

Diagonalization in Parallel Space

Kenneth Sundberg
kasundberg@cc.usu.edu
Dept. of Computer Science

Dan Watson
Dan.Watson@usu.edu
Dept. of Computer Science

David Farrelly
davidfarrelly@yahoo.com
Dept. of Chemistry

Utah State University
Logan UT, 84322-4205

Abstract

Matrix diagonalization is an important component of many aspects of computational science. There are a variety of algorithms to accomplish this task. Jacobi's algorithm is a good choice for parallel environments.

Jacobi's algorithm consists of a series of matrix plane rotations, the ordering of which can dramatically affect performance. We show a new ordering which cuts the number of necessary operations approximately in half.

Additionally, Jacobi's algorithm can be made parallel in space as well as time. This is an advantage when dealing with very large matrices, such as those found in quantum chemistry.

The eigenvalues and corresponding eigenvectors of a matrix are the key targets of many matrix operations. The efficient computation of these quantities is of great use in many applications, such as signal processing [4], computer graphics [8], and computational chemistry [3], among other fields.

The motivation of this study is the use of this technique to calculate the quantum mechanical wave functions of noble gas complexes by solving the Schrödinger equation using basis set methods. The matrices for problems of this type can grow to be very large, on the order of 10^{12} elements for non-linear molecules and 10^7 for simple linear molecules, so parallelization in space as well as in time is highly desirable.

1 Matrix Diagonalization

There are two major families of matrix diagonalization methods, those based on Jacobi rotations and those based on householder reflections. It has long been noted that while Householder methods are faster in a serial environment [1], Jacobi rotation-based methods have much more inherent parallelism, as well as better numerical properties and, therefore, greater accuracy [5]. Furthermore, there are refinements to the Jacobi algorithm available in a parallel environment that have been largely unused in serial approaches. The careful and selective use of these refinements has the potential to significantly improve the performance of the algorithm over the serial version, and makes the appearance of super-linear speedups for parallel diagonalization possible.

The basic problem in matrix diagonalization is this: given a matrix \mathbf{A} , to find matrices \mathbf{Q} and \mathbf{S} such that

$$\mathbf{A} = \mathbf{Q}\mathbf{S}\mathbf{Q}^{-1} \quad (1)$$

where \mathbf{S} has all zero off-diagonal elements. The diagonal elements of \mathbf{S} are the eigenvalues of \mathbf{A} , and the columns of \mathbf{Q} are the corresponding eigenvectors.

2 Jacobi Rotations

Jacobi's method consists of successive application of plane rotation matrices and their transposes in this fashion:

$$\mathbf{A}_{k+1} = \mathbf{J}_k \mathbf{A}_k \mathbf{J}_k^T \quad (2)$$

$$\lim_{k \rightarrow \infty} \mathbf{A}_k = \mathbf{S} \quad (3)$$

$$\lim_{k \rightarrow \infty} \prod_{n=1}^k \mathbf{J}_n = \mathbf{Q} \quad (4)$$

The matrix multiplications in (4) are postmultiplications. Obviously, it is not practical to carry out an infinite number of rotations, nor is it necessary. Jacobi's method terminates when a sufficiently accurate approximation \mathbf{J}_k to \mathbf{S} is found.

Each rotation matrix \mathbf{J}_k is a plane rotation matrix of the

form:

$$\mathbf{J}_k = \begin{bmatrix} 1 & 0 & \cdot & 0 & \cdot & 0 & \cdot & 0 & 0 \\ 0 & 1 & \cdot & 0 & \cdot & 0 & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cos\theta_{k_{pp}} & \cdot & \sin\theta_{k_{pq}} & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & -\sin\theta_{k_{qp}} & \cdot & \cos\theta_{k_{qq}} & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 0 & \cdot & 0 & \cdot & 1 & 0 \\ 0 & 0 & \cdot & 0 & \cdot & 0 & \cdot & 0 & 1 \end{bmatrix} \quad (5)$$

Here, θ_k is an angle of rotation in one of the subplanes of \mathbf{A}_k chosen to annihilate one pair of off-diagonal elements. The algorithm can arbitrarily select two symmetric elements \mathbf{A}_{pq} and \mathbf{A}_{qp} to annihilate, so long as they are off-diagonal elements. The only restrictions in selecting p and q are

$$p \neq q \quad (6)$$

$$p < n \quad (7)$$

$$q < n \quad (8)$$

The rotation matrix to annihilate \mathbf{A}_{pq} and \mathbf{A}_{qp} is the identity matrix, except for elements $\mathbf{J}_{k_{pp}}, \mathbf{J}_{k_{pq}}, \mathbf{J}_{k_{qp}}, \mathbf{J}_{k_{qq}}$ which will instead be $\sin\theta_k$ and $\cos\theta_k$. Such a rotation is often denoted \mathbf{J}_{pq} indicating which element it will annihilate.

Unfortunately, although each rotation annihilates a pair of off-diagonal elements, subsequent rotations can make these elements non-zero again. Thus, the progress of the algorithm cannot be measured solely in terms of annihilated elements. Instead, the progress of the algorithm is best measured by the quantity σ :

$$\sigma(\mathbf{A}) = \sqrt{\sum_{i=0}^n \sum_{j=0}^n \begin{cases} \mathbf{A}_{ij}^2 & i \neq j \\ 0 & i = j \end{cases}} \quad (9)$$

Even though each rotation may undo a zero created by a previous rotation, it is guaranteed that σ will not increase with any Jacobi rotation [6]. Execution proceeds by selecting an element to annihilate, calculating the proper rotation matrix, performing the multiplication, and repeating until σ is below some desired tolerance.

Two very important properties of the rotation \mathbf{J}_{pq} are (i) postmultiplication by this matrix only affects the p^{th} and q^{th} columns of the matrix, and (ii) premultiplication only affects the p^{th} and q^{th} rows. Any pair of successive columns or rows can be calculated without reference to any other column or row in the \mathbf{A} matrix with the following relations:

$$\mathbf{A}_{k+1_p} = \cos\theta \mathbf{A}_{k_p} - \sin\theta \mathbf{A}_{k_q} \quad (10)$$

$$\mathbf{A}_{k+1_q} = \cos\theta \mathbf{A}_{k_p} + \sin\theta \mathbf{A}_{k_q} \quad (11)$$

where k refers to the iteration, and p refers to a column or row.

Thus, each matrix multiplication can be performed in $\mathbf{O}(n)$ steps. Furthermore, $\frac{n}{2}$ multiplications can be performed in parallel for an $n \times n$ matrix, with the condition that no rows or columns are shared by the parallel computations. Additionally, it is also possible to work backwards from (10) and (11) to calculate the proper θ to annihilate $\mathbf{A}_{k_{pq}}$.

3 Choosing p and q

One choice that remains is the selection of p and q . This choice can be arbitrary, and in traditional implementations of Jacobi's algorithm, it is. There are two major variants of the serial algorithm. The first is one in which p and q are chosen randomly, respecting only constraints (6), (7), and (8). This method works reasonably well. A second choice is a cyclic scheme, wherein possible combinations of p and q are enumerated. The cyclic Jacobi then sweeps through this set of pairings. Both of these methods can waste significant effort, i.e., it takes just as many operations, $\mathbf{O}(n)$ to process a bad pairing as it does to process a good one. However, the effect on σ can be drastically different. Specifically,

$$\Delta\sigma \propto \mathbf{A}_{pq}^2 \quad (12)$$

Furthermore, as the algorithm progresses, there are more and more small elements, meaning that the likelihood of choosing a bad pairing rises during the execution of either of these algorithms.

The ideal suggested by Jacobi [7] (and implied by eq. 12) is to annihilate the largest off-diagonal element. However, this has not been considered practical in serial implementations, as each rotation only requires $\mathbf{O}(n)$ steps, but searching for the largest element requires $\mathbf{O}(n^2)$ steps. This is an unacceptable cost.

The ideal in a parallel implementation is also to annihilate the largest off-diagonal element. However, the other columns can also participate in rotations in parallel to the maximal annihilation rotation. In particular, $\frac{n}{2} - 1$ other rotations are possible, as any pair of columns p and q has a corresponding rotation \mathbf{J}_{pq} . This makes maximum element annihilation feasible.

The parallel algorithm can accomplish the search for the largest element in $\mathbf{O}(n)$ steps for each column, done in parallel followed by $\mathbf{O}(\log n)$ communication steps, which is *the same complexity as the multiplication step*.

This pairing of columns can be scaled to situations wherein $p \ll n$, even back to the serial case. After every $\frac{n}{2p}$ rotations are processed in $\mathbf{O}(\frac{n^2}{p})$ time, another set is calculated by finding the maximum element and pairing all non-participating columns into non-conflicting ro-

tations. Finding such a set of rotations also takes $O(\frac{n^2}{p})$ time.

Depending on the specific enumeration and matrix, random selection may have better or worse performance than a cyclic implementation, but it is statistically similar in performance. For this study, the random selection method was used as a comparison.

Figure 1 shows the advantage gained in required multiplications by using maximum element annihilation versus random element annihilation. The graph plots the progress of the algorithm expressed as $\log(\sigma)$ against the log of the required rotations before convergence. Natural logarithms were used, so each unit along the y-axis represents an approximate doubling of work to be done, and each unit along the x-axis represents an approximate doubling of the steps that it took to accomplish that work.

Each algorithm was used to diagonalize a set of 100 random matrices. After each rotation, σ was calculated and noted together with the number of rotations used to arrive at that point. After each run, when the total number of rotations for a particular matrix was known, the number of rotations associated with each σ value was updated to reflect the required rotations to finish rather than the number of rotations to reach that point. The data was then aggregated by the number of required rotations, and the average and standard deviation of σ was calculated for each these sets of matrices. The largest standard deviations occur at the start of the diagonalization process; this is due to the random nature of the data set. The deviations in σ approach zero as the diagonalization of the matrices progresses.

As any particular diagonalization progresses, it moves from the upper left toward the lower right portion of the graph. Note that both series have initially similar slopes of 2, indicating, as expected that both methods lead to quadratic convergence.

Also note that the maximum element strategy is shifted to the right by an entire unit, indicating that only about half as many rotations are required (the units are expressed as natural logarithms). Also surprising is the way that the maximum element strategy avoids stagnation, indicated by the lack of plateaus in its progression. This is a problem that can plague both the random and cyclic strategies wherein the chosen rotation effects very little change in σ . By selecting the largest element and sweeping through many other elements in parallel, this problem has been largely avoided.

The gains are significant enough that even without a parallel architecture it is advantageous to use the parallel maximum element annihilation algorithm for determining a good ordering of Jacobi rotations.

4 Space and Time Parallelism

Equations 10 and 11 are the key to the parallelism inherent in Jacobi's algorithm. The time parallel aspects have been well explored [2, 6], although in this study we show that there are still some improvements to be made in the details. What has not been fully explored is the utility of the space parallelism of Jacobi's method.

During the postmultiplication steps, only two columns \mathbf{A}_p and \mathbf{A}_q need to be known to calculate the result \mathbf{A}_{k+1} of \mathbf{J}_{pq} . This means that no processing node ever needs to hold the entire matrix, only columns \mathbf{A}_p and \mathbf{A}_q and, if

desired, their corresponding partial eigenvectors $\prod_1^k \mathbf{J}_{k_p}$ and $\prod_1^k \mathbf{J}_{k_q}$.

It might seem that the costly step of spreading and gathering columns into rows to perform the premultiplication step of the algorithm would be difficult to overcome. However, this is not necessary. Instead, all that needs to be gathered is a column consisting of θ_{pq} and the p and q to which the angle corresponds. This data is all that is spread to all nodes. With this angle information, each column can independently perform one of the n calculations for each premultiplication. This step is made for all of the elements in the column. In aggregate, this results in n processors performing the premultiplication step in $O(n)$ time without a costly transpose step.

5 Conclusion

Jacobi's method is tremendously useful in the realm of computational science. Along with other matrix diagonalization methods, it provides the means to analyze many important systems.

Jacobi's method is composed of a series of planar rotations. Traditionally, these rotations are calculated in a cyclic or a random fashion. We have shown a more effective ordering of calculations that can reduce the number of rotations by approximately a factor of two.

This ordering, which exploits the content of the matrix, can, along with Jacobi's method, be effectively and efficiently parallelized in time and also in space. This parallelization is critical, as the matrices we wish to diagonalize are typically very large.

Another key aspect that makes Jacobi's method attractive is that it is well conditioned and, thus, introduces less error than other methods. In future work, we would like to explore adding Davidson preconditioning [9] to the matrices to further speed up the diagonalization process.

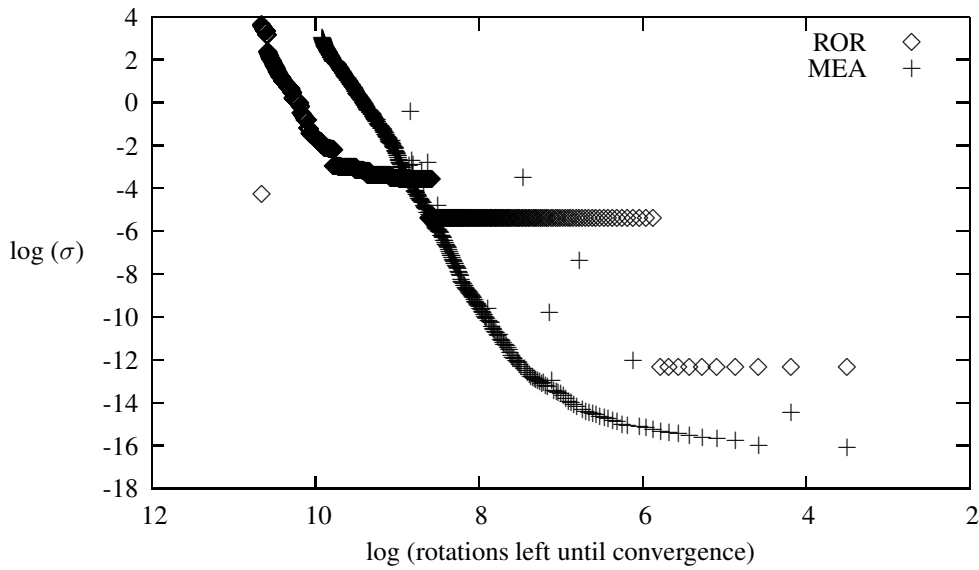


Figure 1: Maximum element annihilation vs. random element annihilation. The graph compares the results of using a random ordering of rotations (ROR) with using a set of rotations containing the maximum element annihilation rotation (MEA). Along the ordinate is plotted $\log(\sigma)$: σ is a measure of the quality of intermediate solutions defined in eq. 9. The lower the value of $\log(\sigma)$ the better the solution: a perfect solution would have a $\log(\sigma)$ of $-\infty$. In practice some threshold value is chosen as a termination point for the algorithm. Along the abscissa is plotted $\log(\text{rotations to convergence})$: each rotation performed moves a data point farther to the right of the graph. Of note are the long plateau areas in the random orderings progression, these are caused by selecting poor rotations. Since each set of rotations in the maximum element annihilation method is guaranteed to contain at least one good rotation these plateaus are not present for this method.

References

- [1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LA-PACK User's Guide*. SIAM, 2nd edition, September 1994.
- [2] U. Burnik, G. Cain, and J. Tasic. On the parallel Jacobi method based eigenfilter. *COST 229 WG4*, 1993.
- [3] D. C. Clary and D. J. Nesbitt. Calculation of vibration-rotation spectra for rare gas-HCl complexes. *J. Chem. Phys.*, 90(12):7000–7013, June 1989.
- [4] P. Comon. Tensor diagonalization, a useful tool in signal processing. *IFAC SYSID*, 1:77–82, 1994.
- [5] Z. Drmac. A posteriori computation of the singular vectors in a preconditioned Jacobi SVD algorithm. *IMA J. of Num. Anal.*, 19(2):191–213, 1999.
- [6] J. Gotze, S. Paul, and M. Sauer. An efficient Jacobi-like algorithm for parallel eigenvalue computation. *IEEE Trans. of Comp.*, 42(9):1058–1065, September 1993.
- [7] C. G. J. Jacobi. Ueber ein leichtes verfahren, die in der theorie der sacularstorungen vorkommenden gleichungen numersch aufzulosen. *J. Reine Angew. Math.*, pages 51–94, 1846.
- [8] K. Shoemake and T. Duff. Matrix animation and polar decomposition. In *Proceedings of the Graphics Interface Conference 1992*, pages 258–264. Conference on Graphics Interface, September 1992.
- [9] G. L. G. Sleijpen and H. A. Van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Review*, 42(2):267–293, 2000.