Estudio experimental de métodos de Jacobi bilaterales en sistemas Intel*†

Domingo Giménez Cánovas ‡

1 Introducción

En este trabajo se realiza un estudio experimental de diferentes métodos de Jacobi bilaterales en diversos sistemas Intel. El trabajo es una continuación del realizado en [1].

Hay multitud de métodos para resolver el problema del que nos ocupamos ([2, 3, 4, 5, 6, 7, 8, 9]) y de la mayoría de ellos existen versiones para Multiprocesadores con Memoria Compartida y Distribuida.

Para la resolución del Problema Simétrico de Valores Propios, normalmente se suele reducir la matriz mediante transformaciones de Householder o de Givens a forma tridiagonal. Posteriormente se puede diagonalizar la matriz tridiagonal por medio de algún método como la iteración QR, bisección e iteración inversa, o el algoritmo divide y vencerás de Cuppen [10, 11]. Estos métodos han sido tradicionalmente los más usados por ser los de mayor eficiencia, en monoprocesadores, de los conocidos, y se han diseñado multitud de algoritmos para Multiprocesadores [12, 13, 14, 15, 16, 17, 18, 19].

Los métodos de Jacobi [20, 21] trabajan reduciendo la matriz a forma diagonal, con lo que los valores propios son los elementos diagonales de esta matriz. Esta reducción se lleva a cabo sin realizar operaciones previas de tridiagonalización o paso a otras formas condensadas intermedias, lo que produce un mayor coste computacional que los métodos que realizan una reducción de la matriz a forma tridiagonal.

A pesar del mayor coste computacional de los métodos tipo Jacobi, estos han vuelto a adquirir importancia en los últimos tiempos debido a su mayor paralelismo implícito [22] y a su mejor estabilidad numérica [23, 24]. Este mayor paralelismo de los métodos tipo Jacobi hace que sea interesante su estudio en Multiprocesadores Masivamente Paralelos (MPP), ya que en estos sistemas podrían llegar a ser competitivos con los que reducen la matriz a forma tridiagonal.

*Parcialmente soportado por la Comisión Interministerial de Ciencia y Tecnología, acción TIC96-1062-C03-02

†Los equipos utilizados han sido: un iPSC/2 (con 4 procesadores) del Centro de Estudios Avanzados de Blanes, del C.S.I.C.; un iPSC/860 (con 8 procesadores) del Argonne National Laboratory y otro (con 32 procesadores) del Center for High Performance Computing de la University of Texas; el Touchstone DELTA (con 512 procesadores) del Californian Institute of Technology; y un Paragon (con 44 procesadores) del Center for High Performance Computing de la University of Texas.

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

De este modo, en este trabajo se continúa el estudio de [1] sobre métodos de Jacobi en MPP, estudiando las prestaciones de distintos métodos en sistemas Intel (iPSC/860, Touchstone DELTA y Paragon) que constituyen un ejemplo típico de MPP.

2 Sistemas y algoritmos

2.1 Comunicaciones en diversos sistemas Intel

Se han realizado experimentos con los diferentes algoritmos estudiados en [1] en diferentes sistemas Intel:

- Un iPSC/2 con 4 procesadores [25].
- Un iPSC/860 con 32 procesadores [26].
- Un Touchstone DELTA con 256 procesadores [27].
- Un Paragon con 44 procesadores [28].

El iPSC/2 y el iPSC/860 tienen topología de hipercubo y el Touchstone DELTA y el Paragon de malla. Cada nodo del iPSC/2 contiene un procesador 80386 y un DCM (Direct-Connect Module) para llevar a cabo las comunicaciones. En el resto de los sistemas cada nodo contiene un procesador i860 y otro para las comunicaciones, que en el caso del iPSC/860 es también un DCM; en el Touchstone DELTA este procesador se llama MRC (Mesh Routing Chip), y en el Paragon MCP (Message Passing Co-processor).

Debido al uso de un coprocesador en cada nodo para gestionar las comunicaciones, el sistema se puede ver como completamente conectado, aunque mensajes entre procesadores vecinos deben producir un menor tiempo de ejecución.

Hay una serie de artículos donde se estudian las prestaciones de estos sistemas [29, 30, 31, 32]. Aquí resumiremos las prestaciones con la intención de poder explicar el comportamiento de los algoritmos en los distintos sistemas en base a una serie de parámetros dependientes del sistema. En los artículos mencionados se hacen estudios detallados de los tiempos de inicio de una comunicación (β) y de envío de un dato (τ) (y en algunos casos también del tiempo aritmético) en diferentes situaciones. En la tabla 1 mostramos unos valores aproximados cuando la comunicación se realiza entre procesadores vecinos. Utilizaremos estos valores para contrastar el tiempo de ejecución teórico con el experimental. Para esta comparación necesitamos también una estimación del coste de una operación aritmética en doble precisión, pues este es el tipo de datos con el que trabajamos. Pero en los algoritmos algunas veces se usa BLAS 1 o BLAS 3, y el coste utilizando rutinas de estos paquetes varía grandemente con el tamaño de los vectores y matrices con los que se opera, y no podemos considerar que al aumentar el tamaño del problema aumente el tamaño de los vectores (matrices) que intervienen en las operaciones, pues durante la ejecución del programa habrá trabajo sobre vectores (matrices) de tamaños muy variables. Por esto, tal como se hace en [33] se utilizará una aproximación al coste de una operación aritmética con el tipo de rutinas que se utilizan y dentro del rango de tamaños con el que se trabaja.

	iPSC/2	iPSC/860	DELTA	Paragon
β (μs)	290	76	61	172
τ (μs)	2.86	2.86	1.20	0.34

Tabla 1: Valores asintóticos del tiempo de inicio de una comunicación (β) y de envío de un dato en doble precisión (τ) en diferentes sistemas, cuando la comunicación es entre procesadores vecinos.

2.2 Métodos de Jacobi analizados

Estudiaremos las prestaciones de distintos algoritmos en diferentes sistemas. Los nombres que usaremos para referirnos a los algoritmos y una breve descripción de sus características es:

- SeSiFi: método Secuencial que explota la Simetría utilizando la ordenación por Filas.
- SeSiFiB1: método Secuencial que explota la Simetría utilizando la ordenación por Filas y BLAS 1 para actualizar la matriz.
- SeSiB3: método Secuencial que explota la Simetría utilizando BLAS 3. Es un método por bloques en el que se genera el orden en que se tratan los bloques con una ordenación par-impar, y el trabajo sobre cada bloque se realiza utilizando BLAS 1 y la ordenación por filas.
- PaNsAnCoPi: método Paralelo que No explota la simetría, trabajando en un Anillo de procesadores, con una distribución de los datos por bloques contiguos de Columnas, y con la ordenación Par-impar.
- PaNsMaCuRr: método Paralelo que No explota la simetría, trabajando en una Malla de procesadores, con distribución de los datos por bloques Cuadrados, y usando la ordenación Round-robin.
- PaNsMaCuRrB1: método Paralelo que No explota la simetría, trabajando en una Malla de procesadores, con distribución de los datos por bloques Cuadrados, usando la ordenación Round-robin, y utilizando BLAS 1 en la actualización de la matriz.
- PaSiAnMaEb: método Paralelo que explota la Simetría, en un Anillo de procesadores, con una distribución de los datos por Marcos, y con la ordenación de Eberlein.
- PaSiAnMaEbB1: método Paralelo que explota la Simetría, en un Anillo de procesadores, con una distribución de los datos por Marcos, con la ordenación de Eberlein, y utilizando BLAS 1 en la actualización de la matriz.
- PaSiMaPlRr: método Paralelo que explota la Simetría, en una Malla de procesadores, con esquema de almacenamiento por Plegamiento, y la ordenación Round-robin.

- PaSiMaPlRrB1: método Paralelo que explota la Simetría, en una Malla de procesadores, con esquema de almacenamiento por Plegamiento, la ordenación Round-robin, y utilizando BLAS 1 en la actualización de la matriz.
- PaSiMaPlB3: método Paralelo que explota la Simetría, en una Malla de procesadores, con esquema de almacenamiento por Plegamiento, la ordenación Round-robin, y con un esquema por bloques con el que se usa BLAS 3 en la actualización de la matriz. El orden en que se tratan los bloques se obtiene con una ordenación par-impar, y el trabajo sobre cada bloque se realiza utilizando BLAS 1 y la ordenación por filas.

En la tabla 2 se muestran los tiempos de ejecución por barrido (aritmético, de inicio de comunicaciones y de envío de datos) de los diferentes algoritmos [1]. En esta tabla, una constante k_1 indica que las operaciones se llevan a cabo usando BLAS 1, una constante k_3 que se llevan a cabo usando BLAS 3, n representa la dimensión de la matriz, y p el número de procesadores.

	aritmético	inicio de comunicación	envío de datos
SeSiFi	$3n^3$		
SeSiFiB1	$k_1 3n^3$		
SeSiB3	$k_3\left(4n^3 - 12n^2t + 8nt^2\right) + k_1 12n^2t$		
PaNsAnCoPi	$\frac{6n^3}{p} + \frac{13n^2}{2p}$	$2np$	$4n^2$
PaNsMaCuRr	$\frac{6n^3}{p} + \left(\frac{11}{2\sqrt{p}} - \frac{4}{p}\right) n^2$	$\left(\sqrt{p} + 8\right) n$	$\left(1 + \frac{8}{\sqrt{p}}\right) n^2$
PaNsMaCuRrB1	$k_1 \frac{6n^3}{p} + \left(\frac{11}{2\sqrt{p}} - \frac{4}{p}\right) n^2$	$\left(\sqrt{p} + 8\right) n$	$\left(1 + \frac{8}{\sqrt{p}}\right) n^2$
PaSiAnMaEb	$\frac{3n^3}{p} + \frac{12n^2}{p}$	$2np$	$\left(6 - \frac{5}{2p}\right) n^2$
PaSiAnMaEbB1	$k_1 \frac{3n^3}{p} + \frac{13n^2}{2p}$	$2np$	$\left(6 - \frac{5}{2p}\right) n^2$
PaSiMaPlRr	$\frac{3n^3}{p} + \left(\frac{23}{2\sqrt{p}} - \frac{2}{p}\right) n^2$	$\left(\sqrt{p} + 12\right) n$	$\left(1 + \frac{8}{\sqrt{p}}\right) n^2$
PaSiMaPlRrB1	$k_1 \frac{3n^3}{p} + \left(1 - 3k_1\right) n^2 + \left(k_1 6 + \frac{11}{2}\right) \frac{n^2}{\sqrt{p}}$	$\left(\sqrt{p} + 12\right) n$	$\left(1 + \frac{8}{\sqrt{p}}\right) n^2$
PaSiMaPlB3	$\frac{4k_3 n^3}{p} + \left(2k_3 + 12k_1\right) \frac{n^2 t}{\sqrt{p}} + 12k_1 \frac{nt^2}{\sqrt{p}}$	$\left(4 + \sqrt{p}\right) \frac{n}{t}$	$\left(\frac{3}{\sqrt{p}} + 2\right) n^2$

Tabla 2: Costes teóricos de diferentes algoritmos.

2.3 Prestaciones de los diferentes sistemas

Para poder comparar los tiempos de ejecución teóricos y experimentales, necesitamos de una estimación del coste de una operación aritmética. Como en los programas se trabaja con datos en doble precisión y en el estudio teórico se cuenta el número de operaciones en coma flotante (ya sean sumas, restas, multiplicaciones o divisiones), haremos una estimación del tiempo de ejecución de una operación en coma flotante. También habrá que estimar los valores de k_1 y k_3.

Para estas estimaciones podríamos utilizar valores de los que se muestran en los artículos antes mencionados y en [34]. En la tabla 3 se muestran los Mflops que se obtienen según los experimentos de [29] (iPSC/2) y [30] (iPSC/860), y de [34], este último utilizando la resolución de un sistema de ecuaciones con $n = 100$ y $n = 1000$. También se muestran los Mflops pico alcanzables de iPSC/860, Touchstone DELTA y Paragon.

	iPSC/2	iPSC/860	DELTA	Paragon
[29] y [30]	0.13	10		
[34] $n = 100$	0.17	9.8	9.8	10
[34] $n = 1000$		34	34	34
pico		40	40	50

Tabla 3: Estimación de los Mflops que se pueden obtener en los distintos sistemas.

Como ya hemos dicho, en los algoritmos con los que trabajamos no siempre se realizan operaciones sobre vectores y matrices del mismo tamaño, dependiendo éste del tamaño de la matriz, sino que se trabaja en un mismo paso con vectores (o submatrices) de distintos tamaños, lo que hará que las prestaciones que se obtienen estén lejos de las obtenidas con el test de Dongarra. Para hacer una estimación que consideramos más acorde con el tipo de algoritmos con el que estamos, estudiamos los Mflops obtenidos con los algoritmos secuenciales SeSiFi, SeSiFiB1 y SeSiB3, para distintos tamaños de problema, y los tomaremos como aproximaciones de los Mflops que se alcanzan al realizar operaciones aritméticas en nuestros algoritmos no usando BLAS, usando BLAS 1 y usando BLAS 3, respectivamente. En la tabla 4 se muestran los Mflops obtenidos con los diferentes algoritmos y diferentes tamaños de matriz en los distintos sistemas.

	iPSC/2	iPSC/860	DELTA	Paragon
SeSiFi $n = 512$	0.14	2.85	2.85	3.78
SeSiFiB1 $n = 512$		4.57	4.57	5.02
SeSiB3 $n = 512$		17.37	17.37	22.60
SeSiB3 $n = 896$		20.31	20.31	26.90
SeSiB3 $n = 960$				28.57
SeSiB3 $n = 1024$		21.80	21.80	

Tabla 4: Prestaciones (en Mflops) de distintos algoritmos secuenciales en los distintos sistemas.

A la vista de los resultados que se muestran en las tablas 3 y 4 se pueden hacer algunas observaciones:

- Las diferentes estimaciones obtenidas en iPSC/2 son muy parecidas.
- En los otros sistemas, en los que cada procesador es un i860, las prestaciones difieren ampliamente dependiendo de que se trabaje con problemas más o menos grandes, que se use o no BLAS 1 y BLAS 3. Esto es debido a la jerarquía de memorias del i860. Se obtiene un uso más eficiente de esa jerarquía si se usa BLAS 1 que si no se usa BLAS,

y el uso de BLAS 3 permite un uso mucho más eficiente de la jerarquía debido a que se realizan $O(n^2)$ accesos a memoria por $O(n^3)$ operaciones aritméticas.

- Las prestaciones obtenidas en el iPSC/860 y el Touchstone DELTA son idénticas al ser los procesadores en los dos casos i860. Sin embargo, aunque el procesador del Paragon es también un i860, en este caso las prestaciones son mayores, reflejándose esta diferencia principalmente en la potencia pico de la máquina (tabla 3) y en los resultados obtenidos con los algoritmos de Jacobi (tabla 4).
- Con el programa SeSiB3 se muestran los Mflops con distintos tamaños de la matriz, y no sabemos a priori cuál de los tamaños nos va a producir una mejor aproximación al comportamiento real de los programas paralelos, pero al aumentar el tamaño de la matriz los valores tienden a estabilizarse.

Teniendo en cuenta lo anterior, mostramos en la tabla 5 estimaciones del tiempo de ejecución (en μs) de una operación sin usar BLAS, usando BLAS 1 (k_1) y usando BLAS 3 (k_3), usando para BLAS 3 $n = 896$. Estos valores serán los que se usen para obtener los tiempos teóricos a comparar con los experimentales.

	iPSC/2	iPSC/860	DELTA	Paragon
sin BLAS	7.14	0.35	0.35	0.26
con BLAS 1 (k_1)		0.21	0.21	0.20
con BLAS 3 (k_3)		0.049	0.049	0.037

Tabla 5: Estimación del tiempo de ejecución de una operación aritmética en los distintos sistemas.

3 Resultados experimentales

El estudio teórico de un algoritmo debe permitirnos predecir su comportamiento experimental, pero esa predicción depende de muchos factores dependientes de la máquina, del software que se utilice, ... Esto hace que muchas veces nos encontremos con una gran diferencia entre los resultados experimentales y los que esperábamos obtener basándonos en el estudio teórico. En esta sección realizaremos una serie de estudios experimentales, y en algunos casos contrastaremos los resultados experimentales con los predichos teóricamente.

3.1 Factores que influyen en la eficiencia

Empezamos analizando los factores que influyen en la eficiencia utilizando el programa PaNsAnCoPi. La eficiencia teórica de este algoritmo (tabla 2) es:

$$\frac{3n^3\gamma}{\left(\frac{6n^3}{p}\gamma + \frac{13n^2}{2p}\gamma + 2np\beta + 4n^2\tau\right)p} = \frac{1}{2 + \frac{13}{6n} + \frac{2p^2\beta}{3n^2\gamma} + \frac{4p\tau}{3n\gamma}} \quad (1)$$

donde γ representa el coste de una operación aritmética.

Claramente, el aumento del número de procesadores hace bajar la eficiencia, y el aumento del tamaño de la matriz manteniendo el número de procesadores fijos hace que aumente la eficiencia. Además, debido a los términos $\frac{\beta}{\gamma}$ y $\frac{\tau}{\gamma}$ que aparecen en el cociente de la fórmula, en un sistema donde los cocientes entre los costes de comunicación y aritmético sean menores se tendrá un menor descenso de la eficiencia al aumentar el número de procesadores. Estos cocientes se muestran en la tabla 6. Se pueden hacer varias observaciones que nos permitirán predecir el comportamiento de los algoritmos en los distintos sistemas:

- Los cocientes son menores en iPSC/2 que en los demás sistemas, por lo que en iPSC/2 la caída de la eficiencia al aumentar el número de procesadores debe ser menor. Desgraciadamente, el iPSC/2 al que hemos tenido acceso tiene únicamente 4 procesadores, lo que hace que los resultados que obtenemos no sean muy significativos.

- Los valores del Touchstone DELTA son algo menores que los del iPSC/860, con lo que, al ser en los dos el procesador de cada nodo igual, los tiempos de ejecución serán muy parecidos con pocos procesadores y los obtenidos en el Touchstone DELTA mejorarán a los obtenidos en iPSC/860 al aumentar el número de procesadores.

- El comportamiento del Paragon es más imprevisible. Los cocientes son peores que los del iPSC/2, pero comparando con el Touchstone DELTA y el iPSC/860 el cociente $\frac{\beta}{\gamma}$ es mayor en el Paragon y el $\frac{\tau}{\gamma}$ es mayor en los otros sistemas. Debido a que el término $\frac{p}{n}$ que afecta a $\frac{\beta}{\gamma}$ va elevado al cuadrado mientras que el que afecta a $\frac{\tau}{\gamma}$ es lineal, las prestaciones caerán muy rápidamente en el Paragon al aumentar el número de procesadores manteniendo n fijo, pero un aumento de n producirá una subida en la eficiencia mayor que en los otros dos sistemas.

	iPSC/2	iPSC/860	DELTA	Paragon
$\frac{\beta}{\gamma}$	40.616	217.143	174.286	661.538
$\frac{\tau}{\gamma}$	0.401	8.171	3.429	1.308

Tabla 6: Cocientes de los costes de comunicación con el aritmético, en diferentes sistemas.

En las tablas 7 y 8 se muestran las eficiencias experimentales (tabla 7) y teóricas (tabla 8) obtenidas con PaNsAnCoPi, con diferentes tamaños de matriz (128, 256 y 512), diferente número de procesadores (2, 4, 8, 16 y 32), y en iPSC/2, iPSC/860 y Paragon.

Vemos que el estudio teórico nos permite predecir el comportamiento experimental del programa en cuanto a la forma en que varía la eficiencia al aumentar el número de procesadores y el tamaño del problema, pero hay muchos factores que no se tienen en cuenta a la hora de calcular los costes teóricos:

- El overhead producido por las comunicaciones (copia de datos, sincronización, ...).

- Los movimientos de columnas internos a cada procesador que se realizan entre cada dos pasos del algoritmo para obtener la nueva agrupación de columnas.

	iPSC/2			iPSC/860			Paragon		
proc	128	256	512	128	256	512	128	256	512
2	0.46	0.47	0.47	0.27	0.39	0.47	0.50	0.58	0.74
4	0.45	0.46	0.47	0.16	0.34	0.46	0.45	0.56	0.73
8				0.10	0.24	0.40	0.33	0.51	0.70
16				0.05	0.14	0.31	0.17	0.26	0.64
32							0.06	0.19	0.45

Tabla 7: Eficiencia experimental de PaNsAnCoPi en distintos sistemas.

	iPSC/2			iPSC/860			Paragon		
proc	128	256	512	128	256	512	128	256	512
2	0.496	0.498	0.499	0.453	0.477	0.489	0.468	0.490	0.496
4	0.489	0.496	0.498	0.402	0.453	0.477	0.402	0.468	0.490
8	0.467	0.489	0.496	0.308	0.402	0.453	0.261	0.402	0.468
16	0.401	0.467	0.489	0.177	0.308	0.402	0.109	0.261	0.402
32	0.261	0.401	0.467	0.072	0.177	0.308	0.033	0.109	0.261

Tabla 8: Eficiencia teórica de PaNsAnCoPi en distintos sistemas.

- Que para distintos tamaños de mensajes los parámetros β y τ son distintos mientras que para realizar el estudio consideramos unos parámetros fijos.
- La jerarquía de memorias propicia que el trabajo con menos datos produzca mejores resultados, por lo que puede pasar que se puedan obtener eficiencias experimentales mayores de las teóricas (como pasa en este caso en el Paragon) al trabajar cada procesador con un volumen menor de datos al aumentar el número de procesadores.

Podemos concluir, por tanto, que el estudio teórico nos servirá para predecir el comportamiento del algoritmo y que esta predicción es tanto más precisa cuanto mayor es el tamaño del problema, ya que estamos haciendo la predicción en base a unos parámetros asintóticos.

3.2 Comparación del tiempo de ejecución teórico y experimental en los algoritmos secuenciales

Comparamos los tiempos teóricos y experimentales con los algoritmos secuenciales: SeSiFi, SeSiFiB1 y SeSiB3.

3.2.1 SeSiFi

Los comparamos en un procesador del iPSC/2, del iPSC/860 y del Paragon. En la tabla 9 se muestran los tiempos teórico y experimental y el cociente entre el tiempo teórico y experimental. Un valor mayor que 1 en el cociente indica que el tiempo previsto es menor que el obtenido experimentalmente, y un valor menor que uno que el tiempo experimental es menor que el previsto. Se observa que los tiempos teóricos son una buena aproximación de los experimentales, siendo la aproximación mejor a mayor tamaño del problema, lo que es debido a que el coste de una operación aritmética se ha estimado con $n = 512$. Por tanto, con tamaño 512 deberíamos tener un cociente igual a 1, mientras que tenemos cocientes ligeramente distintos de 1 debido a los redondeos. En el iPSC/2 el resultado experimental es siempre mejor que el previsto, y se aproximan cuando aumenta el tamaño del problema. En los otros dos sistemas (donde el procesador es un i860) los dos valores están muy próximos para tamaños grandes, mientras que para tamaños pequeños hay una mayor diferencia que puede ser debida al uso de la caché que puede hacer que el valor estimado del coste dependa más del tamaño del problema que en el iPSC/2.

sistema	64	128	192	256	320	384	448	512
	Tiempo teórico							
iPSC/2	5.61	44.92	151.60	359.36	701.89	1212.86	1925.98	2874.94
iPSC/860	0.275	2.20	7.43	17.61	34.40	59.45	94.41	140.92
Paragon	0.204	1.63	5.52	13.08	25.55	44.16	70.13	104.68
	Tiempo experimental							
iPSC/2	5	42	143	342	668	1156	1838	2745
iPSC/860	0.308	1.88	6.33	15.78	32.51	57.25	94.47	141.28
Paragon	0.195	1.18	4.01	10.66	22.69	41.88	69.71	106.59
	Tiempo teórico/Tiempo experimental							
iPSC/2	1.12	1.06	1.06	1.05	1.05	1.04	1.04	1.04
iPSC/860	0.89	1.17	1.17	1.11	1.05	1.03	0.99	0.99
Paragon	1.04	1.38	1.37	1.22	1.12	1.05	1.00	0.98

Tabla 9: Tiempos de ejecución (por barrido y en segundos) teórico y experimental de SeSiFi en distintos sistemas.

3.2.2 SeSiFiB1

Los comparamos en un procesador del iPSC/860 y del Paragon. En la tabla 10 se muestran los tiempos teórico y experimental y el cociente entre el tiempo teórico y experimental, tal como en la tabla 9 se hacía con SeSiFi. Se observa un comportamiento similar al de SeSiFi, pero en este caso las diferencias entre el tiempo teórico y experimental son mayores debido al comportamiento irregular de BLAS 1 para distintos tamaños de vector, siendo típicamente la ganancia que se obtiene con el uso de BLAS 1 con respecto a no usar BLAS mayor para vectores de tamaño mediano y estabilizándose para tamaños grandes en torno al 30%.

sistema	64	128	192	256	320	384	448	512
	Tiempo teórico							
iPSC/860	0.16	1.32	4.45	10.56	20.64	35.67	56.64	84.55
Paragon	0.157	1.25	4.24	10.06	19.66	33.97	53.94	80.53
	Tiempo experimental							
iPSC/860	0.39	1.14	3.80	9.36	20.04	35.35	60.23	88.19
Paragon	0.106	0.805	2.69	7.24	16.14	31.23	52.42	80.42
	Tiempo teórico/Tiempo experimental							
iPSC/860	0.41	1.15	1.17	1.12	1.02	1.00	0.94	0.95
Paragon	1.48	1.55	1.57	1.38	1.21	1.08	1.02	1.00

Tabla 10: Tiempos de ejecución (por barrido y en segundos) teórico y experimental de SeSiFiB1 en distintos sistemas.

3.2.3 SeSiB3

Los comparamos en un procesador del iPSC/860 y del Paragon. En la tabla 11 se muestran los tiempos teórico y experimental y el cociente entre el tiempo teórico y experimental, tal como en la tabla 9 se hacía con SeSiFi. Para el cálculo del tiempo teórico se ha utilizado un tamaño de bloque 16, mientras que al aumentar el tamaño del problema el tamaño de bloque óptimo crece. Esto hace que la diferencia entre los tiempos experimental y teórico crezca al aumentar el tamaño del problema.

sistema	128	256	384	512	640	768	896
	Tiempo teórico						
iPSC/860	0.930	5,33	15.69	34.46	64.10	107.09	165.89
Paragon	0.832	4.55	13.02	28,10	51.66	85.55	131.65
	Tiempo experimental						
iPSC/860	1.39	5.28	14.48	31.74	57.09	91.23	141.39
Paragon	0.570	3.65	10.29	23.69	41.87	68.39	105.34
	Tiempo teórico/Tiempo experimental						
iPSC/860	0.66	1.00	1.08	1.08	1.12	1.17	1.17
Paragon	1.45	1.24	1.26	1.18	1.23	1.25	1.24

Tabla 11: Tiempos de ejecución (por barrido y en segundos) teórico y experimental de SeSiB3 en distintos sistemas.

3.3 Prestaciones de PaNsAnCoPi

En las tablas 12, 13 y 14 se muestra el cociente del tiempo de ejecución teórico y experimental en iPSC/2, iPSC/860 y Paragon, respectivamente.

Vemos que las predicciones en el iPSC/2 son muy precisas. No pasa lo mismo en el

procesadores	64	128	192	256	320	384	448	512
2	0.97	0.99	0.98	0.98	0.98	0.98	0.98	0.98
4	1.01	0.88	0.98	0.98	0.98	0.98	0.98	0.98

Tabla 12: Cociente de los tiempos de ejecución teórico y experimental de PaNsAnCoPi en iPSC/2.

procesadores	64	128	192	256	320	384	448	512
2	0.31	1.08	1.40	1.50	1.58	1.59	1.62	1.53
4	0.15	0.72	1.27	1.33	1.44	1.50	1.53	1.49
8	0.19	0.53	0.84	1.03	1.21	1.29	1.36	1.34
16	0.22	0.50	0.73	0.72	0.87	0.96	1.08	1.12

Tabla 13: Cociente de los tiempos de ejecución teórico y experimental de PaNsAnCoPi en iPSC/860.

iPSC/860 y el Paragon. En estos sistemas, la ejecución paralela produce resultados mejores de los esperados (en el iPSC/860 al aumentar el tamaño de la matriz) debido a que cada procesador utiliza menos memoria y se explota mejor la jerarquía de memorias que en el programa secuencial.

Por otro lado, el overhead asociado a las comunicaciones (sincronización, copia de datos, ...) produce que al aumentar el número de procesadores tengamos una caída adicional de las prestaciones. Esta caída se nota en el iPSC/860, pero no en el Paragon, donde al aumentar el número de procesadores aumenta la mejora del tiempo experimental respecto al teórico. Esto debe ser a causa de que el mejor uso de la jerarquía de memoria con más procesadores no se ve contrarrestado con el overhead de las comunicaciones.

3.4 Explotación de la simetría en anillo

Comparamos las prestaciones de un algoritmo que explota la simetría en un anillo de procesadores (PaSiAnMaEb) con el que no explota la simetría (PaNsAnCoPi). En la tabla 15 se muestra la eficiencia teórica de los dos algoritmos en iPSC/860, y en la tabla 16 en el Paragon.

procesadores	64	128	192	256	320	384	448	512
2	1.21	1.50	1.50	1.47	1.48	1.48	1.48	1.47
4	1.25	1.56	1.55	1.48	1.48	1.48	1.47	1.47
8	1.44	1.77	1.64	1.57	1.51	1.50	1.48	1.47
16	1.76	2.14	1.85	1.77	1.68	1.64	1.57	1.55
32	2.13	2.40	2.16	2.11	1.95	1.86	1.75	1.71

Tabla 14: Cociente de los tiempos de ejecución teórico y experimental de PaNsAnCoPi en Paragon.

Se observa que el algoritmo que explota la simetría tiene una eficiencia teórica mayor que la del que no la explota. En las tablas 17 y 18 se muestran las eficiencias experimentales en iPSC/860 y Paragon, respectivamente. Se comprueba que, aunque la eficiencia experimental del algoritmo simétrico es mayor que la del no simétrico, la diferencia entre los dos algoritmos es menor que la que se desprende del estudio teórico. Esto se debe al mayor overhead del algoritmo simétrico al tener una representación más irregular de los datos.

método/procesadores	64	128	192	256	320	384	448	512
PaNsAnCoPi 2	0.43	0.47	0.48	0.48	0.49	0.49	0.49	0.49
PaSiAnMaEb 2	0.61	0.78	0.85	0.88	0.90	0.92	0.93	0.94
PaNsAnCoPi 4	0.36	0.44	0.46	0.47	0.48	0.48	0.48	0.48
PaSiAnMaEb 4	0.37	0.59	0.70	0.76	0.80	0.83	0.85	0.87
PaNsAnCoPi 8	0.22	0.36	0.41	0.44	0.45	0.46	0.47	0.47
PaSiAnMaEb 8	0.18	0.38	0.51	0.59	0.66	0.70	0.74	0.77
PaNsAnCoPi 16	0.09	0.22	0.31	0.36	0.39	0.41	0.43	0.46
PaSiAnMaEb 16	0.07	0.18	0.29	0.38	0.45	0.51	0.56	0.60

Tabla 15: Eficiencia teórica de PaNsAnCoPi y PaSiAnMaEb en iPSC/860.

método/procesadores	64	128	192	256	320	384	448	512
PaNsAnCoPi 2	0.39	0.46	0.48	0.48	0.49	0.49	0.49	0.49
PaSiAnMaEb 2	0.64	0.84	0.91	0.94	0.95	0.96	0.97	0.97
PaNsAnCoPi 4	0.25	0.39	0.44	0.46	0.47	0.48	0.48	0.48
PaSiAnMaEb 4	0.34	0.64	0.79	0.86	0.89	0.92	0.93	0.94
PaNsAnCoPi 8	0.10	0.25	0.35	0.40	0.42	0.44	0.44	0.46
PaSiAnMaEb 8	0.12	0.34	0.53	0.65	0.73	0.79	0.83	0.86
PaNsAnCoPi 16	0.03	0.10	0.19	0.26	0.31	0.35	0.38	0.40
PaSiAnMaEb 16	0.03	0.12	0.23	0.34	0.44	0.53	0.60	0.65

Tabla 16: Eficiencia teórica de PaNsAnCoPi y PaSiAnMaEb en Paragon.

3.5 Uso de BLAS 1

El uso de BLAS 1 permite reducir el tiempo de ejecución en alrededor de un 30%. Esto ocurre para tamaños de vector suficientemente grandes. En los programas paralelos cada procesador trabaja con menos datos, lo que produce un mejor uso de la jerarquía de memorias, alcanzándose en muchos casos eficiencias superiores al 100%. Además, al trabajar cada procesador con menos datos, los vectores sobre los que se trabaja son menores que en los programas secuenciales y, por tanto, la mejora relativa al usar BLAS 1 es menor en los programas paralelos que en los secuenciales. En la tabla 19 se muestran los tiempos de PaSiAnMaEb y PaSiAnMaEbB1 en el iPSC/860, y en la tabla 20 se muestran en el Paragon. En

método/procesadores	64	128	192	256	320	384	448	512
PaNsAnCoPi 2	0.20	0.27	0.35	0.39	0.45	0.47	0.51	0.47
PaSiAnMaEb 2	0.24	0.53	0.81	0.94	1.07	1.16	1.25	1.25
PaNsAnCoPi 4	0.10	0.16	0.30	0.34	0.40	0.43	0.48	0.46
PaSiAnMaEb 4	0.09	0.30	0.49	0.62	0.79	0.90	1.02	1.02
PaNsAnCoPi 8	0.06	0.10	0.18	0.24	0.32	0.36	0.41	0.40
PaSiAnMaEb 8		0.13	0.26	0.36	0.51	0.62	0.74	0.79
PaNsAnCoPi 16	0.03	0.05	0.12	0.14	0.20	0.24	0.30	0.31
PaSiAnMaEb 16				0.19		0.35		0.49

Tabla 17: Eficiencia experimental de PaNsAnCoPi y PaSiAnMaEb en iPSC/860.

método/procesadores	64	128	192	256	320	384	448	512
PaNsAnCoPi 2	0.45	0.50	0.52	0.58	0.64	0.69	0.73	0.74
PaSiAnMaEb 2	0.47	0.78	0.87	1.04	1.16	1.28	1.36	1.37
PaNsAnCoPi 4	0.31	0.45	0.50	0.56	0.62	0.68	0.71	0.73
PaSiAnMaEb 4	0.22	0.60	0.73	0.91	1.06	1.18	1.27	1.31
PaNsAnCoPi 8	0.15	0.33	0.42	0.51	0.58	0.64	0.67	0.70
PaSiAnMaEb 8		0.28	0.52	0.69	0.87	0.99	1.11	1.17
PaNsAnCoPi 16	0.05	0.16	0.25	0.37	0.46	0.54	0.59	0.63
PaSiAnMaEb 16				0.35		0.58		0.75

Tabla 18: Eficiencia experimental de PaNsAnCoPi y PaSiAnMaEb en Paragon.

las dos tablas se muestran también los tiempos de los programas secuenciales y los cocientes entre los tiempos de PaSiAnMaEb y PaSiAnMaEbB1. Se confirma la observación anterior.

En las tablas 21 y 22 se muestran los cocientes de los tiempos teóricos de ejecución de los dos algoritmos en iPSC/860 y Paragon, respectivamente. También en el caso teórico se comprueba la reducción en la ganancia del programa que usa BLAS 1 respecto al que no lo usa.

3.6 Escalabilidad de algoritmos en malla

En general, los algoritmos para malla son más escalables que los algoritmos para anillo. Esto se puede comprobar comparando los tiempos de algoritmos en las dos topologías. En la tabla 23 se muestra el cociente de los tiempos teóricos de ejecución de PaNsMaCuRr y PaNsAnCoPi en diferentes sistemas. En la tabla 24 se muestran los mismos cocientes con los tiempos experimentales. Un valor menor que 1 indica que el algoritmo para malla tiene un menor tiempo de ejecución.

método/procesadores	64	128	192	256	320	384	448	512
SeSiFiB1	0.39	1.14	3.80	9.36	20.04	35.35	60.23	88.19
SeSiFi	0.28	1.67	5.85	13.96	29.13	51.54	86.26	126.67
BLAS/no BLAS	1.35	0.68	0.64	0.67	0.68	0.68	0.69	0.69
PaSiAnMaEbB1 2	0.59	1.39	2.83	5.60	9.60	15.85	24.08	35.10
PaSiAnMaEb 2	0.59	1.59	3.59	7.43	13.63	22.23	34.60	50.64
BLAS/no BLAS	1.00	0.87	0.78	0.75	0.70	0.71	0.69	0.69
PaSiAnMaEbB1 4	0.66	1.57	2.89	4.59	7.19	11.30	16.22	22.85
PaSiAnMaEb 4	0.59	1.39	2.99	5.59	9.19	14.39	21.21	30.97
BLAS/no BLAS	1.11	1.12	0.96	0.82	0.78	0.78	0.76	0.73
PaSiAnMaEbB1 8		1.60	2.77	4.46	6.19	9.40	12.16	16.78
PaSiAnMaEb 8		1.64	2.80	4.90	7.21	10.40	14.59	19.99
BLAS/no BLAS		0.97	0.98	0.91	0.85	0.90	0.83	0.83
PaNSiAnMaEbB1 16				4.67		8.59		14.99
PaSiAnMaEb 16				4.74		9.36		15.97
BLAS/no BLAS				0.98		0.91		0.93

Tabla 19: Tiempos de ejecución (por barrido y en segundos) de SeSiFiB1, SeSiFi, PaSiAnMaEbB1 y PaSiAnMaEb, y cociente de los tiempos usando y no usando BLAS. En iPSC/860.

3.7 Explotación de la simetría en malla

Ya hemos comparado un algoritmo que no explota la simetría con otro que sí la explota, en un anillo. En este apartado realizamos la misma comparación en una malla. En las tablas 25 y 26 se comparan los tiempos de los algoritmos PaNsMaCuRr y PaSiMaPlRr en iPSC/860 y Touchstone DELTA, respectivamente. Se muestran también los cocientes de los tiempos de ejecución entre el método que explota la simetría y el que no la explota. En el Touchstone DELTA, para tamaños grandes de la matriz y 4 procesadores, el cociente llega a ser menor de 0.5, lo que quiere decir que explotando la simetría se acelera la ejecución más de lo que teóricamente se aceleraría. Esto puede ser debido, de nuevo, al uso de la memoria, pues con el esquema de almacenamiento por plegamiento cada procesador trabaja con menos datos y estos están divididos en submatrices (dos o tres dependiendo del procesador). El mismo fenómeno se observa acentuado en la tabla 27, donde se muestran tiempos de los algoritmos usando BLAS 1 y en el Paragon.

3.8 El algoritmo por bloques

Analizaremos el comportamiento del algoritmo PaSiMaPlB3 en iPSC/860, Touchstone DELTA y Paragon.

método/procesadores	64	128	192	256	320	384	448	512
SeSiFiB1	0.10	0.80	2.69	7.24	16.14	31.23	52.42	80.19
SeSiFi	0.19	1.18	4.01	10.66	22.69	41.88	69.71	106.59
BLAS/no BLAS	1.35	0.68	0.64	0.67	0.68	0.68	0.69	0.69
PaSiAnMaEbB1 2	0.19	0.55	1.62	3.64	6.69	11.64	18.10	27.12
PaSiAnMaEb 2	0.20	0.75	2.30	5.11	9.71	16.35	25.60	38.85
BLAS/no BLAS	0.95	0.73	0.70	0.71	0.68	0.71	0.70	0.69
PaSiAnMaEbB1 4	0.23	0.42	1.13	2.23	3.87	6.47	9.65	14.47
PaSiAnMaEb 4	0.22	0.48	1.37	2.92	5.34	8.86	13.63	20.33
BLAS/no BLAS	1.04	0.87	0.82	0.76	0.72	0.73	0.70	0.71
PaSiAnMaEbB1 8		0.51	0.83	1.61	2.70	4.30	5.97	8.26
PaSiAnMaEb 8		0.52	0.96	1.91	3.26	5.24	7.83	11.38
BLAS/no BLAS		0.98	0.86	0.84	0.82	0.82	0.76	0.72
PaNSiAnMaEbB1 16				1.58		3.23		6.31
PaSiAnMaEb 16				1.87		4.47		8.86
BLAS/no BLAS				0.84		0.72		0.71

Tabla 20: Tiempos de ejecución (por barrido y en segundos) de SeSiFiB1, SeSiFi, PaSiAnMaEbB1 y PaSiAnMaEb, y cociente de los tiempos usando y no usando BLAS. En Paragon.

3.8.1 Uso de BLAS 3 y BLAS 1

El uso de BLAS 3 nos permite obtener algoritmos más eficientes al hacer un mejor uso de la jerarquía de memorias. Para poder usar BLAS 3 es necesario rediseñar los algoritmos para obtener algoritmos por bloques, y esto produce en algunos casos un mayor coste computacional. Así, el algoritmo SeSiB3 tiene un coste $4n^3$ por barrido y el SeSiFiB1 un coste $3n^3$ también por barrido. A pesar del mayor coste computacional del algoritmo por bloques, con éste se tienen menores tiempos de ejecución que usando BLAS 1, llegándose, para tamaños de matriz grandes, a una reducción de hasta el 75% en el tiempo de ejecución [1]. Con los algoritmos paralelos también se obtiene una reducción en el tiempo de ejecución con el al-

método/procesadores	64	128	192	256	320	384	448	512
Secuencial	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60
Paralelo 2	0.73	0.67	0.65	0.64	0.63	0.62	0.62	0.62
Paralelo 4	0.83	0.75	0.71	0.68	0.67	0.66	0.65	0.64
Paralelo 8		0.84	0.79	0.75	0.73	0.71	0.70	0.68
Paralelo 16				0.84		0.79		0.75

Tabla 21: Cociente de los tiempos teóricos de ejecución usando y no usando BLAS. En iPSC/860.

método/procesadores	64	128	192	256	320	384	448	512
Secuencial	0.76	0.76	0.76	0.76	0.76	0.76	0.76	0.76
Paralelo 2	0.83	0.79	0.77	0.77	0.77	0.77	0.77	0.77
Paralelo 4	0.91	0.84	0.81	0.79	0.78	0.78	0.78	0.77
Paralelo 8		0.91	0.87	0.84	0.82	0.81	0.80	0.79
Paralelo 16				0.91		0.87		0.84

Tabla 22: Cociente de los tiempos teóricos de ejecución usando y no usando BLAS. En Paragon.

procesadores	64	128	192	256	320	384	448	512
	iPSC/860							
4	1.08	1.04	1.02	1.01	1.01	1.01	1.01	1.00
16	0.54	0.69	0.78	0.84	0.88	0.90	0.92	0.93
	Paragon							
4	1.10	1.03	1.01	1.00	1.00	1.00	1.00	1.00
16	0.42	0.52	0.63	0.71	0.78	0.83	0.86	0.89

Tabla 23: Cociente de los tiempos teóricos de ejecución de PaNsMaCuRr y PaNsAnCoPi.

goritmo por bloques, pero esta reducción es menor que en los algoritmos secuenciales debido al coste adicional de las comunicaciones y al trabajo en cada procesador con submatrices de menor tamaño. Esto se ve en la tabla 28, donde se muestran los cocientes de los tiempos de PaSiMaPlB3 y PaSiMaPlRrB1 en los distintos sistemas, con distinto tamaño de matriz y distinto número de procesadores. Se observa que la ganancia con el uso de BLAS 3 aumenta al aumentar el tamaño de la matriz, disminuye al aumentar el número de procesadores, y aumenta en sistemas con mejores comunicaciones (en Touchstone DELTA mejor que en el iPSC/860) y en sistemas con mejor uso de la jerarquía de memorias (en Paragon mejor que en iPSC/860 y Touchstone DELTA).

procesadores	64	128	192	256	320	384	448	512
	iPSC/860							
4	0.66	0.88	1.13	1.07	1.11	1.13	1.17	1.14
16	1.01	0.84	0.90	0.78	0.80	0.87	0.92	0.92
	Paragon							
4	0.87	1.15	1.24	1.22	1.30	1.35	1.41	1.40
16	0.56	0.55	0.66	0.92	0.97	1.08	1.07	1.15

Tabla 24: Cociente de los tiempos experimentales de ejecución de PaNsMaCuRr y PaNsAnCoPi.

	64	128	192	256	320	384	448	512
PaNsMaCuRr 4	0.795	1.534	3.553	7.451	13.825	23.167	36.978	54.963
PaSiMaPlRr 4	0.650	1.211	2.395	4.779	7.797	13.012	18.980	28.684
Cociente 4	0.82	0.79	0.67	0.64	0.56	0.56	0.51	0.52
PaNsMaCuRr 16	0.796	1.012	1.822	3.293	5.027	8.031	11.693	16.413
PaSiMaPlRr 16	0.648	1.136	1.769	2.778	3.955	6.001	8.028	11.235
Cociente 16	0.81	1.12	0.97	0.84	0.79	0.75	0.69	0.68

Tabla 25: Tiempos de ejecución por barrido (en segundos) de los algoritmos PaNsMaCuRr y PaSiMaPlRr. En iPSC/860.

	64	128	192	256	320	384	448	512
PaNsMaCuRr 4	0.151	1.008	3.012	6.831	13.074	22.769	37.221	56.716
PaSiMaPlRr 4	0.172	0.703	1.807	4.042	6.949	12.014	17.767	27.384
Cociente 4	1.14	0.70	0.60	0.59	0.53	0.53	0.48	0.48
PaNsMaCuRr 16	0.120	0.387	1.018	2.204	3.738	6.230	9.282	13.853
PaSiMaPlRr 16	0.175	0.463	0.938	1.735	2.751	4.219	6.031	8.979
Cociente 16	1.46	1.20	0.92	0.79	0.74	0.68	0.65	0.65

Tabla 26: Tiempos de ejecución por barrido (en segundos) de los algoritmos PaNsMaCuRr y PaSiMaPlRr. En Touchstone DELTA.

3.8.2 Comparación de resultados teóricos y experimentales

Este algoritmo es, de entre todos los analizados, en el que más factores intervienen: el coste de una operación aritmética con BLAS 1 y BLAS 3 y el coste de las comunicaciones. Además de estos factores que se tienen en cuenta en el estudio teórico hay otros factores que no aparecen en este estudio: que el coste de las operaciones usando BLAS es distinto a tamaños distintos de vector o de matriz, el uso de la jerarquía de memorias, el overhead debido a las comunicaciones por sincronización o copias de datos, el solapamiento de las operaciones aritméticas y de comunicaciones, que no se ha tenido en cuenta en el estudio teórico pero que existe al hacerse las difusiones desde procesadores en la antidiagonal principal a los demás procesadores en la misma fila y columna, ... Esto hará que los resultados teóricos estén lejos de los experimentales y que la diferencia entre ellos dependa ampliamente del sistema. En la tabla 29 se muestran el tiempo teórico y experimental de PaSiMaPlB3 en distintos sistemas. Aunque, como ya hemos dicho, el tiempo experimental no se ajusta al teórico, éste sirve para comparar el comportamiento del programa en los distintos sistemas.

	64	128	192	256	320	384	448	512
PaNsMaCuRrB1 4	0.137	0.753	2.460	5.837	11.520	20.527	34.180	50.615
PaSiMaPlRrB1 4	0.190	0.411	1.202	2.877	5.170	9.172	14.028	21.633
Cociente 4	1.38	0.54	0.48	0.49	0.44	0.44	0.41	0.42
PaNsMaCuRrB1 16	0.120	0.240	0.651	1.659	2.972	5.218	7.855	11.968
PaSiMaPlRrB1 16	0.217	0.259	0.528	0.991	1.701	2.674	3.982	6.045
Cociente 16	1.80	1.07	0.79	0.60	0.57	0.51	0.50	0.50

Tabla 27: Tiempos de ejecución por barrido (en segundos) de los algoritmos PaNsMaCuRrB1 y PaSiMaPlRrB1. En Paragon.

3.8.3 Escalabilidad

Se puede comprobar experimentalmente que el algoritmo presenta buena escalabilidad. En la tabla 30 se muestran los Mflops por nodo obtenidos en iPSC/860, Touchstone DELTA y Paragon. Aparecen prestaciones con dos tamaños fijos de problema por procesador: 128×128 y 256×256. En las tablas aparecen datos para distinto número de procesadores. Se observa que el descenso en el número de Mflops por procesador al aumentar el tamaño del sistema es pequeño, con lo que concluimos que la escalabilidad experimental es satisfactoria. Además, la reducción en el número de Mflops por nodo es sustancialmente menor en el Touchstone DELTA que en el iPSC/860, lo que es normal al tener los dos sistemas el mismo tipo de procesador i860, y el Touchstone DELTA mejores comunicaciones. La reducción en las prestaciones al aumentar el número de procesadores es similar en el Touchstone DELTA y el Paragon.

Referencias

[1] Domingo Giménez Cánovas. Métodos de Jacobi para la resolución del Problema Simétrico de Valores Propios en Multicomputadores. Tesis Doctoral. 1995.

[2] C. Conde Lázaro y G. Winter Althaus. *Métodos y algoritmos básicos del álgebra numérica.* Reverté, 1990.

[3] B. S. Garbow, J. M. Boyle, J. J Dongarra y C. B. Moler. *Matrix Eigensystem Routines-EISPACK Guide Extension.* Springer-Verlag, 1972.

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

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

[6] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema y C. B. Moler. *Matrix Eigensystem Routines-EISPACK Guide.* Springer-Verlag, 1976.

procesadores	128	256	384	512
	iPSC/860			
1	1.21	0.56	0.42	0.35
4	0.77	0.58	0.49	0.43
16	0.57	0.57	0.58	0.56
	Touchstone DELTA			
1	1.21	0.56	0.42	0.35
4	0.61	0.50	0.44	0.39
16	0.48	0.53	0.55	0.50
	Paragon			
1	0.70	0.34	0.32	0.29
4	0.67	0.47	0.38	0.34
16	1.62	0.57	0.51	0.45

Tabla 28: Cociente de los tiempos experimentales de ejecución de PaSiMaPlB3 y PaSiMaPlRrB1. En el caso de un procesador se muestra el cociente entre SeSiB3 y SeSiFiB1.

[7] Antonio M. Vidal. Algoritmos Matriciales Paralelos y Arquitecturas Asociadas para el cálculo de la SVD. Ph. Tesis. D.S.I.C. Univ Politécnica de Valencia, 1990.

[8] David S. Watkins. *Matrix Computations.* John Wiley & Sons, 1991.

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

[10] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov y D. Sorensen. *LAPACK Users' Guide.* SIAM, 1992.

[11] James W. Demmel, Michael T. Heath y Henk A. van der Vorst. Parallel numerical linear algebra. In *Acta Matematica*, pages 111–197. Cambridge University Press, 1993.

[12] Zhaojun Bai y James Demmel. On a block implementation of Hessenberg multishift QR iteration. *International Journal of High Speed Computing*, 1(1):97–112, 1989.

[13] Greg Henry y Robert van de Geijn. Parallelizing the QR Algorithm for the Unsymmetric Algebraic Eigenvalue Problem: myths and reality. Technical Report LAPACK WN 79.

[14] Thomas Schreiber, Peter Otto y Fridolin Hofmann. A new efficient parallelization strategy for the QR algorithm. *Parallel Computing*, 20:63–75, 1994.

[15] Robert A. van de Geijn. Implementing the QR-algoritm on an array of processors. Technical Report TR-1897, University of Maryland, August 1987.

[16] Robert A. van de Geijn. Storage schemes for Parallel Eigenvalue Algorithms. In G. H. Golub y P. Van Dooren, editor, *Numerical Linear Algebra. Digital Signal Processing and Parallel Algorithms*, volume 70 of *NATO ASI Series.* Springer-Verlag, 1991.

tiempo/procesadores	128	256	384	512
	iPSC/860			
teórico/4	0.654	2.940	7.473	14.870
teórico/16	0.351	1.458	3.474	6.553
experimental/4	0.938	2.804	6.402	12.392
experimental/16	0.649	1.601	3.522	6.320
	Touchstone DELTA			
teórico/4	0.558	2.558	6.614	13.344
teórico/16	0.275	1.157	2.798	5.352
experimental/4	0.434	2.039	5.318	10.839
experimental/16	0.225	0.934	2.348	4.526
	Paragon			
teórico/4	0.468	2.091	5.331	10.656
teórico/16	0.227	0.926	2.212	4.203
experimental/4	0.227	1.359	3.577	7.402
experimental/16	0.422	0.569	1.386	2.763

Tabla 29: Tiempos teórico y experimental (por barrido y en segundos) de PaSiMaPlB3.

[17] Robert A. van de Geijn. Deferred shifting schemes for parallel QR methods. *SIAM J. Matrix Anal. Appl.*, 14(1):180–194, 1993.

[18] Robert A. van de Geijn y D. G. Hudson III. Efficient parallel implementation of the nonsymmetric QR algorithm. In *Proceedings of the Fourth Conference on Hypercube Concurrent Computers and Applications*, March 1991.

[19] David S. Watkins. Shifting strategies for the parallel QR algorithm. *SIAM J. Sci. Comput.*, 15(4):953–958.

[20] C. G. J. Jacobi. Uber ein leichtes Verfahren die in der Theorie der Säcularstörungen vorkommenden Gleichungen numerisch aufzulösen. *Journal für die Reine and Anbewante Mathematic*, 30:51–94, 1846.

[21] C. G. J. Jacobi. On a New Way of Solving the Linear Equations that Arise in the Method of Least Squares. Traducción de G. W. Stewart. Technical Report TR-92-42,TR-2877, Institute for Advanced Computer Studies. Department of Computer Science. University of Maryland, April 1992.

[22] 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.

[23] J. Demmel y K. Veselić. Jacobi's method is more accurate than QR. *SIAM J. Matrix Anal. Appl.*, 13:1204–1245, 1992.

procesadores:	1	4	9	16	25	36	49	64	256
	iPSC/860								
128×128	7.16	5.99		5.30					
256×256	12.71	10.83		8.08					
	Touchstone DELTA								
128×128	7.16	8.26	7.74	7.42	7.20	7.39	7.32	7.32	7.19
256×256	12.71	12.39	11.75	11.40	10.92	11.34	11.25	11.25	11.07
	Paragon								
128×128	14.71	12.42	12.45	12.15	12.01	12.12			
256×256	18.38	18.13	18.10	17.87	17.66	17.85			

Tabla 30: Mflops por nodo de PaSiMaPlB3 en distintos sistemas, con distinto número de procesadores, y con tamaño de problema por procesador fijo.

[24] Walter F. Mascarenhas. A note on Jacobi being more accurate than QR. *SIAM J. Matrix Anal. Appl.*, 15(1):215–218, 1994.

[25] *iPSC/2 Users´ s Guide.* Intel Corporation, October 1989.

[26] *iPSC/860 C Compiler Users´ s Guide.* Intel Corporation, April 1991.

[27] *Touchstone DELTA System Description.* Intel Corporation, February 1991.

[28] Artur Klauser. YAP$_P^P$. Yet Another Paragon Programming Primer. Technical report, Graz University of Technology. Computing and Information Services Center, 1995.

[29] Luc Bomans y Dick Roose. Benchmarking the iPSC/2 hypercube multiprocessor. *Concurrency: Practice and Experience*, 1(1):3–18, 1989.

[30] T. H. Dunigan. Performance of the Intel iPSC/860 and the Ncube6400 hypercubes. *Parallel Computing*, 17:1285–1302, 1987.

[31] Roger W. Hockney y Edward A. Carmona. Comparison of communications on the Intel iPSCiPSC/860 and Touchstone Delta. *Parallel Computing*, 18:1067–1072, 1992.

[32] Roger W. Hockney. The communication challenge for MPP: Intel Paragon and Meiko CS-2. *Parallel Computing*, 20:389–398, 1994.

[33] Domingo Giménez, Vicente Hernández, Robert van de Geijn y Antonio M. Vidal. A Jacobi method by blocks on a mesh of processors. *To appear in Concurrency: Practice and Experience.*

[34] Jack J. Dongarra. Performance of various computers using standard linear equations software. Technical Report CS-89-85, January 1994.

ESTUDIO EXPERIMENTAL DE MÉTODOS DE JACOBI BILATERALES
EN SISTEMAS INTEL

Domingo Giménez Cánovas

Departamento de Informática y Sistemas
Universidad de Murcia