.18) para $e_i$. Definiendo el conjunto de variables de interpolación $P_i$ como la unión de toda $P_j$ para $j \in S_i \cap F^*$, se obtiene una fórmula de interplación como en el caso de la interpolación estandar. Si todas las variables $i$ han sido procesadas, se actualiza $F^*$ con todas las variables que han sido obtenidas en la fórmula de interpolación durante este paso.

3. Si $F^* = F$ parar. En otro caso, ir al paso 2.

**Interpolación de Jacobi:** Dada cualquiera de las fórmulas de interpolación que se han señalado anteriormente, se puede obtener un mejora adicional si a porteriori se aplica una relajación tipo Jacobi. Específicamente, una iteración de esta relajación procede como sigue: Para todas las variables $i \in F$, tomar la $i$-ésima ecuación (lado dereho de (2.14)) y, para toda $j \in F_i$, sustituir

$$e_j \longrightarrow \sum_{k \in P_j} w_{jk}^{(\mu-1)} e_k$$

la cual es la nueva ecuación (2.18) para $e_i$. Definiendo el conjunto de variables de interpolación $P_i$ como la unión de $C_i$ y toda $P_j$ para $j \in F_i$, la $\mu$-formula de interplación es obtenida como en el caso de la interpolación estandar.

Sólo uno o dos pasos de Jacobi son prácticos. Dependiendo de la situación, la relajación de la interpolación puede mejorar la convergencia considerablemente. Esta mejora del operador de interpolación fue propuesta en 1997 por Krechel y Stuben [KS97].

**Truncamiento de la interpolación:** Cuando se utiliza el engrosamiento estandar o la interpolación de Jacobi, el conjunto $P_i$ de variables de inerpolación tiende a ser muy grande. Esto es particularmente cierto para la interpolación de Jacobi ya que en cada paso de relajación se introducen adicionales $C$-variables que son usadas en la interpolación. En consecuencia, si un paso de Jacobi es aplicado en cada nivel AMG los operadores de Galerkin resultantes crecen sustancialmente hacia los niveles gruesos. Este proceso, sin un truncamiento razonable, genera un costo computacional muy alto. Por lo tanto, antes de calcular el operador de Galerkin del nivel grueso se tiene que truncar el operador de interpolación ignorando todas las conexiones interpolatorias las cuales sean mucho más pequeñas que la más grande (en valor absoluto) por un factor de truncamiento. Luego, se tiene que reescalar las variables restantes para que la suma total por filas, del operador de interpolación, no cambie. En la práctica, el factor de truncamiento es, usualmente, 0.2.

Luego de haber construido el operador de interpolación $I$ usando cualquiera de los métodos anteriores, se obtiene el operador de diezmado $R$, que usualmente es $R = I^t$, si $A_h$ es una matriz simétrica. Para problemas no simétricos $R$ es la traspuesta de un interpolador diferente, correspondiente a $A_h^T$. Sin embargo, el uso de $R = I^t$, donde $I$ es construido desde $A_h$, en problemas no simétricos no degrada mucho el rendimiento de un AMG. Luego, $A_H = RA_hI$.

Aunque se ha mencionado que en los AMG el proceso de relajación o suavizado es fijo, en este método los autores han presentado algunas variantes en el relajador,

pero siempre basado en relajadores tipo Gauss-Seidel o Jacobi. Hasta los momentos han utilizado $F$-relajación (relajar sólo las $F$-variables), $CF$-relajación (relajar primero las $C$-variables y luego las $F$-variables) y relajación sin orden específico de variables. También han propuesto combinaciones entre estas. Sin embargo, la $CF$-relajación Gauss-Seidel es muy eficiente en la práctica. En particular, para problemas positivos definidos usualmente es más eficiente que la relajación Gauss-Seidel sin un orden específico de variables. La $CF$-relajación está relacionada con la relajación *red-black* en el método multinivel geométrico.

Los resultados que presentan Ruge y Stuben, muestran el buen rendimiento de su AMG para diferentes problemas, incluso para $M$-matrices no simétricas pero siempre diagonal dominante débil. Una explicación más detallada de este método revisar [RS87] [Stu99].

En 1991, W. Huang [Hua91] se apoya en los resultados de Ruge y Stuben para demostrar la convergencia del AMG para matrices simétricas positivas definidas con diagonal dominante débil. En 1994 Reusken [Reu94] propone un método similar al de Ruge y Stuben basado en una aproximación al complemento de Schur. En este método, la matriz $A_h$ es particionada en cuatro bloques acorde a alguna estrategia de etiquetado $F/C$ de los nodos. Luego, una matriz $M$ es construida de la siguiente manera:

$$A_h = \begin{bmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{bmatrix} \rightarrow \begin{bmatrix} M_{FF} & M_{FC} \\ A_{CF} & A_{CC} \end{bmatrix} = M$$

Los bloques $M_{FF}$ y $M_{FC}$ están dados por:

$$M_{FF} = diag(A_{FF})$$

$$m_{ij} = a_{ij} + \sum_{k \in D_i} \frac{a_{ik}}{|C_i \cap N_k|}$$

$M$ es invertida exactamente utilizando una eliminación a bloques:

$$M^{-1} = \begin{bmatrix} I_F & -M_{FF}^{-1} M_{FC} \\ 0 & I_C \end{bmatrix} \begin{bmatrix} M_{FF}^{-1} & 0 \\ 0 & A_S^{-1} \end{bmatrix} \begin{bmatrix} I_F & 0 \\ -A_{CF} M_{FF}^{-1} & I_C \end{bmatrix}$$

$$A_S = A_{CC} - A_{CF} M_{FF}^{-1} M_{FC}$$

Entonces, los operadores de interpolación $I$, diezmado $R$ y la matriz $A_H$ están dados por:

$$I = \begin{bmatrix} -M_{FF}^{-1} M_{FC} \\ I_C \end{bmatrix}$$

$$R = \begin{bmatrix} -A_{CF} M_{FF}^{-1} & I_C \end{bmatrix}$$

$$A_H = RMI = A_{CC} - A_{CF} M_{FF}^{-1} M_{FC}$$

En 1997, Wagner [WKW91] presenta una versión modificada del método de Reusken. En esta versión se resuelve un sistema lineal modificado $Mx = Ff$ con una simple eliminación a bloques, como en el método de Reusken. Aquí, los coeficientes de $M$ son definidos de forma diferente. Ambas propuestas trabajan bien para matrices simétricas positivas definidas con diagonal dominante débil.

### 2.3.2 Métodos de Agregación

Existen otros tipos de AMG basados en el concepto de agregación o aglomeración. La principal diferencia entre estos métodos y los métodos de interpolación es la construcción de los operadores de transferencia y las mallas gruesas. En estos métodos, se pierde la noción de interpolación clásica, ya que un nodo no es eliminado en una fase de engrosamiento. Aquí, la idea es agrupar nodos utilizando algún criterio de agrupamiento con el objetivo de generar un cubrimiento del conjunto inicial de nodos. Por lo tanto, un nodo en la malla gruesa generada por agregados no permite identificar de forma simple nodos de la malla fina, al contrario de los métodos de interpolación en donde cada nodo de la malla gruesa tiene asociado de forma directa nodos de la malla fina.

En 1995, D. Braess [Bra95] propone un método de agregación que agrupa varios nodos en grupos de uno a cuatro nodos. Los grupos son construidos en dos pasos. En el primer paso, se agrupan parejas de nodos fuertemente conectados. El criterio usado para definir los conjuntos de conexiones fuertes es: dos nodos $i$ y $j$ están fuertemente conectados si

$$\frac{a_{ij}^2}{a_{ii} a_{jj}} \geq \theta, \quad \text{donde} \quad \theta \in (0, 1]. \tag{2.19}$$

En el segundo paso, se agrupan parejas formadas en el paso anterior. Para dar una idea del método supongamos que se han agrupados los nodos en $j$ grupos o agregados. El operador de interpolación $I$ es definido como:

$$I_{ij} = \begin{cases} 1 & i \in G_j \\ 0 & otro \ \ caso \end{cases}$$

Luego, $R = I^t$ y $A_H = \frac{1}{c} R A_h I$. Donde $c$ es una constante.

La estimación del valor de $c$ fue hecha estudiando la ecuación de Poisson en 2D y 3D, asumiendo una malla uniforme y que los agregados son construidos por bloques de nodos vecinos 2x2 y 2x2x2, respectivamente. En casos de problemas más generales y diferentes mallas el valor de $c$ no es mayor que 2. Sin embargo, se ha demostrado en [Bra95] que $c = 1.8$ lleva a una mejora sustancial de la convergencia del V-ciclo para varios tipos de problemas, si el ciclo es usado como precondicionador del CG y si el número de niveles es fijo (en [Bra95] siempre son usados cuatro niveles). El suavizado es hecho usando relajaciones tipo SSOR y los resultados de sus experimentos son comparados con el CG precondicionado con un SSOR.

Claramente, la robustez y eficiencia de esta aproximación está limitada ya que un óptimo valor de $c$ depende de varios aspectos, tales como: el tipo de problema, el tipo de malla y, en particular, del tamaño de los agregados. Para una revisión más detallada de este método revisar [Bra95].

En 1996, P. Vanek, J. Mandel y M. Brezina [VMB96] desarrollan un AMG basado en interpolación por suavizado de agregados. Ellos utilizan su algoritmo para resolver problemas elípticos de segundo y cuarto orden. Al igual que el método de Braess, sólo se necesita la matriz del sistema. Ellos utilizan una técnica basada en conexiones fuertes para construir los grupos o agregados. De la misma manera que el método anterior la conexión fuerte entre nodos se define como en (2.19). La fase de construcción de los agregados consta de tres pasos. En el paso 1, se selecciona como una inicial aproximación del cubrimiento a conjuntos disjuntos de nodos fuertemente conectados. En el paso 2, los nodos restantes son adicionados hacia los conjuntos del cubrimiento para los cuales estos nodos estén fuertemente conectados, si tales conjuntos existen. Finalmente, en el paso 3, los nodos que aun no hayan sido agrupados son colocados dentro de agregados que están compuestos de subconjuntos de nodos vecinos fuertemente conectados. Para dar una idea del método supongamos que se han agrupados los nodos en $j$ grupos o agregados. Entonces se define un interpolador tentativo como:

$$P_{ij} = \begin{cases} 1 & i \in G_j \\ 0 & otro \ \ caso \end{cases}$$

Luego, este interpolador es mejorado utilizando un suavizado para obtener el interpolador $I$. Ellos utilizan el método de Jacobi como suavizador y obtienen:

$$I = (I - wD^{-1}A^F)P$$

donde $A^F = (a_{ij})^F$ es la matriz original filtrada y está dada por:

$$a_{ij}^F = \left\{ \begin{array}{cc} a_{ij} & j \in N_i(\epsilon) \\ 0 & otro \ \ caso \end{array} \right\}; (i \neq j)$$

$$a_{ii}^F = a_{ii} - \sum_{j=1, j \neq i} (a_{ij} - a_{ij}^F)$$

donde $N_i$ es el conjunto de conexiones fuertes del nodo $i$ y $D$ denota la diagonal de $A^F$. Es decir, dado algun vector $e^H$ en el nivel grueso, $e^h = Ie^H$ es definido aplicando un paso de relajación $w$-Jacobi al sistema de ecuaciones homogéneo $A^F v^h = 0$ con aproximación inicial $Ie^H$. Luego, $R = I^t$ y $A_H = RA_hI$. Para una explicación más detallada de este método revisar [Vane96].

En 1997, F. Kickinger [Kic97] propone un AMG en donde la estrategia de engrosamiento es independiente de las conexiones fuertes. Esto significa que su método, al igual que los anteriores, está basado en el grafo de la matriz únicamente. La estrategia de engrosamiento es agresiva y rápida. Esta se basa en un coloreado del grafo de la matriz de manera que un nodo etiquetado como grueso tendrá todos sus vecinos etiquetados como finos. En la figura 2.7 se muestra un ejemplo de esta estrategia.

En la figura 2.7.(a) se muestra el grafo original de la matriz. En la figura 2.7.(b) se puede ver como queda el grafo luego de aplicar la estrategia de coloreado. Los nodos gruesos están coloreados en negro y su nueva numeración también.

La principal propiedad de este método es la rápidez para construitr los conjuntos $C$ y $F$, de forma agresiva. Además, se obtiene una secuencia anidada de grafos o mallas gruesas correspondientes a la jeraquía de la malla dada.

Una vez que se ha especificado la malla o grafo grueso, o lo que es equivalente, se han construido los conjuntos $C$ y $F$, se obtiene el operador de interpolación, $I$.

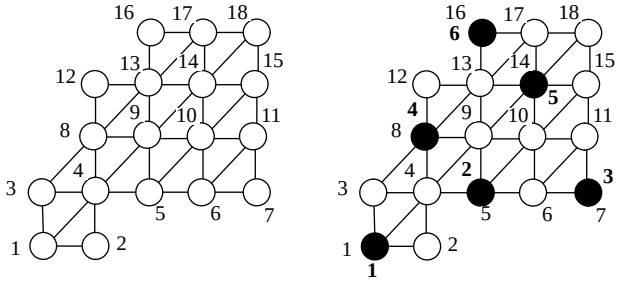



Figura 2.7: (a) Grafo original. (b) Grafo resultante después del coloreado.

El autor presenta dos estrategias para construir este operador:

Estrategia 1:

$$I = (a_{ij}) := \begin{cases} a_{ij} = 1 & si\ j \in C \\ a_{ij} = 0 & otro\ caso \end{cases}$$

Para el ejemplo de la figura 2.7, se tiene que:

$$I = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^t$$

Estrategia 2:

$$I = (a_{ij}) := \begin{cases} a_{ij} = 1 & si\ j \in C \\ a_{ij} = \frac{1}{n} & si\ j \in F\ y\ n\ es\ la \\ & cantidad\ de\ vecinos\ gruesos \\ a_{ij} = 0 & otro\ caso \end{cases}$$

Para el ejemplo de la figura 2.7, se tiene que:

$$I = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.33 & 1 & 0.5 & 0 & 0 & 0.33 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 1 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.33 & 0 & 0 & 0 & 1 & 0.33 & 0 & 0 & 0.5 & 0.33 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.33 & 0.5 & 0.5 & 0 & 0.33 & 1 & 1 & 0 & 0.5 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.33 & 0 & 0 & 1 & 0.5 & 0 \end{bmatrix}^t$$

La estrategia 1 produce una interpolación constante. La estrategia 2 genera una interpolación lineal. La segunda estrategia, en general, obtiene mejores resultados.

Al igual que la mayoría de los método descritos, el operador de diezmado, $R$, es obtenido como $R = I^t$ y $A_H = RA_hI$.

### 2.3.3 Operadores de Complejidad

Existen dos operadores de complejidad que indican aproximadamente los requerimientos de memoria necesarios al usar cualquiera de los AMG. Estos requerimientos están expresados en términos del operador de complejidad de malla, $C_G$, y del operador de complejidad algebraico, $C_A$,

$$C_G = \sum_l \frac{n_l}{n_1} \tag{2.20}$$

y

$$C_A = \sum_l \frac{nz_l}{nz_1}, \tag{2.21}$$

donde $n_l$ y $nz_l$ denotan el número de variables y el número de elementos no ceros de la matriz $A$ en el nivel $l$. El nivel $l = 1$ corresponde al nivel más fino. Aunque el requerimiento verdadero de memoria para un AMG no está completamente reflejado en estas cantidades, estas medidas dan una idea muy aproximada del requerimiento de memoria necesario en una aplicación.

Cuando estén (2.20) y (2.21) más cercanas de 1, la estrategia de engrosamiento es más óptima en cuanto a requerimiento de memoria.

## 2.4 Precondicionador AMG

Originalmente los AMG fueron diseñados para ser usados *stand-alone*. Debido a experiencias prácticas, se puede observar que los AMG también pueden ser muy buenos precondicionadores, mejores que los precondicionadores tipo $ILU$. Heurísticamente, la mayor razón de esto es debido al hecho de que los AMG operan sobre todas las componentes del error, las de corto rango y las de largo rango. Sin embargo, la fase de *setup* puede requerir un costo computacional alto, por lo tanto, usar engrosamientos de tipo agresivo son útiles si los AMG son usados como precondicionadores.

Aunque un AMG trata de capturar toda la información relevante del problema usando apropiadas técnicas de engrosamiento e interpolación, la relación entre éstas

deberá ser muy óptima. La interpolación puede ayudar a que la reducción del error sea significativamente menos eficiente para algunas componentes muy específicas del error. Esto puede causar que unos pocos autovalores de la matriz de iteración AMG estén considerablemente mucho más cercanos a 1 que el resto de ellos. Por lo tanto, el factor de convergencia del AMG estará limitado por la lenta convergencia de unas pocas componentes del error mientras que la mayoría de éstas son reducidas muy rápidamente. Si se usa un método iterativo como acelerador estas frecuencias particulares son reducidas muy eficientemente. Otra alternativa es prevenir tales situaciones desarrollando interpoladores más complejos, pero esto aumenta el costo computacional.

Al ser usado un AMG como precondicionador, la fase de *setup* es realizada una sola vez. De esta manera, el método iterativo aplicado al sistema precondicionado deberá usar ciclos multinivel para resolver el sistema ($y = M^{-1}z$) en cada iteración del método iterativo. Usualmente, con uno o dos ciclos multinivel es suficiente en cada iteración del método iterativo seleccionado.

## 2.5 Problemas en los AMG

Todos estos métodos (AMG basados en interpolación y AMG basados en agregación) han sido evaluados para la ecuación de Poisson y para algunos otros problemas. La mayoría de los métodos manipulan la discretización del problema para obtener como resultado una $M$-matriz simétrica positiva definida débilmente diagonal dominante o una matriz simétrica positiva definida débilmente diagonal dominante o una $M$-matriz no simétrica con diagonal dominancia débil. Esto es debido a que la cosntrucción de los espacios gruesos se basa en caracterizar sobre el grafo de la matriz como se aproximan los errores de baja frecuencia (errores suaves), y esto esta bien estudiado para casos especificos, como los mencionados con anterioridad.

No está claro que usando las discretizaciones clásicas se pueda obtener un buen rendimiento de estos métodos. En general, ambas metodologías tienen rendimiento comparables, en aquellos casos que pueden ser aplicadas. En algunos casos, el AMG basado en interpolación es más rápido y en otros casos, los AMG basados en agregación son más rápidos. La mayor ventaja de los métodos de agregación es que necesitan mucho menos requerimientos de memoria que los métodos de interpolación. Por otro lado, los métodos de agregación requieren ser acelerados por algún método iterativo para mantener su eficiencia y robustez en muchas situaciones complejas. Hasta el momento el único resultado con demostración formal en cuanto a convergencia lo han dado Ruge y Stuben [RS87].

| Nombre | $n$ | $n_z$ | Descripción |
|---|---|---|---|
| C1 | 27.000 | 183.600 | ecuación 3D escalar de convección-difusión malla 30x30x30, $Cc = 10^3$ |
| C2 | 64.000 | 438.400 | ecuación 3D escalar convección-difusión malla 40x40x40, $Cc = 10^3$ |
| C3 | 125.000 | 860.000 | ecuación 3D escalar convección-difusión malla 50x50x50, $Cc = 10^3$ |
| D1 | 21.200 | 1.488.768 | Raefsky3 desde colección Davis |
| D2 | 19.779 | 1.328.611 | Raefsky4 desde colección Davis |
| D3 | 16.146 | 1.015.156 | Olafu desde colección Davis |
| P1 | 11.783 | 1.587.323 | Bearing desde código PERMAS |
| P2 | 25.906 | 786.380 | Railway desde código PERMAS |
| P3 | 48.162 | 2.386.056 | Methan desde código PERMAS |

Tabla 2.1: Descripción de los problemas de prueba.

En principio estos métodos son presentados en la literatura como muy prometedores, pero no está claro que puedan ser usados como una caja negra que sea independiente de los otros modulos de un paquete de simluación. Por tal motivo, nosotros a continuación haremos un estudio del AMG propuesto en [RS87] para varios problemas de test. Es importante mencionar que estos problemas están discretizados de forma clásica.

Se desea resolver el sistema lineal no singular $Ax = f$, donde $A$ proviene de un operador 3D escalar de convección-difusión, de la colección de Davis [Dav97] y del

código PERMAS [AJL+97]. El vector lado derecho, $f$, es generado de tal manera que el vector solución, $x$, tiene valores aleatorios en $[0, 1]$. De esta manera, se puede medir el vector residuo y el vector error.

En la tabla 2.1 está la descripción de cada problema. En la primera columna se encuentra el nombre del problema. En la segunda columna se encuentra el número de filas de la matriz. En la tercera columna se encuentra la cantidad de elementos no nulos de la matriz. En la cuarta columna se encuentra una breve descripción del problema.

Las matrices C1, C2 y C3 provienen de un operador 3D escalar eliptico discretizado por diferencias finitas. Las matrices D1, D2 y D3 proviene de problemas de dinámica de fluidos y su descretización es mediante elementos finitos. Las matrices P1, P2 y P3 proviene de un paquete industrial para probelmas de mecánica estructural y su discretización es mediante elementos finitos. Es importante mencionar que las primeras 6 matrices son no simétricas y las matrices P1, P2 y P3 son simétricas positivas definidas. Además, ninguna de las matrices de prueba son debilmente diagonal dominantes.

| Nombre | AMG |
|---|---|
| C1 | * |
| C2 | * |
| C3 | * |
| D1 | * |
| D2 | * |
| D3 | * |
| P1 | >> 500 |
| P2 | >> 500 |
| P3 | >> 500 |

Tabla 2.2: Convergencia del AMG.

En la tabla 2.2 se puede observar la convergencia del AMG usando un esquema de V-ciclo con dos pasos de relajación $CF$ en cada nivel. Se construyó un engrosamiento con conexiones fuertes positivas y se uso una interpolación estandar.

El criterio de parada para este esquema es $||r^k||_2/||f||_2 < 10^{-12}$, donde $r^k$ es el

vector residuo en la $k$-ésima iteración y $f$ es el vector lado derecho. Se ha fijado el espacio más grueso en 1000 nodos.

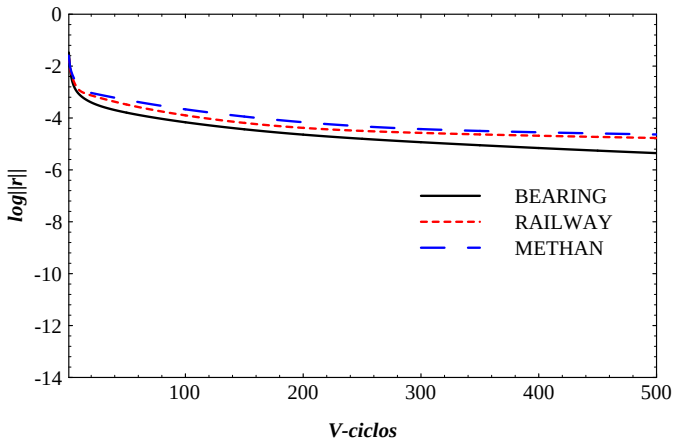



Figura 2.8: Convergencia del AMG.

Para los casos C1, C2, C3, D1, D2, D3 el AMG diverge debido a la no diagonal dominancia de las matrices. Para los casos P1, P2 y P3 la convergencia es demasiado lenta. Esto es debido a la mala disposición de las dos descomposiciones del espacio $V^h$. En estos casos, el trabajo de convergencia lo está haciendo solamente el método relajador. En la figura 2.8 podemos observar las gráficas de convergencia para los casos P1, P2 y P3.

Estos resultados experimentales demuestran que estos métodos no pueden ser usados como cajas negras en un paquete de simulación, porque su implementación requiere usar discretizaciones específicas para cada problema con el fin de obtener matrices con una diagonal dominancia significativa. Lo cual implicaría modificar sustancialmente el software del paquete de simulación.

# Capítulo 3

# Sistema Complemento de Schur

En este capítulo se presenta una breve introducción del Sistema Complemento de Schur. Se describe como se obtiene el sistema de Schur. También, se presentan algunas técnicas algebraicas para precondicionar dicho sistema.

## 3.1 Introducción

Los métodos de descomposición en dominios no son únicamente un esquema de paralelización de grano grueso, sino que en muchas ocasiones también permiten reducir la complejidad numérica de la resolución del sistema lineal, incluso en códigos secuenciales. Esto se debe a que partiendo de la distribución de datos que induce una descomposición en dominios es posible explotar características numéricas que con otras ordenaciones pasan desapercibidas.

El desarrollo de las técnicas de descomposición en dominios va históricamente unida a la resolución de EDPs. El primer trabajo conocido sobre descomposición en dominios data de 1870 y fue realizado por H. A. Schwarz. Derivado de este trabajo se generó una familia de métodos conocidos como métodos de Schwarz o métodos con solape. Dichos métodos descomponen la región donde se define la EDP en un conjunto de subdominios solapados. Por solape entendemos superposición de elementos en el caso de elementos finitos, o de intervalos de discretización en el caso de diferencias finitas. La formulación original del método es la siguiente, partiendo de una primera aproximación a la solución, se resuelve de forma alternativa el problema original en cada subdominio, usando como condición de contorno en la frontera que está en el interior de otros subdominios la aproximación de la solución en la iteración anterior en esos subdominios.

Existen diferentes mejoras del algoritmo original que generan dos familias de

métodos conocidos como métodos aditivos de Schwarz [DW92] y métodos multiplicativos de Schwarz [DSW93]. Cuando estos métodos son usados como precondicionadores en problemas tridimensionales simétricos definidos positivos, se demuestra que $\kappa(M^{-1}A) \geq C(\frac{H}{h})$, donde $H$ es el diámetro de los dominios, $h$ es el diámetro de los elementos y $C$ es una función dependiente únicamente del solape entre dominios [Drij93]. Para un análisis detallado sobre los aspectos prácticos de los métodos de Schwarz ver [GS93a].

Estos métodos han sido usados principalmente en dinámica de fluidos dentro de los llamados métodos multibloque [Ber89] [DJF90] [Ece88] [Ewi91] [GS93a]. Sin embargo, el interés en los métodos de Schwarz es cada vez menor, ya que se ha demostrado que todo método con solape se puede formular de forma equivalente como un método sin solape, y los métodos sin solape tienen la ventaja de no replicar el trabajo que se hace en las zonas solapadas, [TC88] [BW89]. Además, los métodos sin solape permiten un enfoque más algebraico, que los hace más fácilmente generalizables a otros campos diferentes de la resolución de EDPs.

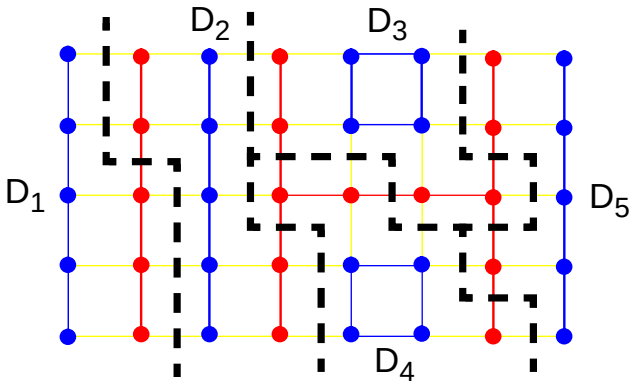



Figura 3.1: Partición en 5 dominios.

Los métodos sin solape se basan en descomponer la región donde se define la EDP en un conjunto de subdominios no solapados. Esta descomposición se realiza mediante algún algoritmo de partición de grafos [SE87] [BB87] [PSL90] [DMM95]. En las aplicaciones de resolución de EDPs la partición se aplica a la malla que discretiza el dominio de resolución, considerando a los elementos de dicha malla unidades indivisibles. La partición se realiza de forma que se obtiene un conjunto de subgrafos, llamados dominios, que no tienen ninguna conexión entre ellos. Los dominios tienen únicamente conexiones con otro subgrafo llamado frontera. En la figura 3.1, se muestra un ejemplo para 5 dominios. Luego, numerando primeramente las incógnitas de los dominios y después las de la frontera, obtenemos un sistema lineal, donde la matriz tiene una estructura de flecha a bloques como la siguiente:

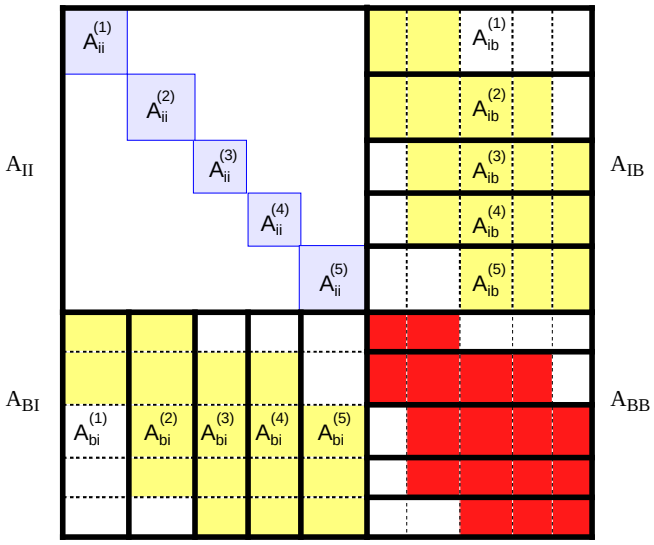



Figura 3.2: Estructura a bloques de la matriz.

$$\begin{bmatrix} A_{ii}^{(1)} & & & A_{ib}^{(1)} \\ & \cdots & & \cdots \\ & & A_{ii}^{(P)} & A_{ib}^{(P)} \\ A_{bi}^{(1)} & \cdots & A_{bi}^{(P)} & A_{BB} \end{bmatrix} \begin{bmatrix} x_I^{(1)} \\ \cdots \\ x_I^{(P)} \\ x_B \end{bmatrix} = \begin{bmatrix} b_I^{(1)} \\ \cdots \\ b_I^{(P)} \\ b_B \end{bmatrix} \tag{3.1}$$

En la figura 3.2 se puede observar las estructura a bloques de la matriz para el ejemplo de 5 dominios. Este sistema puede ser reducido por eliminación gaussiana a bloques al sistema de complemento de Schur:

$$\begin{bmatrix} A_{ii}^{(1)} & & & A_{ib}^{(1)} \\ & \cdots & & \cdots \\ & & A_{ii}^{(P)} & A_{ib}^{(P)} \\ & & & S \end{bmatrix} \begin{bmatrix} x_I^{(1)} \\ \cdots \\ x_I^{(P)} \\ x_B \end{bmatrix} = \begin{bmatrix} b_I^{(1)} \\ \cdots \\ b_I^{(P)} \\ \tilde{b}_B \end{bmatrix} \tag{3.2}$$

donde, $S = A_{BB} - \sum_{k=1}^{P} A_{bi}^{(k)} A_{ii}^{(k)-1} A_{ib}^{(k)}$ y $\tilde{b}_B = b_B - \sum_{k=1}^{P} A_{bi}^{(k)} A_{ii}^{(k)-1} b_i^{(k)}$.

El resto del capítulo esta organizado como sigue. En primer lugar, se describe como queda expresado el sistemas reducido de Schur. Luego, se presentan algunas técnicas algebraicas, las más populares, para construir precondicionadores para dicho sistema.

## 3.2 Sistema de Schur

Para una manipulación más sencilla del sistema lineal (3.1) lo vamos a representar de la siguiente manera:

$$\begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix} \begin{bmatrix} x_I \\ x_B \end{bmatrix} = \begin{bmatrix} b_I \\ b_B \end{bmatrix} \tag{3.3}$$

donde $A_{II}$ es una matriz diagonal a bloques. Como se explicó anteriormente este sistema lineal puede ser reducido a:

$$Sx_B = \tilde{b}_B \tag{3.4}$$

$$S = A_{BB} - A_{BI}A_{II}^{-1}A_{IB} = A_{BB} - \sum_{k=1}^{P} A_{bi}^{(k)} A_{ii}^{(k)-1} A_{ib}^{(k)} \tag{3.5}$$

$$\tilde{b}_B = b_B - A_{BI}A_{II}^{-1}b_I = b_B - \sum_{k=1}^{P} A_{bi}^{(k)} A_{ii}^{(k)-1} b_i^{(k)} \tag{3.6}$$

Por lo tanto, para resolver el sistema (3.3) en primer lugar se resuelve el sistema (3.4) para obtener $x_B$, y finalmente se calcula $x_I$ en una sustitución paralela hacia atrás:

$$x_I = A_{ii}^{(k)-1}(b_i^{(k)} - A_{ib}^{(k)}x_B) \tag{3.7}$$

Una primera opción para resolver (3.4) es usar un método directo. Esto implica calcular de forma explícita los coeficientes de $S$, lo que supone factorizar la matriz $A_{II}$ y resolver tantos sistemas triangulares como incógnitas haya en la frontera. Seguidamente, se debería factorizar la matriz $S$ y resolver los sistemas triangulares pertinentes. Claramente esto es muy costoso en operaciones, y un cuello de botella secuencial en la fase de factorización de $S$ [Cha87] [CR87]. Esta opción tiene muchas más operaciones que calcular la factorización de $A$ usando una ordenación que minimice el llenado.

Una segunda opción para resolver (3.4) es usar un método iterativo. Dado que en un método iterativo sólo se requiere la matriz de coeficientes para multiplicarla por un vector, esta operación puede hacerse usando la expresión (3.5), sin necesidad de calcular explícitamente los coeficientes de $S$. De esta manera, la secuencia de operaciones para calcular $w = Sv$, donde $w$ y $v$ son vectores se puede observar en la figura 3.3.

| MATRIZ-VECTOR DE SCHUR |
|---|
| 1.- CALCULAR $p := A_{IB}v$ |
| 2.- RESOLVER $A_{II}q = p$ |
| 3.- CALCULAR $w := A_{BB}v - A_{BI}q$ |

Figura 3.3: Algoritmo matriz-vector de schur.

El uso de la matriz de Schur presenta notables ventajas desde el punto de vista numérico en la resolución del sistema lineal. Para EDPs bidimensionales elípticas autoadjuntas definidas positivas se demuestra que el número de condición de $A$ es $O(h^{-2})$, y en el caso de problemas tridimensionales $O(h^{-3})$, donde $h$ es el diámetro de los elementos. Sin embargo, en estos mismos problemas el número de condición de $S$ es respectivamente $O(H^{-1}h^{-1})$ y , $O(H^{-1}h^{-2})$ donde $H$ es el diámetro de los dominios [CM94]. Por ejemplo, para problemas de electromagnetismo tridimensionales $H$ puede estar en el orden de las decenas de centimetros y $h$ puede estar en el orden las decimas de milimetros. Es decir, $H = 10^{-1}$ y $h = 10^{-4}$. Entonces, se tiene que:

$$k(A) = O(10^{12})$$
$$k(S) = O(10^{9})$$

Aunque la mejora del número de condición es notable frente a la matriz completa, puede no ser suficiente, y por lo tanto será necesario algún tipo de precondicionador para $S$. No es posible usar una factorización incompleta, ya que $S$ no se calcula de forma explícita. Una opción sería usar un precondicionador polinomial, que tan sólo requiere para su aplicación la operación matriz por vector. Sin embargo, tales precondicionadores requieren tener una buena estimación de los autovalores máximo y mínimo de la matriz para ser competitivos [AMO92]. Salvo en problemas muy específicos donde esta estimación puede hacerse de forma analítica, en la mayoría de los casos sólo pueden emplearse las cotas de Gerschgorin, resultando entonces que el coste de la aplicación del precondicionador supera el ahorro en iteraciones que proporciona.

## 3.3 Precondicionamiento

Utilizando un precondicionamiento por la derecha, explicado en el capítulo §1, para el sistema de Schur (3.4). El sistema precondicionado queda expresado como:

$$\tilde{S}\tilde{x}_B = \hat{f}_B \tag{3.8}$$

donde,

$$\tilde{S} = SM^{-1} \quad \text{y} \quad \tilde{x}_B = Mx_B. \tag{3.9}$$

La matriz $M$ es el precondicionador. Para realizar el producto matriz por vector, $w = \tilde{S}v$, en cada iteración del método iterativo se deberá resolver un sistema lineal con $M$ y seguidamente multiplicar por la matriz $S$. Esto se puede ver en la figura 3.4.

| MATRIZ-VECTOR CON PRECONDICIONADOR |
|---|
| 1.- RESOLVER $Mu = v$ |
| 2.- CALCULAR $w := Su$, USANDO EL ALGORITMO 3.3 |

Figura 3.4: Algoritmo matriz-vector de Schur con precondicionador.

Existen varias opciones para definir la matriz $M$. Estas opciones deberán basarse en alguna interpretación física que se pueda dar a la matriz $S$. Por ello, vamos en primer lugar a relacionar la matriz $S$ con el problema continuo. Para mayor claridad en la exposición haremos uso del siguiente ejemplo:

$$Lu \equiv -\nabla(\alpha(x,y,z)\nabla u) = f(x,y,z) \quad \text{en} \quad \Omega \quad \text{y} \quad u = 0 \quad \text{en} \quad \partial\Omega \tag{3.10}$$

Supongamos para simplificar la notación que sólo tenemos dos dominios. Denotamos por $\Omega_1$ y $\Omega_2$ los dos dominios en que ha sido descompuesta la región $\Omega$. Denotamos por $\Gamma$ la frontera entre $\Omega_1$ y $\Omega_2$. Así mismo, denotamos por $u_1$, $u_2$ y $u_B$ a la solución $u$ de (3.10) restringida a $\Omega_1$, $\Omega_2$ y $\Gamma$, respectivamente.

Integrando por partes la formulación débil de (3.10) obtenemos que $u_1$, $u_2$ y $u_B$ cumplen los siguientes problemas locales:

$$\begin{cases} Lu_1 = f & \text{en} \quad \Omega_1 \\ u_1 = 0 & \text{en} \quad \partial\Omega_1 \setminus \Gamma \\ u_1 = u_2 & \text{en} \quad \Gamma \end{cases} \tag{3.11}$$

$$\begin{cases} Lu_2 = f & \text{en} \quad \Omega_2 \\ u_2 = 0 & \text{en} \quad \partial\Omega_2 \setminus \Gamma \\ u_2 = u_B & \text{en} \quad \Gamma \end{cases} \tag{3.12}$$

y la condición de continuidad de flujo a través de $\Gamma$:

$$\vec{n}_1 \cdot (\alpha\nabla u_1) = -\vec{n}_2 \cdot (\alpha\nabla u_2) \quad \text{en} \quad \Gamma \tag{3.13}$$

donde $\vec{n}_1$ y $\vec{n}_2$ son respectivamente las normales a $\Gamma$ en $\Omega_1$ y $\Omega_2$.

Sea $g$ una condición de contorno tipo Dirichlet arbitraria en $\Gamma$, y sean $v_1$ y $v_2$ las soluciones de los siguientes problemas:

$$\begin{cases} Lv_1 = f & \text{en} \quad \Omega_1 \\ v_1 = 0 & \text{en} \quad \partial\Omega_1 \setminus \Gamma \\ v_1 = g & \text{en} \quad \Gamma \end{cases} \tag{3.14}$$

$$\begin{cases} Lv_2 = f & \text{en} \quad \Omega_2 \\ v_2 = 0 & \text{en} \quad \partial\Omega_2 \setminus \Gamma \\ v_2 = g & \text{en} \quad \Gamma \end{cases} \tag{3.15}$$

Naturalmente, $v_1$ y $v_2$ no cumplirán la condición de continuidad de flujo a través de $\Gamma$ (3.13), salvo si $g = u_B$.

Definimos el operador $T$ como:

$$T : g \rightarrow \vec{n}_1 \cdot (\alpha \nabla v_1) + \vec{n}_2 (\alpha \nabla v_2) \tag{3.16}$$

Es decir, $T$ mapea $g$ en el salto de flujo a través de $\Gamma$. Entonces $u_B$ satisface la ecuación:

$$Tu_B = 0 \tag{3.17}$$

El operador $T$ se conoce como el operador de Steklov-Poincaré [Ago88] [QV91]. Si $L$ es un operador simétrico positivo definido se puede demostrar que $T$ es también un operador simétrico positivo definido respecto al producto interno $L^2$ en $\Gamma$.

La versión discreta de la ecuación (3.17) se obtiene al reducir el sistema lineal que discretiza (3.10) al sistema de complemento de Schur. Por lo tanto las propiedades numéricas de $S$ están relacionadas con las del operador de Steklov-Poincaré. Casi todos los precondicionadores propuestos para $S$ en el contexto de la resolución de EDPs se basan en analizar las propiedades del operador $T$ en los espacios de Sobolev pertinentes [Dry82] [BW93] [Elm89]. La formulación analítica de $T$, o simplemente el análisis de sus propiedades (autovalores), sólo es posible en algunos problemas, esencialmente en problemas elípticos en dominios regulares. Por ello casi todos los precondicionadores propuestos para $S$ son de aplicación práctica muy limitada. Sin embargo, un punto muy interesante de esta línea de trabajos es que clarifican la relación entre los algoritmos multinivel o multimalla y la descomposición en dominios

[AV89a]. Para una recopilación de todo el conjunto de precondicionadores propuestos para $S$ siguiendo este esquema de trabajo ver [SIA88] [SIA89] [SIA90] [SIA91] [SIA92].

### 3.3.1 Precondicionador con Vectores Prueba

El primer intento de formular un precondicionador de carácter más general para la matriz de Schur fue realizado por T. Chan. La propuesta se basó en la observación experimental de que en muchos problemas, el operador diferencial que discretizado generaría el sistema de Schur presenta un acoplo global débil y un acoplo local fuerte [Cha89] [CK90]. Esto significa que el coeficiente $s_{ij}$ de la matriz de Schur es tanto más grande en magnitud cuanto más cercanos entre si estén los nodos $i$-ésimo y $j$-ésimo.

Basado en esta observación T. Chan propuso el precondicionador MSC($k$). La idea básica de este precondicionador es aproximar la matriz de Schur por una matriz en banda donde el número de diagonales incluidas en la banda lo controla el parámetro $k$ ($k = 0, 1, 2, \ldots$). En concreto se pretende calcular $E_k$ de forma que:

$$E_k v_i = S v_i \quad \text{para} \quad i = 1, \ldots, L \tag{3.18}$$

En problemas no simétricos $L = 2k + 1$ y en problemas simétricos $L = k + 1$. Es decir, forzamos que el producto del precondicionador por un conjunto de vectores prueba sea idéntico al producto de la matriz $S$ por el mismo conjunto de vectores prueba. Por lo tanto, este precondicionador tan sólo requiere la operación matriz por vector para ser calculado. Los vectores prueba, para el caso no simétrico con 2 dominios, son:

$$E_0 : v_1 = [1, 1, 1, 1, \ldots]^t \qquad E_1 : \begin{array}{l} v_1 = [1, 0, 0, 1, 0, 0, 1, \ldots]^t \\ v_2 = [0, 1, 0, 0, 1, 0, 0, \ldots]^t \\ v_3 = [0, 0, 1, 0, 0, 1, 0, \ldots]^t \end{array}$$

y en el caso simétrico con 2 dominios son:

$$E_0 : v_1 = [1, 1, 1, 1, \ldots]^t \qquad E_1 : \begin{array}{l} v_1 = [1, 0, 1, 0, 1, 0, 1, \ldots]^t \\ v_2 = [0, 1, 0, 1, 0, 1, 0, \ldots]^t \end{array}$$

Para $k = 0$ el precondicionador MSC($k$) aproxima la matriz de Schur por una matriz diagonal cuyo elemento $i$-ésimo es la suma de la fila $i$-ésima de $S$. Para $k = 1$ se tiene una matriz tridiagonal cuyo elemento $e_{ij}$ es la suma alternada en columnas de los elementos de la fila $i$-ésima de $S$. Intuitivamente se observa que

los vectores de prueba tratan de muestrear la matriz $S$ y comprimirla pero respetando la disposición relativa de los números más significativos. En problemas sobre mallas estructuradas y con particiones regulares el precondicionador MSC($k$) puede llegar a captar los números más significativos de la matriz de Schur, ya que estos están dispuestos en forma regular. Desafortunadamente, en problemas genéricos de elementos finitos usando mallas no estructuradas este precondicionador deja de ser eficaz debido a que los elementos más significativos de $S$ se disponen de forma irregular, y en consecuencia, los vectores prueba que deben ser usados acaban siendo la base canónica, lo que es equivalente a calcular toda la matriz $S$.

Hoy en día, a este método de obtener un precondicionador también es llamado *probing*. En [Saa96] también hay una explicación breve de esta técnica.

### 3.3.2 Precondicionador $A_{BB}$

O. Axelsson [Axe94] ha abierto una línea puramente algebraica de análisis de los precondicionadores para $S$. Para lograr precondicionadores de $S$ propone aproximar la ecuación (3.5) de la siguiente forma:

$$M = A_{BB} - A_{BI} D_{II}^{-1} A_{IB} \tag{3.19}$$

donde $D_{II}$ es una matriz diagonal a bloques con la misma estructura que $A_{II}$, pero más sencilla de invertir. Este precondicionador es de especial interés en aquellos problemas en los que es posible tomar $D_{II} = 0$, en cuyo caso nos queda $M = A_{BB}$. Esto implica que de los dos términos que forman la matriz de Schur, $A_{BB}$ y $A_{BI} A_{II}^{-1} A_{IB}$, el primero sea dominante frente al segundo.

Notar que el primer termino, $A_{BB}$, representa la influencia entre nodos frontera de forma directa (acoplo local), mientras que el segundo termino, $A_{BI} A_{II}^{-1} A_{IB}$, representa la influencia entre nodos de la frontera a través de nodos internos (acoplo global). Es decir, la propuesta de precondicionar la matriz de Schur mediante la matriz $A_{BB}$ es válida en los mismos casos que lo es el precondicionador MSC($k$), con la ventaja de que ahora la malla puede ser irregular. Evidentemente, no todos los operadores diferenciales presentan la propiedad de acoplo local fuerte y acoplo global débil. Esta propiedad es típica de los operadores que modelan fenómenos de difusión, mientras que los operadores que modelan fenómenos de convección presentan justamente la propiedad opuesta. Por lo tanto, este tipo de precondicionadores se restringe al caso de EDPs con operadores diferenciales de difusión.

En [Axe94] se analiza el uso de la matriz $A_{BB}$ como precondicionador para

problemas simétricos positivos definidos, en los que $A$ define una norma, $\|x\|_A = \sqrt{< x, Ax >}$.

Dados los espacios vectoriales $V_1$ y $V_2$ de dimensiones $n_1$ y $n_2$, definimos $W = V_1 \times V_2$ de dimensión $n = n_1 + n_2$. Particionando según $V_1$ y $V_2$ la matriz $A$, que es simétrica positiva definida, tenemos:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{3.20}$$

Definimos también los espacios $W_1 = \{v = [v_1, 0]^t,\ v_1 \in V_1\}$, $W_2 = \{v = [0, v_2]^t,\ v_2 \in V_2\}$, de forma que $W_1$ y $W_2$ cumplen $W_1 \cap W_2 = [0, 0]^t$.

La desigualdad de Cauchy-Schawrz-Bunyakowski, que es una generalización de la desigualdad de Schwarz, afirma que existe $\gamma \leq 1$ dependiente únicamente de la elección de $W_1$ y $W_2$ tal que:

$$| < w_1, Aw_2 > | \leq \gamma \sqrt{< w_1, Aw_1 > \cdot < w_2, Aw_2 >} \quad \forall w_1 \in W_1 \ \text{ y } \ \forall w_2 \in W_2 \tag{3.21}$$

Al valor mínimo de $\gamma$ se le denomina coseno del ángulo entre $W_1$ y $W_2$ [Ipse95]. Para un análisis detallado de la desigualdad Cauchy-Schawrz-Bunyakowski ver [EV91]. De (3.20) y (3.21) tenemos que:

$$\gamma = sup_{w_1 \in W_1,\ w_2 \in W_2} \frac{< w_1, Aw_2 >}{\sqrt{< w_1, Aw_1 > \cdot < w_2, Aw_2 >}} = sup_{v_1 \in V_1,\ v_2 \in V_2} \frac{< v_1, A_{12} v_2 >}{\sqrt{< v_1, A_{11} v_1 > \cdot < v_2, A_{22} v_2 >}}$$

En [Axe94] se demuestra que si se usa como precondicionador de $S$ la matriz $A_{BB}$ el número de condición del sistema precondicionado es:

$$\kappa(A_{BB}^{-1} S) \leq \frac{1}{1 - \gamma^2} \tag{3.22}$$

Este estudio analítico no se puede extender a problemas no simétricos ya que entonces $A$ no define una norma, aunque el precondicionador sigue siendo aplicable en sistemas no simétricos.

### 3.3.3 Precondicionador Inducido

Un precondicionador más general es el llamado *precondicionador inducido* [Saa96]. Este precondicionador usa una aproximación al término global $A_{bi}^{(k)} A_{ii}^{(k)-1} A_{ib}^{(k)}$ basada

en una factorización $ILU$. Es decir, se obtiene una aproximación de cada $A_{ii}^{(k)}$, en paralelo, como:

$$\tilde{A}_{ii}^{(k)} = L_{ii}^{(k)} U_{ii}^{(k)} - R_{ii}^{(k)} \tag{3.23}$$

Entonces, la aproximación de la matriz de Schur $S$ queda:

$$\tilde{S} = A_{BB} - A_{bi}^{(k)} \tilde{A}_{ii}^{(k)-1} A_{ib}^{(k)} \tag{3.24}$$

De esta manera, con los paramétros adecuados de la factorización $ILU$ se puede obtener la matriz $S$ de forma exacta. La matriz $\tilde{S}$ es densa, lo que implica aplicar alguna estrategia de diezmado para eliminar elementos no significativos de dicha matriz. Sea $T$ la matriz obtenida luego del diezmado,

$$T = \tilde{S} - E. \tag{3.25}$$

Por último, se debe calcular una factorización $ILU$ de $T$ para ser usada como precondicionador en el método iterativo seleccionado.

Esta técnica de construir un precondicionador tiene la desventaja de que la factorización (3.23) puede ser muy densa para problemas en donde el acoplo global de los nodos es fuerte. Ya que como mínimo se puede hacer una $ILU(0)$ que implicaría tener en la factorización la misma cantidad de elementos de $A_{ii}^{(k)}$ en las mismas posiciones que en $A_{ii}^{(k)}$.

# Capítulo 4

# Precondicionador Algebraico

En este capítulo se presentan varios precondicionadores algebraicos paralelos para el sistema complemento de Schur. El primer precondicionador propuesto, llamado DFP, se basa en una factorización fuertemente diezmada. El segundo precondicionador propuesto, llamado AMGP, se basa en técnicas multinivel algebraicas. Se usa paralelismo de paso de mensaje y *threads* para explotar dos niveles de paralelismo. Los precondicionadores son evaluados con una ecuación de convección-difusión, con un conjunto de casos industriales provenientes de un paquete de elementos finitos llamado PERMAS [AJL+97] y un conjunto de problemas obtenidos desde la colección de Davis [Dav97]. Se ha obtenido una aceleracion cuasi-lineal hasta 32 procesadores.

## 4.1 Introducción

La resolución de sistemas lineales es el núcleo computacional que más tiempo consume en una simulación de elementos finitos, pero éste no es el módulo de software más grande (más costoso en cuanto a desarrollo de software). Debido a razones económicas, el método de resolución del sistema lineal debe ser tan independiente como sea posible del esquema de discretización, del generador de malla y del sistema CAD que envuelven la simulación completa. Además, el método de resolución lineal debe ser tan robusto como sea posible. Hoy en día, los métodos directos (factorizaciones *LU* o *Cholesky*) son los principales métodos de resolución de un paquete comercial de elementos finitos. También, en algunos casos, los paquetes comerciales ofrecen como alternativa métodos iterativos que usan precondicionadores basados en factorizaciones incompletas. Aunque, los métodos directos tienen varias desventajas (llenado, cantidad de operaciones, bajo grado de paralelismo), son seleccionados como el principal método de resolución debido a su efectividad y robustez para problemas de tamaño moderado. Además, ellos son una perfecta caja negra sin parámetros de usuario.

Para los métodos directos existe una estrecha relación entre el llenado producido por la factorización y el paralelismo, debido a la estructura de la matriz. Usualmente, el factor dominante es la cantidad de llenado. Por tal razón, hay que seleccionar un orden que minimice el llenado. La escalabilidad de un método directo no es muy alta ($O(10)$ procesadores), debido a los limites impuesto por el orden elegido. Además, son requeridos módulos complejos de software (*parallel scheduler*) para explotar este paralelismo potencial [AJL+97].

Deseamos desarrollar un método de resolución lineal que sea competitivo con los métodos que actualmente usan los paquetes comerciales, es decir, los métodos directos. Nuestro método deberá tener las siguientes propiedades:

- Pueda ser usado en un amplio rango de aplicaciones.
- El número de operaciones deberá ser similar al de un método directo.
- Sea independiente de otras partes de la simulación
- Este deberá tener pocos parámetros de usuario, y estos parámetros deberán poderse sintonizar fácilmente.
- El paralelismo deberá ser explotado fácilmente, y la escalabilidad deberá ser al menos hasta $O(100)$ procesadores.

El esquema propuesto se basa en la formulación del complemento de Schur del sistema lineal. Se construyen dos precondicionadores paralelos para la matriz complemento de Schur. Estos precondicionadores sólo requieren información algebraica. El primero, llamado DFP, usa una factorización fuertemente diezmada. El segundo, llamado AMGP, usa técnicas de métodos multinivel algebraicas. Además, se ha explotado dos niveles de paralelismo (procesos y *threads*) usando PVM [GBDJ93] y openMP [Ope98]. Esto permite una mayor escalabilidad [LC00], [LC01a], [LC01b], [LCra].

El resto del capítulo está organizado como sigue. En primer lugar, se presenta la formulación del problema. Se describe la forma a bloques del sistema y los componentes de la matriz de Schur. En segundo lugar, se proponen dos precondicionadores algebraicos paralelos para este sistema. Finalmente, se presentan los resultados experimentales para un conjunto de problemas de prueba.

## 4.2 Formulación del Problema

Se desea resolver el siguiente sistema lineal no singular de ecuaciones,

$$Ax = f \tag{4.1}$$

donde $A$ es una matriz de gran dimensión y dispersa, de orden $N \times N$. Se asume que un ordenamiento en descomposición en dominios ha sido aplicado al grafo de la matriz o a la malla original, de forma tal que son obtenidos $P$ dominios no solapados. Entonces, la estructura del sistema lineal es la siguiente:

$$\begin{pmatrix} A_{ii}^{(1)} & & & A_{ib}^{(1)} \\ & \ddots & & \vdots \\ & & A_{ii}^{(P)} & A_{ib}^{(P)} \\ A_{bi}^{(1)} & \cdots & A_{bi}^{(P)} & A_{BB} \end{pmatrix} \begin{pmatrix} x_i^{(1)} \\ \vdots \\ x_i^{(P)} \\ x_B \end{pmatrix} = \begin{pmatrix} f_i^{(1)} \\ \vdots \\ f_i^{(P)} \\ f_B \end{pmatrix} \tag{4.2}$$

Con el fin de simplificar la notación, el conjunto de bloques $A_{ii}^{(k)}$, $A_{ib}^{(k)}$, $A_{bi}^{(k)}$, $x_i^{(k)}$ y $f_i^{(k)}$ serán denotados por $A_{II}$, $A_{IB}$, $A_{BI}$, $x_I$ y $f_I$ El sistema lineal (2) puede ser reducido al sistema complemento de Schur,

$$Sx_B = \hat{f}_B \tag{4.3}$$

donde,

$$S = A_{BB} - A_{BI}A_{II}^{-1}A_{IB} \quad \text{y} \quad \hat{f}_B = f_B - A_{BI}A_{II}^{-1}f_I$$

Cuando el sistema (4.3) es resuelto, el vector solución $x_I$ se obtiene como:

$$x_I = A_{II}^{-1}(f_I - A_{IB}x_B)$$

Puede ser usado un método iterativo para resolver el sistema complemento de Schur (4.3). De esta manera, sólo es necesario un método de resolución lineal para cada matriz $A_{ii}^{(k)}$ cuando se requiera multiplicar la matriz $S$ por un vector. Nosotros usamos un método directo (factorización $LU$ o $Cholesky$) para las matrices $A_{ii}^{(k)}$.

Un ordenamiento basado en una descomposición en dominios genera una manera natural de introducir una paralelización de pasos de mensajes, creando un proceso por dominio. Este ordenamiento produce una forma óptima de asignación estática de datos en procesadores, es decir, esta asignación resuelve el problema de secuenciamiento dinámico (*parallel scheduler*). Sin embargo, un orden basado en una descomposición en dominios genera más llenado que otras técnicas como *minimun degree* o *multilevel nested dissection.* Por este motivo, los métodos directos nunca

usan ordenamientos basados en una descomposición en dominios, y ellos necesitan un secuenciamiento dinámico para una paralelización de paso de mensaje.

En términos del número de operaciones, la formulación de Schur puede ser competitiva con respecto a un método directo sólo si el método iterativo converge muy rápido. Debido a esto, es obligatorio tener un eficiente y robusto precondicionador para la matriz de Schur.

## 4.3 Precondicionador Algebraico Paralelo

Nosotros usamos un precondicionador por la derecha del sistema de Schur. Entonces, el sistema (4.3) se transforma en:

$$SM^{-1}Mx_B = \hat{f}_B \tag{4.4}$$

La matriz $M$ es el precondicionador. Para multiplicar $SM^{-1}$ por un vector, se necesita resolver un sistema lineal con la matriz $M$, y luego, multiplicar el vector solución de ese sistema por la matriz $S$.

La matriz de Schur, $S$, está formada por dos términos:

$$S = T_{local} - T_{global} = A_{BB} - A_{BI}A_{II}^{-1}A_{IB} \tag{4.5}$$

El estado del arte de los precondicionadores para la matriz de Schur consiste en técnicas de aproximar el término local y global de la expresión (4.5). El término local, $T_{local}$, permite capturar los acoplos fuertes que aparecen entre los nodos vecinos de la malla. El término global, $T_{global}$, permite capturar los acoplos fuertes entre nodos que no son vecinos directos en la malla.

En [Axe94] se propone usar sólo el término local como un precondicionador algebraico para $S$, es decir, la matriz $A_{BB}$ o alguna aproximación a ésta. Desafortunadamente, esto no es suficiente para un amplio rango de aplicaciones. Por lo tanto, es necesario alguna técnica algebraica para obtener una aproximación del término global.

En [Saa96] son presentadas varias aproximaciones algebraicas para el término global como técnicas con vectores de prueba y precondicionadores inducidos, ya explicadas en el capítulo anterior.

Otra posibilidad es usar una técnica de aproximación dispersa de la inversa (SPAI) [SC98] [BMT96] [GH97]. De esta manera, se puede obtener una aproximación de $A_{II}^{-1}A_{IB}$ utilizando operaciones matriz por vector únicamente. En [CS97] se propone un precondicionador basado en SPAI.

Todas estas técnicas mencionadas son efectivas si existe un acoplo global débil. La más general de todas es usar un precondicionador inducido, pero la cantidad de computo a realizar puede ser excesivo para problemas con acoplo global fuerte. Además, desafortunadamente la gran mayoría de los problemas industriales presentan un acoplo global fuerte. Por lo tanto, es necesario construir precondicionadores que permitan capturar estos acoplos de forma efectiva.

Nuestro precondicionador $M$ tiene la siguiente forma:

$$M = A_{BB} - A_{BI}\tilde{A}_{II}^{-1}A_{IB}. \tag{4.6}$$

donde $\tilde{A}_{II}^{-1}$ es una aproximación de $A_{II}^{-1}$. El término local $A_{BB}$ no es aproximado porque no es significante el costo computacional asociado a este término. Nosotros calculamos explícitamente la matriz $M$.

En general, la matriz $M$ en (4.6) tiene una densidad alta, aunque la matriz $\tilde{A}_{II}^{-1}$ sea una matriz muy dispersa. Deseamos que $M$ tenga una densidad similar que la matriz $A_{BB}$ para reducir el número de operaciones en el método de resolución. Por lo tanto, debemos usar algún criterio de diezmado que debe ser aplicado a la matriz $M$. Además, como es necesario resolver sistemas lineales con $M$, entonces esta matriz debe ser factorizada. Nosotros usamos una factorización incompleta, especificamente usamos la $ILUt$ [Saa94b]. Los parámetros de usuario de esta factorización pueden ser sintonizados para obtener como mínimo una $ILU(0)$ y como máximo los factores completos. En el Capítulo §1, se describe con detalle esta factorización incompleta. Debido a todas estas aproximaciones, nuestro precondicionador tiene tres fuentes independientes de error: la aproximación $\tilde{A}_{II}^{-1}$, la matriz diezmada y la factorización incompleta de la matriz diezmada.

La matriz $M$ es construida explícitamente usando una paralelismo de paso de mensajes. Entonces, existe un proceso por cada dominio trabajando independientemente con las submatrices $A_{bi}^{(k)}$, $\tilde{A}_{ii}^{(k)}$ y $A_{ib}^{(k)}$. Se requiere alguna comunicación entre procesos para construir $M$ porque la matriz $A_{BB}$ también está distribuida. El $k$-ésimo proceso debe resolver con $\tilde{A}_{ii}^{(k)}$ tantos sistemas lineales como columnas no nulas de la matriz $A_{ib}^{(k)}$. La cantidad de columnas no nulas de la matriz $A_{ib}^{(k)}$ es igual a la longitud de la frontera del $k$-ésimo dominio. La figura 4.1 muestra los pasos del

algoritmo paralelo para calcular la matriz $M$.

CÁLCULO PARALELO DE $M$
*Para cada dominio k: (procesos paralelos)*
1.- Obtener $\tilde{A}_{ii}^{(k)-1}$
2.- Para cada columna no nula en $A_{bi}^{(k)}$ j: (*threads* paralelos)
2.1.- $M_j = A_{bi}^{(k)} * \tilde{A}_{ii}^{(k)-1} * A_{bi\ j}^{(k)}$,
2.2.- Reducción $M_j = A_{BBj} - M_j$
2.3.- Diezmar $M_j$

Figura 4.1: Algoritmo paralelo para calcular $M$.

Las soluciones de todos estos sistemas lineales pueden ser calculadas en paralelo. Para explotar este segundo nivel de paralelismo nosotros usamos *threads*. Entonces, dentro de cada proceso paralelo un conjunto de *threads* son creados para este cálculo. Es importante mencionar que no existen puntos de sincronización entre *threads*, que reducen la aceleración. La figura 4.2 se observa el esquema paralelo para 2 procesos PVM y 3 threads openMP por proceso.

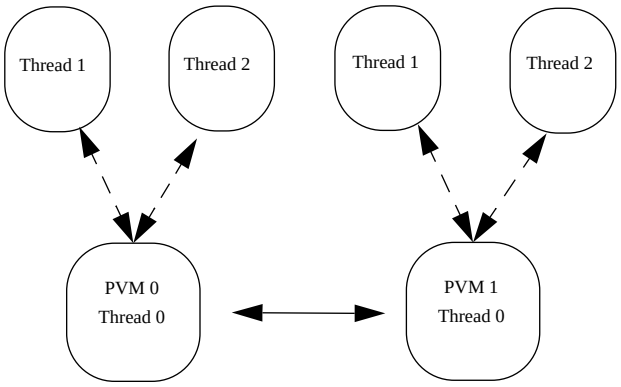



Figura 4.2: Esquema Paralelo.

En general, esta aproximación debe tener más operaciones que la factorizació $LU$ de la matriz completa usando un ordenamiento de llenado mínimo. Sin embargo, la cantidad de paralelismo y la reducción de operaciones debido a las aproximaciones es lo suficientemente grande para obtener una reducción significativa del tiempo de ejecución.

Como se menciono con anterioridad, algún criterio de diezmado debe ser aplicado a la matriz $M$ para tener un razonable número de operaciones en (4.6). Nosotros usamos un doble criterio, el primer criterio se basa en un umbral numérico relativo al máximo valor absoluto de los elementos de cada columna de la matriz $M$. Para definir este umbral, es necesario un parámetro de usuario llamado $tol_M$. Esta estrategia no garantiza que la cantidad de elementos resultante sea lo suficientemente pequeña. Por lo tanto, se usa un segundo criterio que establece: si el número de elementos es más grande que un número máximo sólo los más grandes serán seleccionados. La cantidad máxima de elementos permitidos en la columna $j$-ésima de $M$, llamado $NZmax$, se obtiene como:

$$NZmax = \text{MIN}(fill_M * \text{elementos no nulos } j\text{-ésima columna de } A_{BB}, n)$$

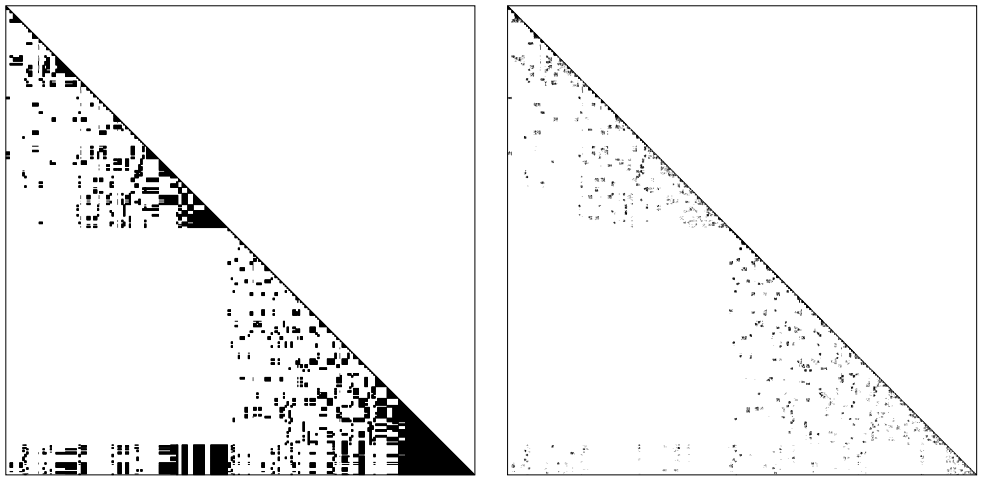
donde $n$ es la dimensión de $A_{BB}$ y $fill_M$ es un parámetro definido por el usuario tal que $fill_M > 0$. Si $fill_M = 1.0$, la matriz $M$ tendrá como máximo tantos elementos por columna que la matriz $A_{BB}$. El mismo criterio puede ser aplicado a filas en vez de columnas.

El punto clave de nuestro precondicionador algebraico es la aproximación usada para construir $\tilde{A}_{II}^{-1}$. Nosotros hemos aproximado la matriz $A_{II}^{-1}$ usando tres técnicas. La primera técnica, que llamaremos DFP (*Dropped Factorization Preconditioner*), usa los factores diezmados de las matrices $A_{ii}^{(k)}$ y la segunda técnica, que llamaremos AGMP (*Algebraic MultiGrid Preconditioner*), usa los espacios gruesos definidos por un método multinivel algebraico.

### 4.3.1 DFP

Esta aproximación usa los factores diezmados de las matrices $A_{ii}^{(k)}$. Estos factores son siempre calculados porque son necesarios para multiplicar la matriz de Schur por un vector. Obviamente, si estos factores no son fuertemente diezmados el costo computacional del precondicionador deberá ser similar que calcular la matriz de Schur explícitamente.

La estrategia de diezmado es aplicada a cada fila o columna de los factores, de tal forma que mantiene sólo un porcentaje pequeño de los elementos más significativos. Sin embargo, el número máximo de elementos no es uniforme para cada fila. Este es proporcional a la cantidad de nuevos elementos (llenado) en la fila correspondiente. Entonces, en las filas iniciales no se elimina cualquier elemento. En las filas finales es en donde se eliminan la mayor parte de elementos.



(a) (b)

Figura 4.3: Diezmado de un factor. (a) Factor original. (b) Factor diezmado.

En la figura 4.3 se puede observar un ejemplo de la operación de diezmado aplicada a un factor de un caso PERMAS [AJL+97]. La figura 4.3.(a) muestra el factor original, supongamos el factor $L_{ii}^{(1)}$. En la figura 4.3.(b) se puede observar el factor diezmado $L_{ii}^{DFP}$. Las zonas negras indican que existe llenado. Notese que las filas con pocos elementos, por ejemplos las filas iniciales, no se elimina cualquier elemento, aunque estos sean numéricamente insignificantes. Nosotros establecemos un número máximo de elementos para cada fila, si en una fila existen más elementos que ese máximo entonces los más grandes son seleccionados.

Este número máximo de elementos por fila deberá ser definido directamente como un parámetro de usuario. Pero, de esta forma no es sencillo sintonizar éste para un problema en específico. Por tal motivo, nosotros calculamos el máximo número de elementos por fila de la siguiente manera:

$$\text{máximo elementos por fila} = fill_F * \frac{NZ}{N}$$

donde $fill_F$ es un parámetro de usuario, $NZ$ es el total de elementos no nulos en el factor y $N$ es el número de filas del factor. Entonces, $\frac{NZ}{N}$ es la cantidad promedio de elementos por fila y $fill_F$ es el porcentaje de esta cantidad promedio. De esta manera, el rendimiento del algoritmo es menos sensitivo a variaciones pequeñas de $fill_F$, y el rango de $fill_F$ está claramente acotado. Nosotros encontramos que valores razonables para $fill_F$ están entre 0.1 y 0.3. Es importante mencionar que el paralelismo de *threads* puede ser usado también en este algoritmo de diezmado.

En la figura 4.4 se presenta el esquema paralelo usado para la construcción de este precondicionador. La estrategia de diezmado es aplicada a cada fila de los fac-

ALGORITMO DFP
*Para cada dominio k: (procesos paralelos)*
1.- Para cada fila $i$ (*threads* paralelos)
Diezmar la $fila_i$ de $L_{ii}^{(k)}$ usando $fill_F$
Diezmar la $fila_i$ de $U_{ii}^{(k)}$ usando $fill_F$

Figura 4.4: Agoritmo paralelo para diezmar los factores.

tores $L_{ii}^{(k)}$ y $U_{ii}^{(k)}$ en paralelo usando directivas openMP.

Es importante mencionar que los factores resultantes después del diezmado tienen menos elementos que una factorización $ILU(0)$.

### 4.3.2 AMGP

En la actualidad existen varios métodos multinivel algebraicos (AMG). Los más populares son presentados en [Stu99] [VMB96]. El AMG más general y robusto es el propuesto por K.Stuben en [Stu99]. En el capítulo §2, se describen estos métodos ampliamente. Como se observó en ese capítulo, la interacción entre los operadores de transferencia entre mallas y el método suavizador es la clave del buen rendimiento de un AMG. Hasta ahora, sólo esta interacción ha sido bien estudiada para M-matrices simétricas con diagonal dominancia débil usando los métodos de Gauss-Seidel o Jacobi como suavizadores. En [Stu99] se presentan algunos resultados experimentales para matrices no simétricas, pero siempre con diagonal dominancia débil.

Para construir nuestro precondicionador AMGP, nosotros usamos los espacios gruesos definidos por un método AMG. Para cada matriz $A_{ii}^{(k)}$ nosotros construimos un conjunto de espacios anidados definidos por un método AMG. Entonces, la matriz $A_{ii}^{(k)}$ es proyectada hacia el espacio más grueso. Para aproximar $x = {A_{ii}^{(k)}}^{-1} y$, se proyecta $y$ hacia el espacio más grueso para resolver un sistema lineal con la matriz más gruesa. Luego, se proyecta hacia el espacio más fino el vector solución, es decir:

$$x = I_{L-1} \cdots I_1 A_c^{-1} R_{L-1} \cdots R_1 y$$

donde $L$ es el número de niveles definidos por un método AMG, $A_c$ es la matriz más gruesa: $A_c = R_{L-1} \cdots R_1 A_{ii}^{(k)} I_1 \cdots I_{L-1}$, y $R_j$, $I_j$ con $j = 1, \cdots, L-1$ son los operadores de diezmado e interpolación en el $j$-ésimo nivel del AMG. La matriz más gruesa $A_c$ es factorizada para resolver el sistema lineal en el espacio más grueso.

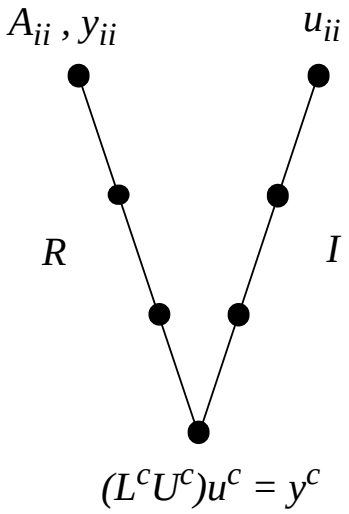



Figura 4.5: Esquema de cuatro niveles para cada dominio.

Estos factores son llamados $L^c$ y $U^c$.

En la figura 4.5 se puede observar un esquema de cuatro niveles para cada dominio. En el nivel más fino están la matriz $A_{ii}$ y el lado derecho $y_{ii}$. Cada punto negro en el esquema es un paso de diezmado, $R$, e interpolación, $I$. En el nivel más grueso se resuelve un sistema triangular con las matrices $L^c$ y $U^c$, como se menciona anteriormente. Luego, mediante los operadores de interpolación se proyecta el vector solución grueso, $u^c$, hacia el espacio más fino para obtener el vector $u_{ii}$, que es una aproximación al vector solución del sistema $A_{ii}x_{ii} = y_{ii}$.

Los métodos AMG completos pueden ser usados para construir una aproximación de las matrices $\tilde{A}_{ii}^{(k)^{-1}}$, pero esta aproximación tiene un número de operaciones inaceptables para los problemas tratados en esta tesis. Además, para varios de los problemas evaluados en este trabajo el AMG falla en convergencia en ambos problemas: en un simple dominio y en el problema completo. Esta falla de convergencia es debido a que la interacción entre el método suavizador y el operador de corrección grueso es deficiente. Sin embargo, los espacios gruesos pueden ser usados como una buena aproximación del problema original.

En la figura 4.6 se presenta el esquema paralelo usado para construir este precondicionador. En el paso 1 de este algoritmo, se realiza una fase inicialización. En esta fase se construyen los niveles gruesos y los operadores de transferencia, como

se explicó en el capítulo §2.

ALGORITMO AMGP
*Para cada dominio k: (procesos paralelos)*
1.- Fase de inicialización del método AMG
2.- Para cada nodo $j$ en el espacio más grueso (*threads* paralelos)
$col_j\ A^c_{ii} = R^c \cdots R^1 A_{ii} I^1 \cdots col_j\ I^c$
3.- Factorizar $A^c_{ii}$

Figura 4.6: Algoritmo paralelo del AMGP.

El tiempo que consume la construcción del conjunto de espacios anidados $(R_j, I_j)$ puede ser significativo comparado con el tiempo requerido para construir y factorizar la matriz $A^c$. Para un método AMG la anterior observación es critica para construir el conjunto de espacios gruesos que interactuan perfectamente con el operador de suavizado. Debido a esto, estrategias muy complejas han sido desarrolladas para la fase de inicialización. Sin embargo, nosotros no necesitamos estas estrategias complejas para mantener una razonable aproximación del espacio original porque no usamos ningún operador de suavizado. Nosotros usamos una fase de inicialización propuesta por Kickinger [Kic97]. En esta fase, la estrategia de engrosamiento es agresiva y sencilla. Aunque este método no es tan robusto como el propuesto por Stuben [Stu99], este construye unos espacios gruesos lo suficiente buenos para nuestros casos de prueba. Esta estrategia de engrosamiento ha sido descrita en el Capítulo §2.

### Método de Proyección basado en Wavelet

Otro posible método basado en proyecciones algebraicas de matrices hacia un conjunto de subespacios gruesos es la transforma wavelet 2D [PTVF94]. La transforma wavelet es un método de multiresolución en que descompone una función en diferentes escalas. Usualmente, esta técnica es usada en áreas de procesamiento de señales sismicas e imágenes.

Nosotros podemos usar este método para proyectar las matrices $A^{(k)}_{ii}$ hacia un espacio lo suficientemente grueso, como en el AMGP.

La transformada wavelet 2D de una matriz $A$ es:

$$WAW^t = \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \tag{4.7}$$

donde $W$ es una matriz ortogonal definida por un nivel de transformada wavelet. De esta manera, la expresión (4.7) es la representación de la matriz $A$ en una base wavelet. Existen diferentes bases, las más populares son las de Daubichies [PTVF94]. El bloque $A_1$ de la expresión (4.7) es una compresión de la matriz $A$ en un nivel de escala más pequeño. El algoritmo de transformación es realmente eficiente si la dimensión de $A$ es potencia de 2. El método se puede aplicar recursivamente al bloque $A_1$ para obtener varios niveles de escala de la matriz $A$. La matriz $A_1$ tiene exactamente la mitad de la dimensión de la matriz $A$. En la figura 4.7, se puede observar como es la estructura de la matriz $A$ después de haber aplicado 3 niveles de transforma wavelet no estandar (NS-forma) [BCR91]. En este caso, la dimensió de $A_3$ es $Dim(A) * 2^{-l}$, donde $l = 3$.

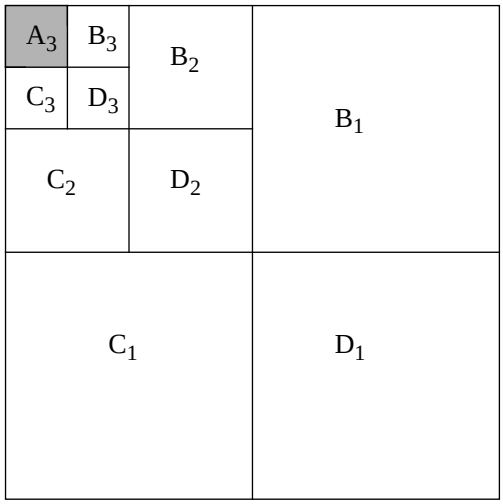



Figura 4.7: Tres niveles de transformada wavelet de la matriz $A$.

Este método no es atractivo para resolver sistemas lineales dispersos, ya que la transformada wavelet de una matriz dispersa es mucho más densa que dicha matriz. Aunque una gran cantidad de elementos de la matriz transformada no son numéricamente significativos. Sin embargo, para resolver sistemas lineales densos esta metodología ha motivado algún interés. Por ejemplo, En [GBD98] los autores usan la representación NS-forma para proponer un método directo para sistemas en donde la matriz asociada es densa.

Nosotros exploramos la posibilidad de construir un precondicionador basado en este método. La idea fue construir un sólo nivel de escala de la matriz $A_{ii}^{(k)}$ aplicandole la transformada wavelet 2D. De esta manera, se utilizó el bloque $A_1^{(k)}$ como una aproximación de la matriz $A_{ii}^{(k)}$. Desafortunadamente, esta aproximación no logró capturar los acoplos globales del problema de forma eficiente, y por lo tanto su aplicabilidad no fue competitiva para nuestros problemas de prueba.

Para más detalle acerca de la transformada wavelet 2D revisar [PTVF94].

### 4.3.3 Algoritmo Paralelo Propuesto

En esta sección vamos a describir con detalles todos los pasos del algoritmo paralelo de nuestro método de resolución.

Supongamos que la matriz $A$ ya está ordenada usando una descomposición en dominios. Además, supongamos que cada procesador tiene almacenado los bloques $A_{ii}^{(k)}$, $A_{ib}^{(k)}$, $A_{bi}^{(k)}$ y $A_{bb}^{(k)}$, donde $k = 1, \cdots, P$. El valor $P$ representa el número total de procesadores.

Se desea resolver el sistema reducido de Schur:

$$\tilde{S}\tilde{x}_B = \hat{f}_B$$

donde $\tilde{S} = SM^{-1}$, $\tilde{x}_B = Mx_B$ y $\hat{f}_B = f_B - A_{BI}A_{II}^{-1}f_I$. La matriz $M^{-1}$ la obtenemos usando cualquiera de los precondicionadores propuestos. Nosotros usamos el GMRES($m$) como método iterativo. La figura 4.8 se presenta el algoritmo paralelo de nuestro método de resolución.

Nosotros queremos ver que comuicación entre procesos requiere el algoritmo paralelo. Para ello, hemos dividido el algoritmo en 6 pasos.

El primer paso es calcular la factorización en paralelo de cada bloque $A_{ii}$. Este paso no requiere ningun tripo de comunicación entre procesos.

El segundo paso es diezmar los factores originales de cada matriz $A_{ii}$ para el DFP usando el algoritmo de la figura 4.4 ó construir los espacios gruesos para el AMGP usando el algoritmo de la figura 4.6. Este cálculo es completamente paralelo y no requiere ningun tipo de comunicación entre procesos.

El tercer paso del algoritmo es calcular la matriz $M$ usando el algoritmo de la figura 4.1. En este paso es necesario la comunicación entre vecinos. La matriz $M$

```
   DO k = 1, P
1)   Calcular:  A_ii^(k) = L_ii^(k) U_ii^(k)
c ...  Cálculo del Precondicionador
2)   IF (DFP) THEN
        Diezmar L_ii^(k), U_ii^(k)
     ELSE IF (AMGP)
        Calcular A_c^(k), R_ii^(k), I_ii^(k)
     ENDIF
3)   Calcular M
4)   Calcular la ILUt de M
c ...  Método Iterativo
     Elegir x_0; calcular r_0 = f̂_B − S̃ x_0
5)   WHILE (no convergencia)
       DO j = 1, m − 1
          DO i = 1, j
             h_ij =< S̃ w_j, w_i >
          ENDDO
          ŵ_{j+1} = S̃ w_j − Σ_{i=1}^{j} h_ij w_i
          h_{j+1,j} = ‖ŵ_{j+1}‖
          w_{j+1} = ŵ_{j+1} / h_{j+1,j}
          Calular la rotación de Givens
          Actualizar g y la matriz R
          IF (convergencia) GOTO 10
       ENDDO
  10   Calcular y_m = R^{-1} g
       Calcular x̃_B = x_0 + W y
     END WHILE
     Calcular:  x_B = M^{-1} x̃_B
c ...  Obtener x_I
6)   Resolver:  (L_ii^(k) U_ii^(k)) x_i^(k) = f_i^(k) − A_ib^(k) x_B
   ENDDO
```

Figura 4.8: Algoritmo paralelo del método.

queda distribuida en bloques de columnas.

El cuarto paso es calcular la factorización incompleta de $M$ para usar ésta como precondicionador. Antes de calcular la factorización incompleta de $M$, es necesa-

rio redistribuir $M$ de forma entrelazada. El tamaño del entrelazado depende de la relación cálculo-comunicación de la máquina usada. En el cálculo de la $ILUt$ de $M$ se necesitan comunicaciones globales. Los factores resultantes $L$ y $U$ quedan distribuidos de forma entrelazada.

El quinto paso es aplicar el método iterativo precondicionado para obtener una solución aproximada $x_B$. En este paso, se necesitan comunicaciones globales al resolver los sistemas triangulares en cada paso del método iterativo. Además, se requieren comunicaciones entre vecinos al multiplicar $S$ por un vector, en el caso de frontera simple. También, los productos escalar requieren una comunicación global.

El sexto paso es resolver en paralelo $k$ sistemas lineales para obtener la solución aproximada $x_I$. Este paso es completamente paralelo y por lo tanto no requiere ningun tipo de comunicación entre procesos.

Una observación importante es que el cálculo de la $ILUt$ y la resolución de sistemas triangulares representan un cuello de botella para nuestra implementación. Sin embargo, aunque esto reduce el paralelismo, el computo y la comunicación que esto requiere es poco en comparación con las otras partes del algoritmo. Por lo tanto, el paralelismo masivo de las otras partes del algoritmo es muy superior que este paralelismo reducido.

## 4.4 Resultados Numéricos

Se desea resolver el sistema lineal no singular $Ax = f$, donde $A$ proviene de un operador 3D escalar de convección-difusión, de la colección de Davis [Dav97] y del código PERMAS [AJL+97]. El vector lado derecho, $f$, es generado de tal manera que el vector solución, $x$, tiene valores aleatorios en $[0, 1]$. De esta manera, se puede medir el vector residuo y el vector error. Todos los cálculos son hechos en un supercomputador SGI Origin2000.

El criterio de parada para el método iterativo es $||r^k||_2/||f||_2 < 10^{-12}$, donde $r^k$ es el vector residuo en la $k$-ésima iteración y $f$ es el vector lado derecho. Hemos usado el paquete METIS [KK95] para el ordenamiento y partición del grafo de la matriz original.

La tabla 4.1 muestra la descripción de los problemas de prueba. En la primera columna se encuentra el nombre de los problemas. En la segunda columna, $n$ es la dimensión de las matrices. En la tercera columna, $n_z$ es el número de elementos

no nulos de las matrices. En la cuarta columna, se observa la procedencia de los problemas.

| Nombre | $n$ | $n_z$ | Descripción |
|---|---|---|---|
| C1 | 27.000 | 183.600 | ecuación 3D escalar de convección-difusión malla 30x30x30, $Cc = 10^3$ |
| C2 | 64.000 | 438.400 | ecuación 3D escalar convección-difusión malla 40x40x40, $Cc = 10^3$ |
| C3 | 125.000 | 860.000 | ecuación 3D escalar convección-difusión malla 50x50x50, $Cc = 10^3$ |
| D1 | 21.200 | 1.488.768 | Raefsky3 desde colección Davis |
| D2 | 19.779 | 1.328.611 | Raefsky4 desde colección Davis |
| D3 | 16.146 | 1.015.156 | Olafu desde colección Davis |
| P1 | 11.783 | 1.587.323 | Bearing desde código PERMAS |
| P2 | 25.906 | 786.380 | Railway desde código PERMAS |
| P3 | 48.162 | 2.386.056 | Methan desde código PERMAS |

Tabla 4.1: Descripción de los problemas de prueba.

El operador para la ecuación 3D escalar de convección-difusión es: $L(u) = \nabla(\alpha\nabla u) + \beta\nabla u$, donde el coeficiente de convección $Cc = \frac{\|\beta\|}{\|\alpha\|}$ mide la influencia de la convección. Nosotros usamos $Cc = 10^3$. Las constantes $\alpha$ y $\beta$ son isotrópicas. El dominio computacional es un cubo. La discretización es hecha usando diferencias finitas con un *7-stencil*. Para esta valor de $Cc$ las matrices obtenidas son fuertemente no simétricas. Las matrices obtenidas desde la colección de Davis también son

no simétricas. La matriz D1 proviene de un problema de interacción de estructuras de fluidos turbulentos. Las matrices D2 y D3 provienen de problemas de análisis estructural. Las matrices obtenidas desde el código de PERMAS son simétricas positivo definidas. Estas provienen de problemas de mecánica estructural. Todas las matrices de prueba no son débilmente diagonal dominantes.

Es importante mencionar que se ha intentado resolver este conjunto de problemas utilizando otros métodos. Específicamente,

1. Dos diferentes métodos AMG [Stu99] [VMB96] usados como métodos de resolución lineales.

2. GMRES [SS86] con diferente parámetro de re-inicialización usando la $ILUt$ [Saa94b] como precondicionador de la matriz completa.

3. GMRES con diferente parámetro de re-inicialización usando varios precondicionadores SPAI [GH97] [BMT96] [SC98] para la mariz completa.

4. Sistema de Schur. Usando el GMRES con diferente parámetro de re-inicialización precondicionado con la $ILUt$ [Saa94b] de la $A_{BB}$.

Los métodos AMG fallan en convergencia para C1, C2, C3, D1, D2 y D3. Para P1, P2 y P3 su convergencia es demasiado lenta. Entonces, la cantidad de trabajo computacional es muy alta. El precondicionador $ILUt$ de la matriz completa trabaja bien en todos los casos, pero la cantidad de llenado que el precondicionador requiere es prácticamente el mismo que en la factorización completa. Con los precondicionadores SPAI siempre es posible llegar a la convergencia, pero en la mayoría de estos problemas de prueba la matriz que se calcula es la inversa completa. Tampoco se tuvo éxito utilizando sólo el termino local para aproximar la matriz de Schur, es decir, un precondicionador $ILUt$ de la matriz $A_{BB}$. Este precondicionador requiere una cantidad de llenado muy grande si se desea una convergencia rápida del método iterativo. En general, ninguno de estos métodos son competitivos con respecto a los métodos directos para el conjunto de problemas usado en este trabajo.

Nosotros queremos ver que tan eficiente es el método de descomposición en dominios. Para ello podemos observar la tabla 4.2. En esta tabla se observa la descomposición en dominios de cada problema de prueba. El número de dominios se denota como $P$. Para $A_{ii}$, $n$ y $n_z$ representan valores promedios de dimensión y elementos no nulos, respectivamente. La desviación de este valor promedio es muy pequeña. Para cada problema el número de dominios es calculado tal que el número de nodos internos este entre 2.000 y 5.000 nodos. En la figura 4.9 hay una escala porcentual

de la tabla 4.2. Se puede observar el porcentaje de nodos internos y frontera generados por el método de descomposiciøn en dominios. El eje horizontal representa el nombre de los problemas y el eje vertical representa una escala porcentual. Se puede ver que el 80% de los nodos son nodos internos y sólo aproximadamente el 20% son nodos frontera. Sin embrago, METIS genera una frontera doble. Si se generará una frontera simple la cantidad de nodos fronteras se reduciría a la mitad. Además, en la figura 4.10 se muestra la eficiencia del método de descomposición en dominios en términos del porcentaje de elementos no nulos de los bloques $A_{II}$, $A_{IB}$, $A_{BI}$ y $A_{BB}$, generados por la descomposición. Se puede observar que también en este caso el método de descomposición es eficiente.

| Nombre | $P$ | $n\ A_{ii}^{(k)}$ | $n_z\ A_{ii}^{(k)}$ | $n\ A_{BB}$ | $n_z\ A_{BB}$ |
|---|---|---|---|---|---|
| C1 | 8 | 2.750 | 18.002 | 4.996 | 28.972 |
| C2 | 16 | 3.098 | 20.265 | 14.424 | 84.742 |
| C3 | 32 | 2.923 | 19.092 | 31.452 | 185.964 |
| D1 | 4 | 4.788 | 324.657 | 2.048 | 100.292 |
| D2 | 4 | 4.270 | 268.142 | 2.696 | 130.466 |
| D3 | 4 | 3.643 | 219.522 | 1.572 | 74.414 |
| P1 | 4 | 2.365 | 271.628 | 2.322 | 226.750 |
| P2 | 8 | 2.880 | 83.079 | 2.861 | 48.987 |
| P3 | 16 | 2.354 | 104.730 | 10.494 | 345.188 |

Tabla 4.2: Descomposición en dominios para cada problema.

Es importante comparar la cantidad de elementos no nulos generados por un método directo aplicado a la matriz $A$ usando orden de llenado mínimo (*multilevel nested dissection*) y un método directo aplicado a la matriz $A$ usando un orden basado en una descomposición en dominios.

En la tabla 4.3 se observa la cantidad de elementos no nulos de la factorización completa de $A$ usando un orden de llenado mínimo (*multilevel nested dissection*) y un orden basado en una descomposición en dominios. Se presentan sólo el número de elementos no nulos del factor $L$. Todas las matrices tienen estructura simétrica. Se observa que la principal parte del llenado adicional en el orden basado en descomposición en dominios está en las matrices $L_{BB}$ y $L_{BI}$. En la quinta columna de esta tabla se puede observar la diferencia entre los elementos no nulos de la matriz $L_{II}$ y la matriz completa $L$ usando un orden de llenado mínimo. Esta cantidad nos da una

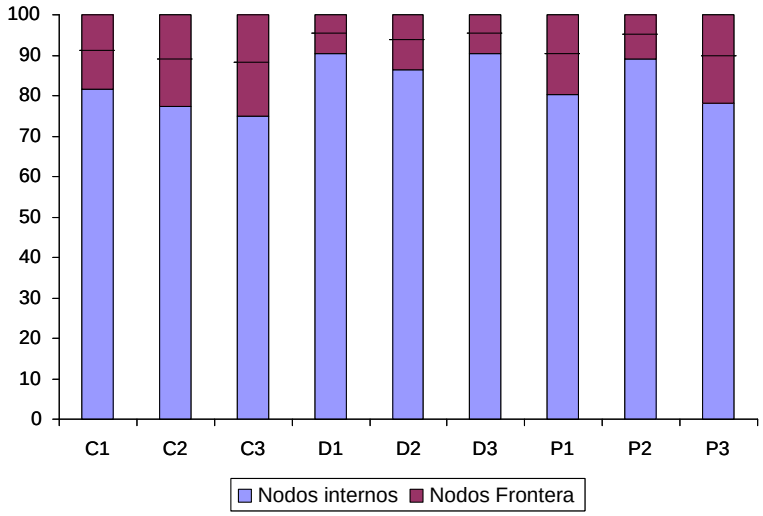



Figura 4.9: Eficiencia de la descomposición en cuanto a nodos.

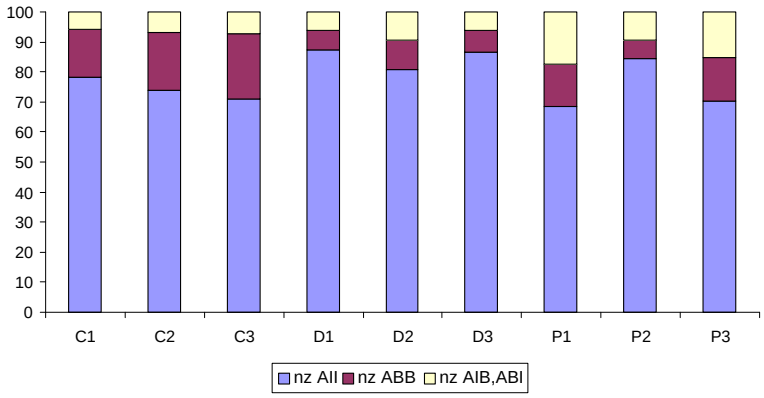



Figura 4.10: Eficiencia de la descomposición en cuanto a no nulos.

cota del número de operaciones que nosotros podemos gastar en el precondicionador y en el método iterativo para que nuestro método de resolución sea competitivo con respecto a un método directo. Esto es una cota pesimista porque la factorización de $A_{II}$ puede ser hecha en paralelo sin ningún tipo de comunicación entre procesos. Entonces el costo computacional es proporcional a los elementos no nulos del factor $L_{ii}^{(k)}$. En la figura 4.11 se muestra la anterior comparación en términos de porcentaje. El eje horizontal de la figura representa los problemas de prueba y el eje vertical una escala porcentual. El 100% representa al método directo usando una estrategia de mínimo llenado. Se puede ver que aplicar un método directo usando un orden basado en una descomposición en dominios nos es competivo en frente de un método directo usando un orden de llenado mínimo. Sin embargo, el porcentaje de no nulos del factor $L_{II}$ es menor que el porcentaje de no nulo del factor $L_{minFil}$. En la figura

4.12 se puede observar esto con más detalle.

| Nombre | $n_z\ L_{minFil}$ | $n_z\ L_{D.D.}$ | $n_z\ L_{ii}^{(k)}$ | $nz\ L_{minFil} - nz\ L_{II}$ |
|---|---|---|---|---|
| C1 | 3.244.154 | 10.870.755 | 135.032 | 2.163.898 |
| C2 | 13.844.258 | 49.973.542 | 153.433 | 11.389.330 |
| C3 | 37.855.331 | >3 Gbytes | 137.489 | 33.455.683 |
| D1 | 4.245.672 | 5.429.814 | 635.974 | 1.701.776 |
| D2 | 5.549.729 | 7.214.817 | 765.702 | 2.486.921 |
| D3 | 2.746.996 | 3.256.444 | 422.310 | 1.057.756 |
| P1 | 3.930.937 | 5.719.192 | 426.623 | 2.224.445 |
| P2 | 3.257.124 | 4.863.817 | 196.419 | 1.685.772 |
| P3 | 16.230.251 | 32.179.292 | 313.120 | 11.220.331 |

Tabla 4.3: Cantidad promedio de los elementos no nulos.

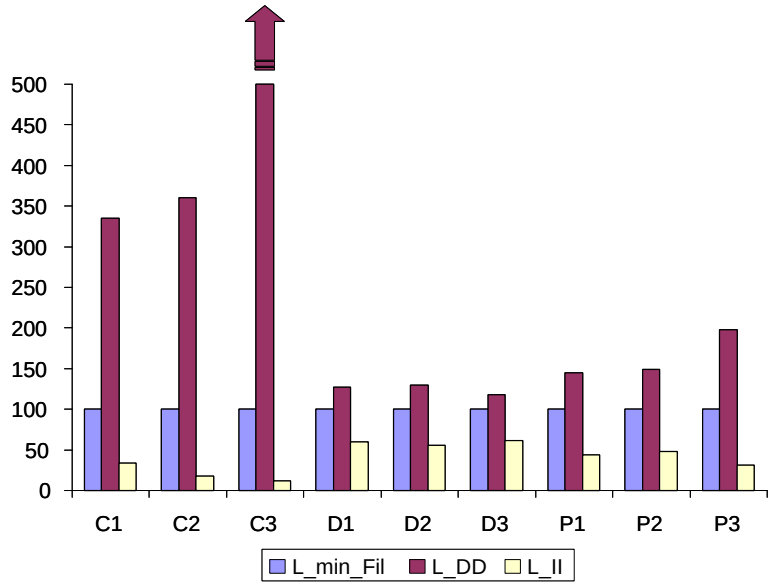



Figura 4.11: Llenado minimo vs DD.

La tabla 4.4 muestra el número de iteraciones hasta la convergencia del GMRES(50) aplicado al sistema de Schur, usando los precondicionadores DFP y AMGP. En ambos casos, $fill_M = 1.5$, y $tol_M = 10^{-4}$. La factorización ILUt de $M$ usa $fil_{ILU} = 10$, $tol_{ILU} = 10^{-3}$.

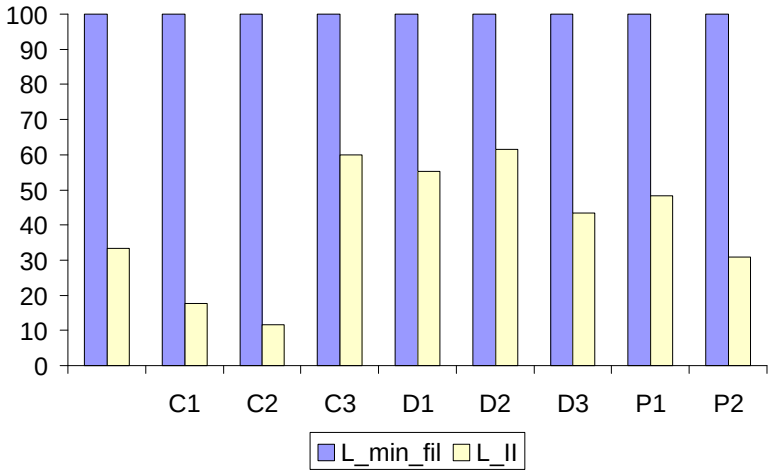



Figura 4.12: Llenado minimo vs $L_{II}$.

El precondicionador DFP usa $fill_F$ entre 0.1 y 0.3. En el AMGP, se ha fijado la dimensión del espacio más grueso menor o igual a $10^3$ nodos. Para los problemas C1, C2 y C3 los espacios más gruesos se alcanza construyendo 3 niveles. Para el resto de los problemas estos espacios se alcanza construyen sólo 2 niveles. Para todos los problemas de PERMAS, se obtiene una mejora en convergencia del GMRES si el AMGP usa un esquema de V-ciclo completo con 2 pasos Gauss-Seidel como método suavizador. Pero, entonces el número total de operaciones es inaceptable.

Aunque nuestro método de resolución tiene varios parámetros de usuario, el método no es muy sensitivo a variaciones en los parámetros $fill_M$, $tol_M$, $fil_{ILU}$ y $tol_{ILU}$. Estos parámetros pueden ser ajustados para un amplio rango de problemas de forma sencilla. Los parámetros criticos son $fill_F$ para el DFP, la dimensión del espacio más grueso para el AMGP y el número de niveles para la transformada wavelet. Estos parámetros necesitan ser sintonizados en cada problema. Para el AMGP, no es sencillo conocer el número de niveles óptimo, ya que esto depende esencialmente de la dimensión de problema. El DFP puede ser sintonizado con menos esfuerzo.

En tabla 4.5 se presenta la cantidad de no nulos de las distintas aproximaciones para $A_{II}$. La primera columna representa la cantidad promedio de los elementos no nulos de los factores $L_{ii}^{(k)}$ de las matrices $A_{ii}^{(k)}$. La segunda columna representa la cantidad promedio de los elementos no nulos de los factores diezmados $L_{ii}^{DFP}$ del DFP. La tercera columna representa la cantidad promedio de los elementos no nulos de los factores gruesos, $L_{ii}^{c}$, usados para construir el precondicionador AMGP. También, se muestran las medidas de complejidad de malla, $C_G$, y algebraica, $C_A$,

| Nombre | DFP | AMGP |
|---|---|---|
| C1 | 22 | 148 |
| C2 | 24 | 95 |
| C3 | 23 | 72 |
| D1 | 44 | 148 |
| D2 | 20 | 280 |
| D3 | 83 | 195 |
| P1 | 42 | 134 |
| P2 | 78 | 330 |
| P3 | 33 | 250 |

Tabla 4.4: Iteraciones GMRES para los diferentes problemas.

para el AMGP. Estas medidas de complejidad son definidas como: $C_G = \sum_j \frac{n_j}{n_1}$ y $C_A = \sum_j \frac{nz_j}{nz_1}$, donde $n_j$ y $nz_j$ denotan el número de variables y el número de no nulos sobre el nivel $j$, respectivamente. Aunque el verdadero requerimiento de memoria para un AMG no está completamente reflejado en estas cantidades, se necesita espacio de trabajo extra, estas cantidades están muy cercanas a la realidad.

| Nombre | $n_z$ $L_{ii}$ | $n_z$ $L_{ii}^{DFP}$ | $n_z$ $L_{ii}^{c}$ | $C_G$ | $C_A$ |
|---|---|---|---|---|---|
| C1 | 135.032 | 35.237 | 7.428 | 1,46 | 2,25 |
| C2 | 153.433 | 33.434 | 8.613 | 1,52 | 2,33 |
| C3 | 137.489 | 31.736 | 7.418 | 1,50 | 2,30 |
| D1 | 635.974 | 110.706 | 1.972 | 1,02 | 1,003 |
| D2 | 765.702 | 96.647 | 4.989 | 1,04 | 1,011 |
| D3 | 422.310 | 85.774 | 2.222 | 1,03 | 1,007 |
| P1 | 426.623 | 56.825 | 2.305 | 1,03 | 1,008 |
| P2 | 196.419 | 49.154 | 21.569 | 1,11 | 1,077 |
| P3 | 313.120 | 51.986 | 8.556 | 1,08 | 1,071 |

Tabla 4.5: Promedio de no nulos para los factores y operadores de complejidad.

Se puede observar que el algoritmo DFP elimina entre de 75% y 90% de los elementos originales de los factores $L_{ii}$. Lo que implica una reducción significativa

de operaciones para construir $M$. Los factores gruesos generados por el AMGP tienen menos elementos que los factores DFP. Por lo tanto, el número de operaciones requeridas para construir $M$ es menor que en el DFP. Además, las medidas de complejidad $C_G$ y $C_A$ indican que la estrategia de engrosamiento es óptima para la mayoría de los casos, ya que lo ideal es que estas medidas estén muy cercanas a 1. Entonces, los requerimientos de memoria son mínimos para estos casos. Sin embargo, observando la tabla 4.4 podemos ver que existe una significativa diferencia en el número de iteraciones entre el DFP y el AMGP.

En la figura 4.13 se puede observar una representación porcentual de la tabla 4.5. El eje horizontal representa los problemas de prueba y el eje vertical representa una escala porcentual. El 100% representa la cantidad de elementos no nulos del factor $L_{II}$, después de usar una estrategia de llenado mínimo. Se puede observar con mayor claridad la reducción significativa de operaciones de nuestras propuestas.

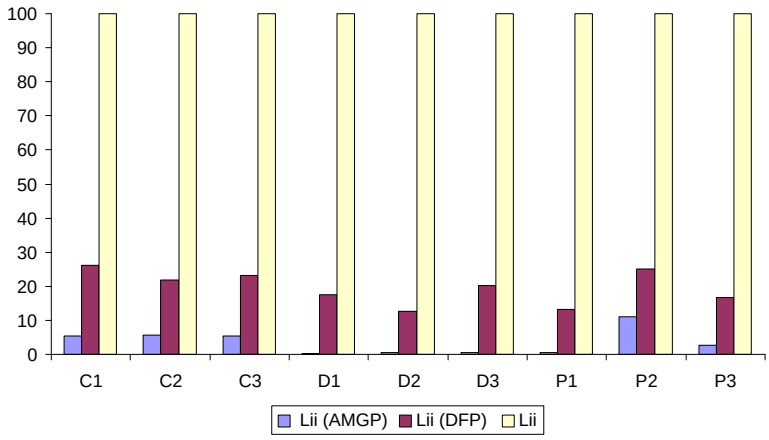



Figura 4.13: Aproximaciones para $A_{II}$.

Es muy importante que nuestra aproximación $M$ tenga una cantidad similar de elementos no nulos con respecto de $A_{BB}$. En la tabla 4.6 se muestra la densidad de la matrices $A_{BB}$, $M_{DFP}$ y $M_{AMGP}$. Se puede observar que el número de no nulos de las matrices $M_{DFP}$ y $M_{AMGP}$ es similar al número de elementos no nulos de la matriz $A_{BB}$. En la figura 4.14 se puede observar una representación en gráfica de barra de esta tabla.

Nuestro método de resolución tiene que ser competitivo en frente de los métodos usados en los paquetes comerciales, como hemos dicho estos métodos son los métodos directos. En la tabla 4.7 se observa el número total de operaciones en Mflops que requiere nuestro método de resolución iterativo ejecutandose de forma secuencial,

| Nombre | $A_{BB}$ | $M_{DFP}$ | $M_{AMGP}$ |
|---|---|---|---|
| C1 | 0,12 | 0,20 | 0,17 |
| C2 | 0,04 | 0,07 | 0,06 |
| C3 | 0,02 | 0,03 | 0,03 |
| D1 | 2,39 | 2,30 | 1,49 |
| D2 | 1,79 | 2,20 | 2,61 |
| D3 | 3,01 | 3,82 | 4,38 |
| P1 | 4,21 | 2,58 | 6,29 |
| P2 | 0,60 | 0,70 | 0,87 |
| P3 | 0,31 | 0,57 | 0,46 |

Tabla 4.6: Densidad (%) de $A_{BB}$, $M_{DFP}$ y $M_{AMGP}$.

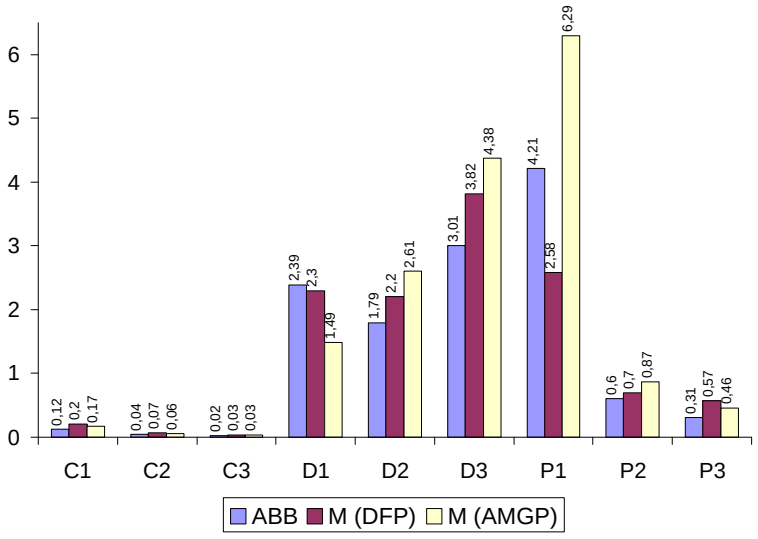



Figura 4.14: Densidad de $M$.

es decir, sin solapamiento de operaciones. En la columna 2, se presenta el número total de operaciones de las factorizaciones $A_{ii}^{(k)}$. En la columna 3 y 4, se presenta el número total de operaciones para obtener la matriz $M$. Esta matriz se obtiene usando las aproximaciones propuestas. En la columna 5, se presenta el número de operaciones para la factorización incompleta de la matriz $M$. Por último, en la columna 6, se presenta el número total de operaciones para una iteración del GMRES.

En la tabla 4.8 se muestra el número total de operaciones (Mflpos) de un método directo (*LU* o *Cholesky* dependiendo del caso) y de nuestro método de resolución

| Nombre | Fact. $A_{ii}^{(k)}$ | $M$ | | ILU($M$) | GMRES |
|---|---|---|---|---|---|
| | | $M_{DFP}$ | $M_{AMGP}$ | | |
| C1 | 259,45 | 520,80 | 461,40 | 0,98 | 4,80 |
| C2 | 586,72 | 1.779,55 | 1.582,70 | 2,86 | 11,24 |
| C3 | 990,30 | 3.997,20 | 3.683,35 | 6,48 | 207,75 |
| D1 | 893,74 | 1.429,90 | 178,70 | 5,97 | 11,46 |
| D2 | 1.537,80 | 2.317,50 | 344,25 | 11,24 | 14,44 |
| D3 | 526,30 | 822,20 | 125,54 | 6,83 | 8,06 |
| P1 | 406,10 | 1.415,65 | 383,56 | 35,42 | 9,83 |
| P2 | 182,70 | 622,40 | 518,52 | 2,58 | 7,28 |
| P3 | 924,78 | 4.401,60 | 1.578,87 | 26,20 | 24,70 |

Tabla 4.7: Número total de operaciones secuenciales.

iterativo, sin solapamiento de operaciones, usando las aproximaciones DFP y AMGP para alcanzar la convergencia observada en la tabla 4.4. Se puede observar que para C1, C2 y C3 nuestro método de resolución tiene una reducción significativa de operaciones. Para D1, D2, D3, P1, P2 y P3 nuestro método de resolución tiene similar número de operaciones que el método directo. Entonces, si nosotros explotamos el paralelismo natural de nuestro método de resolución, se puede obtener una significante reducción del tiempo de ejecución.

| Nombre | Directo | Solver | |
|---|---|---|---|
| | | DFP | AMGP |
| C1 | 3.064,92 | 886,83 | 1.432,23 |
| C2 | 30.150,25 | 2.638,89 | 3.240,08 |
| C3 | 131.600,45 | 9.772,23 | 19.638,13 |
| D1 | 2.637,42 | 2.833,85 | 2.274,49 |
| D2 | 4.785.16 | 4.155,34 | 5.936,49 |
| D3 | 1.496,96 | 2.024,31 | 2.230,37 |
| P1 | 1.783,73 | 2.270,03 | 2.142,30 |
| P2 | 854,88 | 1.375,52 | 3.106,20 |
| P3 | 9.745,82 | 6.167,68 | 8.704,85 |

Tabla 4.8: Número total de operaciones (Mflops) para convergencia.

En la figura 4.15 se puede observar una representación porcentual de la tabla 4.8. El eje horizontal representa los problemas de test y el eje vertical representa una escala porcentual. El 100% representa la cantidad de operaciones del método directo.

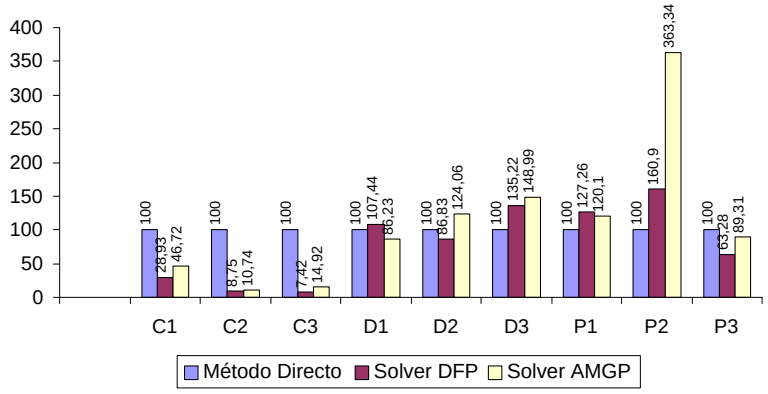



Figura 4.15: Operaciones secuencial.

En la tabla 4.9, se muestra el número de operaciones en paralelo, es decir, con solapamiento de operaciones para cada parte de nuestro método de resolución asumiendo un perfecto solapamiento de operaciones entre procesos paralelos, donde este solapamiento es teóricamente posible. Se asume que existen tantos procesos paralelos como dominios. El número de operaciones de la factorización incompleta de la matriz $M$ es el mismo que en la tabla 4.7, ya que esta factorización se realiza de forma secuencial. También, todas las operaciones en el GMRES son consideradas secuenciales excepto el producto matriz por vector. Esta tabla no considera que existe un importante segundo nivel de solapamiento de operaciones debido a los *threads* paralelos.

En la figura 4.16 se puede observar un representación porcentual de esta tabla. El eje horizontal representa los problemas de prueba y el eje vertical representa una escala porcentual. En esta figura se puede ver la reducción significativa de operaciones de nuestro método en comparación con un método directo.

Es importante conocer como están distribuidas las operaciones de ambos precondicionadores. En la figura 4.17 y 4.18, se puede observar esta distribución. Para el DFP la mayor cantidad de operaciones están en la construcción de la matriz $M$ y para el $AMGP$ la mayor cantidad de operaciones están en el método iterativo. Esto indica que el DFP es mejor precondiconador que el AMGP. Además, la factorización incompleta de $M$ es despreciable en número de operaciones.

| Nombre | Fact. $A_{ii}^{(k)}$ | $M$ | | GMRES | Total | |
|---|---|---|---|---|---|---|
| | | $M_{DFP}$ | $M_{AMGP}$ | | DFP | AMGP |
| C1 | 32,43 | 65,10 | 57,67 | 0,98 | 99,49 | 92,06 |
| C2 | 36,67 | 111,22 | 98,91 | 1,90 | 152,65 | 140,34 |
| C3 | 30,94 | 124,91 | 115,10 | 3,39 | 165,72 | 155,91 |
| D1 | 223,43 | 357,47 | 44,67 | 3,63 | 590,50 | 277,70 |
| D2 | 384,45 | 579,37 | 86,06 | 4,96 | 980,02 | 486,71 |
| D3 | 131,57 | 205,55 | 31,38 | 2,81 | 346,76 | 172,59 |
| P1 | 101,52 | 353,91 | 95,89 | 3,97 | 494,82 | 236,80 |
| P2 | 22,83 | 77,80 | 64,81 | 3,82 | 107,03 | 94,04 |
| P3 | 57,79 | 275,10 | 98,67 | 4,65 | 363,74 | 187,31 |

Tabla 4.9: Número de operaciones (Mflops) en paralelo.

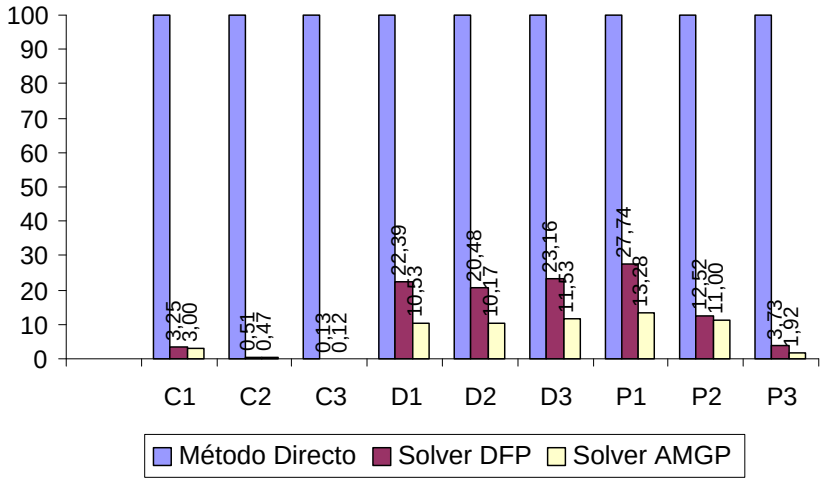



Figura 4.16: Operaciones en paralelo.

La tabla 4.10 muestra el número de iteraciones requeridas, aproximadamente, por el GMRES para que el método de resolución paralelo DFP o AMGP tengan un número total de operaciones similar a un método directo. Un valor negativo en la tabla indica que nuestro método no es competitivo, y en ese caso tiene mucho más operaciones que un método directo. Esto se puede observar para P1 usando el DFP sin solapamiento. Se puede observar que al solapar operaciones, nuestro método otorga un margen amplio de holgura para aumentar el parámetro $fill_F$ de la aproximación DFP, aumentar el llenado de la matriz $M_{DFP}$ o $M_{AMGP}$ y/o aumentar el llenado de la factorización incompleta de la matriz $M$. Comparando las tablas 4.3

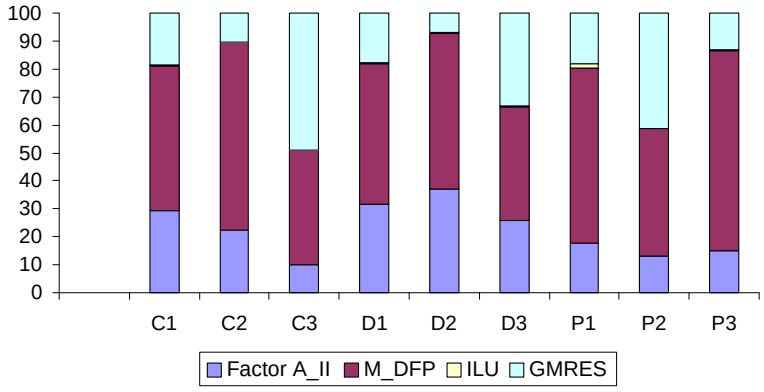



Figura 4.17: Distribución de operaciones para el DFP.

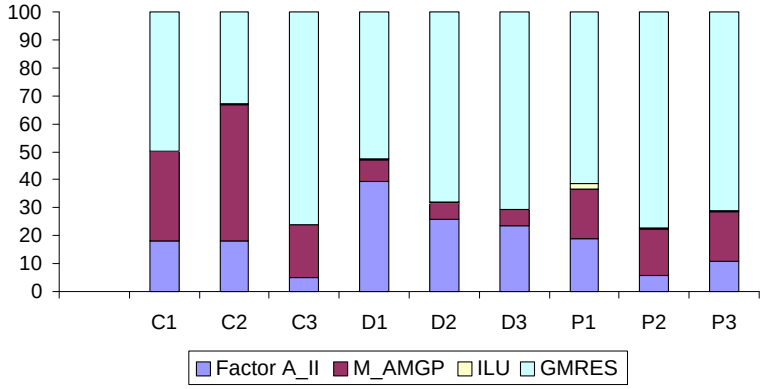



Figura 4.18: Distribución de operaciones para el AMGP.

y 4.10 se puede observar que nuestro método es competitivo para todos los casos de prueba. Además, tanto el DFP como el AMGP requieren similar cantidad de trabajo.

Es importante mencionar que, el AMGP realiza más trabajo en el método iterativo y el DFP realiza más trabajo en la construcción del precondicionador $M$. Nosotros preferimos al DFP porque es más sencillo de sintonizar para un problema especifico. Además, en el limite siempre es posible alcanzar convergencia con éste, aunque el número de operaciones sea muy grande.

En la tabla 4.11 se muestra la aceleración obtenida en la construcción del precondicionador DFP. Para el AMGP la aceleración obtenida es similar. La tabla muestra la aceleración usando diferentes números de *threads* por proceso. Para cada problema, el número de procesos se muestra en la primera columna. Nosotros estamos limitados a 32 CPUs.

| Nombre | DFP | | AMGP | |
|---|---|---|---|---|
| | Sin solapamiento | Con solapamiento | Sin solapamiento | Con solapamiento |
| C1 | 475 | 3.026 | 488 | 3.034 |
| C2 | 2.471 | 15.789 | 2.489 | 15.795 |
| C3 | 609 | 38.772 | 610 | 38.775 |
| D1 | 26 | 564 | 136 | 651 |
| D2 | 63 | 768 | 200 | 867 |
| D3 | 17 | 410 | 104 | 472 |
| P1 | -7 | 325 | 97 | 390 |
| P2 | 6 | 196 | 20 | 200 |
| P3 | 177 | 2.018 | 292 | 2.056 |

Tabla 4.10: Máximo número de iteraciones GMRES sin y con solapamiento.

| Nombre | 1 *Thread* | 2 *Thread* | 4 *Thread* | 8 *Thread* |
|---|---|---|---|---|
| C1 (8 procesos) | 7,45 | 15,30 | 30,25 | - |
| C2 (16 procesos) | 15,35 | 31,95 | - | - |
| C3 (32 procesos) | 31,75 | - | - | - |
| D1 (4 procesos) | 3,80 | 7,90 | 15,85 | 27,70 |
| D2 (4 procesos) | 3,90 | 7,60 | 15,95 | 30,45 |
| D3 (4 procesos) | 3,95 | 7,72 | 15,80 | 31,35 |
| P1 (4 procesos) | 3,65 | 7,95 | 15,70 | 31,05 |
| P2 (8 procesos) | 7,55 | 15,35 | 30,35 | - |
| P3 (16 procesos) | 15,85 | 30,60 | - | - |

Tabla 4.11: Aceleración obtenida para el precondicionador DFP.

El número de CPUs usados es igual al número de procesos por el número de *threads* por proceso. Esto es:

$$\text{Total CPUs} = \text{total procesos} \times (\text{total threads por proceso})$$

La aceleración es medidad con respecto al mismo programa ejecutado en un procesador. Esta medida muestra la escalabilidad de nuestro método, que como se

puede observar es cuasi-lineal hasta 32 CPUs.

Una medida más interesante es comparar los tiempos de ejecución de nuestro método y un método directo paralelo con una buena estrategia de *parallel scheduler*. Desafortunadamente, no tenemos disponible un buen método directo paralelo.

Nosotros usamos un secuenciamiento dinámico openMP para los bucles paralelos. Esto es porque en nuestra implementación, el bucle, en toda las columnas de la matriz $A_{ib}^{(k)}$, es hecho sobre la frontera completa, y no sobre la frontera del $k$-ésimo dominio. Sin embargo, esto no introduce un *overhead* significativo ya que el balanceo de carga es muy bueno. Con un código escrito más eficiente es posible usar un secuenciamiento estático. Esto es importante especialmente si esta técnica es implementada en ambientes más complejos, como métodos *out of core*, donde los *threads* son manejados directamente por el programador. Además, los *threads* acceden las rutinas de paso de mensaje con exclusión mutua, ya que existe la necesidad de comunicación entre *threads* de distintos procesos.

# Conclusiones

Esta tesis trata sobre la resolución de sistemas lineales de ecuaciones mediante descomposición en dominios. Estos sistemas son provenientes de la resolución de ecuaciones diferenciales en derivadas parciales.

La descomposición en dominios permite un doble enfoque numérico para la resolución de sistemas lineales: usar la matriz completa o usar la matriz de complemento de Schur. De estos dos enfoques numéricos, el primero está limitado por el uso de precondicionadores basados en factorizaciones incompletas que reducen la eficiencia debido a la secuencialidad de la resolución de los sistemas triangulares. El segundo enfoque, está limitado por la carencia de precondicionadores de propósito general. Para el primer enfoque, existen precondicionadores basados en factorizaciones incompletas que permite al usuario seleccionar que tipo de dependencia de datos son aceptadas, y en consecuencia permite reducir la secuencialidad de las comunicaciones que deben realizarse a la hora de resolver los sistemas triangulares. En cambio, para el segundo enfoque el problema aun no esta resuelto. En tal sentido, en esta tesis se proponen varios precondicionadores de carácter general para la matriz de complemento de Schur.

Específicamente, en esta tesis se proponen dos precondicionadores algebraicos paralelos para el Sistema Complemento de Schur, llamados DFP y AMGP. Estos precondicionadores permiten capturar información de los acoplos globales del problema. El precondicionador DFP es construido usando los factores fuertemente diezmados de las matrices $A_{ii}^{(k)}$, y el precondicionador AMGP se construye usando los espacios gruesos generados por un método multinivel algebraico. Para la construcción de estos precondicionadores se explotan dos niveles de paralelismo: paso de mensajes y *threads*. Esto permite explotar al máximo el rendimiento de los supercomputadores actuales.

Los dos precondicionadores propuestos han demostrado ser robustos para el conjunto de problemas usados en esta tesis, frente al otro precondicionador propuestos, y frente a las técnicas algebraicas de precondicionamiento que existen en la actuali-

dad, usadas cuando es necesario obtener una aproximación del término global de la matriz de Schur. Además, la matriz $M$ obtenida tiene similar cantidad de elementos no nulos que la matriz $A_{BB}$.

El precondicionador DFP es más robusto que el precondicionador AMGP. Además, nuestro método de resolución es competitivo con respecto a los métodos directos.

Las principales aportaciones de este tesis son:

- Se ha realizado un estudio experimental de viabilidad de los precondicionadores SPAI.

- Se ha realizado un estudio de convergencia de los AMG en casos industriales.

- Se ha hecho un análisis de efectividad de la Transformada wavelet como precondicionador para el sistema complemento de Schur.

- Propuesta de dos precondicionadores algebraicos para el sistema complemento de Schur, el DFP y el AMGP. El primero, está basado en una factorización fuertemente diezmada. Este diezmado es controlado por medio del parámetro $fill_F$. Mientras $fill_F$ es más grande, los factores diezmados tienden a ser los factores originales. De esta manera, se puede obtener una muy buena aproximación de los factores originales. Este precondicionador posee una alta escalabilidad. El segundo, está basado en técnicas multinivel algebraicas. En general, este precondicionador es menos robusto que el DFP, pero puede ser atractivo en muchos casos. Ambos precondicionadores son de carácter algebraico.

- Propuesta de un esquema de paralelización para la construcción del precondicionador. Este esquema está formado por dos niveles de paralelismo: paso de mensaje y *threads*. Esto permite obtener una mayor escalabilidad y explotar al máximo el rendimiento de los supercomputadores actuales.

El trabajo futuro que se abre a la vista de los resultados de esta tesis son:

- Estudiar otras heuristicas para diezmar los factores del DFP.

- Realizar más pruebas experimentales con problemas industriales.

- Implementar un código más eficiente.

# Bibliografía


[Ago88] V. I. Agoshkov. Poincar-Steklov's operators and Domain Decomposition Methods in Finite Dimensional Spaces. In *1st International Symposium on Domain Decomposition Methods for Partial Differential Equations*, number ISBN 0-89871-220-3. SIAM, 1988.

[AJL+97] M. Ast, T. Jerez, J. Labarta, H. Manz, A. Perez, V. Schulz, and J. Sole. Runtime Parallelization of the Finite Element Code PERMAS. *Int. Jour. of Supercomputing Applications and High Performance Computing*, 11(4), 1997.

[AMO92] S. Ashby, T. Manteuffel, and J. Otto. A Comparison of Adaptive Chebyshev and Least Squares Polynomial Preconditioning for Hermitian Positive Definite Linear Systems. *SIAM J. Sci. Stat. Comput.*, 13(1):1–29, 1992.

[AMS89] S. Ashby, T. Manteuffel, and P. Saylor. Adaptive Polynomial Preconditioning for Hermitian Indefinite Linear Systems. *BIT*, 29:583–609, 1989.

[AMS90] S. Ashby, T. Manteuffel, and P. Saylor. A Taxonomy for Conjugate Gradient Methods. *SIAM Jour. Numer. Analysis*, 27(6):1542–1568, 1990.

[Arn51] W. E. Arnoldi. The Principle of Minimized Iteration in the Solution of Matrix Eigenvalue Problem. *Quart. Appl. Math*, 9:17–29, 1951.

[AV89a] O. Axelsson and P.S. Vassilevski. A Survey of Multilevel Preconditioned Iterative Methods. *BIT*, 29:769–793, 1989.

[AV89b] O. Axelsson and P.S. Vassilevski. Algebraic Multilevel Preconditioning Methods I. *Numer. Math.*, 56:157–177, 1989.

[AV90] O. Axelsson and P.S. Vassilevski. Algebraic Multilevel Preconditioning Methods II. *SIAM Journal on Numer. Anal.*, 27(6):1564–1590, 1990.

[AV91] O. Axelsson and P.S. Vassilevski. Asymptotic Work Estimates for AML Methods. *Applied Numerical Mathematics*, 7:437–451, 1991.

[Axe85] O. Axelsson. A Survey of Preconditioned Iterative Methods for Linear Systems of Algebraic Equations. *BIT*, 25:166–187, 1985.

[Axe94] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, 1994.

[Bak66] N. S. Bakhvalov. On the Convergence of a Relaxation Method with Natural Constrains on the Elliptic Operator. *URSS Computational Math. and Math. Phys.*, 6:101–135, 1966.

[BB87] M. Berger and S. Bokhari. A Partitioning Strategy for Nonuniform Problems on Multiprocessors. *IEEE Trans. on Computers*, C-36(5), 1987.

[BCMM95] R. Bru, C. Corral, A. Martinez, and J. Mas. Multisplitting Preconditioners based on Incomplete Choleski Factorizations. Technical report, Departamento de Matemtica Aplicada, Universidad de Valencia, 1995.

[BCR91] G. Beylkin, R. Coifman, and V. Rokhlin. Fast Wavelet Transform and Numerical Algorithms I. *Comm. on Pure and Applied Mathematics*, 44:141–183, 1991.

[Ber89] T. Berglind. Multi-Block Euler Method using Patched Grids. In *SIAM 2th International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1989.

[BMT94] M. Benzi, C. Meyer, and M. Tuma. A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method. Technical report, Mathematics Department, North Carolina State University, 1994.

[BMT96] M. Benzi, C. Meyer, and M. Tuma. A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method. *SIAM J. Scientific Computing*, 17(3):1135–1151, 1996.

[Bra77] A. Brandt. Multi-Level Adaptive Solutions to Boundary-Value Problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[Bra95] D. Braess. Towards Algebraic Multigrid for Elliptic Problems of Second Order. *Computing*, (55):379–393, 1995.

[Bri87] W. L. Briggs. *A Multigrid Tutorial.* SIAM, 1987.

[BW89] P. Bjorstad and O. Widlund. To Overlap or not to Overlap: A note on a Domain Decomposition Method for Elliptic Problems. *SIAM Jour. Sci. and Stat. Computing*, 10(5):1093–1061, 1989.

[BW93] P. Bjorstad and O. Widlund. Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures. *SIAM Journal on Numerical. Analysis*, 23(6):1093–1120, 1993.

[Cha87] T. Chan. Analysis of Preconditioners for Domain Decomposition. *SIAM Jour. on Numer. Analysis*, 24(2), 1987.

[Cha89] T. Chan. Boundary Probe Domain Decomposition Preconditioners for Fourth Order Problems. In *SIAM 2th International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1989.

[CIKM94] S. Campbell, I. Ipsen, C. Kelley, and C. Meyer. GMRES and the Minimal Polynomial. Technical report, Center for Research in Scientific Computing, North Carolina State University, 1994.

[CK90] T. Chan and D. Keyes. Interface Preconditionings for Domain-Decomposed Convection-Diffusion Operators. In *SIAM 3th International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1990.

[CM94] T. Chan and T. Mathew. *Domain Decomposition Algorithms*, volume 2. Acta Numerica, Cambridge University Press, 1994.

[CN92] J.M. Cela and J. J. Navarro. Performance Model for Algebraic Multilevel Preconditioner on a Shared Memory Multicomputers. In *PACTA92*, 1992.

[CR87] T. Chan and D. Resasco. A Domain-Decomposed Fast Poisson Solver on a rectangle. *SIAM Jour. on Sci. Stat. Comput.*, 8(1), 1987.

[CS97] E. Chow and Y. Saad. Approximate Inverse Techniques for Block-Partitioned Matrices. *SIAM J. Sci. Comput.*, 18(6):1657–1675, 1997.

[Dav97] T. Davis. University of Florida Sparse Matrix Collection. *NA Digest 1997, also available online at http://www.cise.ufl.edu/∼davis/sparse*, 1997.

[DGK84] J. Dongarra, F. Gustavson, and A. Karp. Implementing Linear Algebra Algorithms for Dense Matrices on a Vector Pipeline Machine. *SIAM Review*, 26:91–112, 1984.

[DJF90] W. Dietz, J. Jacocks, and J. Fox. Application of Domain Decomposition to the Analysis of Complex Aerodynamic Configurations. In *SIAM 3th International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1990.

[DMM95] R. Diekmann, D. Meyer, and B. Monien. Parallel Decomposition of Unstructred FEM-Meshes. Technical report, Department of Mathematics and Computer Science, University of Padervorn, 1995.

[Dry82] M. Dryja. A Capacitance Matrix Method for Dirichlet Problem on Polygon Region. *Numerische Mathematik*, 39:51–64, 1982.

[DSW93] M. Dryja, B. F. Smith, and O. B. Widlund. Schawrz Analysis of Iterative Substructuring Algorithms for Elliptic Problems in Three Dimensions. Technical Report 638, Department of Computer Science, Courant Institute, 1993.

[dV82] H. A. Van der Vost. A Vectorizable Variant of some ICCG Methods. *SIAM J. Sci. Stat. Comput.*, 3(3), 1982.

[dV92] H. A. Van der Vost. Bi-CGstab: A Fast and Smoothly Converging Variant of the BiCG for the Solution of Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 13(2), 1992.

[DW92] M. Dryja and O. B. Widlund. Additive Schawrz Methods for Elliptic Finite Element Problems in Three Dimensions. In *SIAM 5th International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1992.

[Ece88] A. Ecer. Block-Structured Solution of Transonic Flows. In *SIAM 1th International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1988.

[Eij90a] V. Eijkhout. Analysis of Parallel Incomplete Point Factorizations. Technical Report 1045, CSRD, 1990.

[Eij90b] V. Eijkhout. Beware of Modified Incomplete Point Factorizations. Technical Report 1048, CSRD, 1990.

[Elm89] H. Elman. Approximate Schur Complement Preconditioners on Serial and Parallel Computers. *SIAM J. Sci. Stat. Comput.*, 10(3), 1989.

[EOV90] S. Eisenstat, J. Ortega, and C. Vaughan. Efficient Polynomial Preconditioning for the Conjugate Gradient Method. *SIAM J. Sci. Stat. Comput.*, 11(5):859–872, 1990.

[EV91] V. Eijkhout and P. S. Vassilevski. The Role of the Strengthened Cauchy-Schawrz-Bunyakowski Inequality in Multilevel Methods. *SIAM Review*, 33(5), 1991.

[Ewi91] R. Ewing. Application of Domain Decomposition Techniques in Large-Scale Fluid Flow Problems. *Applied Numerical Mathematics*, 8:375–388, 1991.

[Fed62] R. P. Fedorenko. A Relaxation Method for Solving Elliptic Difference Equations. *URSS Computational Math. and Math. Phys.*, 1:1092–1096, 1962.

[Fle76] R. Flecher. Conjugate Gradient Methods for Indefinite Systems. *Lecture Notes in Mathematics 506*, 1976.

[FM84] V. Faber and T. Manteuffel. Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method. *SIAM J. Numer. Analysis*, 21(2):352–362, 1984.

[GBD98] D.L. Gines, G. Beylkin, and J. Dunn. LU Factorization of Non-Standard Forms and Direct Multiresolution Solvers. *Applied and Computational Harmonic Analysis*, 5:156–201, 1998.

[GBDJ93] A. Geist, A. Beguelin, J. Dongarra, and W. Jiang. PVM3 User's Guide and Reference Manual. Technical report, Oak Ridge National Laboratory, 1993.

[GH97] M. Grote and T. Huckle. Parallel Preconditioning with Sparse Approximate Inverse. *SIAM Scientific Computing*, 18(3):838–853, 1997.

[GL89] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1989.

[GS93a] W. Gropp and B. F. Smith. Experiences with Domain Decomposition in Three Dimensions: Overlapping Schwarz Methods. In *SIAM 6th International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1993.

[GS93b] M. Grote and H. Simon. Parallel Preconditioning and Approximate Inverses on the Connection Machine. In *6th SIAM Conference on Parallel Processing for Scientific Computing*, pages 519–523. SIAM, 1993.

[GS94] M. Grote and H. Simon. A New Approach to Parallel Preconditioning with Sparse Approximate Inverses. Technical Report SCCM-94-03, Department of Sci. Comput. and Comput. Math., Stanford University, 1994.

[GS95] M. Grote and H. Simon. Effective Parallel Preconditioning with Sparse Approximate Inverses. In *7th SIAM Conference on Parallel Processing for Scientific Computing*, pages 466–471. SIAM, 1995.

[Hac85] W. Hackbusch. *Multigrid Methods and Applications.* Springer-Verlag, 1985.

[Hig96] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, 1996.

[HS52] M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[Hua91] W.Z. Huang. Convergence of Algebraic Multigrid Methods for Symmetric Positive Definite Matrices with Weak Diagonal Dominance. *Appl. Math. Comp.*, 46:145–164, 1991.

[Kic97] F. Kickinger. Algebraic Multi-grid for Discrete Elliptic Second-Order Problems. Technical report, Institute for Mathematics, Johannes Kepler Uiversity Linz, Austria, 1997.

[KK95] G. Karypis and V. Kumar. Metis: Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical report, Department of Computer Science, University of Minnesota, also available online at http://www.cs.umn.edu/~karypis/metis, 1995.

[KS97] A. Krechel and K. Stuben. Operator Dependent Interpolation in Algebraic Multigrid. Technical report, GMD Report 1048, 1997.

[KY93] L. Yu Kolotilina and A. Yu Yeremin. Factorized Sparse Approximate Inverse Preconditionings I. theory. *SIAM Jour. Matrix Analy. Appli.*, 14:45–58, 1993.

[Lan50] C. Lanczos. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *Journal of Research of the National Bureau of Standards*, 45:255–282, 1950.

[Lan52] C. Lanczos. Solutions of Linear Equations by Minimized Iterations. *Journal of Research of the National Bureau of Standards*, 49:33–53, 1952.

[LC99] G. Larrazábal and J. M. Cela. Study of SPAI Preconditioner for Convective Problems. In *5th International Conference on Numerical Methods in Engineering and Applied Sciences, (CIMENICS2000)*, Caracas, Venezuela, 1999.

[LC00] G. Larrazábal and J. M. Cela. Two Approximation to Global Term of the Schur Matrix. In *First SIAM Conference on Computational Science and Engineering*, Washington D.C., USA, 2000.

[LC01a] G. Larrazábal and J. M. Cela. Two Parallel Algebraic Preconditioner for the Schur Complement System. In *IEEE 15th International Parallel and Distributed Processing Symposium, (IPDPS2001)*, San Francisco, USA, 2001.

[LC01b] G. Larrazábal and J. M. Cela. A New Algebraic Preconditioner for the Schur Complement System. In *International Conference on Preconditioning Technique for Large Sparse Matrix Problems in Industrial Applications, (Preconditioning2001)*, Tahoe City, USA, 2001.

[LCra] G. Larrazábal and J. M. Cela. A Parallel Iterative Solver for the Schur Complement System. In *IEEE International Conference on Parallel Processing, (ICPP2001)*, Valencia, Spain, 2001. Summitted to Journal on Numerical Linear Algebra.

[McC89] S. F. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations*, volume 6. SIAM, Frontiers in Applied Mathematics, 1989.

[MdV77] J. A. Meijerink and H. A. Van der Vost. An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-matrix. *Math. Compt.*, 31:148–162, 1977.

[NJL94] J.Ñavarro, T. Juan, and T. Lang. MOB Forms: A Class of Multilevel Block Algorithms for Dense Linear Algebra Operations. In *ICS 94*, 1994.

[NRT92] N.Ñachtigal, S. Reddy, and L. Trefethen. How Fast are Nonsymmetric Matrix Iterations? *SIAM J. Matrix Anal. Appl.*, 13(3), 1992.

[Ope98] OpenMP Organization. *OpenMP C/C++ Application Program Interface version 1.0*, available online at: www.openmp.org, 1998.

[PSL90] A. Pothen, H. Simon, and K. Liou. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM Mat. Anal. Appl.*, 11(3), 1990.

[PSS92] B. Philippe, Y. Saad, and W. J. Stewart. Numerical Methods in Markov Chain Modeling. *Operations Research*, 40(6), 1992.

[PTVF94] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing*. Second Edition, Cambridge University Press, 1994.

[QV91] V.A. Quarternoi and A. Valli. Theory and Application of the Steklov-Poincare Operators for Boundary-Value Problems: The heterogeneous operator case. In *SIAM 4th International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1991.

[Reu94] A. Reusken. Multigrid with Matrix-Dependent Transfer Operators for Convection-Diffusion Problems. In *Seventh International Symposium on Domain Decomposition Methods for Partial Equations*, 1994.

[RS87] J.W. Ruge and K. Stuben. Algebraic Multigrid (AMG). *In Multigrid Methods (McCormick, S.F., ed.), SIAM, Frontiers in Applied Mathematics*, 5, 1987.

[Rud93] U. Rude. *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, volume 13. SIAM, Frontiers in Applied Mathematics, 1993.

[Saa81] Y. Saad. Krylov Subspace Methods for Solving Large Unsymmetric Linear Systems. *Math. Compt.*, 37:105–126, 1981.

[Saa85] Y. Saad. Practical Use of Polynomial Preconditionings for the Conjugate Gradient Method. *SIAM J. Sci. Stat. Comput.*, 6(4), 1985.

[Saa89] Y. Saad. Krylov Subspace Methods on supercomputers. *SIAM Journal on Sci. Sta. Comput.*, 10(6):1200–1232, 1989.

[Saa94a] Y. Saad. Highly Parallel Preconditioners for General Sparse Matrices. In *Recent Advances in Iterative Methods*, volume 60. Springer-Verlag, 1994.

[Saa94b] Y. Saad. ILUT: a Dual Threshold Incomplete LU Factorization. *Numerical Linear Algebra with Applications*, 1(4), 1994.

[Saa96] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWD Publishing, New York, 1996.

[SC98] Y. Saad and E. Chow. Approximate Inverse Preconditioners via sparse-sparse Iterations. *SIAM J. Scientific Computing*, 19, 1998.

[SE87] P. Sadayappan and F. Ercal. Nearest-Neighbor Mapping of Finite Element Graphs onto Processors Meshes. *IEEE Trans. on Computers*, C-36(12), 1987.

[SF93] G. Sleijpen and D. Fokkema. BIGCstab(l) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum. *Electronic Transactions on Numerical Analysis*, 1:11–32, 1993.

[SIA88] SIAM. *1st International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1988.

[SIA89] SIAM. *2st International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1989.

[SIA90] SIAM. *3st International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1990.

[SIA91] SIAM. *4st International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1991.

[SIA92] SIAM. *5st International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1992.

[Son89] P. Sonneveld. CGS, A Fast Lanczos Type Solver for Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 10(1), 1989.

[SS86] Y. Saad and M. H. Sultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 7(3), 1986.

[Ste91] G. W. Stewart. Lanczos and Linear Systems. Technical Report CS-TR 2641, Department of Computer Science, University of Maryland, 1991.

[Stu99] K. Stuben. Algebraic Multigrid (AMG): An Introduction with Applications. Technical report, GMD Report 53, 1999.

[TC88] D. Goovaerts T. Chan. Schwarz=Schur: Overlapping versus No Overlapping Domain Decomposition. Technical Report CAM 88-21, Department of Mathematics, UCLA, 1988.

[Ton90] C. Tong. *Parallel Preconditioned Conjugate Gradient Methods for Elliptic Partial Diferential Equations.* PhD thesis, Computer Science Department, UCLA, 1990.

[Tum89] R. Tuminaro. *Multigrid Algorithms on Parallel Processing Systems.* PhD thesis, Computer Science Department, Stanford University, 1989.

[Var62] R. Varga. *Matrix Iterative Analysis.* Prentice-Hall International, 1962.

[VMB96] P. Vanek, J. Mandel, and M. Brezina. Algebraic Multigrid by Smoothed Aggregation for Second Order and Fourth Order Elliptic Problems. *Computing*, (56):179–196, 1996.

[Wat81] J. Watt. A Conjugate Gradient Truncated Direct Method for the Iterative Solution of the Reservoir Simulation Pressure Equation. *Society of Petroleum Engineer Journal*, 21:345–353, 1981.

[Wei94] R. Weiss. Minimization Properties and Short Recurrences for Krylov Subspace Methods. *Electronic Transactions on Numerical Analysis*, 2:57–65, 1994.

[WKW91] C. Wagner, W. Kinzelbach, and G. Wittum. Schur-Complement Multigrid- a Robust Method for Groundwater Flow and Transport Problems. *Numerishe Mathematik*, (75):523–545, 1991.

[Yea89] D. Young and et al. Application of Sparse Matrix Solvers as Effective Preconditioners. *SIAM J. Sci. Stat. Comput.*, 10(6):1186–1199, 1989.

[Yse82] H. Yserentant. On the Multilevel Splitting of Finite Element Spaces. *Numer. Math.*, 49:379–412, 1982.

[Zla82] Z. Zlatev. Use of Iterative Refinement in the Solution of Sparse Linear Systems. *SIAM Jour. Numer. Analysis*, 19:381–399, 1982.