Accelerating the Convergence of Blocked Jacobi Methods [1]

D. Giménez, M. T. Cámara, P. Montilla
Departamento de Informática y Sistemas. Univ de Murcia.
Aptdo 4021. 30001 Murcia. Spain.
e-mail: {domingo,cpmcm,cppmm}@dif.um.es

Keywords: Symmetric Eigenvalue Problem, Jacobi methods

ABSTRACT

In this work we study the possible combination of two techniques to reduce the execution time when solving the Symmetric Eigenvalue Problem by Jacobi methods: acceleration of convergence, and work by blocks.

INTRODUCTION

The Symmetric Eigenvalue Problem (SEP) appears in the solution of a lot of problems on science and engineering [5]. In some of these applications the problems to solve are of great size, making it neccessary to use highly efficient methods.

The Jacobi method was the most widely used to solve the SEP for more than a century [9], but in the 60's it was surpassed by methods based on reduction of the initial matrix to tridiagonal form [6]. More recently Jacobi methods have become important again due to better stability properties [4] and straightforward parallelization [13,8], and in some cases the Jacobi methods can surpass other methods based on reduction to tridiagonal form [7].

A Jacobi method for the SEP consists in the generation of a succession $\{A_s\}$ through:

$$A_{s+1} = Q_s A_s Q_s^t, \; s = 1, 2, \ldots$$

with $A_1 = A$, and Q_s represents Givens rotations in the plane (i, j), with $1 \leq i, j \leq n$, nullifying a_{ij} and a_{ji}.

There are two very different strategies to reduce the execution time when solving the SEP by Jacobi methods: acceleration of the convergence, and work by blocks.

To accelerate the convergence the idea is to work element by element choosing the element to be nullified between the elements of largest absolute value, which reduces the number of nullifications needed to reach the convergence.

Another possibility to reduce the execution time consists of redesigning the method to obtain algorithms by blocks which perform more of the computation with matrix-matrix operations (typically matrix multiplications). In this way the better use of the memory hierarchy produces a reduction on the execution time.

In this work we study the possible combination of these two methods. The two methods work in a very different way: to accelerate the convergence the work is done element by element, and with algorithms by blocks the work is done working by blocks of elements. Thus, the two techniques cannot be easily combined.

We will begin analysing different techniques of acceleration of the convergence on methods non-working by blocks, and after that we will study the possible combination of these techniques with an algorithm by blocks.

---

[1] Partially supported by Comisión Interministerial de Ciencia y Tecnología, project TIC96-1062-C03-02; and Consejería de Cultura y Educación, Dirección General de Universidades, project FI-con 96/9. This work has been performed in part on the 44 node Intel Paragon operated by the University of Texas Center of High Performance Computing.

ACCELERATION OF THE CONVERGENCE ON JACOBI METHODS

The classical Jacobi method [9] proceeds by choosing in each iteration, as element to be nullified, that of greatest absolute value from among the nondiagonal elements. Because in each iteration the element of greatest absolute value is chosen, the number of iterations is small but the execution time is very long. Other Jacobi methods proceed by performing successive sweeps, nullifying in each sweep each nondiagonal element once (so, each sweep consists of $n(n-1)/2$ steps), using a certain order to nullify the elements. In this way the calculation of the maximum is avoided and an order $O(n^3)$ is obtained per sweep, while the classical method has an order $O(n^4)$ in $n(n-1)/2$ steps. However, more steps are needed to reach the convergence than in the classical method.

Different techniques have been proposed to reduce the number of nullifications (and consequently the execution time) avoiding obtaining the maximum on each step:

- Threshold strategies:

  With these methods the nondiagonal elements are nullified by sweeps, but the nullification of an element is avoided when it is small in absolute value. In this way only elements of large absolute value (the elements whose absolute value is bigger than the threshold) are nullified. There are different possibilities when choosing the threshold [14,12]:

  - The threshold can be fixed, with a value ensuring that when no elements are nullified in a sweep the method converges ($Off(A) \leq tolerance$).
  - The threshold can vary, using first a threshold of large value (when the nondiagonal elements on the matrix are great) and reducing the threshold when the values of the nondiagonal elements decrease. There are different possibilities but a good strategy is that of Kahan and Corneil:
    Initially $\omega = \sum_{i=1}^{n-1} \sum_{j=1,i<j}^{n} a_{ij}^2$, and after each nullification ω is updated by substracting a_{ij}^2. A rotation is applied to a_{ij} if

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

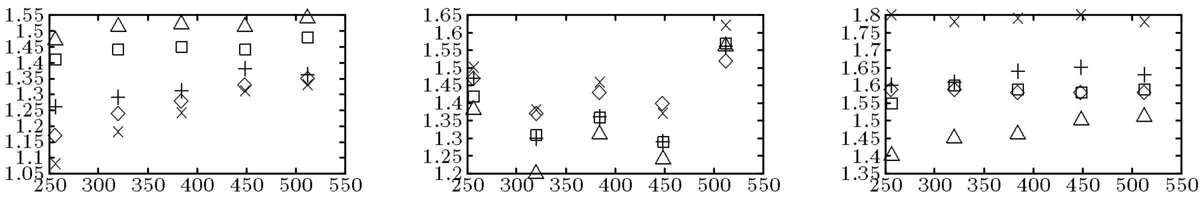
    which means that elements nullified are those whose square is bigger than the mean of the squares.

- Other methods do not nullify the nondiagonal elements in a predetermined order. These elements are preprocessed arranging them in such a way that ensures the elements to be nullified are of high absolute value. That produces a reduction on the number of nullifications to reach the convergence, but can produce or not (depending on the characteristics of the machine and the matrix) a reduction on the execution time. Two of these methods are the Karp-Greenstadt [10] and the semiclassical method [3].

  - In the Karp-Greenstadt method a set of non-conflicting rotations that includes the largest nondiagonal elements (in absolute value) is obtained before each step. To obtain this set, the maxima of each column are obtained and sorted. After that, the elements to be nullified are chosen from this set from the largest to the lowest element, but an element is not chosen if a previous element in the same row has been chosen. In this way the nullifications could be performed in parallel (this is the idea of Karp-Greenstadt), but also the elements in the set have not changed and the initial sorting in this set remains.

– On the semiclassical method the nondiagonal elements are preprocessed in a different way. Before each sweep they could be sorted from the largest to the least absolute value and nullified in this order. But the last elements are elements of low absolute value and their nullification contributes little to the convergence, and when the first elements are nullified the values change and the elements are not ordered as initially. For these reasons, it is preferable not to nullify all the nondiagonal elements and not to sort them. What is better is to "semisort" the elements and nullify only a part of them. The elements are "semisorted" following the Quicksort plan [11]: one element is chosen and the other elements are divided into two sets, one with the elements whose absolute value is bigger than that of the chosen element and another with these elements whose absolute value is smaller. Working with the first set only, succesive steps of this type are made until the greatest element is obtained. After that, the first $(n(n-1)/2)/d$ elements in the "semisorting" are nullified. And the method works by making succesive steps until the convergence is reached. With a big d the number of nullifications would be small, but the number of steps big; and with small d the number of steps would be small but the number of nullifications big. Thus, the optimum value of d depends on the machine we are using.

The different acceleration techniques can be combined in different ways, and which technique is preferred depends on the machine we are using. In figure 1 we compare different techniques of acceleration. The figure shows the quotient of the execution time of a Jacobi method (using a cyclic-by-rows ordering and without threshold strategy) with respect to the execution times obtained with different Jacobi methods using acceleration techniques.



i860 HP Apollo 700 Silicon Graphic Power Challenge XL

Figure 1: *Comparison of different techniques of acceleration. Quotient of the execution time of a Jacobi method (using a cyclic-by-rows ordering and without threshold strategy) with respect to the execution times obtained with different Jacobi methods using acceleration techniques.* $\diamond$: *Kahan-Corneil,* $+$: *semiclassical,* $\square$: *semiclassical+fixed threshold,* $\times$: *semiclassical+Kahan-Corneil,* $\triangle$: *semiclassical+Karp-Greenstadt.*

JACOBI METHODS BY BLOCKS

Recently, to solve efficiently problems of linear algebra on machines with a hierarchical memory, the technique of redesigning the algorithms to work by blocks has been used [1]. Some algorithms have been developed for the SEP or related problems [13,2,8], but in these papers the only reference we have found to a possible acceleration of the convergence on Jacobi methods by blocks is in [2]. This is the motivation of our work: we think it is interesting to analize the possible acceleration of the convergence on Jacobi methods working by blocks.

In the methods by blocks the elements of the matrix are grouped on square blocks and these blocks are treated in some order (as the elements in the methods non-working by blocks). The work in each block can consist of performing a sweep on the elements of the block accumulating the rotations in a rotation matrix, and after that the initial matrix is updated premultiplying and postmultiplying rows and columns of blocks by the rotation matrix. In that way the method has a cost of $4n^3$ flops per sweep, and the methods non-working by blocks have a cost of $3n^3$ flops per sweep, but when working by blocks the updating of the matrix is done with matrix-matrix multiplications using BLAS 3, and the methods by blocks are quicker than those non-working by blocks.

ACCELERATION OF THE CONVERGENCE ON JACOBI METHODS BY BLOCKS

To accelerate the convergence on the Jacobi methods by blocks what we intend to do is to reduce the number of sweeps (not the number of nullifications) because a reduction on the number of nullifications can produce an increment in the number of sweeps, and the cost of the algorithm is $4n^3$ times the number of sweeps.

The combination of the two techniques can be achieved by applying some acceleration technique to each subsweep on each block on the algorithm by blocks. It can produce a reduction on the number of nullifications but not always on the number of global sweeps, as we can see in table 1. The combination of the two techniques is not very promising because only a small reduction on the number of sweeps is achieved in some cases. But we can obtain some conclusions:

	128	256	384	512	640	768
cyclic	10	10	11	11	11	11
cyclic, two subsweeps	8	8	9	9	9	10
var threshold	9	10	10	11	11	12
var threshold, two subsweeps	8	10	9	10	10	10
fixed threshold	10	10	11	11	11	11
Kahan-Corneil threshold	14	14	14	14	14	14
Kahan-Corneil threshold, two subsweeps	12	12	13	14	14	14
semiclassical	8	9	10	10	10	11
semiclassical+fixed threshold	8	10	10	11	12	11

Table 1: *Number of sweeps necessary to reach the convergence for different methods without using an acceleration strategy (cyclic) or using some acceleration strategies.*

- The use of a threshold strategy on the sweeps on each block reduces the number of nullifications, but can produce an increment in the number of sweeps because less nullifications are performed on each sweep.
- It may be preferred to make more computation on each block, using a semiclassical strategy or performing more than one sweep, before updating the matrix, but this work must not be very time consuming because the small reduction on the number of sweeps could not compensate the time of the additional work.

In figure 2 we compare different combinations of acceleration techniques with a scheme by blocks. The figure shows the quotient of the execution time of a Jacobi method by blocks (using an odd-even ordering to generate the order in which the blocks are treated and a cyclic-by-rows ordering to perform the subsweeps on each block) with respect to the execution times obtained with different Jacobi methods by blocks using acceleration techniques on the subsweep on the blocks.

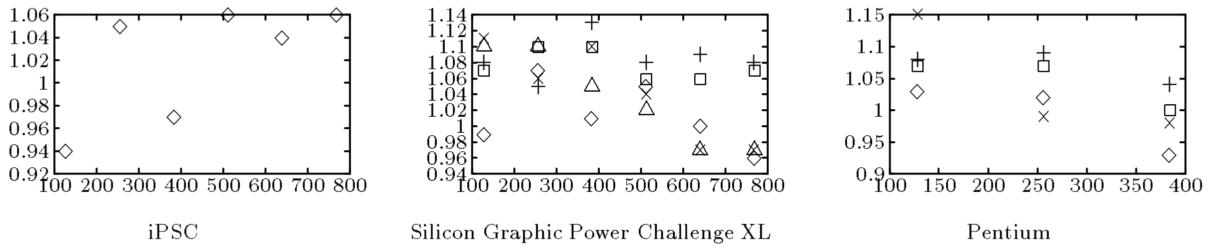


Figure 2: *Comparison of different techniques of acceleration. Quotient of the execution time of a Jacobi method by blocks with respect to the execution times obtained with different Jacobi methods by blocks using acceleration techniques.* $\diamond$*: semiclassical,* $+$*: variable threshold-two subsweeps,* $\square$*: cyclic-two subsweeps,* $\times$*: semiclassical+fixed threshold-two subsweeps,* $\triangle$*: Kahan-Corneil-two subsweeps.*

SPECIAL CASES

There are reasons to think the combination of the two techniques could be more successful in some special cases. We have performed some experiments in which more favourable results have been obtained:

- When solving the SEP obtaining the eigenvalues and the eigenvectors the computation per sweep increases and the additional work to "semisort" the nondiagonal elements on a semiclassical method is less important. Thus, a bigger reduction on the execution time can be achieved. In table 2 we compare the execution time of an algorithm by blocks without acceleration with a method in which a semiclassical strategy is used on each block.

	eigenvalues		eigenvalues+eigenvectors	
	without acceleration	with semiclassical	without acceleration	with semiclassical
128	7.40	6.24	10.14	7.21
256	59.90	52.12	101.78	77.00
384	192.78	189.61	349.13	257.27

Table 2: *Comparison of a Jacobi method by blocks without acceleration with a method in which a semiclassical strategy is used on each block. Execution time when only eigenvalues or eigenvalues and eigenvectors are computed. On a Pentium.*

- In distributed memory algorithms, on each sweep we have arithmetic cost and cost due to communications. As in the previous situation, the time consumed working with each block is less important. In table 3 the execution time of different distributed memory algorithms are compared.
- With some special matrices, which need a bigger number of sweeps to reach the convergence, it is possible to obtain a bigger reduction on the number of sweeps, and consequently on the execution time. In table 4 we compare the number of sweeps and the execution time of some algorithms when applied to one special matrix.

CONCLUSIONS

When solving the SEP by Jacobi methods it is possible to combine techniques of acceleration of the convergence and techniques of work by blocks. The combination of these two classes of techniques produce in some cases (depending on the characteristics of the machine and the matrix) a small reduction on the execution time.

	Processors								
	3			10			36		
	no acc	n a,2 sw	sc	no acc	n a,2 sw	sc	no acc	n a,2 sw	sc
128	2.98	2.75	2.84	1.35	1.26	1.30	0.67	0.61	0.64
256	16.69	15.52	15.74	6.42	6.00	6.00	2.66	2.48	2.72
384	48.87	48.52	49.89	17.26	17.16	17.65	6.61	6.60	6.71
512	117.84	107.74	109.43	38.85	35.68	36.45	14.00	13.10	13.46

Table 3: *Comparison of distributed memory Jacobi methods by blocks: without acceleration (no acc), without acceleration and two subsweeps per block (n a,2 sw), and with semiclassical in each block (sc). On a Paragon.*

	128		256		384	
	time	sweeps	time	sweeps	time	sweeps
without acceleration	7.50	12	62.61	14	232.49	16
with semiclassical	7.44	11	65.09	13	214.30	13
variable threshold	4.97	10	37.86	11	172.72	14
Kahan-Corneil threshold	4.31	9	36.26	10	156.83	14

Table 4: *Comparison of Jacobi methods by blocks when solving the SEP of a special matrix (eigenvalues very close to 1, -1, 2 and -2). On a Pentium.*

REFERENCES

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

[2] C. H. Bischof, *Computing the singular value decomposition on a distributed system of vector processors*, Parallel Computing 11, p. 171 (1989).

[3] M. T. Cámara, D. Giménez, *On the Semiclassical Jacobi Algorithm*, In John G. Lewis, editor, Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, p. 285 (1994).

[4] J. Demmel, K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl. 13, p. 1204 (1992).

[5] A. Edelman, *Large dense linear algebra in 1993: The parallel computing influence*, The International Journal of Supercomputer Applications 7(2), p. 113 (1993).

[6] J. G. F. Francis, *The QR Transformation*, Computer J., p. 265 (1961).

[7] D. Giménez, *A comparison of the solution of the Symmetric Eigenvalue Problem with ScaLAPACK and Jacobi methods*, In Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, (1997).

[8] D. Giménez, V. Hernández, R. van de Geijn, A. M. Vidal, *A Jacobi method by blocks on a mesh of processors*, To appear in Concurrency: Practice and Experience.

[9] 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, p. 51 (1846).

[10] A. H. Karp, J. Greenstadt, *An improved parallel Jacobi method for diagonalizing a symmetric matrix*, Parallel Computing 5, p. 281 (1987).

[11] D. E. Knuth, *The Art of Computer Programming. Vol 3: Sorting and Searching*, Addison-Wesley, (1973).

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

[13] R. Schreiber, *Solving eigenvalue and singular value problems on an undersized systolic array*, SIAM J. Sci. Stat. Comput. 7(2), p. 441 (1986).

[14] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, (1965).