UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

Paralelización de datos para el cálculo del Pseudoespectro.

Trabajo Especial de Grado presentado ante la ilustre Universidad Central de Venezuela por el **Br. Rafael David Guevara** para optar al título de Licenciado en Computación

Tutores: Profa. Zenaida Castillo.
Prof. Reinaldo Astudillo.

Caracas, Venezuela
Mayo 2012

Índice general

Introducción

El espectro de matrices ha sido una poderosa herramienta de trabajo para diversos problemas dentro de las áreas de ciencia e ingeniería. Sin embargo, cuando las matrices están sujetas a perturbaciones el análisis del espectro no es suficiente y se requiere de una herramienta complementaria llamada el pseudoespectro. Esta herramienta nos da más información y está siendo muy utilizada actualmente. En este trabajo se analiza e implementa un esquema de partición de dominio para el cálculo de pseudoespectro el cual es implementado eficientemente en paralelo.

El cálculo del pseudoespectro es uno de los problemas más recientes dentro del análisis numérico. Sus orígenes se remontan al año 1990 y desde entonces se han abierto muchas líneas de investigación para su creciente estudio. El pseudoespectro tiene muchas aplicaciones dentro de las áreas de ciencia e ingeniería como dinámica estructural, redes eléctricas, procesos de combustión, termodinámica, macroeconomía, análisis de reacciones químicas y ha sido objeto de muchas investigaciones desde la década de los 90 por autores como A.E Trefethen, L.N Trefethen, S.C Reddy, M. Embree y T.A Driscoll.

En el presente trabajo de grado se desarrollan dos algoritmos para realizar el cálculo del pseudoespectro en paralelo, un algoritmo para matrices densas y otro algoritmo para matrices dispersas y de gran tamaño.

Seguidamente dichos algoritmos se someten a experimentos numéricos y de tiempo sobre una arquitectura paralela y se llega a la conclusión de que los algoritmos planteados son correctos y escalables.

Este trabajo tiene la siguiente estructura: un primer capítulo donde se muestra la definición de espectro junto a las definiciones de pseudoespectro y sus áreas de aplicación además de un ejemplo que muestra sus diferencias.

Un segundo capítulo donde se describen los métodos numéricos utilizados actualmente para realizar el cálculo del pseudoespectro tanto para matrices densas y como para matrices dispersas de gran magnitud.

Seguidamente se tiene un tercer capítulo donde se dan a conocer los trabajos realizados anteriormente sobre el cálculo del pseudoespectro en paralelo junto al esquema de paralelismo propuesto para ser realizado en este trabajo.

Luego en el cuarto capítulo se muestran los resultados de los experimentos realizados

sobre matrices densas y dispersas con el fin de medir la correctitud de los algoritmos elaborados junto a su comportamiento a nivel de tiempo computacional.

Finalmente en el quinto y último capítulo se dan a conocer las conclusiones obtenidas a partir de los resultados experimentales junto a algunas recomendaciones para trabajos futuros.

Capítulo 1

Espectro y pseudoespectro de matrices

El espectro de una matriz A de $n \times n$ es el conjunto formado por todos sus autovalores, siendo éstos todos los escalares λ tales que $Ax = \lambda x$ para algún vector x no nulo. Este conjunto puede ser descrito como:

$$\Lambda(A) = \{z \in \mathbb{C} : \|(zI - A)^{-1}\| = \infty\} = \{z \in \mathbb{C} : (zI - A) \text{ es singular}\}$$

Al par (x, λ) que satisface $Ax = \lambda x$, se le denomina autopar, donde λ es un autovalor y x es su correspondiente autovector.

La importancia del espectro de matrices $n \times n$ surge como consecuencia de que dicho conjunto contiene información que resulta de interés en múltiples problemas de la ciencia y la ingeniería, tales como dinámica estructural, teoría de control, ingeniería química, dinámica poblacional, etc. (ver por ejemplo [29]).

En la resolución del problema $Ax = b$, el radio espectral y la distribución de los autovalores en el plano complejo proveen información sobre la convergencia o divergencia de los métodos iterativos utilizados para resolver dicho problema; por ejemplo, para el caso de los métodos estacionarios como Jacobi, Gauss-Seidel y SOR si el radio espectral es menor que 1 se dice que el método converge, de lo contrario diverge (ver por ejemplo [15]); otro ejemplo surge en aquellos problemas modelados por ecuaciones diferenciales, en el cual se calcula el espectro de la matriz Jacobiana evaluada en soluciones estacionarias, para conocer si estas soluciones son estables o no; específicamente, si todos los autovalores de esa matriz tienen parte real negativa entonces el sistema se denomina estable en esta solución, en caso contrario el sistema es inestable.

Los problemas mencionados anteriomente tienen muchas aplicaciones dentro las ciencias e ingeniería, en áreas como: estabilidad de sistemas dinámicos [10], problemas de elasticidad [31], cadenas de Markov [22], hidrodinámica [36], láseres [26], mecánica cuántica [3] y otras. En la práctica las matrices generadas pueden ser muy grandes, dispersas y no-simétricas, en cuyo caso, la información obtenida con el cálculo de autovalores no es suficiente para realizar una análisis real y completo, ya que éste no

considera las perturbaciones presentes en los datos y en los cálculos. En estos casos, es deseable calcular el pseudoespectro, para complementar el análisis del espectro.

1.1. Definiciones

El concepto de *pseudoespectro* fué introducido por L. Treffethen en la década de los 90 (L. Trefethen en [34]), algunas definiciones equivalentes de pseudoespectro o ϵ-pseudoespectro, para un ϵ dado, no negativo, se mencionan a continuación:

Definición 1. Para toda matriz $A \in C^{n\times n}$ y para todo número real $\epsilon > 0$ se define el ϵ-pseudoespectro de A, y se denota por $\Lambda_\epsilon(A)$ como el conjunto:

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : \|(zI - A)^{-1}\| > \epsilon^{-1}\}$$

.

Definición 2. Para cualquier matriz $A \in C^{n\times n}$ y para cada número real $\epsilon > 0$ se define el ϵ-pseudoespectro de A como el conjunto:

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : z \in \Lambda(A + E) \text{ para } E \in \mathbb{R}^{nxn} \text{ y } \|E\| < \epsilon\}$$

donde$(\Lambda(A))$ es el espectro de A.

Definición 3. Para cada matriz $A \in C^{n\times n}$ y para cada número real $\epsilon > 0$ se define el ϵ-pseudoespectro de A como el conjunto:

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : \|(zI - A)v\| < \epsilon \text{ para } v \in \mathbb{C}^n \text{ con } \|v\| = 1\}.$$

Todas las definiciones anteriores son matemáticamente equivalentes, y si se usa la norma euclideana puede deducirse una cuarta definición.

Definición 4. Para toda matriz $A \in C^{n\times n}$ y para todo número real $\epsilon > 0$ se define el ϵ-pseudoespectro de A, y se denota por $\Lambda_\epsilon(A)$ como:

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C} : \sigma_{min}(A - zI) < \epsilon\}$$

donde $\sigma_{min}(A - zI)$ es el valor singular mínimo de $(A - zI)$.

El pseudoespectro es una poderosa herramienta para los problemas sensibles a las perturbaciones, en los cuales el análisis del espectro no es suficiente. Un caso particular lo constituyen los sistemas dinámicos. Por esta razón, el desarrollo de algoritmos eficientes para el cálculo del pseudoespectro se ha convertido en un área activa de investigación durante los últimos años. A continuación se puede ver el pseudoespectro de la matriz Smoke de 70×70 en la figura 1.1:

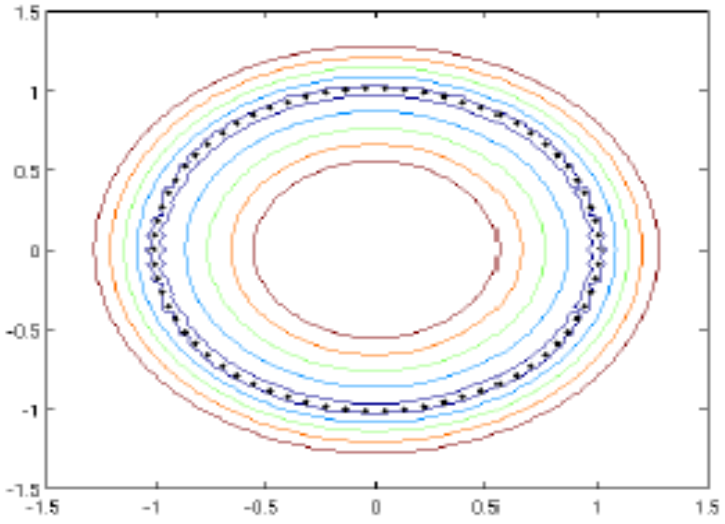

Figura 1.1: Gráfica del pseudoespectro de la matriz Smoke de dimensión 70

1.2. Diferencias entre el espectro y el pseudoespectro

Para mostrar la diferencia existente entre el análisis realizado del espectro y aquel realizado sobre el pseudoespectro, se presenta un ejemplo tomado del artículo: "Algorithms for the computation of the pseudoespectral radius and the numerical radius of a matrix"de E. Mengi M. Overton [28]. En el mismo se desea analizar la convergencia del sistema dinámico:

$$x_k = Ax_{k-1} \tag{1.1}$$

donde $x \in \mathbb{C}^n$ y $A \in \mathbb{C}^{n\times n}$. Este análisis depende en gran medida de la norma de las potencias de A. De forma asintótica, cuando $k \to \infty$ los autovalores proveen de toda la información necesaria para analizar (1.1), específicamente es bien conocido que $\lim_{k\to\infty} ||x_k|| = 0$ para todo x_0 si sólo si todos los autovalores de A se encuentran dentro del circulo unitario, es decir, A es una matriz convergente.

Más aún podemos medir la velocidad de convergencia con el radio espectral de A, definido como:

$$\rho(A) = max\{|\lambda| : \forall\lambda \in \Lambda(A)\}$$

Es conocido que mientras más cercano a cero sea el radio espectral de A, entonces más rápida será la convergencia asintótica de la ecuación (1.1). Sin embargo, para valores finitos de k, el sólo cálculo de los autovalores no aporta suficiente información al análisis de (1.1), a menos que la matriz A sea normal ($AA^t = A^tA$). Para una matriz A no-normal, aún cuando sus autovalores estén dentro del círculo unitario, la norma

de la k-eśima potencia $||A^k||_2$ puede ser arbitrariamente grande y los autovalores no dan ninguna cota sobre la misma. En este caso, es útil , un concepto relacionado al pseudoespectro como lo es el *radio pseudoespectral*, definido como:

$$\rho_\epsilon(A) = max\{|z| : z \in \Lambda_\epsilon(A)\},$$

se puede deducir que:

$$\sup_{\epsilon>0} \frac{\rho_\epsilon(A) - 1}{\epsilon} \leq \sup_{\epsilon>0} ||A^k|| \leq en \sup_{\epsilon>0} \frac{\rho_\epsilon(A) - 1}{\epsilon} \tag{1.2}$$

donde e es el número irracional definido por $e = \sum_{i=0}^{\infty} 1/i!$ y n el orden de A.

La expresión (1.2) proporciona cotas, superior e inferior, para determinar el valor máximo de la norma de las potencias de A, en términos del radio pseudoespectral $\rho_\epsilon(A)$. La cota inferior es particularmente útil como un indicador de cuán grande puede crecer la norma de las potencias de la matriz A. Por ejemplo, consideremos A como una matriz Toeplitz de orden 100×100, con la siguiente estructura:

$$a_{ij} = \begin{cases} -0{,}4 & \text{si } j = i+1; \\ 0{,}4 & \text{si } j \leq i; \\ 0 & \text{en otro caso.} \end{cases}$$

Esta matriz tiene radio espectral igual a $\rho(A) = 0,90517$, por lo tanto A es una matriz convergente, sin embargo, se tiene que el radio pseudoespectral es $\rho_\epsilon(A) = 1,05725$ para $\epsilon = 10^{-7}$, nótese que aunque el espectro está dentro del círculo unitario el ϵ-pseudoespectro no lo está (ver figura 1.2). Para este valor de ϵ el número $\frac{\rho_\epsilon(A)-1}{\epsilon}$ que acota superiormente la norma de las potencias de A, es aproximadamente $5,72 \times 10^5$, es decir, que gracias al análisis del pseudoespectro se puede saber que $||A^k||$ alcanza valores de orden 10^5 o superior. Es así, como el pseudoespectro nos permite hacer un análisis más completo que el espectro cuando la matriz es no-normal. La figura 1.3 confirma este comportamiento.

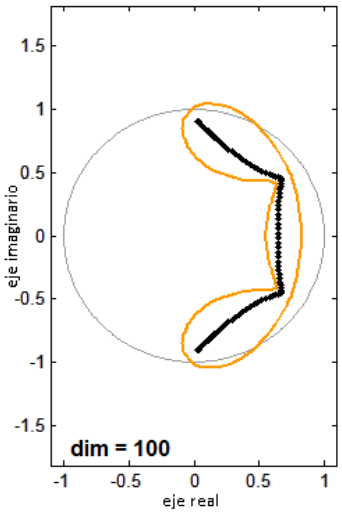

Figura 1.2: Espectro y 10^{-7} Pseudoespectro de la matriz Toeplitz 100×100

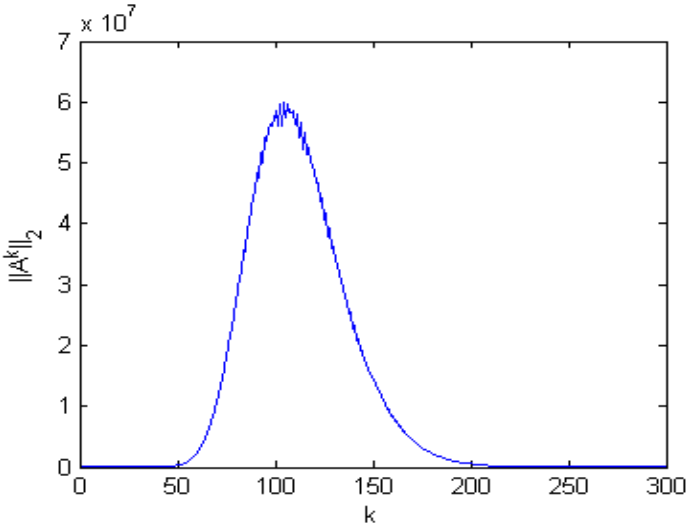

Figura 1.3: Gráfica del comportamiento de $||A^k||$ a medida que k incrementa

Capítulo 2

Métodos numéricos para el cálculo del Pseudoespectro

Tomando en cuenta las definiciones planteadas en el capítulo anterior se puede construir dos algoritmos básicos para realizar el cálculo del pseudoespectro de A. Un algoritmo para matrices densas y otro algoritmo para matrices dispersas y de gran tamaño. Estos algoritmos son:

Algoritmo 2.1 Algoritmo básico para el cálculo del Pseudoespectro para matrices densas

1: Determinar una región $\mathbb{K}$ de interés en $\mathbb{C}$
2: Discretizar $\mathbb{K}$
3: Para cada punto $z \in \mathbb{K}$ calcular $\|(zI - A)^{-1}\|$
4: Visualizar el contorno de $\mathbb{K}$

Algoritmo 2.2 Algoritmo básico para el cálculo del Pseudoespectro para matrices dispersas y de gran tamaño

1: Proyectar A sobre un subespacio de menor dimensión. Denotado por H, $H \in C^{p \times p}$ con $p << n$
2: Determinar una región $\mathbb{K}$ de interés en $\mathbb{C}$
3: Discretizar $\mathbb{K}$
4: Para cada punto $z \in \mathbb{K}$ calcular $\|(zI - H)^{-1}\|$
5: Visualizar el contorno de $\mathbb{K}$

Existen diferentes esquemas que nos permiten implementar el paso 3 del algoritmo 2.1 y los pasos 1 y 4 del algoritmo 2.2; en particular para el algoritmo 2.1 se puede utilizar en el paso 3 el método básico, la iteración inversa y el método de Lanczos inverso junto a una técnica de aceleración llamada triangularización preliminar mientras que en el paso 1 del algoritmo 2.2 se puede utilizar el algoritmo de Arnoldi y los métodos de

Lanczos no simétrico, JDQR y BLIRAM, por otro lado en el paso 4 del algoritmo 2.2 se puede utilizar el método de Lanczos inverso para matrices rectangulares.

2.1. Métodos para matrices densas

Los métodos para matrices densas están basados en calcular el mínimo valor singular de la matriz $(zI - A)$ para distintos escalares complejos z. Entre ellos podemos encontrar el método básico , el método de la iteración inversa y el método de Lanczos, adicionalmente se tiene una técnica para acelerar dichos métodos llamada triangularización preliminar.

2.1.1. Método básico

Dado que $\|(zI - A)^{-1}\|_2 = s_{min}(zI - A)$, en el paso 3 del algoritmo base se puede utilizar la descomposición en valores singulares (SVD) (ver [35]), tomando una región discretizada que consista de m puntos (x, y) en la dirección real e imaginaria, con coordenadas indexadas por los vectores x y y. Este método tiene una complejidad computacional de $O(m^2n^3)$ siendo m la dimensión de la malla y n la dimensión de la matriz $(zI - A)$ (ver [35]) y puede ser implementado en Matlab como lo muestra el algoritmo 2.3.

Algoritmo 2.3 Algoritmo del método básico

```
1: for k = 1 : m do
2:     for j = 1 : m do
3:         sigmin(j, k) = min(svd((A − (x(k) + y(j) ∗ i) ∗ eye(n))));
4:     end for
5: end for
6: contour(x, y, log10(sigmin));
```

El comando contour dibuja el borde o la frontera del conjunto que define el ϵ-pseudoespectro de la matriz A. En esta región se encontrarán todos los autovalores de matrices perturbadas $A + E$, con $||E|| < \epsilon$.

2.1.2. Cálculo usando la iteración inversa

Una desventaja del método básico es que hace operaciones operaciones innecesarias para el cálculo de $s_{min}(zI-A)$ ya que para conseguir el mínimo valor singular de $(zI-A)$ calcula primero todos los valores singulares de $(zI - A)$ y luego selecciona el de menor magnitud. Se puede conseguir una reducción significativa del costo computacional si se usa el método de la iteración inversa sobre $(zI-A)^t(zI-A)$ (ver [35] y [37]). La idea de usar el método de la iteración inversa es calcular el mínimo valor singular de $(zI-A)$ sin

tener que calcular todos los valores singulares. Este método tiene el siguiente algoritmo codificado en MATLAB y puede ser visto en la figura 2.4:

Algoritmo 2.4 Algoritmo del método de iteración inversa

```
1: for k = 1 : m do
2:     for j = 1 : m do
3:         B = A - (x(k) + y(j)*i)*eye(n);
4:         u = rand(n,1) + i*rand(n,1);
5:         [L,U] = lu(B);
6:         for p = 1 : maxit do
7:             u = U\L\L'\U'\ u;
8:             sig = 1/norm(u);
9:             if abs(sigold/sig − 1) < 1e − 2 then
10:                break;
11:            end if
12:            u = sig*u;
13:            sigold = sig;
14:        end for
15:        sigmin(j, k) = sqrt(sig);
16:    end for
17: end for
18: contour(x, y, log10(sigmin));
```

2.1.3. Cálculo usando el método de Lanczos

Considerando la misma filosofía del método de la iteración inversa y tomando en cuenta que calcular el mínimo valor singular de $(zI - A)$ conlleva intrínsecamente el cálculo del mínimo autovalor de la matriz $(zI - A)^t(zI - A)$, se puede mejorar aún más el rendimiento usando un método iterativo más sofisticado: la iteración de Lanczos [9], que se aplica a la matriz $((zI - A)^t(zI - A))^{-1}$. Esta iteración calcula el autovalor de módulo mínimo de $(zI - A)^t(zI - A)$ y el valor singular mínimo se consigue al tomar la raíz cuadrada de éste número. Su implementación en MATLAB es:

Algoritmo 2.5 Algoritmo del método de Lanczos

```
1: for k = 1 : m do
2:     for j = 1 : m do
3:         B1 = A − (x(k) + y(j) ∗ i) ∗ eye(n);
4:         u = rand(n, 1) + i ∗ rand(n, 1);
5:         sigold = 0;
6:         gold = zeros(n, 1);
7:         beta = 0;
8:         T = [];
9:         q = rand(n, 1) + i ∗ rand(n, 1);
10:        q = q/norm(q);
11:        for p = 1 : maxit do
12:            v = B1\(B1′\q) − beta ∗ qold;
13:            alpha = real(q′ ∗ v);
14:            v = v − alpha ∗ q;
15:            beta = norm(v);
16:            qold = q;
17:            q = v/beta;
18:            T(p + 1, p) = beta;
19:            T(p, p + 1) = beta;
20:            T(p, p) = alpha;
21:            sig = max(eig(H(1 : p, 1 : p)));
22:            if abs(sigold/sig − 1) < 1e − 2 then
23:                break;
24:            end if
25:            sigold = sig;
26:        end for
27:        sigmin(j, k) = sqrt(sig);
28:    end for
29: end for
30: contour(x, y, log10(sigmin));
```

2.1.4. Triangularización preliminar

Se pueden diseñar variantes de los algoritmos anteriores, como por ejemplo las propuestas por S. Liu en su artículo "Computation of pseudospectra by continuation" [27] para mejorar el tiempo de cómputo que consumen los métodos iterativos de iteración inversa y de Lanczos , a saber:

• Una triangularización preliminar de la matriz A o factorización de Schur no altera el espectro de A y reduce a $O(n^2)$ la resolución de sistemas lineales que aparecen en el algoritmo 2.4 con las matrices B^t y B.

• Para el cálculo del vector singular en algún punto de la región $\mathbb{K}$, usar como iterado inicial el vector singular calculado en el punto adyacente, a este procedimiento se le conoce como continuación numérica.

2.2. Métodos para matrices dispersas y de gran tamaño

Los métodos para matrices dispersas y de gran tamaño están basados en proyectar la matriz A sobre un subespacio de menor dimensión y luego aplicar alguno de los métodos para matrices densas sobre el resultado de dicha proyección. A continuación se presentan algunos métodos para calcular el pseudoespectro de matrices dispersas y de gran tamaño utilizando proyecciones sobre el subespacio de Krylov.

2.2.1. Proyección sobre el subespacio de Krylov usando el algoritmo de Arnoldi

En muchas aplicaciones se requiere el cálculo del pseudoespectro de matrices de dimensión $n >> 1000$, para las cuales, los métodos descritos anteriormente no pueden ser aplicados debido a su alto costo computacional, para estos casos se sugiere hacer una proyección sobre el subespacio de Krylov (ver Toh y Trefethen en [32] y Wright y Trefethen en [38]) de dimensión p (por lo general, $p << n$) como la mostrada a continuación:

$$\mathcal{K}_p(A, x) = \{x, Ax, A^2x, ..., A^{p-1}x\}$$

Esta proyección puede hacerse usando el algoritmo de Arnoldi (planteado en [2] y usado en [32] para el cálculo del pseudoespectro por Toh y Trefethen), el cual crea una base ortogonal $V_p = [V_{p-1}, v_p]$ del subespacio de Krylov, y una matriz rectangular Hessenberg $\bar{H}_p$ de dimensión $(p + 1 \times p)$, tal que:

$$AV_p = V_{p+1}\bar{H}_p$$

En [32] los autores Toh y Trefethen demuestran la siguiente inclusión:

$$\Lambda_\epsilon(\bar{H}_p) \subseteq \Lambda_\epsilon(A),$$

es decir, al calcular el pseudoespectro de $\bar{H}_p$ se puede aproximar el pseudoespectro de A. Esta idea es clave, ya que nos permite aproximar el pseudoespectro de matrices de grandes dimensiones, calculando el pseudoespectro de matrices mucho más pequeñas, usando los métodos clásicos señalados en la sección anterior.

El método de proyección IRAM (por sus siglas en inglés Implicit Restarted Arnoldi Method) surge como variante del algoritmo de Arnoldi, fue propuesto por Sorensen en [30] y usado para el cálculo del pseudoespectro por Wright y Trefethen en [38].

2.2.2. Otros métodos de proyección sobre el subespacio de Krylov

Existen otros métodos para proyectar la matriz A sobre un espacio de menor dimensión, entre ellos están el método de Lanczos no simétrico (planteado en [24] y [25] por C. Lanczos y usado por Astudillo en [4] para el cálculo del pseudoespectro), el algoritmo JDQR (propuesto por Fokkema, Sleijen y Van der Vorst en [14] y usado en [39] por Wright para el cálculo del pseudoespectro) y el método BLIRAM (propuesto por Castillo en [12] y utilizado en [5] por Astudillo y Castillo para calcular el pseudoespectro).

El método de Lanczos no simétrico es un proceso iterativo que aplicado a una matriz A, con vector inicial $v_1 \in C^n$ y luego de p pasos genera: una matriz $V_{p+1} = [v_1, v_2, \ldots, v_p, v_{p+1}]$ base para el subespacio de Krylov $\mathcal{K}_p(A, v_1)$ y una matriz tridiagonal $\bar{T}_p \in C^{p+1 \times p}$ tal que se cumple lo siguiente:

$$AV_p = V_{p+1}\bar{T}_p \tag{2.1}$$

El algoritmo JDQR genera una descomposición parcial de Schur de la forma:

$$AQ_k \simeq Q_k R_k$$

donde Q_k es una matriz con conlumnas ortonormales de $n \times k$ y R_k es una matriz triangular superior de $k \times k$ tal que:

$$\Lambda_\epsilon(R_k) \subseteq \Lambda_{\epsilon + k^{\frac{3}{2}}\epsilon_j}(A)$$

El algoritmo BLIRAM es una extensión del método IRAM que produce una factorización de Arnoldi en bloque tal que:

$$AV_{[m]} = V_{[m]}H_{[m]} + F_m E_m^T$$

donde A es una matriz real de orden n, $V_{[m]} = [V_1, V_2, \ldots, V_m]$ es una base ortogonal para el subespacio en bloque de Krylov.

$$K_m(AV_1) = Span\{V_1, AV_1, A^2V_1, \ldots, A^{m-1}V_1\}.$$

y $E_m^T = [Z_b, Z_b, \ldots, Z_b, I_b]$ es una matriz de $b \times (m \cdot b)$ donde Z_b representa la matriz zero de orden b, mientras que I_b representa la matriz identidad.

Capítulo 3

Esquema de paralelización para el cálculo del Pseudoespectro

Calcular el pseudoespectro es de por si una tarea costosa a nivel de tiempo computacional, ya que se deben hacer muchas operaciones aritméticas para calcular los valores singulares asociados a los puntos de la región de interés. Esta tarea adquiere un costo aún más elevado cuando crece la cantidad de puntos de dicha región. Para resolver este problema es recomendable aprovechar la independencia de datos presente entre los puntos de la región de interés aplicando paralelismo de datos sobre dichos puntos y así disminuir el costo computacional existente al aumentar la velocidad de cálculo del pseudoespectro.

Los métodos planteados anteriormente pueden ser enmarcados dentro de un esquema que permita el cálculo en paralelo del pseudoespectro tanto de matrices densas como de matrices dispersas de gran tamaño. A continuación se presentan algunos trabajos previos donde se realiza el cálculo del pseudoespectro en paralelo y luego se plantean dos algoritmos para efectuar dicho cálculo usando algunos de los métodos planteados en el capítulo anterior.

3.1. Antecedentes

En esta sección se mencionan algunos trabajos realizados por otros autores donde se realiza la paralelización de datos en el cálculo del pseudoespectro. Además se deja en claro la diferencia entre los trabajos realizados anteriormente y el presente trabajo.

En el año 1993 A.E Trefethen, L.N Trefethen, S.C Reddy y T.A Driscoll realizaron un artículo sobre hidrodinámica donde calcularon el pseudoespectro de matrices en paralelo en lenguaje FORTRAN sobre matrices de tamaño cercano a 80×80 (ver [36]).

Luego en el año de 1996 T. Braconnier hizo un reporte en el cual desarrolló una librería para realizar el cálculo del pseudoespectro en paralelo usando rutinas en FORTRAN y la librería PVM [8]. En ese mismo año V Frayssé, L Giraud y V Toumazou elaboraron una herramienta de software para cuantificar la

sensibilidad de autovalores frente a perturbaciones de tamaño variable la cual realiza el cálculo del pseudoespectro de matrices en paralelo [16].

Para el año de 1999 A.E Trefethen, L.N Trefethen y P.J Schmid en [33] calcularon el pseudoespectro en paralelo para resolver problemas de dinámica de fluidos en una tubería. Para ese mismo año C Bekas y E Gallopoulos realizaron el cálculo del pseudoespectro en paralelo utilizando el método del descenso mínimo [7].

En este trabajo se desarrolla un esquema que permite complementar el cálculo del pseudoespectro al proponer una paralelización de datos en el dominio de la región donde se desea la información del pseudoespectro.

3.2. Propuesta

En este trabajo hemos utilizado dos esquemas de procesamiento. El primer esquema es aplicado cuando queremos calcular todo el pseudoespectro de la matriz A (Algoritmo 1), este esquema es para matrices densas y el segundo esquema lo utilizamos cuando queremos calcular porciones del pseudoespectro de A (Algoritmo 2), dicho esquema es adecuado para matrices dispersas y de gran tamaño.

3.2.1. Descripción de Algoritmo 1

El Algoritmo 1 recibe como entrada la matriz densa A y genera como salida una gráfica del pseudoespectro de A. Este algoritmo consiste en dos módulos, uno para realizar el cálculo del pseudoespectro en paralelo donde se implementa el algoritmo de Lanczos simétrico utilizando la factorización de Schur y otro para graficar los contornos que forman el pseudoespectro de A. El primer módulo recibe como entrada la matriz A y genera como salida los valores singulares que serán graficados por el segundo módulo el cual genera una gráfica que contiene el pseudoespectro de A. El primer módulo calcula los valores singulares de la matriz $(zI - A)$ en paralelo usando el algoritmo de Lanczos simétrico. El algoritmo de Lanczos simétrico calcula el máximo autovalor de la matriz B realizando la factorización $BQ = QT$ donde B es simétrica, Q es ortogonal y T es tridiagonal, este algoritmo fue creado por Cornelius Lanczos y su estructura se puede ver en el algoritmo 3.1.

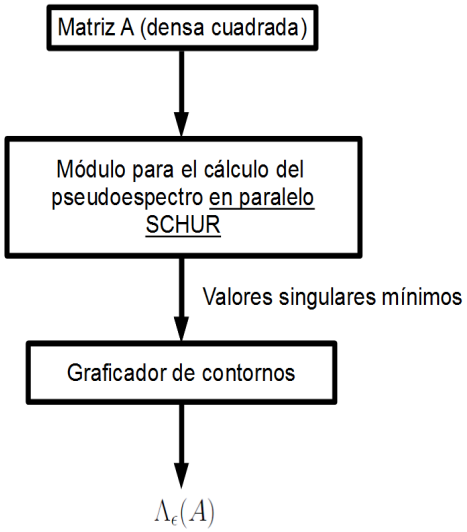

Figura 3.1: Algoritmo para calcular todo el pseudoespectro de A.

En este algoritmo se usa la factorización de Schur sobre la matriz B ($B = Q^t T Q$). Esta factorización genera una matriz triangular superior T tal que:

$$\Lambda_\epsilon(T) = \Lambda_\epsilon(B),$$

Esto nos permite reducir el tiempo que toma la resolución de los sistemas lineales en la línea 7 del algoritmo 3.1, ya que sólo se resuelven sistemas triangulares.

Algoritmo 3.1 Algoritmo del método de Lanczos simétrico

1: $sigold = 0$;
2: $gold = zeros(n, 1)$;
3: $beta = 0$;
4: $q = rand(n, 1)$;
5: $q = q/norm(q)$;
6: **for** $p = 1 : maxit$ **do**
7: $v = B \backslash B^t \backslash q - beta * qold$;
8: $alpha = real(q' * v)$;
9: $v = v - alpha * q$;
10: $beta = norm(v)$;
11: $qold = q$;
12: $q = v/beta$;
13: $T(p + 1, p) = beta$;
14: $T(p, p + 1) = beta$;
15: $T(p, p) = alpha$;
16: $sig = max(eig(T(1 : p, 1 : p)))$;
17: **if** $abs(sigold/sig - 1) < 1e - 2$ **then**
18: break;
19: **end if**
20: $sigold = sig$;
21: **end for**

3.2.2. Descripción de Algoritmo 2

El Algoritmo 2 recibe como entrada la matriz dispersa y de gran tamaño A, genera como salida una gráfica del pseudoespectro de A y consiste en tres módulos: uno para obtener la matriz $\bar{H}_p$ realizando una proyeccion sobre el subespacio de Krylov usando el método IRAM implementado por la biblioteca ARPACK, otro para realizar el cálculo del pseudoespectro de $\bar{H}_p$ en paralelo donde se implementa el método de Lanczos utilizando la factorización QR y otro para graficar los contornos que forman el pseudoespectro de A. El primer módulo recibe como entrada la matriz A y genera como salida la matriz $\bar{H}_p$, el segundo módulo recibe la matriz $\bar{H}_p$ y genera los valores singulares que serán graficados por el tercer módulo el cual recibe como entrada dichos valores singulares y genera como salida una gráfica que contiene el pseudoespectro de A.

Para este algoritmo el primer módulo extrae la matriz $\bar{H}_p$ rectangular utilizando una implementación del método IRAM proveniente de la biblioteca ARPACK. La matriz $\bar{H}_p$ fue extraida de un arreglo unidimensional haciendo uso de las subrutinas dnaupd y atmux. El segundo módulo realiza el cálculo de los valores singulares de la matriz $(zI - A)$ en paralelo usando el algoritmo de Lanczos simétrico adaptado para matrices rectangulares. Dicha adaptación utiliza factorización QR para poder trabajar con matrices rectangulares.

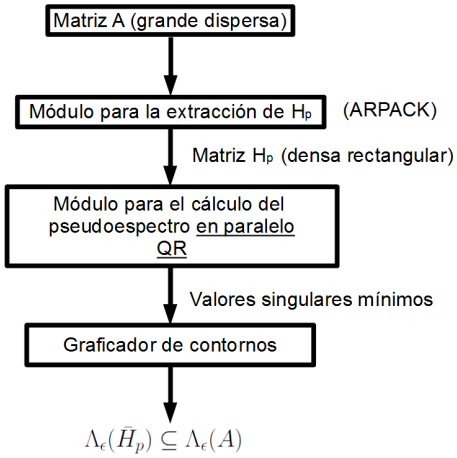

Figura 3.2: Algoritmo para calcular parte del pseudoespectro de A.

3.2.3. Descripción del particionamiento de la región de interés en subregiones

Supongamos que tenemos la malla de puntos de dimensión $mx \times my$ que representa a la región de interés $\mathbb{K}$ que ilustra la figura 3.3 donde mx y my son la cantidad de elementos de los ejes x e y respectivamente y z_{ij} es el número complejo con parte real igual a x_i y parte imaginaria igual a y_j. Supongamos también que tenemos una arquitectura paralela de n procesos entonces particionamos $\mathbb{K}$ en n subregiones etiquetadas desde 0 hasta $n-1$. Dichas subregiones se muestran delimitadas por una linea vertical en la figura 3.4. Entonces cada subregión se asocia a un proceso y cada proceso calcula los valores singulares correspondientes a su subregión en paralelo para luego enviar los valores singulares calculados al proceso maestro. De esta manera se obtiene un ahorro significativo del tiempo de cómputo y se aprovecha la independencia de datos para acelerar el cálculo del pseudoespectro.

Es importante mencionar que cada subregión contendrá $\frac{mx}{n} * my$ puntos dado que mx sea divisible entre n, de lo contrario la subregión 0 pasa a tener $\frac{mx}{n} * my$ puntos y las subregiones restantes pasan a tener $(\frac{mx}{n} + 1) * my$ puntos siempre y cuando la etiqueta de dichas subregiones sea menor o igual que mx **mod** n, sino estas subregiones tendrán $\frac{mx}{n} * my$ puntos.

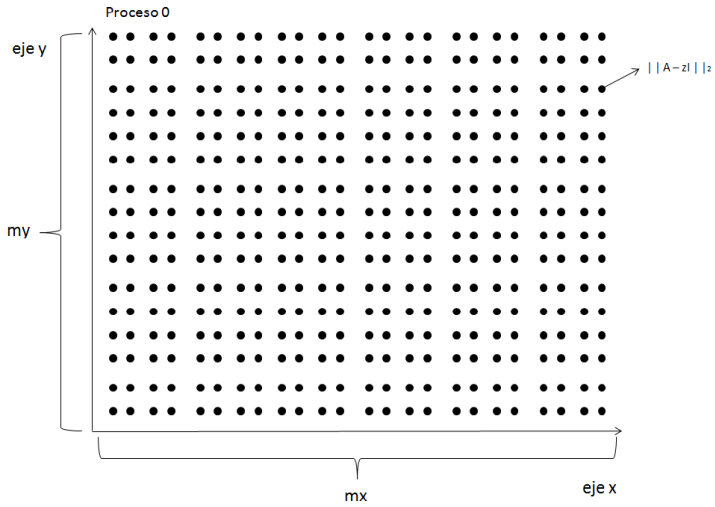

Figura 3.3: Región de interés $\mathbb{K}$ sin particionar asignada al proceso 0.

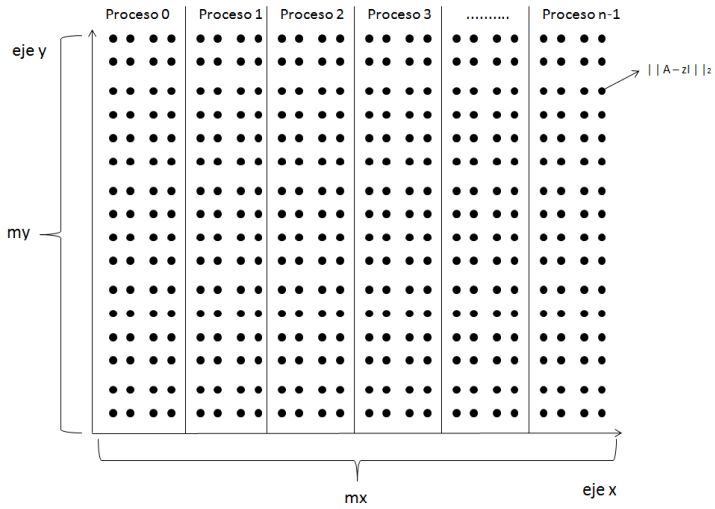

Figura 3.4: Región de interés $\mathbb{K}$ particionada en n subregiones asignadas a n procesos.

Capítulo 4

Experimentación Numérica

Los algoritmos 1 y 2 descritos en el capítulo anterior han sido sometidos a diversos experimentos numéricos con el objetivo de medir los siguientes puntos:

1. Correctitud de los algoritmos 1 y 2 junto a los tiempos de ejecución del módulo paralelo.

2. Tiempos de ejecución del módulo paralelo y sus respectivos comportamientos junto al comportamiento de la aceleración.

3. Escalabilidad del módulo paralelo.

Todos los experimentos fueron realizadas en un computador tipo cluster MIMD a memoria distribuida compuesto por 5 nodos con 2 procesadores Intel quad core Intel Xeon CPU E5440 con 2.83GHz y 8 GB de RAM por nodo. Las gráficas de $\Lambda(A)$ y $\Lambda(\bar{H}_p)$ fueron elaboradas usando un computador AMD con procesador Turion de 64 bits.

Las herramientas de software utilizadas para el desarrollo de los algoritmos 1 y 2 son el sistema de álgebra computacional MATLAB versión 7.8.0.347 (R2009a), el lenguaje de programación FORTRAN 95 con compiladores gfortran versión 4.5.3 y mpif90 versión 4.5.3 y la librería de paso de mensajes MPI para FORTRAN.

Los tiempos de ejecución correspondientes al módulo paralelo fueron obtenidos calculando el promedio de los tiempos de ejecución de cada proceso.

4.1. Pruebas Numéricas

En esta sección se presentan experimentos numéricos realizados sobre matrices densas y dispersas con el fin de mostrar la correctitud de los resultados obtenidos al ejecutar los algoritmos 1 y 2 con 2, 5 y 16 procesos para luego comparar los resultados obtenidos por la versión secuencial (1 proceso). Estos experimentos son similares a los realizados por Trefethen L. y Wright T. en [38], Braconnier y Higham en [9] y Wright T en [39], el máximo de iteraciones para obtener el mínimo autovalor de $(A-zI)^*(A-zI)$ por cada z es 100, la tolerancia escogida es 10^{-3} y la resolución de la malla es de 50×50 puntos igualmentes espaciados para todos los experimentos.

Las matrices sobre las cuales se ponen en práctica estos experimentos han sido usadas con mucha frecuencia en varios trabajos dentro del análisis numérico y fueron elegidas por mostrar diferencias de forma en sus pseudoespectros. El valor de la tolerancia escogido corresponde con el valor utilizado en la mayoría de las implementaciones conocidas de los métodos para calcular el pseudoespectro.

4.1.1. Pruebas numéricas para matrices densas

Estos experimentos se hicieron utilizando el Algoritmo 1 para realizar el cálculo de $\Lambda_\epsilon(A)$ con A densa.

Para el primer experimento se considera la matriz Kahan de dimensión 70 en la región [-0.7 , 1.2] × [-1.0 , 1.0]. Esta matriz surge en pruebas de estimación de rango de una matriz [23] y se encuentra en la colección MMDELI de Matrix Market [1]. En la figura 4.1 se ve que para 1, 2, 5 y 16 procesos los resultados obtenidos son exactamente iguales. Los tiempos de ejecución pueden ser vistos en el cuadro 4.1.

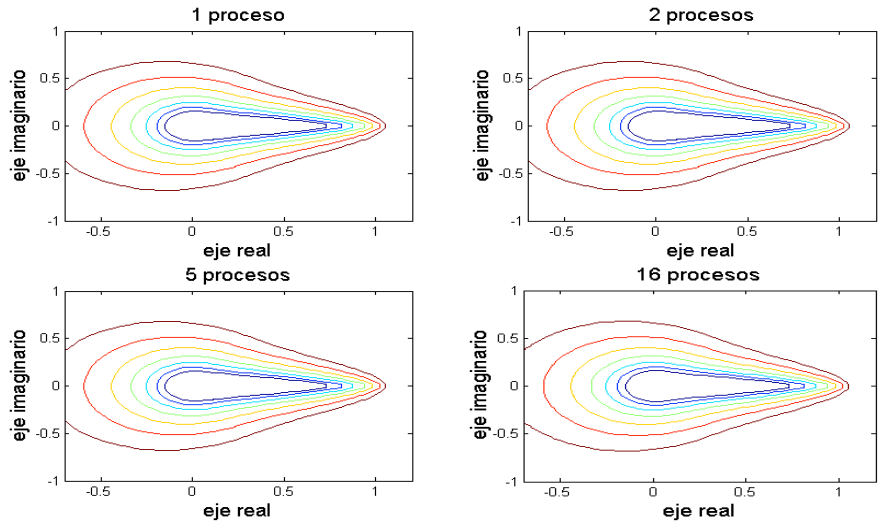

Figura 4.1: Pseudoespectro de la matriz Kahan con $\epsilon = 10^{-2}, 10^{-2,5}, \ldots, 10^{-5}$. Se puede ver claramente que para 1,2,5 y 16 procesos el Algoritmo 1 genera exactamente los mismos resultados

número de procesos	tiempo de ejecución
1	0.1730 seg.
2	0.0990 seg.
5	0.0438 seg.
16	0.0289 seg.

Cuadro 4.1: Tiempos de ejecución para la matriz Kahan usando 1, 2, 5 y 16 procesos.

En el segundo experimento se toma en cuenta la matriz Grcar de dimensión 80 en la región [-1.2 , 3.1] $\times$ [-4.6 , 4.6]. Se trata de una matriz Toeplitz no simétrica definida en [18] y disponible en la colección NEP de Matrix Market [1]. Para este experimento se puede ver que se mantiene la correctitud del programa para 1, 2, 5 y 16 procesos en la figura 4.2 pese a lo no-normal de la matriz Grcar. Los tiempos de ejecución se muestran en el cuadro 4.2.

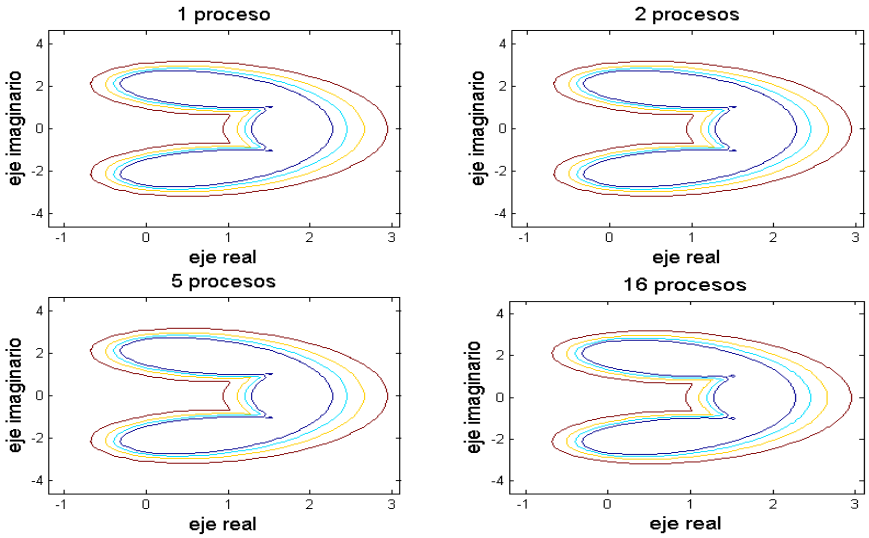

Figura 4.2: Pseudoespectro de la matriz Grcar para 1,2,5 y 16 procesos con $\epsilon = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$. Nótese lo correcto de los resultados generados por el Algoritmo 1 en la similitud de las aproximaciones calculadas

número de procesos	tiempo de ejecución
1	0.3979 seg.
2	0.2190 seg.
5	0.1086 seg.
16	0.0560 seg.

Cuadro 4.2: Tiempos de ejecución para la matriz Grcar usando 1, 2, 5 y 16 procesos.

Para el tercer experimento se usa la matriz Tolosa de dimensión 80 y se calcula el pseudoespectro de esta matriz en la región [-300 , 200] × [-250 , 250]. Esta matriz surge en el análisis de estabilidad de un modelo de aeroplano en vuelo [21], forma parte de la colección NEP de Matrix Market [1] y en la figura 4.3 se puede ver la aproximación a su pseudoespectro. Nótese lo correcto de los resultados para 1, 2, 5 y 16 procesos. Los tiempos de ejecución se encuentran en el cuadro 4.3.

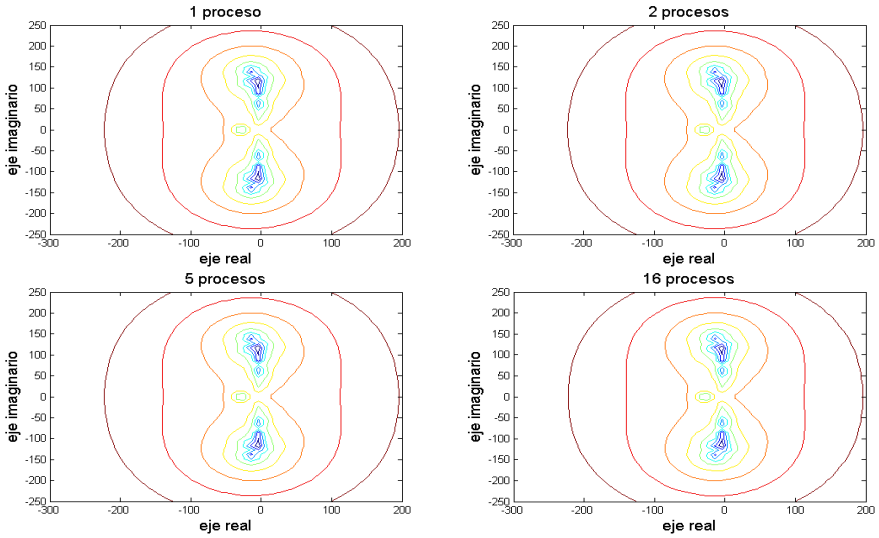

Figura 4.3: Pseudoespectro de la matriz Tolosa para 1,2,5 y 16 procesos. Los resultados generados por el Algoritmo 1 son exactamente iguales. Los niveles deseados del pseudoespectro son: $\epsilon = 10^{0,5}, 10^{0,25}, 10^{0}, \ldots, 10^{-1,5}$

número de procesos	tiempo de ejecución
1	0.5139 seg.
2	0.2850 seg.
5	0.1228 seg.
16	0.0618 seg.

Cuadro 4.3: Tiempos de ejecución para la matriz Tolosa usando 1, 2, 5 y 16 procesos.

El último experimento de esta sección se realizó sobre la matriz Toeplitz de dimensión 100 en la región [-2.5 , 2.5] $\times$ [-2.5 , 2.5]. Esta matriz se encuentra estrechamente relacionada con la transformada de Fourier [19] y es generada con el comando $A = gallery('toeppen', n, 0, 1/2, 0, 0, 1)$; en Matlab. Al igual que en los experimentos anteriores se puede notar la correctitud de los resultados obtenidos en la figura 4.4. Los tiempos de ejecución pueden ser vistos en el cuadro 4.4.

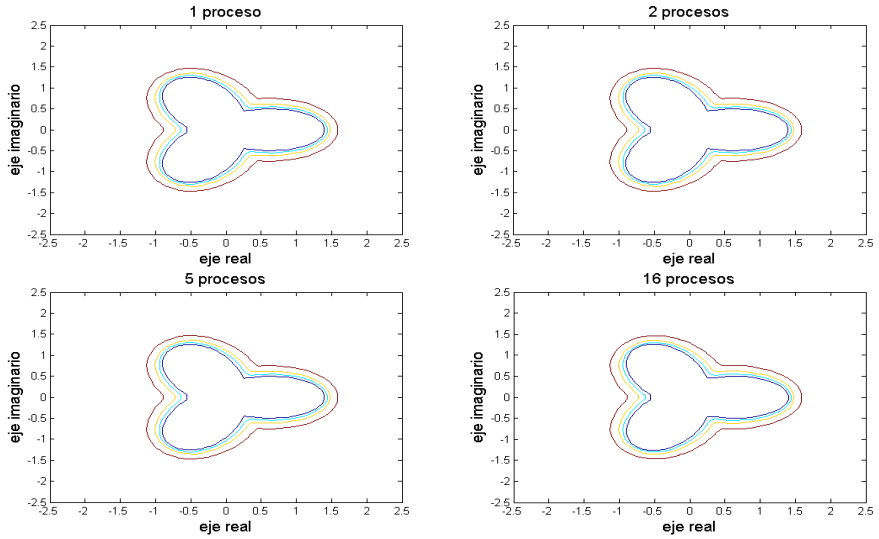

Figura 4.4: Pseudoespectro de la matriz Toeplitz usando 1, 2, 5 y 16 procesos para $\epsilon = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$. Al igual que en los casos anteriores, los resultados generados por el Algoritmo 1 son exactamente iguales

número de procesos	tiempo de ejecución
1	1.2848 seg.
2	0.6689 seg.
5	0.2986 seg.
16	0.1199 seg.

Cuadro 4.4: Tiempos de ejecución para la matriz Toeplitz usando 1, 2, 5 y 16 procesos.

4.1.2. Pruebas numéricas para matrices dispersas y de gran tamaño

Estos experimentos se realizaron utilizando el Algoritmo 2 para realizar el cálculo de $\Lambda_\epsilon(A)$ con A dispersa y de gran tamaño. Para obtener la matriz $\bar{H}_p$ se utilizó un máximo de 30000 iteraciones y se tomaron en cuenta los autovalores de mayor parte real (LR). El valor de la tolerancia fue escogido para lograr mejores aproximaciones del pseudoespectro. Todas las matrices pertenecen a la colección NEP de Matrix Market [1].

El primer experimento se realizó sobre la matriz de Jacobi (rbd800l) de dimensión 800. En la figura 4.5 se muestra una aproximación al pseudoespectro de dicha matriz en la región [-1.1 , 1.1] $\times$ [-0.2 , 2.5]. Esta matriz surge de un modelo de reacción difusión en ingeniería química [6]. El número de iteraciones para obtener la matriz $\bar{H}_p$ fue 179 en 0,9209 segundos con $p = 50$ con una tolerancia de $1{,}0^{-20}$. Los tiempos de ejecución pueden ser vistos en el cuadro 4.5.

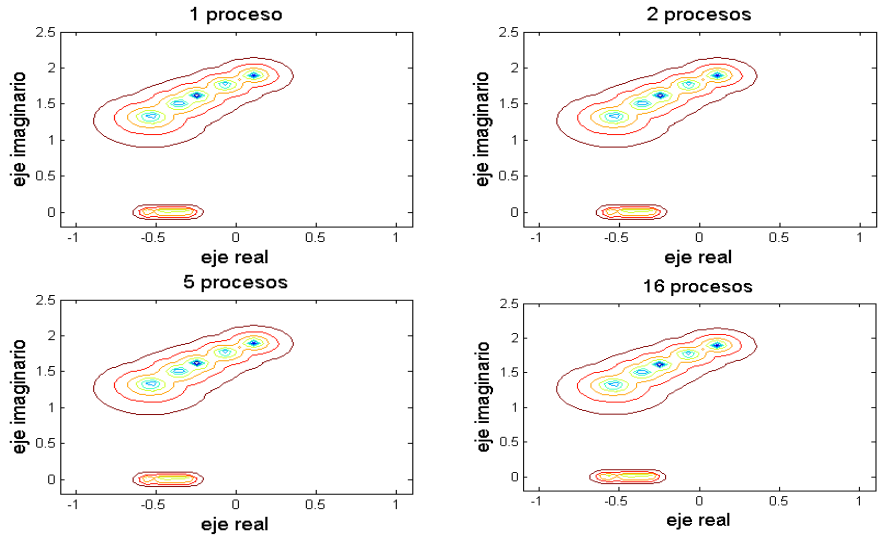

Figura 4.5: Aproximación del pseudoespectro de la matriz de Jacobi (rdb800l) con $p = 50$ y $\epsilon = 10^{-1}, 10^{-1{,}2}, \ldots, 10^{-2{,}4}$. Nótese que los resultados generados por el Algoritmo 2 son exactamente iguales para 1, 2, 5 y 16 procesos

número de procesos	tiempo de ejecución
1	0.3349 seg.
2	0.1795 seg.
5	0.0786 seg.
16	0.0314 seg.

Cuadro 4.5: Tiempos de ejecución para la matriz de Jacobi usando 1, 2, 5 y 16 procesos.

Para el segundo experimento se tomó en cuenta la matriz Bwm de dimensión 2000. En la figura 4.6 se puede ver el pseudoespectro de $\bar{H}_p$ con $p = 100$ en la región [-22 , 5] $\times$ [-7 , 7]. Se trata de una matriz utilizada para el análisis de reacciones químicas [20]. El número de iteraciones para obtener la matriz $\bar{H}_p$ es 64 en 5,7421 segundos con una tolerancia de $1{,}0^{-14}$. Los tiempos de ejecución pueden ser vistos en el cuadro 4.6.

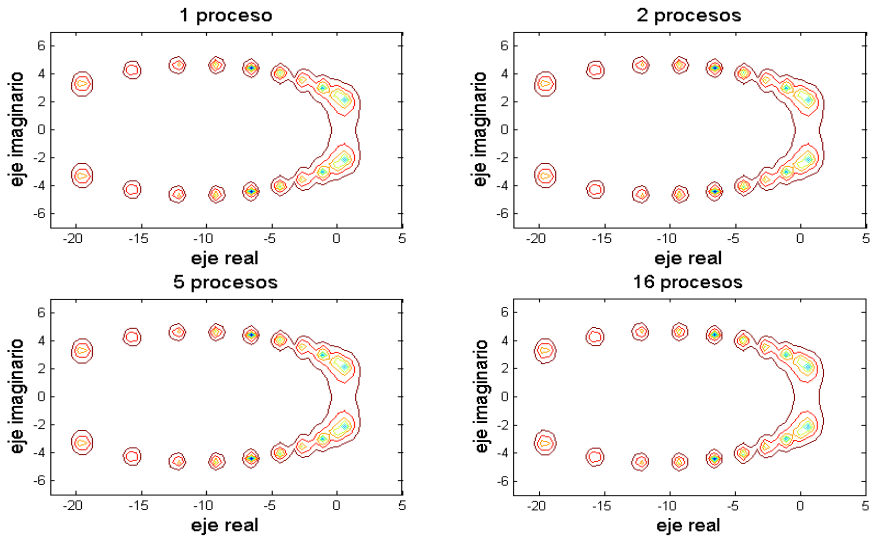

Figura 4.6: Aproximación del procesos de la matriz Bwm con $p = 100$ y $\epsilon = 10^{-0,2}, 10^{-0,2}, \ldots, 10^{-1,6}$. Los resultados generados por el Algoritmo 2 se mantienen iguales para 1,2,5 y 16 procesos

número de procesos	tiempo de ejecución
1	0.9848 seg.
2	0.5129 seg.
5	0.2074 seg.
16	0.0797 seg.

Cuadro 4.6: Tiempos de ejecución para la matriz Bwm usando 1, 2, 5 y 16 procesos.

En el tercer experimento se considera la matriz Crystal de dimensión 10000 la cual surge en la simulación del crecimiento de cristales [11]. Para este experimento se muestra el pseudoespectro de $\bar{H}_p$ con $p = 80$ en la región [-3 , 7] $\times$ [-1.5 , 1.5] en la Figura 4.7. El número de iteraciones para obtener la matriz $\bar{H}_p$ es 458 en 121,9995 segundos siendo $1{,}0^{-14}$ el valor de la tolerancia seleccionada. Los tiempos de ejecución pueden ser vistos en el cuadro 4.7.

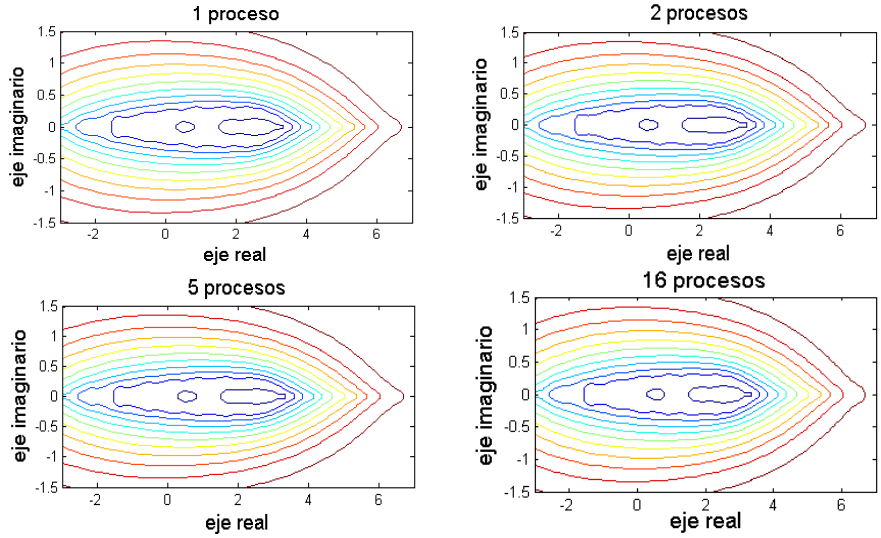

Figura 4.7: Aproximación del pseudoespectro de la matriz Crystal con $p = 80$ $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-13}$. Todos los resultados generados por el Algoritmo 2 son exactamente iguales, prueba de la correctitud del programa para 1,2,5 y 16 procesos

número de procesos	tiempo de ejecución
1	0.5919 seg.
2	0.3100 seg.
5	0.1286 seg.
16	0.0500 seg.

Cuadro 4.7: Tiempos de ejecución para la matriz Crystal usando 1, 2, 5 y 16 procesos.

El último experimento fue realizado sobre la matriz Airfoil de dimensión 23560. En el cuadro 4.8 se muestra el pseudoespectro de $\bar{H}_p$ con $p = 80$ en la región [-1.5 , 0] $\times$ [-1.5 , 1.5]. Dicha matriz es aplicable en dinámica de fluidos sobre superficies de sustentación [13]. El número de iteraciones para obtener la matriz $\bar{H}_p$ es 2312 en 2993,3020 segundos con una tolerancia de $1{,}0^{-16}$. Los tiempos de ejecución se muestran en el cuadro 4.8.

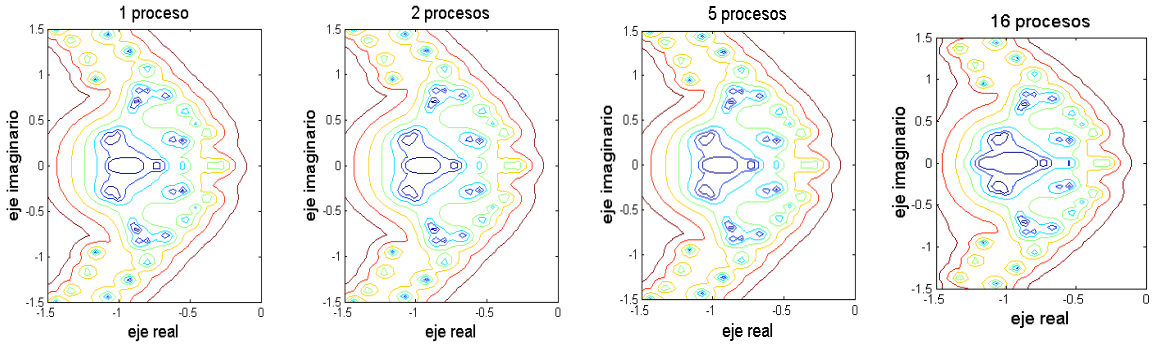

Figura 4.8: Aproximación del pseudoespectro de la matriz Airfoil para $\epsilon = 10^{-1}, 10^{-0,25}, \ldots, 10^{-2,5}$ con $p = 80$. Se puede ver claramente que los resultados obtenidos son iguales para todos los casos de este experimento

número de procesos	tiempo de ejecución
1	0.6859 seg.
2	0.3609 seg.
5	0.1526 seg.
16	0.0597 seg.

Cuadro 4.8: Tiempos de ejecución para la matriz Airfoil usando 1, 2, 5 y 16 procesos.

Se puede concluir rápidamente que los algoritmos 1 y 2 son correctos ya que los resultados son los esperados y se mantienen fijos al variar el número de procesadores así como también se puede concluir que los tiempos de cómputo obtenidos mejoran notablemente con respecto al algoritmo secuencial.

4.2. Pruebas de tiempo

En esta sección se muestran los resultados de realizar experimentos de tiempo sobre las matrices Kahan y Grcar de 64×64 con la finalidad de dar a conocer los tiempos de ejecución junto a su comportamiento, además de mostrar el comportamiento de la aceleración para mallas de 100×100, 200×200, 300×300, 400×400, 500×500 y $1, 2, 4, 8, 10, 12, 16$ procesos. La matriz Kahan fue escogida como representante de las matrices normales y la matriz Grcar fue escogida como representante de las matrices no-normales.

4.2.1. Pruebas de tiempo para la matriz Kahan

A continuación se muestran los tiempos de ejecución junto a su comportamiento además del comportamiento de la aceleración para los experimentos aplicados sobre la matriz Kahan.

resolución de malla	**tiempo de ejecución (segundos)**						
	p = 1	**p = 2**	**p = 4**	**p = 8**	**p = 10**	**p = 12**	**p = 16**
100×100	0,6319	0,3424	0,1792	0,0945	0,0757	0,0643	0,0567
200×200	2,5166	1,3378	0,6650	0,3372	0,2741	0,2335	0,1767
300×300	5,6581	2,9785	1,4849	0,7387	0,5957	0,4947	0,3740
400×400	10,0725	5,3247	2,6132	1,3025	1,0442	0,8717	0,6555
500×500	15,7286	8,2722	4,0836	2,0333	1,6181	1,3514	1,0135

Cuadro 4.9: Tiempos de ejecución asociados a la matriz Kahan para $1, 2, 4, 8, 10, 12$ y 16 procesos con mallas de 100×100, 200×200, 300×300, 400×400 y 500×500 correspondientes al módulo encargado de calcular los valores singulares.

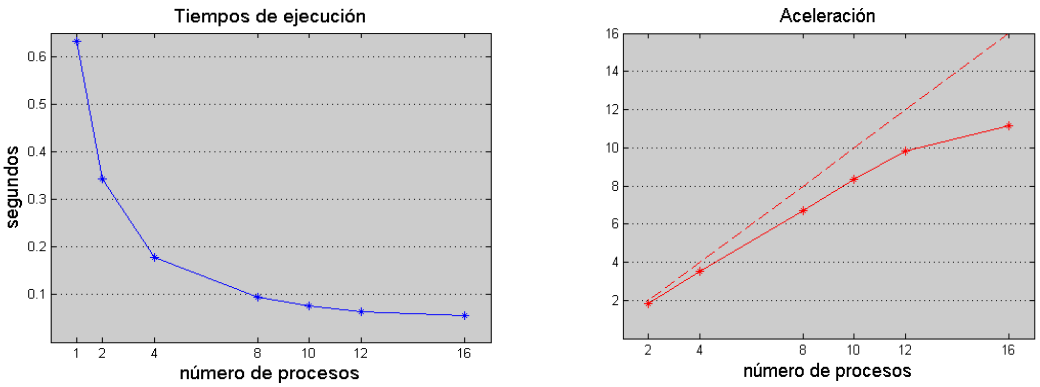

Figura 4.9: Tiempos de ejecución y aceleración para mallas de 100×100

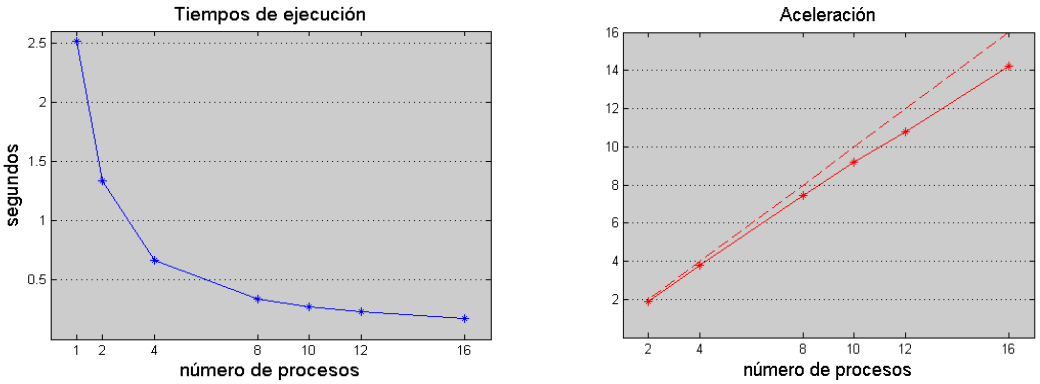

Figura 4.10: Tiempos de ejecución y aceleración para mallas de 200×200

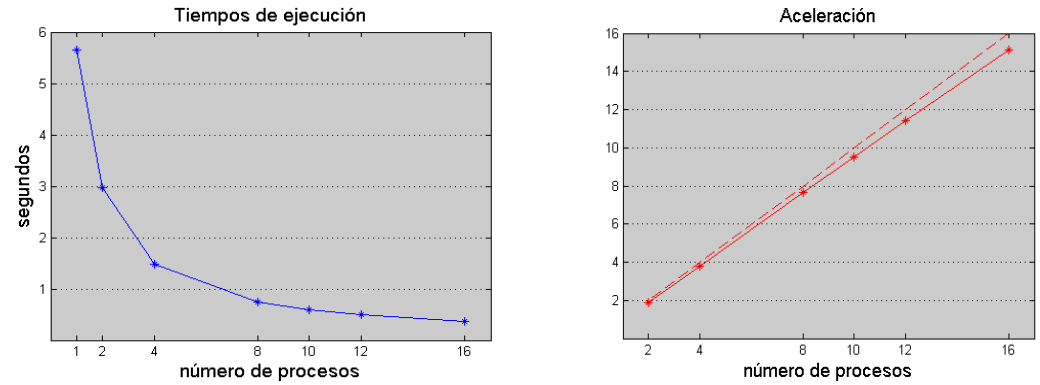

Figura 4.11: Tiempos de ejecución y aceleración para mallas de 300×300

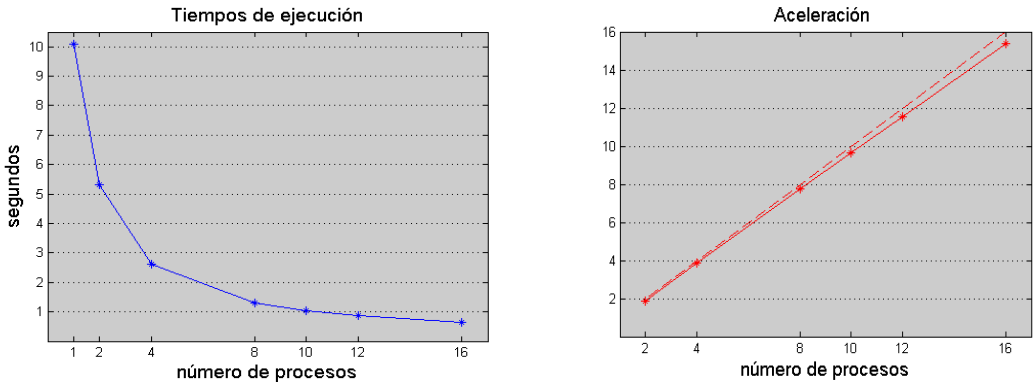

Figura 4.12: Tiempos de ejecución y aceleración para mallas de 400×400

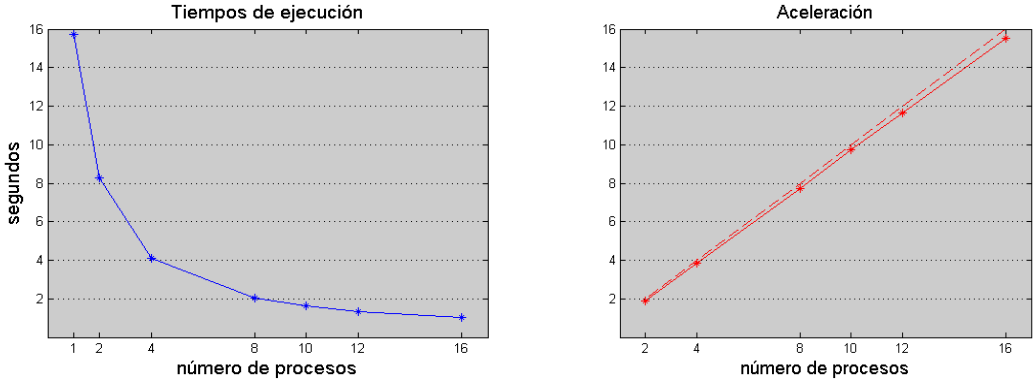

Figura 4.13: Tiempos de ejecución y aceleración para mallas de 500×500

Se puede ver que los tiempos de ejecución disminuyen notablemente mientras que la aceleración aumenta a medida que crece el número de procesos para todas las resoluciones de malla. Nótese que en general, el comportamiento de la aceleración se va haciendo lineal a medida que crece la resolución de la malla.

4.2.2. Pruebas de tiempo para la matriz Grcar

A continuación se muestran los tiempos de ejecución junto a su comportamiento además del comportamiento de la aceleración para los experimentos aplicados sobre la matriz Grcar.

resolución de malla	**tiempo de ejecución (segundos)**						
	p = 1	**p = 2**	**p = 4**	**p = 8**	**p = 10**	**p = 12**	**p = 16**
100 × 100	1,1748	0,6554	0,3807	0,1912	0,1648	0,1394	0,1097
200 × 200	4,6733	2,5711	1,4292	0,7289	0,5908	0,5051	0,3811
300 × 300	10,4784	5,7046	3,1303	1,6393	1,3497	1,1333	0,8394
400 × 400	18,6112	10,2249	5,5559	2,9068	2,3720	1,9863	1,4866
500 × 500	29,0576	15,9636	8,6829	4,5428	3,7118	3,0570	2,3000

Cuadro 4.10: Tiempos de ejecución asociados a la matriz Grcar para $1, 2, 4, 8, 10, 12$ y 16 procesos con mallas de 100×100, 200×200, 300×300, 400×400 y 500×500 correspondientes al módulo encargado de calcular los valores singulares.

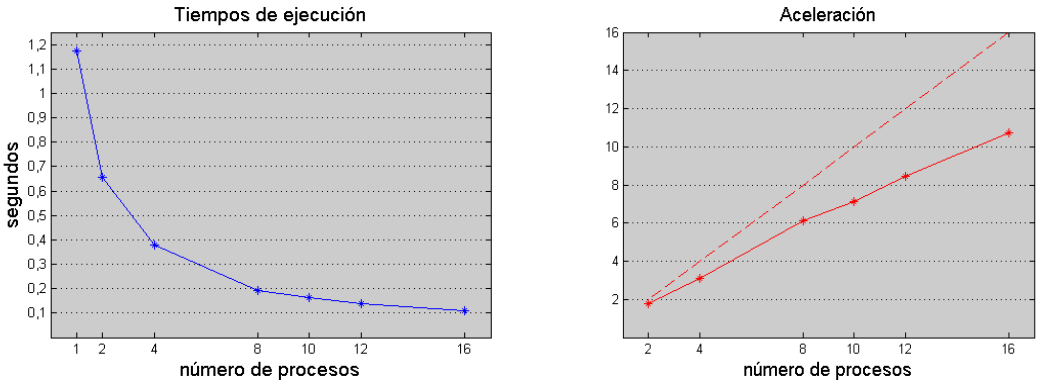

Figura 4.14: Tiempos de ejecución y aceleración para mallas de 100×100

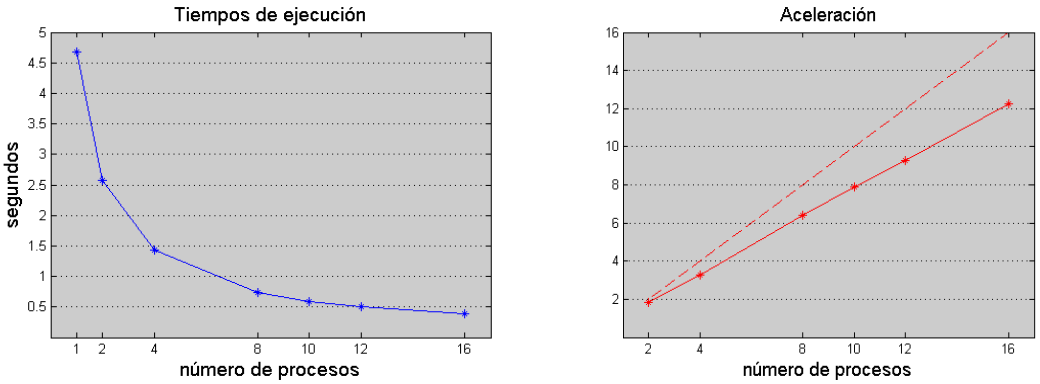

Figura 4.15: Tiempos de ejecución y aceleración para mallas de 200×200

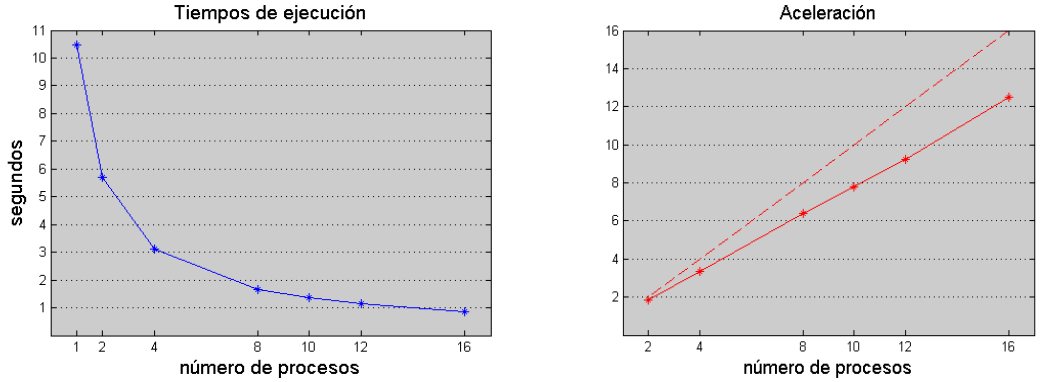

Figura 4.16: Tiempos de ejecución y aceleración para mallas de 300×300

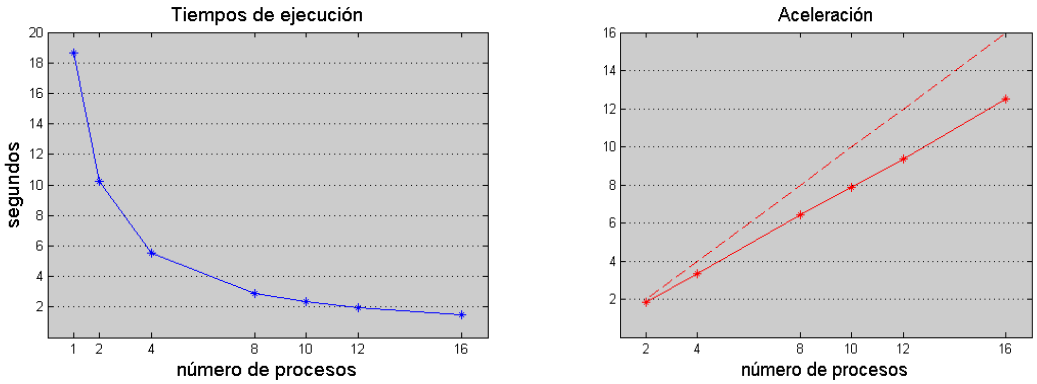

Figura 4.17: Tiempos de ejecución y aceleración para mallas de 400×400

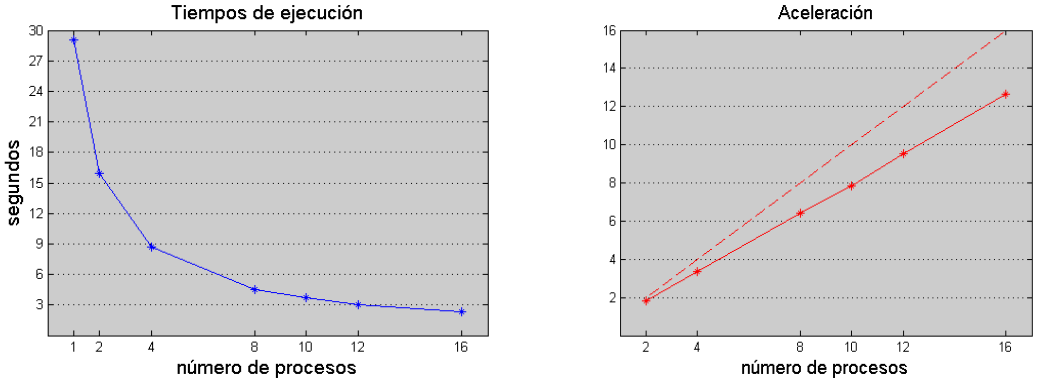

Figura 4.18: Tiempos de ejecución y aceleración para mallas de 500×500

Al igual que con la matriz Kahan se puede ver que los tiempos de ejecución disminuyen notablemente mientras que la aceleración aumenta a medida que crece el número de procesos para todas las resoluciones de malla. Es importante mencionar que el comportamiento de la aceleración se mantiene por debajo del lineal a medida que crece la resolución de la malla. Esto se debe a que la matriz Grcar es una matriz muy no-normal y su pseudoespectro es muy difícil de calcular a diferencia de la matriz Kahan, cuyo pseudoespectro es mucho más fácil de calcular.

4.2.3. Escalabilidad del módulo paralelo

Un programa paralelo es escalable siempre y cuando su aceleración crezca proporcionalmente a algún factor al aumentar el número de procesos para un determinado tamaño de datos [17]. Las siguientes gráficas muestran dicho crecimiento para las matrices Kahan y Grcar como prueba de la escalabilidad del módulo paralelo.

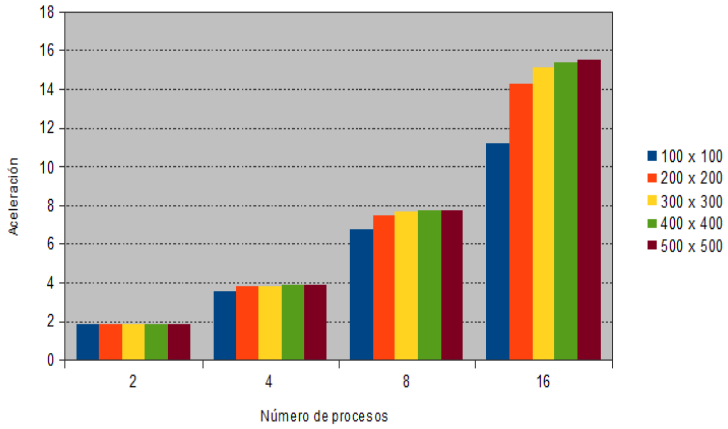

Figura 4.19: Aceleración del módulo paralelo en función del número de procesos para la matriz Kahan aumentando la resolución de la malla.

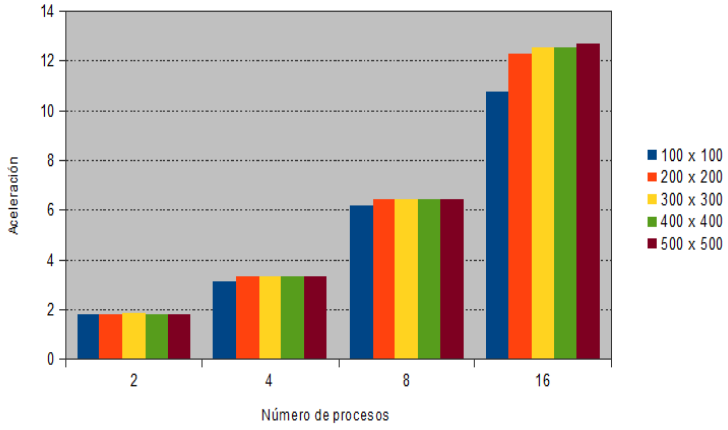

Figura 4.20: Aceleración del módulo paralelo en función del número de procesos para la matriz Grcar aumentando la resolución de la malla.

En ambas figuras se puede ver que para cada resolución de malla la aceleración crece con cierto factor a medida que el número de procesos se duplica, por lo tanto el módulo paralelo es escalable. A continuación se muestra un gráfico donde se encuentran dichos factores de crecimiento para ambas matrices en función de la resolución de la malla:

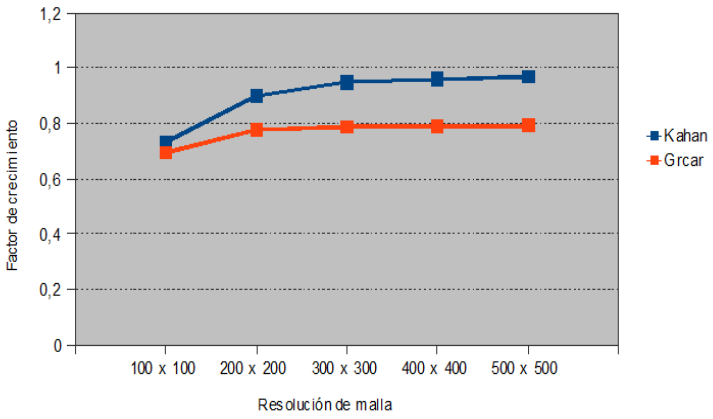

Figura 4.21: Factor de crecimiento del rendimiento del módulo paralelo en función de la resolución de la malla para las matrices Kahan y Grcar.

A partir de los resultados obtenidos se puede concluir que el módulo paralelo tiene un desempeño muy alto para la matriz Kahan, ya que su aceleración aumenta notablemente a medida que aumenta el número de procesos y la resolución de la malla, lo cual se ve reflejado en la figura 4.19, cosa que no ocurre en el caso de la matriz Grcar ya que para esta matriz la aceleración crece mucho menos que la aceleración de la matriz Kahan, esto se puede ver reflejado en la figura 4.20.

Además se puede ver claramente en la figura 4.21 que el factor de crecimiento de la aceleración del módulo paralelo aumenta considerablemente al aumentar la resolución de la malla para el caso de la matriz Kahan, a diferencia de la matriz Grcar donde dicho factor aumenta ligeramente para luego mantenerse cerca de 0,8; lo cual prueba la influecia de la resolución de la malla sobre la aceleración del módulo paralelo ya que para ambas matrices dicha aceleración no sólo crece al aumentar el número de procesos en paralelo, sino que también crece al aumentar la resolución de la malla.

Todo lo señalado anteriormente resulta suficiente para probar no sólo la escalabilidad del módulo para el cálculo del pseudoespectro en paralelo, sino también que la aceleración del mismo depende de la resolución de la malla y de la estructura espectral de la matriz de entrada.

Capítulo 5

Conclusiones y recomendaciones

Los fundamentos teóricos del cálculo del pseudoespectro junto a los métodos numéricos utilizados actualmente para realizar dicho cálculo han sido escritos en este documento, así como también el esquema de paralelización para el cálculo del pseudoespectro. La importancia del tema se puede ver claramente en varios ejemplos de aplicaciones reales y cuenta con el soporte de las referencias indicadas en este documento. Como resultado de esta investigación se han generado dos algoritmos para realizar el cálculo del pseudoespectro en paralelo tanto para matrices densas como para matrices dispersas haciendo paralelismo de datos. Además, se analizan los resultados obtenidos luego de una rigurosos experimentos a través de numerosas pruebas numéricas y de tiempo. Dicho análisis genera las siguientes conclusiones:

Con respecto a los trabajos realizados con anterioridad en el área de cálculo en paralelo del pseudoespectro se puede decir que en éste trabajo se logró realizar paralelización de datos en el cálculo del pseudoespectro, algo que no se había hecho en ninguno de los trabajos anteriores. Esto abre nuevas líneas de investigación tanto en el área de análisis numérico como en el área de paralelismo. En cuanto al área de análisis numérico resulta interesante paralelizar nuevos métodos de cálculo del pseudoespectro mientras que en el área de paralelismo es de interés realizar comparaciones de rendimiento entre los mismos.

En cuanto a los métodos planteados en el capítulo 2, se puede decir que el método de lanczos para calcular el pseudoespectro, el cual surge como variante del método de la iteración inversa ha sido paralelizado de forma exitosa, esto se ve reflejado en los resultados obtenidos de las pruebas numéricas del capítulo 4.

El método de proyección de Arnoldi planteado en el capítulo 2 ha sido implementado con éxito como se prueba en el capítulo 4. Con este avance se puede realizar el cálculo del pseudoespectro de matrices grandes y dispersas en poco tiempo lo cual resulta de mucha utilidad en áreas de la ciencia e ingeniería donde haya que

resolver problemas de estabilidad de sistemas los cuales involucren matrices grandes y dispersas.

Acerca de los módulos implementados en el capítulo 3 es importante mencionar que:

1. El módulo para realizar el cálculo de los valores singulares en paralelo planteado en el capítulo 3 genera nuevas investigaciones en torno al paralelismo funcional a pesar de que sólo hace paralelismo de datos. Dichas investigaciones resultan de interés para obtener una versión del módulo la cual permita nuevos niveles de paralelismo. Para este fin es recomendable estudiar diferentes esquemas de paralelismo funcional y adaptar el más conveniente al módulo encargado de realizar el cálculo del pseudoespectro en paralelo.

2. El módulo encargado de realizar la extracción de la matriz $\bar{H}_p$ pudiera mejorar aún más su rendimiento si se hace una paralelización del mismo que aproveche las bondades del paralelismo funcional y del paralelismo de datos. Para ello se recomienda estudiar la biblioteca PARPACK y utilizarla para realizar una versión del módulo que realice la extracción de la matriz $\bar{H}_p$ en paralelo.

La principal ventaja del esquema de particionamiento de datos planteado para arquitecturas paralelas en el presente trabajo de grado es la mejora en los tiempos de ejecución y la escalabilidad que presenta. Dicho esquema puede utilizarse con cualquier otro método para el cálculo del pseudoespectro, así como también se pueden utilizar otros esquemas de distribición de datos para realizar dicho cálculo. Adicionalmente existen otras arquitecturas de alto rendimiento que pudieran ejecutar el esquema propuesto, tales como tarjetas gráficas programables. Por tanto es recomendable realizar implementaciones de este esquema en tarjetas gráficas programables utilizando otros métodos para el cálculo del pseudoespectro además del método del algoritmo de Lanczos a fin de comparar los tiempos de ejecución y la alta escalabilidad de cada implementación. También es recomendable plantear otros esquemas de particionamiento de datos a fin de mejorar aún más los tiempos de ejecución y el rendimiento del módulo paralelo.

Bibliografía

[1] 2007. Matrix Market. `http://math.nist.gov/MatrixMarket/`.

[2] Arnoldi, W. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quartely of applied mathematics 9* (1951), 17–29.

[3] Aslanyan, A., and Davies, E. B. Spectral instability for some Schrödinger operators. *Numerische Mathematik 85*, 4 (2000), 525–552.

[4] Astudillo, R. Análisis e implementación de nuevos esquemas para el cálculo del pseudoespectro. Master's thesis, Universidad Central de Venezuela, 2011.

[5] Astudillo, R., and Castillo., Z. Computing pseudospectra using block implicitly restarted Arnoldi iteration. *Mathematical and Computer Modelling*, doi:10.1016/j.mcm.2011.03.022.

[6] B.D. Hassard, N. K., and Wan, Y. *Theory and applications of Hopf bifurcation*. Cambridge, UK, 1981.

[7] Bekas, C., and Gallopoulos, E. Parallel computation of pseudospectra by fast descent. *Parallel Computing 28*, 2 (2002), 223–242.

[8] Braconnier, T. Fvpspack: A Fortran and PVM package to compute the field of values and pseudospectra of large matrices. Numerical Analysis Report No. 293, Department of Mathematics, University of Manchester, Manchester M13 9PL, England, 1996.

[9] Braconnier, T., and Higham, N. Computing the field of values and pseudospectra using the Lanczos method with continuation. Numerical Analysis Report No. 279, Manchester Centre for Computational Mathematics, Manchester, M13 9PL, England, Nov. 1995.

[10] Burke, J. V., Lewis, A. S., and Overton, M. L. Optimization and pseudospectra, with applications to robust stability. *SIAM Journal on Matrix Analysis and Applications 25*, 1 (2003), 80.

[11] C. Yang, D. Sorensen, D. M., and Wedeman, B. Numerical computation of the linear stability of the diffusion model for crystal growth simulation. Technical Report TR96-04, Department of Comp. & App. Mathematics, Rice University, Houston, TX, 1996.

[12] Castillo, Z. *A New Algorithm for Continuation and Bifurcation Analysis of Large Scale Free Surface Flows*. PhD thesis, Rice University, 2004. ISBN 9729961506.

[13] D. Mahajan, E. H. D., and Blis, D. Eigenvalue calculation procedure for an Euler-Navier-Stokes solver with applications to flows over airfoils. *J. Comput. Phys.*, 97 (1991), 398–413.

[14] D. R. Fokkema, G. L. G. S., and der Vorst, H. A. V. Jacobi-davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM j. Sci. Comput.*, 20 (1999), 94–125.

[15] Datta, B. N. *Numerical Linear Algebra and Applications*. Brooks/Cole, Pacific Grove, CA, 1995.

[16] Frayssé, V., Giraud, L., and Toumazou, V. Parallel computation of spectral portraits on the Meiko CS2. Tech. Rep. TR/PA/96/02, CERFACS, Toulouse, France, 1996. Preliminary version of proceeding in High-Performance Computing and Networking, 1996.

[17] Giusti, L. D., and Tarrío, D. Escalabilidad en algoritmos paralelos de cálculo del costo mínimo de camino en grafos. Tech. rep., Laboratorio de Investigación y Desarrollo en Informática, Facultad de Informática, Universidad Nacional de la Plata.

[18] Grcar, J. Operator coefficient methods for linear equations. Report SAND89-8691, Sandia National Laboratory, 1989.

[19] H. Dai, Z. Geary, L. P. K. Asymptotics of eigenvalues and eigenvectors of Toeplitz matrices. *Journal of Statistical Mechanics: Theory and Experiment*, 05 (2009), 1742–5468.

[20] Holmes, P. J. *Waves in distributed chemical systems: experiments and computations*. Philadelphia. SIAM. Proceedings of the Asilomar Conference Ground, Pacific Grove, California, 1979.

[21] J. Barconnier, F. Chatelin, J. C. D. Highly nonnormal eigenproblems in the aeronautical industry. *Japan J. Indust. Appl. Math.*, 12 (1995), 123–136.

[22] Jonsson, G. F., and Trefethen, L. N. A numerical analyst looks at the cutoff phenomenon in card shuffling and other Markov chains. Tech. rep., Center for Applied Mathematics, Cornell University, Ithaca, New York, USA, 1996.

[23] Kahan, W. Numerical linear algebra. *Canad. Math. Bull.*, 9 (1966), 757–801.

[24] Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *j. Res. Natl. Bur. Stand 45* (1950), 225–280.

[25] Lanczos, C. Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand 49* (1952), 33–53.

[26] Landau, H. J. Loss in unstable resonators. *J. Opt. Soc. Am. 66*, 6 (1976), 525–529.

[27] Lui, S. Computation of pseudospectra by continuation. *SIAM J. Sci. Comput. 18*, 2 (1997), 565–573.

[28] Mengi, E., and Overton, M. L. Algorithms for the computation of the pseudospectral radius and the numerical radius of a matrix. *IMA Journal of Numerical Analysis 25*, 4 (2005), 648–669.

[29] Saad, Y. *Numerical Methods for Large Eigenvalue Problems.* Halstead Press, New York, 1992.

[30] Sorensen, D. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl. 13*, 1 (1992), 357–385.

[31] Talbot, C., and Crampton, A. Pseudo spectral methods applied to problems in elasticity. *J. Sci. Comput. 27* (2006), 443–454.

[32] Toh, K., and Trefethen, L. Calculation of pseudospectra by the Arnoldi iteration. *SIAM Journal on Scientific Computing 17*, 1 (1996), 1–15.

[33] Trefethen, A. E., Trefethen, L. N., and Schmid, P. J. Spectra and pseudospectra for pipe Poiseuille flow. *Computer Methods in Applied Mechanics and Engineering 175*, 3 (1999), 413–420.

[34] Trefethen, L. Pseudospectra of matrices. In *Numerical analysis* (Harlow, Essex, 1991), D. F. Griffiths and G. A. Watson, Eds., vol. 260, Longman Scientific and Technical, pp. 234–266.

[35] Trefethen, L. Computation of pseudospectra. *Acta Numerica 8*, 1 (1999), 247–295.

[36] Trefethen, L., A. Trefethen, S. R., and Driscoll, T. Hydrodynamic stability without eigenvalues. *Science 261*, 5121 (1993), 578–584.

[37] Trefethen, L., and Embree, M. *Spectra and Pseudospectra: the behavior of nonnormal Matrices and Operators.* Princeton University Press, New Jersey, USA, 2005.

[38] Trefethen, L., and Wright, T. Large-scale computation of pseudospectra using ARPACK and eigs. *SIAM J. Sci. Comput. 23* (2001), 591–605.

[39] Wright T. G. Algorithms and software for pseudospectra, 2002. D. Phil. thesis, Oxford University.