Aceleración de la convergencia en métodos de Jacobi para el Problema Simétrico de Valores Propios

María T. Cámara Javier Cuenca Domingo Giménez *

Abstract

En este trabajo comparamos dos técnicas de aceleración de la convergencia en métodos de Jacobi para la solución del Problema Simétrico de Valores Propios: uso de umbrales y métodos semiclásicos. Se analizan las dos técnicas y su posible combinación con métodos de Jacobi por bloques.

1 INTRODUCCION

El Problema Simétrico de Valores Propios se puede resolver por métodos de Jacobi. Para reducir el tiempo de ejecución de estos métodos se puede acelerar la convergencia utilizando técnicas de umbralización [1] o métodos semiclásicos [2]. Otra posibilidad de reducción del tiempo de ejecución consiste en usar métodos por bloques [3] con los que se pueden obtener programas eficientes tanto en procesadores secuenciales [4] como en Multiprocesadores [5].

En la sección 2 compararemos las técnicas de umbralización con los métodos semiclásicos y en la sección 3 analizaremos la posible combinación de estos métodos con métodos por bloques.

Previamente analizaremos las características principales de los métodos de Jacobi que utilizamos.

*Departamento de Informática y Sistemas. Univ de Murcia. Aptdo 4021. 30001 Murcia. (domingo@dif.um.es)

1.1 Métodos de Jacobi

Un método de Jacobi trabaja construyendo una secuencia de matrices $\{A_l\}$ por medio de

$$A_{l+1} = Q_l A_l Q_l^t \ , \ l = 1, 2, \ldots \tag{1}$$

donde $A_1 = A$, y Q_l es una rotación de Givens en el plano (p, q), con $1 \leq p, q \leq n$. Bajo ciertas condiciones [6, 7], la secuencia $\{A_l\}$ converge a una matriz diagonal D,

$$D = Q_k Q_{k-1} \ldots Q_2 Q_1 A Q_1^t Q_2^t \ldots Q_{k-1}^t Q_k^t \tag{2}$$

cuyos elementos diagonales son los valores propios de A, y los vectores propios son las columnas de $Q_1^t Q_2^t \ldots Q_{k-1}^t Q_k^t$.

Cada producto $Q_l A_l Q_l^t$ representa una transformación de semejanza que anula un par de elementos no diagonales, a_{ij} y a_{ji}, de la matriz A_l. La matriz Q_l coincide con la identidad excepto en los elementos $q_{ii} = c$, $q_{ij} = s$, $q_{ji} = -s$, y $q_{jj} = c$, donde $c = \cos\ \theta$, $s = \sin\ \theta$, y [7]

$$\tan\ 2\theta = \frac{2a_{ij}}{a_{ii} - a_{jj}} \tag{3}$$

1.1.1 Método clásico

El método de Jacobi clásico procede eligiendo en cada iteración como elemento a anular el de mayor valor absoluto de entre los no diagonales. Esto produce que el número de iteraciones sea pequeño pero que el tiempo de ejecución sea muy grande debido al trabajo adicional del cálculo del máximo.

Un esquema del método sería:

Algoritmo 1 *Esquema del método de Jacobi clásico.*

MIENTRAS $off(A) > cota$
  Calcular a_{ij} con máximo valor absoluto de entre los no diagonales
  Calcular Rotación $Q(i, j, \theta)$ tal que $(QAQ^t)_{ij} = 0$
  $A = QAQ^t$
FINMIENTRAS

donde $off(A) = \sqrt{\sum_{j=1,j\neq i}^{n} \sum_{i=1}^{n} a_{ij}^2}$.

El número de flops necesario para la actualización de la matriz en cada pasada por el núcleo del bucle es de orden $O(n)$, pero el cálculo del máximo produce que el orden de cada pasada por el núcleo del bucle teniendo en cuenta las comparaciones sea $O(n^2)$. De este modo, las $\frac{n(n-1)}{2}$ anulaciones suponen un coste de $O(n^3)$ flops, y de $O(n^4)$ comparaciones (es frecuente denominar barrido a la realización de $\frac{n(n-1)}{2}$ anulaciones, especialmente en los métodos cíclicos que se estudian a continuación).

1.1.2 Métodos cíclicos

Los métodos cíclicos proceden realizando sucesivos barridos, anulando en cada barrido una vez cada elemento no diagonal (con lo que cada barrido constará de $\frac{n(n-1)}{2}$ anulaciones) hasta que se cumple algún criterio de convergencia. Normalmente hasta que $off(A) < cota$. Para ello se utiliza un cierto orden de anulación de los elementos, por ejemplo, el método cíclico por filas anula los elementos en el orden dado por los pares:

$$(1,2),(1,3),\ldots,(1,n),(2,3),(2,4),\ldots,(2,n),\ldots,(n-1,n) \quad . \qquad (4)$$

Un esquema en este caso podría ser:

Algoritmo 2 *Esquema de un método de Jacobi cíclico.*

```
MIENTRAS off(A) > cota
    MIENTRAS queden pares en la ordenación
        Obtener el siguiente par (i, j)
        Calcular Rotación Q(i, j, θ) tal que (QAQ^t)_ij = 0
        A = QAQ^t
    FINMIENTRAS
FINMIENTRAS
```

De esta forma se evita el cálculo del máximo y se obtiene un orden $O(n^3)$ por barrido, pues cada anulación necesita de 11 flops en el cálculo de los parámetros de las rotaciones y $6n + 12$ flops en la actualización de A, lo que hace $6n+23$ flops por anulación y $\frac{(6n+23)n(n-1)}{2}$ flops por barrido para el cálculo de los valores propios, lo que da un valor de $3n^3$ flops para los términos de mayor orden. El cálculo de los valores y vectores propios tendrá un coste por barrido de $6n^3$ flops, pues habrá que acumular las rotaciones. Por otro lado, ya sabemos que el coste del método clásico en $\frac{n(n-1)}{2}$ anulaciones, teniendo en cuenta las comparaciones, es $O(n^4)$. Sin embargo, con un método cíclico, al eliminarse los elementos en un orden predeterminado, para alcanzar la convergencia se necesita de más iteraciones que en el método clásico.

1.1.3 Comparación clásico-cíclicos

En la tabla 1 comparamos los tiempos de ejecución con el método clásico (JCLA) y con distintos métodos cíclicos (JFIL es el método por filas [6], JEBE con la ordenación de Eberlein [8], JRR con el Round-Robin [9]). Los resultados se muestran en segundos y han sido obtenidos en una HP Apollo 700, utilizando matrices reales, simétricas y densas con elementos de doble precisión generados aleatoriamente con valores entre -10 y 10.

tamaño	128	192	256	320	384	448	512
JCLA	479	2426	7868	20941			
JFIL	17	55	176	324	626	1004	1754
JEBE	17	55	177	339	634	1015	1772
JRR	17	55	179	340	642	1144	1787

Tabla 1: Comparación de métodos cíclicos y clásico. Tiempos de ejecución en segundos en una HP Apollo 700.

En la tabla 2 comparamos la velocidad de convergencia con distintos métodos cíclicos y el método clásico. En los métodos cíclicos se representa el número de barridos y en el método clásico el número de anulaciones dividido por el número de anulaciones de un barrido ($\frac{n(n-1)}{2}$).

Como se puede ver en las tablas 1 y 2, los resultados obtenidos con distintos métodos cíclicos son muy similares (el número de barridos se considera que es del orden $O(log_2\, n)$ [9]).

Se puede comprobar también que el método clásico es mucho peor que los métodos cíclicos en tiempo de ejecución, pero que necesita de menos anulaciones para alcanzar la convergencia.

tamaño	128	192	256	320	384	448	512
JCLA	4	4	4	4			
JFIL	8	8	9	9	9	9	9
JEBE	8	8	9	9	9	9	9
JRR	8	8	9	9	9	10	9

Tabla 2: Velocidad de convergencia de métodos cíclicos y clásico (se representa el número de barridos en el caso de los métodos cíclicos y en el método clásico el número de anulaciones dividido por el número de anulaciones de un barrido). Resultados obtenidos en una HP Apollo 700.

2 Aceleración de la convergencia

2.1 Técnicas de umbralización

Para acelerar la convergencia es posible utilizar **umbrales**, de modo que cuando el valor absoluto del elemento a anular esté por debajo del umbral no se realice la anulación ni la actualización de la matriz. De este modo se evitarán anulaciones de elementos pequeños que contribuirían poco a la convergencia.

Utilizando una técnica de umbralización adecuada se asegura la convergencia del método [10] y se disminuye el tiempo de ejecución llegándose a obtener reducciones del 33% en los tiempos de ejecución dependiendo de la técnica de umbralización utilizada y de la máquina en que se ejecuten los programas. En [1, 10] se enumeran algunas posibles técnicas de umbralización:

1. Una primera posibilidad es usar un umbral fijo para todas las iteraciones. Si la condición de fin es que $off(A) < cota$, tomando el umbral como $\frac{cota}{n}$ se asegura que se cumple la condición de fin cuando se realiza un barrido completo sin anular ningún elemento.

2. La técnica de Rutishauser consiste en utilizar umbral variable durante los 4 primeros barridos, y en los restantes usar umbral fijo. El umbral variable viene dado por la fórmula:

$$\frac{1}{5}\left(\sum_{i=1}^{n-1}\sum_{j=1,i<j}^{n}\frac{|a_{ij}|}{n}\right)^2 \tag{5}$$

3. Utilizar una sucesión de umbrales, uno para cada uno de los barridos, disminuyendo el valor del umbral al aumentar el número de barrido, por ejemplo 2^{-3}, 2^{-6}, 2^{-10}, ...

4. La técnica de umbral variable de Kahan y Corneil, donde inicialmente se calcula ω como:

$$\omega = \sum_{i=1}^{n-1}\sum_{j=1,i<j}^{n} a_{ij}^2 \tag{6}$$

   y tras cada actualización se recalcula ω restándole a_{ij}^2. Para decidir si se aplica o no una rotación al elemento a_{ij} estudiaremos si

$$\frac{n(n-1)}{2}a_{ij}^2 > \omega \tag{7}$$

   lo que quiere decir que anularemos sólo los elementos cuyo cuadrado esté por encima de la media de los cuadrados.

Un esquema para los métodos de Jacobi cíclicos con umbral podría ser:

Algoritmo 3 *Esquema de un método de Jacobi cíclico con umbral.*

MIENTRAS $off(A) > cota$
  MIENTRAS queden pares en la ordenación
    Obtener el siguiente par (i, j)
    SI $|a_{ij}| > umbral$
      Calcular Rotation $Q(i, j, \theta)$ tal que $(QAQ^t)_{ij} = 0$
      $A = QAQ^t$
      Actualizar $umbral$
    FINSI
  FINMIENTRAS
FINMIENTRAS

Se han realizado algunos experimentos con las diferentes técnicas de umbralización que hemos enumerado utilizando la ordenación cíclica por filas, en los que se observa que se reduce el tiempo de ejecución con el uso de umbrales. Los mejores resultados los hemos obtenido con el umbral de Kahan y Corneil, por lo que en lo sucesivo utilizaremos esta técnica.

En la tabla 3 se comparan los tiempos de ejecución con el orden cíclico por filas no utilizando y utilizando umbral, y en la tabla 4 se compara la velocidad de convergencia. En el caso en que se usa umbral se representa el cociente del número de anulaciones llevadas a cabo entre el número de elementos por encima de la diagonal, y cuando no se usa umbral se representa el número de barridos (que es también en este caso el cociente entre el número anulaciones y el número de elementos por encima de la diagonal). Los resultados que se muestran han sido obtenidos en un procesador i860.

tamaño	256	320	384	448	512
sin umbral	134.80	279.87	493.09	825.54	1230.12
con umbral	114.39	224.88	383.82	618.49	907.56

Tabla 3: Uso de umbral (de Kahan y Corneil). Tiempos de ejecución en segundos en el procesador i860.

tamaño	256	320	384	448	512
sin umbral	10	10	10	10	10
con umbral	5.93	6.03	6.08	6.10	6.14

Tabla 4: Uso de umbral (de Kahan y Corneil). Número de anulaciones partido por $\frac{n(n-1)}{2}$. En un procesador i860.

En la tabla 5 se muestran los cocientes de los tiempos de ejecución y del número de barridos obtenidos usando umbral y no usándolo. Se puede observar que la ganancia en el número de barridos es mucho mayor (alrededor de un 39%) que en el tiempo de ejecución (hasta un 27%), lo que es debido al coste adicional de la actualización del umbral, la comparación para decidir si se anula el elemento, etc...

tamaño	256	320	384	448	512
tiempo	0.84	0.80	0.77	0.74	0.73
barridos	0.59	0.60	0.60	0.61	0.61

Tabla 5: Cociente entre métodos usando umbral (de Kahan y Corneil) y no usando umbral. En un procesador i860.

En las tablas 6 y 7 se muestran los resultados obtenidos ejecutando los programas en una HP Apollo 700. La reducción en el número de barridos es igual que la obtenida en el procesador i860, por lo que no se muestra. Sin embargo, se observa que en diferentes máquinas una misma reducción en los barridos produce reducciones distintas en los tiempos de ejecución debido a la diferencia de los costes de las operaciones aritméticas, de comparación y de acceso a memoria.

tamaño	256	320	384	448	512
sin umbral	87	151	319	506	1087
con umbral	59	110	222	359	713

Tabla 6: Uso de umbral (de Kahan y Corneil). Tiempos de ejecución en segundos. En una HP Apollo 700.

tamaño	256	320	384	448	512
tiempo	0.68	0.72	0.69	0.71	0.65
barridos	0.59	0.60	0.60	0.61	0.61

Tabla 7: Cociente entre métodos usando umbral (de Kahan y Corneil) y no usando umbral. En una HP Apollo 700.

2.2 Método semiclásico

2.2.1 Idea general

Los métodos cíclicos producen mucho mejores tiempos de ejecución que el método clásico debido a que realizan un barrido según una determinada ordenación evitando la obtención del máximo antes de cada anulación, obteniéndose de este modo un coste $O(n^3)$ por barrido mientras que con el método clásico el coste por barrido (por $\frac{n(n-1)}{2}$ anulaciones) es $O(n^4)$. Sin embargo, el método clásico necesita de un número de anulaciones pequeño para alcanzar la convergencia, y esto es debido a que en cada paso del algoritmo se anula el elemento de mayor valor absoluto que será el que más contribuya a la convergencia. En esta subsección pretendemos estudiar un método que utilice características de los métodos clásico y cíclicos de manera que se acelere la convergencia (se reduzca el número de barridos que se necesita con los métodos cíclicos) aumentando algo el tiempo de ejecución por barrido. Se trata de encontrar un punto intermedio en el que este aumento en el tiempo de ejecución por barrido se vea compensado por la reducción en el número de barridos, obteniéndose de este modo mejores tiempos de ejecución que con los métodos cíclicos. Ideas similares se encuentran en [11] donde se diseña un método para Multiprocesadores con Memoria Compartida y en [12] para métodos de Jacobi unilaterales.

Una primera idea consiste en ordenar antes de cada barrido los elementos no diagonales de mayor a menor valor absoluto y realizar las anulaciones en este orden, con lo que necesitamos en cada barrido de una ordenación previa de $\frac{n(n-1)}{2}$ datos, lo que haciéndolo con un método óptimo como el Quicksort [13], representa un orden promedio $O(n^2 log_2\ n)$. Como el orden del método de Jacobi en cada barrido es $O(n^3)$ esta ordenación previa no modificará el orden por barrido. Por otra parte, la ordenación previa hará que en cada barrido anulemos primero los elementos de mayor valor absoluto, con lo que es posible que se rebaje el número de barridos necesarios para la convergencia.

Una vez hecha la ordenación las primeras iteraciones se hacen sobre los elementos con mayor valor absoluto, pero al anular estos elementos se modifican los valores iniciales de los elementos no diagonales, con lo que estos elementos ya no están ordenados según el orden obtenido al principio del barrido. Por este motivo, en nuestro método no haremos la ordenación de los datos sino que, siguiendo el esquema del Quicksort, tomamos el primer elemento y obtenemos los mayores y los menores que él en valor absoluto, haciendo una nueva llamada a este procedimiento de "semi-ordenación" con los elementos mayores, y así recursivamente hasta que hagamos una llamada con un único elemento. Esta "semi-ordenación" tiene un coste promedio de $O(n^2)$. Anulando en cada barrido los elementos en el orden obtenido con la "semi-ordenación" obtenemos unos tiempos de ejecución que se muestran en la tabla 8 y el número de barridos de la tabla 9 (datos obtenidos en un procesador i860).

tamaño	256	320	384	448	512
cíclico por filas	134.80	279.87	493.09	825.54	1230.12
semiclásico	129.05	267.69	465.66	777.96	1167.05

Tabla 8: Método semiclásico. Tiempos de ejecución en segundos en un procesador i860.

tamaño	256	320	384	448	512
cíclico por filas	10	10	10	10	10
semiclásico	9	9	9	9	9

Tabla 9: Método semiclásico. Número de barridos en un procesador i860.

En estas dos tablas se observa que utilizando el método semiclásico que hemos explicado se obtiene una pequeña reducción en el tiempo de ejecución, y que esta reducción es debida a que se necesita de un barrido menos para alcanzar la convergencia. Sin embargo, con este método no se mejoran los tiempos obtenidos con el uso de umbrales (tabla 3).

Intentaremos modificar este método para obtener mejores resultados. Al anularse en las últimas etapas de cada barrido términos que inicialmente son los de menor valor absoluto, estas anulaciones deben contribuir poco a la convergencia del método, por lo que consideramos preferible hacer en cada barrido $\frac{n(n-1)}{2DIVISION}$ anulaciones, indicando $\frac{1}{DIVISION}$ la porción de elementos no diagonales que anulamos en cada barrido (llamaremos a cada uno de estos barridos un **subbarrido**). Cuando $DIVISION = 1$ obtenemos el método anterior y cuando $DIVISION = \frac{n(n-1)}{2}$ el método es el de Jacobi clásico donde hemos obtenido el máximo haciendo una "semi-ordenación".

Compararemos experimentalmente en distintos casos (distintas máquinas, uso de umbral y de BLAS 1) los resultados obtenidos con este segundo método semiclásico y distintos valores de $DIVISION$ con los obtenidos con el método cíclico por filas. Lo que pretendemos es comprobar en qué casos se obtiene una mejora sobre los métodos hasta ahora presentados, y si se puede determinar de una manera empírica un valor óptimo de $DIVISION$.

2.2.2 Resultados experimentales sin uso de umbral

En la tabla 10 se compara la velocidad de convergencia del método cíclico por filas y el semiclásico para distintos valores de $DIVISION$. Se representa el número de anulaciones realizadas partido por el número de elementos que hay por encima de la diagonal, coincidiendo este valor con el número de barridos en el método cíclico por filas. Los resultados se han obtenido en un procesador i860, pero serían similares en otras máquinas. Se observa que el número de anulaciones disminuye al aumentar $DIVISION$ y que tiende a estabilizarse entre 5 y 6 veces $\frac{n(n-1)}{2}$ según el tamaño de la matriz.

tamaño	256	320	384	448	512
cíclico por filas	10	10	10	10	10
semiclásico 2	7.50	8.00	9.50	8.00	9.00
semiclásico 4	6.75	8.00	9.50	8.00	9.00
semiclásico 6	6.17	6.50	6.67	6.50	6.50
semiclásico 8	5.88	6.25	6.38	6.25	6.25
semiclásico 10	5.60	5.90	6.00	6.00	6.10
semiclásico 12	5.42	5.58	5.67	5.67	5.83
semiclásico 14	5.43	5.36	5.57	5.64	5.78

Tabla 10: Comparación de la velocidad de convergencia del método cíclico por filas y el método semiclásico con distintos valores de $DIVISION$. Número de anulaciones partido por número de elementos por encima de la diagonal. En un procesador i860.

Como se puede comprobar, se llega a obtener un número de barridos ligeramente menor que el que se obtenía usando el umbral de Kahan y Corneil (tabla 4). Pero con el método semiclásico se divide cada barrido en un número de subbarridos igual a $DIVISION$ realizándose en cada subbarrido una semiordenación, con lo que el coste por barrido será $O(n^3) + DIVISION\ O(n^2)$. De este modo, se obtiene que una reducción en el número de barridos no implicará siempre una reducción en el tiempo de ejecución, sino que dependerá del valor de $DIVISION$ y del coste de las diferentes operaciones (aritméticas, de comparación y de acceso a memoria) en la máquina en que se ejecuten los programas.

Mostraremos los resultados obtenidos experimentalmente en diferentes casos.

En la tabla 11 se muestran los tiempos obtenidos en una HP Apollo 700, y en la tabla 12 los obtenidos en un procesador i860. En ambas tablas se marcan los mejores tiempos obtenidos. Se observa que el valor óptimo de $DIVISION$ aumenta con el tamaño de la matriz y es distinto para distintos sistemas.

tamaño	256	320	384	448	512
cíclico por filas	87	151	319	506	1087
semiclásico 2	76	131	307	507	937
semiclásico 4	64	123	256	438	791
semiclásico 6	61	121	253	412	750
semiclásico 8	**59**	118	235	401	736
semiclásico 10	**59**	**116**	236	398	715
semiclásico 12	61	122	**234**	395	699
semiclásico 14	62	125	236	392	701
semiclásico 16	64	122	237	**391**	**695**

Tabla 11: Método semiclásico con distintos valores de $DIVISION$. Tiempo de ejecución en segundos en una HP Apollo 700.

Se puede ver, comparando los valores óptimos obtenidos en la tabla 3 y los obtenidos en la tabla 12 que este método mejora en tiempo de ejecución muy ligeramente los resultados obtenidos con un método cíclico usando umbral. Intentaremos, por tanto, combinar el uso de umbral con el método semiclásico.

tamaño	256	320	384	448	512
cíclico por filas	134.80	279.87	493.09	825.54	1230.12
semiclásico 2	114.30	245.80	491.58	711.74	1158.88
semiclásico 4	109.87	221.75	393.70	649.02	987.96
semiclásico 6	106.86	223.42	393.15	629.75	927.83
semiclásico 8	**106.29**	226.44	393.96	623.41	919.83
semiclásico 10	107.95	220.81	379.42	616.63	927.87
semiclásico 12	108.96	216.58	**375.63**	**597.92**	**902.46**
semiclásico 14	113.82	**216.31**	376.57	611.24	917.39

Tabla 12: Método semiclásico con distintos valores de $DIVISION$. Tiempo de ejecución en segundos en un procesador i860.

2.2.3 Resultados experimentales usando umbral

Como ya hemos visto, las mejoras relativas con el método semiclásico pueden depender de la máquina en que se ejecuten los programas. Además, hemos visto que usando el método semiclásico se puede reducir en algunos casos ligeramente el tiempo de ejecución que se obtenía usando umbral, por lo que puede ser conveniente combinar las dos técnicas. Aquí estudiaremos la combinación de estas dos técnicas presentando los resultados obtenidos en un procesador i860 y en una HP Apollo 700. En el procesador i860 los mejores resultados se obtienen con el método semiclásico y umbral fijo, y en la HP Apollo 700 los mejores resultados se obtienen con el método cíclico usando el umbral variable de Kahan y Corneil y con el método semiclásico con el mismo umbral variable y con $DIVISION = 1$.

En algunos casos es preferible no usar el mejor umbral junto con el semiclásico porque el uso de umbral hace que en cada barrido o subbarrido anulemos menos elementos que si no usáramos umbral, lo que produciría con el método semiclásico un mayor número de subbarridos y por tanto una mayor cantidad de semiordenaciones, empeorando así el tiempo de ejecución mientras se reduce el número de anulaciones.

Esto se muestra en las tablas 13, donde se muestran los resultados en la HP Apollo 700, 14, donde se muestran los de i860 sin usar BLAS 1, y 15, donde se muestran los de i860 usando BLAS 1.

tamaño	256	320	384	448	512
cíclico con umbral	59	107	216	348	696
semiclásico 1, umbral variable	58	109	218	368	670
semiclásico 2, umbral fijo	67	135	272	478	926
semiclásico 4, umbral fijo	63	119	249	411	837
semiclásico 6, umbral fijo	**61**	116	238	407	749
semiclásico 8, umbral fijo	**61**	**115**	238	406	712
semiclásico 10, umbral fijo	62	117	234	393	707
semiclásico 12, umbral fijo	62	121	**233**	398	705
semiclásico 14, umbral fijo	64	119	237	395	**689**
semiclásico 16, umbral fijo	65	123	240	**390**	691

Tabla 13: Método semiclásico con umbral. Tiempo de ejecución en segundos en una HP Apollo 700.

tamaño	256	320	384	448	512
cíclico con umbral	114.39	224.88	383.82	618.49	907.56
sc 2, umbral fijo	110.65	237.35	419.25	738.36	1099.50
sc 4, umbral fijo	103.01	212.83	381.59	616.19	997.19
sc 6, umbral fijo	98.53	202.06	359.24	613.45	900.06
sc 8, umbral fijo	**95.51**	196.68	354.67	592.14	857.55
sc 10, umbral fijo	96.63	197.48	345.17	**571.00**	851.49
sc 12, umbral fijo	95.71	200.86	340.74	583.95	850.12
sc 14, umbral fijo	97.76	**193.89**	**339.12**	572.76	**826.20**
sc 1, umbral variable	131	246	406	659	956
sc 2, umbral variable	128	242	417	643	928
sc 3, umbral variable	**123**	**236**	**395**	**629**	**921**
sc 4, umbral variable	127	244	414	640	932

Tabla 14: Método semiclásico con umbral. Tiempo de ejecución en segundos en un procesador i860 sin BLAS 1.

tamaño	256	320	384	448	512
cíclico con umbral	80.09	153.77	260.44	424.38	615.70
sc 4, umbral fijo	68.18	141.77	245.07	424.23	620.59
sc 6, umbral fijo	**63.64**	134.09	232.38	394.31	620.91
sc 8, umbral fijo	64.30	137.46	233.60	402.80	570.84
sc 10, umbral fijo	63.79	131.94	**228.49**	395.24	567.76
sc 12, umbral fijo	65.31	**130.22**	231.15	397.25	578.66
sc 14, umbral fijo	67.66	132.13	237.56	393.43	**556.30**
sc 16, umbral fijo	68.21	137.15	235.98	**382.06**	566.23
sc 1, umbral variable	96.29	180.76	295.52	476.59	685.41
sc 2, umbral variable	93.76	176.51	301.01	463.41	660.60
sc 3, umbral variable	**88.98**	**171.21**	**291.66**	**456.87**	**655.93**

Tabla 15: Método semiclásico con umbral. Tiempo de ejecución en segundos en un procesador i860 con BLAS 1.

En la tabla 16 se muestran los cocientes entre los valores óptimos mostrados en las tablas 13 a 15 y el tiempo de ejecución con el método cíclico por filas correspondiente. Se puede observar que los resultados que se obtienen son muy diferentes en las dos máquinas en que se ha experimentado. Esto es debido a la diferencia en el coste de las operaciones aritméticas, de comparación, de acceso a memoria, etc. Lo que se puede deducir es que en algunos casos el uso de un método semiclásico en combinación con una estrategia de umbralización adecuada puede dar lugar a algoritmos que reducen el tiempo de ejecución con respecto a los que sólo usan umbral. Así, los resultados obtenidos en la HP son muy parecidos usando umbral o usando el método semiclásico en combinación con umbral variable, y lo más adecuado parece ser no usar un método semiclásico. Sin embargo, en un procesador i860 lo más conveniente parece ser usar el método semiclásico en combinación con un umbral fijo, tanto si se usa como si no se usa BLAS 1.

tamaño	256	320	384	448	512
HP: umbral/cíclico	0.67	**0.70**	**0.67**	**0.68**	0.64
HP: sc umb fijo/cíclico	0.70	0.76	0.73	0.77	0.63
HP: sc umb var/cíclico	**0.66**	0.72	0.68	0.72	**0.61**
i860 sin BLAS: umbral/cíclico	0.95	0.85	0.82	0.71	0.78
i860 sin BLAS: sc umb fijo/cíclico	**0.71**	**0.69**	**0.69**	**0.69**	**0.67**
i860 sin BLAS: sc umb var/cíclico	0.91	0.84	0.80	0.76	0.75
i860 con BLAS: umbral/cíclico	0.95		0.82		0.78
i860 con BLAS: sc umb fijo/cíclico	**0.76**	**0.72**	**0.72**	**0.64**	**0.70**
i860 con BLAS: sc umb var/cíclico	1.06	0.95	0.92	0.76	0.83

Tabla 16: Ganancia con los distintos métodos respecto al cíclico por filas.

2.2.4 Algunas variaciones

Se han realizado algunos otros experimentos intentando acelerar la convergencia del método:

1. Ordenando los elementos que se van a anular (en vez de semiordenando) se consigue una mayor reducción en el número de anulaciones necesarias para alcanzar la convergencia, pero esta reducción no compensa (al menos en los experimentos realizados) el tiempo de ejecución adicional necesario para realizar la ordenación, con lo que los tiempos de ejecución que se obtienen son peores ordenando.

2. Utilizando el método propuesto en [11] se reduce aún más el número de anulaciones necesarias para alcanzar la convergencia, pero a costa de un mayor tiempo de ejecución en cada subbarrido, con lo que se obtienen peores tiempos de ejecución. Este método consiste en tomar el máximo de cada columna en valor absoluto, ordenar estos valores máximos y aplicar las rotaciones en este orden, no anulando elementos en una fila o columna donde ya se ha anulado previamente otro elemento (el método está pensado para Multiprocesadores con Memoria Compartida, por lo que se toma una secuencia de pares de índices que permita aplicar las rotaciones en paralelo). Esta manera de proceder hace que cada

barrido se divida en al menos n subbarridos, y el trabajo adicional en cada subbarrido tiene un coste de $\frac{n^2}{2} + n\ log_2\ n$, con lo que el coste por barrido que hay que sumar a los $3n^3$ flops es del orden $O(n^3)$. Combinando este método con el método semiclásico se obtiene una pequeña reducción en el tiempo de ejecución en algunos casos.

Comparamos estas nuevas variaciones con los métodos estudiados anteriormente. En la tabla 17 se muestra el número de anulaciones partido por el número de elementos por encima de la diagonal, utilizando el valor de $DIVISION$ óptimo en un i860. Se comparan los métodos: por filas (filas), con umbral variable de Kahan-Corneil (kc), semiclásico con umbral fijo (sem. umb. fijo), semiclásico con umbral variable de Kahan-Corneil (sem. umb. var.), semiclásico ordenando los elementos a anular y con umbral fijo (ord. umb. fijo), semiclásico ordenando los elementos a anular y con umbral variable de Kahan-Corneil (ord. umb. var.), y semiclásico combinado con la idea de Karp-Greenstadt (sem. k-g). Se observa que con todos los métodos de aceleración de la convergencia se reduce sustancialmente el número de anulaciones habiendo pequeñas diferencias entre ellos: el uso del umbral variable, la ordenación de los elementos a anular, o la combinación con el método de Karp-Greenstadt, producen una mayor reducción en el número de anulaciones. Esta reducción no siempre implica una reducción en el tiempo de ejecución. En la figura 1 mostramos el cociente entre los tiempos de ejecución de los distintos métodos semiclásicos analizados respecto al que usa el umbral de Kahan-Corneil, en una HP Apollo 700, y en la figura 2 mostramos los mismos cocientes en un i860. Se observa que los resultados son muy diferentes en las dos máquinas:

- En la HP el comportamiento es muy irregular, pero el umbral de Kahan-Corneil no es superado por ningún método semiclásico, aunque los resultados obtenidos con el método semiclásico usando umbral variable son similares a los obtenidos usando únicamente umbral.

- En i860 algunos métodos semiclásicos mejoran el uso de umbrales: los que usan umbral fijo y el que usa la idea de Karp-Greenstadt, siendo el método con el que se obtienen mejores resultados el semiclásico con Karp-Greenstadt.

tamaño	256	320	384	448	512
filas	10	10	10	10	10
kc	5.93	6.03	6.08	6.10	6.14
sem. umb. fijo	5.88	5.36	5.67	5.67	5.83
sem. umb. var.	5.18	5.32	5.44	5.31	5.40
ord. umb. fijo	5.57	5.48	5.57	5.66	5.42
ord. umb. var.	5.09	5.16	5.20	5.25	5.28
sem. k-g	5.38	5.30	5.40	5.35	5.40

Tabla 17: Número de anulaciones partido por número de elementos por encima de la diagonal, utilizando el valor de $DIVISION$ óptimo en un i860.

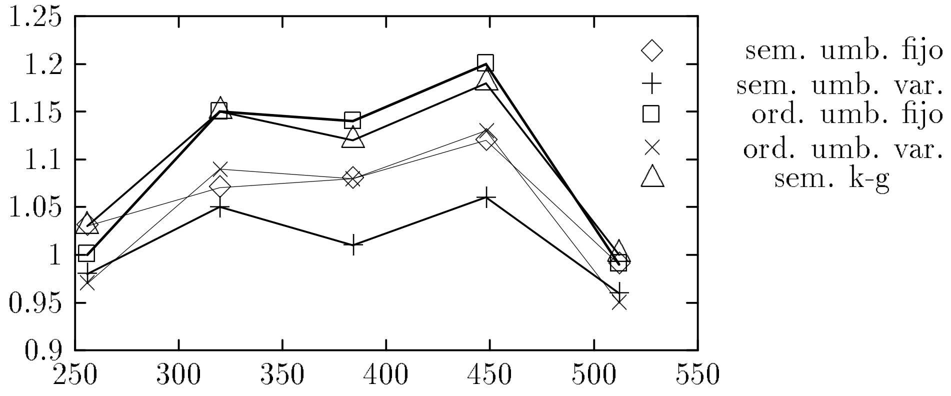

Figura 1: Cociente de los tiempos de ejecución de los distintos métodos respecto al de umbral de Kahan-Corneil. En HP Apollo 700.

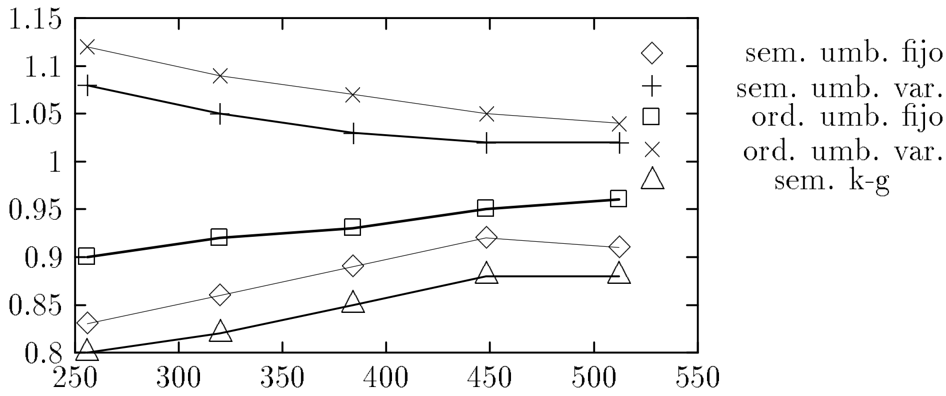

Figura 2: Cociente de los tiempos de ejecución de los distintos métodos respecto al de umbral de Kahan-Corneil. En i860.

2.2.5 Problemas con distintas condiciones

Hasta ahora hemos realizado experimentos generando los elementos de la matriz de manera uniforme, pero puede ser interesante estudiar también el comportamiento de problemas con otro tipo de condiciones. En particular, nos interesará conocer el comportamiento (en cuanto a número de anulaciones y barridos necesarios para la convergencia) de los algoritmos cuando los valores propios de la matriz se concentran alrededor de uno o varios valores.

Generación de los problemas Para generar los problemas con las condiciones mencionadas hemos seguido los siguientes pasos:

1. Se genera una matriz diagonal D con valores propios entre -1 y 1 distribuidos uniformemente.

2. Se elevan esos valores propios a un *exponente*, con lo que se obtiene, a mayor *exponente*, valores propios más cercanos a 0. Los valores usados de *exponente* han sido 1, 3, 5, 7 y 9.

3. Opcionalmente, se suman a distintas partes de la diagonal de D distintos valores v_i, con lo que en las partes donde se haya sumado v_i se tendrán valores cercanos a v_i y más cercanos a mayor valor de *exponente*. Se han realizado experimentos con matrices con valores propios cercanos a 1 y -1, a 1, 2, -1 y -2, y a 1, 2, 4, 8, -1, -2, -4 y -8.

4. Por último, y para obtener una matriz no diagonal con los mismos valores propios que D, se genera un vector unitario u y se realiza la actualización de rango dos de D utilizando u:

$$D' = \left(I - 2uu^t\right) D \left(I - 2uu^t\right) \tag{8}$$

A continuación mostraremos algunos de los resultados obtenidos con las matrices así generadas.

En las tablas se mostrarán resultados obtenidos con distintos métodos: FIL es el método de Jacobi con ordenación cíclica por filas y sin usar umbral, SCd será el método semiclásico con valor de $DIVISION$ indicado por el número d, KC será el método cíclico por filas cuando se usa el umbral variable de Kahan y Corneil, SCUFd será el método semiclásico con umbral fijo indicando d el valor de $DIVISION$, y SCKCd será el método semiclásico con umbral de Kahan-Corneil indicando d el valor de $DIVISION$.

Debido a que, como ya hemos visto, los tiempos de ejecución de las distintas versiones del método semiclásico dependen en gran medida de las características de la máquina en que se esté trabajando, no estudiaremos los tiempos de ejecución, sino el número de anulaciones y barridos y la precisión.

Todos los experimentos se han realizado en una HP Apollo 700 pero, al no obtener tiempos de ejecución, los resultados serían similares en otras máquinas.

Número de anulaciones En la tabla 18 se muestra el número de anulaciones, dividido por el número de elementos no diagonales, necesarias para alcanzar la convergencia con distintos métodos y con distintos tamaños de matriz (con FIL coincidirá con el número de barridos) cuando la matriz A tiene todos sus valores propios cercanos a 0. Se utilizan valores de *exponente* 1, 3, 5 y 7 (al aumentar el valor de *exponente* se obtienen valores propios más próximos).

El valor de $DIVISION$ que se utiliza en los métodos semiclásicos es 12, que es un valor intermedio que nos permite considerar que el número de anulaciones con el valor de $DIVISION$ óptimo estará alrededor del que aquí se muestra.

Se puede observar que al estar los valores propios más cerca aumenta el número de anulaciones con todos los métodos, siendo este aumento considerable en el caso de FIL, y bastante más pequeño en los demás casos, necesitando los métodos semiclásicos con umbral de menos anulaciones que el método cíclico que utiliza el umbral de Kahan-Corneil. Además, el método que menos anulaciones necesita es el semiclásico con umbral de Kahan-Corneil. Aunque de esto no se pueda concluir que con los métodos semiclásicos se mejore el uso de umbral, sí se deduce que cuando los valores propios son cercanos tenemos resultados similares (en cuanto a número de anulaciones) a los que obteníamos generando la matriz A de manera uniforme, por lo que, también en este caso, cuál sea el método más rápido dependerá de la máquina a utilizar.

Resultados similares se obtienen cuando hay varios puntos donde se acumulan los valores propios. En la tabla 19 se muestra el número de anulaciones partido por el número de elementos no diagonales, cuando los valores propios están cercanos a 1, 2, -1 y -2. A mayor valor de $exponente$ los valores propios estarán más cercanos a esos cuatro puntos.

En este caso también aumenta el número de anulaciones al aumentar $exponente$, aunque en menor medida que lo hace en la tabla 18 al estar los valores propios más dispersos. Hay un gran aumento en el número de anulaciones en FIL y un aumento pequeño en los demás métodos. Este aumento es especialmente pequeño en KC, lo que hace que al aumentar la acumulación de los valores propios alrededor de 1, 2, -1 y -2 KC sea preferible a los métodos semiclásicos, mientras que con $exponente = 1$ el menor número de anulaciones se obtiene con el método semiclásico usando el umbral de Kahan-Corneil.

Número de barridos Aunque en los métodos que hasta ahora hemos visto una disminución en el número de anulaciones representa una disminución en el número de flops, no siempre es lo más interesante reducir el número de anulaciones. Esto pasa por ejemplo con los métodos semiclásicos y como veremos en la sección siguiente con un método por bloques. Algunas veces

exponente=1	128	192	256	320
FIL	8	8	9	9
KC	4.21	4.25	4.25	4.22
SCUF12	3.81	3.80	3.81	3.64
SCUV12	3.52	3.43	3.47	3.46

exponente=3	128	192	256	320
FIL	11	13	11	14
KC	4.35	4.45	4.62	4.44
SCUF12	3.90	3.76	3.82	4.05
SCUV12	3.65	3.79	3.67	3.50

exponente=5	128	192	256	320
FIL	13	17	18	16
KC	4.48	4.71	4.70	4.85
SCUF12	4.08	4.11	4.10	4.03
SCUV12	3.70	3.75	3.88	3.83

exponente=7	128	192	256	320
FIL	17	16	19	19
KC	4.93	5.02	5.00	5.45
SCUF12	4.05	4.45	4.32	5.28
SCUV12	3.82	3.76	3.99	3.89

Tabla 18: Número de anulaciones partido por número de elementos no diagonales, con distintos métodos de Jacobi y distintos tamaños de matriz, cuando los valores propios se acumulan alrededor de 0. En una HP Apollo 700.

exponente=1	128	192	256	320
FIL	8	9	9	9
KC	3.79	3.81	3.77	3.77
SCUF12	3.77	3.67	3.80	3.74
SCUV12	3.64	3.55	3.60	3.46
exponente=3	128	192	256	320
FIL	10	10	10	12
KC	3.75	3.88	3.84	3.92
SCUF12	3.87	4.08	4.05	4.24
SCUV12	3.63	3.61	3.77	3.68
exponente=5	128	192	256	320
FIL	11	12	13	12
KC	3.69	3.80	3.96	3.74
SCUF12	4.20	4.13	4.44	4.49
SCUV12	3.80	3.93	3.88	4.00
exponente=7	128	192	256	320
FIL	11	12	13	16
KC	3.86	3.87	4.04	4.09
SCUF12	4.50	4.58	4.81	4.83
SCUV12	4.02	3.95	4.14	4.16

Tabla 19: Número de anulaciones partido por número de elementos no diagonales, con distintos métodos de Jacobi y distintos tamaños de matriz, cuando los valores propios se acumulan alrededor de 1, 2, -1 y -2. En una HP Apollo 700.

puede ser interesante reducir el número de barridos (el número de veces que se trata cada uno de los elementos no diagonales ya sea anulándolo o no).

En la tabla 20 representamos el número de barridos necesarios para alcanzar la convergencia con distintos métodos, distintos tamaños de matriz y distintos valores de *exponente* cuando los valores propios se acumulan alrededor de 0. En los métodos semiclásicos se utiliza $DIVISION = 1$ porque este valor es el que nos da menor número de barridos que es lo que pretendemos reducir.

Se observa que el uso del umbral de Kahan-Corneil, aunque reduce el número de anulaciones, aumenta el número de barridos, mientras que la semiordenación de los elementos para anularlos en ese orden reduce el número de barridos, especialmente si se usa umbral fijo para evitar anular elementos de valor absoluto muy pequeño. Este hecho lo aplicaremos en la sección siguiente para combinar el método semiclásico con un método por bloques de manera que se reduzca en algunos casos el número de barridos y por tanto el tiempo de ejecución.

En la tabla 21 representamos el número de barridos necesarios para alcanzar la convergencia cuando los valores propios se acumulan alrededor de 1, 2, -1 y -2. También en este caso es preferible un método semiclásico, y no está claro si es preferible usar umbral fijo o no usar umbral. En cambio, sí está claro que el uso de umbral variable aumenta el número de barridos.

3 Aceleración de la convergencia en métodos por bloques

Las ideas de la aceleración de la convergencia y del trabajo por bloques son totalmente contrapuestas pues en la aceleración de la convergencia se trabaja elemento a elemento y en el otro caso se reduce el tiempo de ejecución trabajando por bloques de elementos. De cualquier modo, hemos intentado combinar estas dos ideas. El método por bloques que hemos utilizado es el de [4, 5], y la combinación que se ha realizado con las técnicas de aceleración consiste en:

- En el caso de umbrales: al realizar un barrido sobre un bloque se anulan los elementos que están por encima del umbral. Se ha utilizado umbral fijo y el umbral variable de Kahan-Corneil.

exponente=1	128	192	256	320
FIL	8	8	9	9
KC	9	9	9	10
SC1	7	7	8	7
SCUF1	7	7	7	7
SCUV1	11	12	12	12
exponente=3	128	192	256	320
FIL	11	13	11	14
KC	14	15	16	16
SC1	10	12	12	14
SCUF1	8	10	11	11
SCUV1	12	14	12	12
exponente=5	128	192	256	320
FIL	13	17	18	16
KC	13	15	18	16
SC1	11	11	13	13
SCUF1	10	12	12	14
SCUV1	13	16	16	15
exponente=7	128	192	256	320
FIL	17	16	19	19
KC	15	17	17	20
SC1	12	13	16	16
SCUF1	15	14	14	15
SCUV1	15	15	16	18

Tabla 20: Número de barridos, con distintos métodos de Jacobi y distintos tamaños de matriz, cuando los valores propios se acumulan alrededor de 0. En una HP Apollo 700.

exponente=1	128	192	256	320
FIL	8	9	9	9
KC	10	9	10	11
SC1	7	7	7	7
SCUF1	7	7	7	7
SCUV1	12	13	13	13

exponente=3	128	192	256	320
FIL	10	10	10	12
KC	10	11	11	12
SC1	8	9	9	9
SCUF1	7	9	10	10
SCUV1	14	13	13	14

exponente=5	128	192	256	320
FIL	11	12	13	12
KC	11	12	14	13
SC1	10	11	10	11
SCUF1	9	10	11	12
SCUV1	13	13	16	15

exponente=7	128	192	256	320
FIL	11	12	13	16
KC	11	14	14	16
SC1	10	10	11	13
SCUF1	10	11	12	12
SCUV1	14	15	15	16

Tabla 21: Número de barridos, con distintos métodos de Jacobi y distintos tamaños de matriz, cuando los valores propios se acumulan alrededor de 1, 2, -1 y -2. En una HP Apollo 700.

- En la combinación con el semiclásico: se han realizado experimentos realizando sobre cada bloque s subbarridos antes de actualizar la matriz, y anulando una porción $\frac{1}{DIVISION}$ de elementos de cada bloque en cada subbarrido. El aumentar $DIVISION$ acelera la convergencia pero produce unos tiempos adicionales por las semiordenaciones. El aumentar s produce un mayor aprovechamiento del uso de los bloques, pero postpone la anulación de los mayores elementos de otros bloques, ralentizando la convergencia. Esto produce que los mejores resultados se han obtenido con $DIVISION = 1$ y $s = 1$.

En la tabla 22 mostramos los tiempos de ejecución y los barridos combinando distintos métodos de aceleración de la convergencia con el método por bloques. Los métodos con los que se ha experimentado son: el método por bloques (bloques), el método por bloques combinado con umbral fijo (blo-uf), el método por bloques combinado con el umbral variable de Kahan-Corneil (blo-uv), y el método por bloques combinado con un método semiclásico sin umbral y con $DIVISION = 1$ y $s = 1$ (blo-sc). Se observa:

- El uso del umbral de Kahan-Corneil produce un aumento en el número de barridos, con lo que empeora el tiempo de ejecución del método por bloques.

- El uso del método semiclásico produce normalmente una reducción en el número de barridos, lo que puede producir (dependiendo de las características de la máquina, de la reducción en el número de barridos, del coste adicional del método semiclásico utilizado y del tipo de matriz) una pequeña reducción en el tiempo de ejecución. En la tabla 23 mostramos resultados obtenidos en un procesador i860 que confirman esta idea.

Se pueden hacer algunas consideraciones heurísticas sobre el comportamiento del método semiclásico al combinarlo con un esquema por bloques:

- Cuando el tamaño de la matriz aumenta, al mantenerse el tamaño de bloque óptimo entre 16 y 32, el uso de un método semiclásico en los bloques pierde efectividad al ser los bloques más pequeños respecto a la matriz.

	barridos				tiempo			
método:	bloques	blo-uf	blo-uv	blo-sc	bloques	blo-uf	blo-uv	blo-sc
64	9	9	13	**8**	**0.27**	**0.27**	0.38	**0.27**
128	10	10	13	**8**	1.47	1.50	1.81	**1.41**
256	**10**	**10**	14	**10**	**7.78**	7.98	9.84	8.39
512	11	11	14	**10**	50.04	50.96	55.96	**47.42**

Tabla 22: Tiempos de ejecución y barridos combinando distintos métodos de aceleración con un método por bloques, en una Sillicon Graphic Power Challenge.

método	bloques				blo-sc			
bloque	16		32		16		32	
	bar.	tiempo	bar.	tiempo	bar.	tiempo	bar.	tiempo
128	9	**7.70**	10	10.31	9	8.18	9	9.10
256	10	49.06	10	53.59	9	**46.62**	9	50.65
384	10	**144.43**	11	162.20	10	148.35	10	156.51
512	11	346.78	11	344.42	10	**325.77**	10	329.23
640			11	628.29	10	604.41	10	**601.55**
768			11	1012.94	10	999.24	10	**956.28**

Tabla 23: Combinación del método por bloques y el semiclásico. Tiempos en segundos y número de barridos obtenidos en un i860.

- Cuando la matriz es pequeña el coste aritmético es pequeño y el coste de las semiordenaciones adquiere una gran importancia, haciendo que una reducción en el número de barridos no siempre produzca una reducción en el tiempo de ejecución.

- Cuando aumenta el coste de computación (por ejemplo cuando se calculan los valores y vectores propios) tiende a ser más interesante el uso de métodos semiclásicos por dos motivos: aumenta el tamaño de bloque óptimo, y el coste de las semiordenaciones pierde importancia.

- En los casos en que aumenta el número de barridos (cuando los valores propios están cercanos) puede ser interesante utilizar métodos semiclásicos. En la tabla 24 se muestra la reducción en el número de barridos con matrices de distintas características: con valores propios cercanos a 0, 10 ó 1, -1, 2 y -2, estando los valores propios próximos a estos valores (se usa un exponente 3 en la generación de las matrices tal como explicamos) o muy próximos (exponente 7). Se observa que normalmente hay una pequeña reducción en el número de barridos, y que para algunas matrices especiales esta reducción llega a ser importante.

	128	192	256	320	384	512
próximos a 0	1	2	1	0	1	1
muy próximos a 0	1	1	1	1	0	0
próximos a 10	1	1	0	0	0	0
muy próximos a 10	4	3	3	2	0	0
próximos a 1, -1, 2, -2	3	2	4	2	1	1
muy próximos a 1, -1, 2, -2	2	3	2	1	2	3

Tabla 24: Reducción en el número de barridos cuando se combina el método semiclásico con el algoritmo por bloques. con valores propios cercanos a 0, a 10 ó a 1, -1, 2 y -2 y mayor o menor proximidad.

4 CONCLUSIONES

Los métodos semiclásicos son una buena alternativa a los métodos de umbralización para acelerar la convergencia en métodos de Jacobi en la solución del Problema Simétrico de Valores Propios.

En el caso de métodos de Jacobi que no trabajan por bloques con los métodos semiclásicos se consigue normalmente una mayor aceleración de la convergencia que usando umbrales, aunque la reducción en el número de anulaciones para alcanzar la convergencia no siempre ocasiona una reducción en el tiempo de ejecución, debido al coste adicional de las semiordenaciones en los métodos semiclásicos. En los experimentos realizados en un procesador i860 el mejor método semiclásico es el que se obtiene combinándolo con la idea de Karp y Greenstadt, y en este caso se llegan a tener unas reducciones en el tiempo de ejecución de entre un 20% y un 15%.

En el caso de métodos de Jacobi por bloques el uso de umbrales, aunque disminuye el número de anulaciones, produce un mayor número de barridos y por tanto un mayor tiempo de ejecución. Sin embargo, la combinación de un método semiclásico con el trabajo por bloques produce normalmente una reducción en el número de barridos y puede producir una reducción en el tiempo de ejecución.

Referencias

[1] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, 1980.

[2] Maria T. Cámara y Domingo Giménez. On the Semiclassical Jacobi Algorithm. In John G. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, pages 285–289. SIAM, 1994.

[3] Gautam Schroff y Robert Schreiber. On the convergence of the cyclic Jacobi method for parallel block orderings. *SIAM J. Matrix Anal. Appl.*, 10(3):326–346, 1989.

[4] Domingo Giménez, Vicente Hernández, Robert A. van de Geijn y Antonio M. Vidal. A Jacobi method by blocks to solve the symmetric eigenvalue problem. Technical Report TR 1-95, Dept de Informática y Sistemas. Universidad de Murcia, 1995.

[5] Javier Cuenca, Domingo Giménez y Javier Zapata. Uso de núcleos computacionales en el método de Jacobi para el cálculo de valores propios en multiprocesadores. IV Congreso Español de Matemática Aplicada. 1995.

[6] G. E. Forsythe y P. Henrici. The cyclic Jacobi method for computing the principal values of a complex matrix. *Trans. Amer. Math. Soc.*, 94:1–23, 1960.

[7] G. H. Golub y C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989. Segunda Edición.

[8] P. J. Eberlein y Haesun Park. Efficient implementation of Jacobi algorithms and Jacobi sets on distributed memory architectures. *Journal of Parallel and Distributed Computing*, 8:358–366, 1990.

[9] Richard P. Brent y Franklin T. Luk. A systolic architecture for almost linear-time solution of the symmetric eigenvalue problem. Technical Report TR-CS-82-10, Department of Computer Science, Australian National University, Camberra, August 1982.

[10] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, 1965.

[11] Alan H. Karp y John Greenstadt. A improved parallel Jacobi method for diagonalizing a symmetric matrix. *Parallel Computing*, 5:281–294, 1987.

[12] P. P. M. Rijk. A one-sided Jacobi algorithm for computing the singular value decomposition on a vector computer. *SIAM J. Sci. Stat. Comput.*, 10(2):359–371, 1989.

[13] D. E. Knuth. *El arte de programar ordenadores. Vol 3: clasificación y búsqueda*. Reverté, 1987.