A Parallel Solver for Extreme Eigenpairs[1]

Leonardo Borges and Suely Oliveira[2]

Computer Science Department, Texas A&M University, College Station, TX 77843-3112, USA.

Abstract. In this paper a parallel algorithm for finding a group of extreme eigenvalues is presented. The algorithm is based on the well known Davidson method for finding one eigenvalue of a matrix. Here we incorporate knowledge about the structure of the subspace through the use of an arrowhead solver which allows more parallelization in both the original Davidson and our new version. In our numerical results various preconditioners (diagonal, multigrid and ADI) are compared. The performance results presented are for the Paragon but our implementation is portable to machines which provide MPI and BLAS.

1 Introduction

A large number of scientific applications rely on the computation of a few eigenvalues for a given matrix A. Typically they require the lowest or highest eigenvalues. Our algorithm (DSE) is based on the Davidson algorithm, but calculates various eigenvalues through implicit shifting. DSE was first presented in [10] under the name RDME to express its ability to identify eigenvalues with multiplicity bigger than one. The choice of preconditioner is an important issue in eliminating convergence to the wrong eigenvalue [14] In the next section, we describe the Davidson algorithm and our version for computing several eigenvalues. In [9] Oliveira presented convergence rates for Davidson type algorithm dependent on the type of preconditioner. These results are summarized here in Section 3. Section 4 addresses parallelization strategies discussing the data distribution in a MIMD architecture, and a fast solver for the projected subspace eigenproblem. In Section 5 we present numerical and performance results for the parallel implementation on the Paragon. Further results about the parallel algorithm and other numerical results are presented in [2].

2 The Davidson Algorithm

Two of the most popular iterative methods for large symmetric eigenvalue problems are Lanczos and Davidson algorithms. Both methods solve the eigenvalue

[1] This research is supported by NSF grant ASC 9528912 and a Texas A&M University Interdisciplinary Research Initiative Award.

[2] Department of Computer Science, Texas A&M University, College Station, TX 77843.
email: suely@cs.tamu.edu.

problem $Au = \lambda u$ by constructing an orthonormal basis $V_k = [v_1, \ldots, v_k]$, at each k^{th} iteration step, and then finding an approximation for the eigenvector u of A by using a vector u_k from the subspace spanned by V_k. Specifically, the original problem is projected onto the subspace which reduces the problem to a smaller eigenproblem $S_k y = \tilde{\lambda} y_k$, where $S_k = V_k^T A V_k$. Then the eigenpair $(\tilde{\lambda}_k, y_k)$ can be obtained by applying a efficient procedure for small matrices. To complete the iteration, the eigenvector y_k is mapped back as $u_k = V_k y_k$, which is an approximation to the eigenvector u of the original problem. The difference between the two algorithms consists on the way that basis V_k is built. The attractiveness of the Lanczos algorithm results from the fact that each projected matrix S_k is tridiagonal. Unfortunately, sometimes this method may require a large number of iterations. The Davidson algorithm defines a dense matrix S_k on the subspace, but since we can incorporate a preconditioner in this algorithm the number of iterations can be much lower than for Lanczos. In Davidson type algorithms, a preconditioner M_{λ_k} is applied to the current residual, $r_k = Au_k - \tilde{\lambda}_k u_k$, and the preconditioned residual $t_k = M_{\lambda_k} r_k$ is orthonormalized against the previous columns of $V_k = [v_1, v_2, \ldots, v_k]$. Although in the original formulation M_λ is the diagonal matrix $(diag(A) - \lambda I)^{-1}$ [6], the Generalized Davidson (GD) algorithm allows the incorporation of different operators for M_λ. The DSE algorithm can be summarized as follows.

Algorithm 1 – Restarted Davidson for Several Eigenvalues *Given a matrix A, a normalized vector v_1, number of eigenpairs p, restart index q, and the minimal dimension m for the projected matrix S $(m > p)$, compute approximations λ and u for the p smallest eigenpairs of A.*

1. *Set* $V_1 \leftarrow [v_1]$. *(initial guess)*
2. *For* $j = 1, \ldots, p$ *(approximation for j-th eigenpair)*
   While $k = 1$ *or* $\|r_{k-1}\| < \epsilon$ *do*
   (a) *Project* $S_k = V_k^T A V_k$.
   (b) *If* $(m + q) \leq dim\ S$ *(restart* S_k*)*
   Reduce $S_k \leftarrow (\Lambda_k)_{(m \times m)}$ *to its m smaller eigenvectors, and update* V_k *for the new basis.*
   (c) *Compute the* j^{th} *smallest eigenpair* λ_k, y_k *of* S_k.
   (d) *Compute the Ritz vector* $u_k \leftarrow V_k y_k$.
   (e) *Check convergence for* $r_k \leftarrow Au_k - \lambda_k u_k$.
   (f) *Apply preconditioner* $t_k \leftarrow M r_k$.
   (g) *Expand basis* $V_{k+1} \leftarrow [V_k, t_k]$ *using modified Gram Schmidt (MGS).*
   End while
3. *End For.*

The core ideas of DSE (Algorithm 1) are based on the projection of A into the subspace spanned by the columns of V_k. The interation number k is not necessarily equal to $dim\ S_k$, since we have incorporated implicit restarts. The matrix S_k is obtained by adding one more column and row $V_k^T A v_k$ to matrix S_{k-1} (step **2.a**). Other important aspects of the DSE algorithm are:

(1) the eigenvalue solver for the subspace matrix S_k (step **2.c**); (2) the use of an auxiliary matrix $W_k = [w_1, \ldots, w_k]$ to provide a residual calculation $r_k = Au_k - \lambda_k u_k = w_k y_k - \lambda_k u_k$ with less computational work (step **2.e**); the choice of a preconditioner M (step **2.f**); and the use of *modified* Gram-Schmidt orthonormalization (step **2.g**) which preserves numerical stability when updating the orthonormal basis V_{k+1}. At each iteration, the algorithm expands the matrix S either until all the first p eigenvalues have been converged, or S reaches a maximum dimension $m + q$; In the latter case, restarting is applied by using the orthonormal decomposition $S_k = Y_k^T \Lambda_k Y_k$ of S. It corresponds to step **2.b** in the algorithm. Because of our choice for m, note that in step **2.c** *dim* S will be always bigger or equal to j.

3 Convergence Rate

A proof of convergence (but without a rate estimate) for the Davidson algorithm is given in Crouzeix, Philippe and Sadkane [5]. A bound on the convergence rate was first presented in [10]. The complete proof is shown in Oliveira [9]. Let A be the given matrix whose eigenvalues and eigenvectors are wanted. The preconditioner M is given for one step, and Davidson's algorithm is used with u_k being the current computed approximate eigenvector. The current eigenvalue estimate is the Rayleigh quotient $\widetilde{\lambda}_k = \rho_A(u_k) = (u_k^T A u_k)/(u_k^T u_k)$. Let the exact eigenvector with the smallest eigenvalue of A be u, and

$$Au = \lambda u.$$

(If λ is a repeated eigenvalue of A, then we can let u be the normalized projection of u_k onto this eigenspace.)

Theorem 1. *Let P be the orthogonal projection onto $ker(A - \lambda I)^{\perp}$. Suppose that A and M are symmetric positive definite. If*

$$\|P - PMP(A - \lambda I)\|_2 \leq \sigma < 1,$$

then for almost any starting value x_1, the convergence of the eigenvalue estimates $\widehat{\lambda}_k$ converge to λ ultimately geometrically with convergence factor bounded by σ^2, and the angle between the computed eigenvector and the exact eigenspace goes to zero ultimately geometrically with convergence factor bounded by σ.

A geometric convergence rate can be found for DSE (which obtains eigenvalues beyond the smallest (or largest) eigenvalue) by modifying Theorem 1. In the following theorem assume that

$$\sigma' = \|P' - P'MP'\,P'(A - \lambda_p I)P'\|_2$$

where P' is the orthogonal projection onto the orthogonal complement of the span of the first $p - 1$ eigenvectors. Then we can shown, in a similar way to Theorem 1 that the convergence factor for the new algorithm is bounded by

$(\sigma')^2$ To prove Theorem 2 we use the fact that $P's_k = s_k$, as s_k is orthogonal to the bottom p eigenvectors. and that although $(A - \lambda_p I)$ is no longer positive semi-definite, $P'(A - \lambda_p I)P'$ is.

Theorem 2. *Suppose that A and M are symmetric positive definite and that the first $p-1$ eigenvectors have been found exactly. Let P' be the orthogonal projection onto the orthogonal complement of the span of the first $p-1$ eigenvectors of A. If*

$$\|P' - P'MP'(A - \lambda_p I)P'\|_2 \leq \sigma' < 1,$$

then for almost any starting value x_1, the eigenvalue estimates $\widehat{\lambda}_k$ obtained by our modified Davidson algorithm for several eigenvalues *converges to λ_p ultimately geometrically with convergence factor is bounded by $(\sigma')^2$, and the angle between the exact and computed eigenvector goes to zero ultimately geometrically with convergence factor bounded by σ'.*

4 Parallel Implementation

Previous implementations for the Davidson algorithm solve the eigenvalue problem in subspace S by using algorithms for dense matrices: early works [3, 4, 17] adopt EISPACK [12] routines, and later implementations [13, 15] use LAPACK [1] or reductions to tridiagonal form. Partial parallelization is obtained through the matrix-vector operations and sparse format storage for matrix A [13, 15]. Here we explore the relationship between two successive matrices S_k which allows us to represent S_k through an arrowhead matrix. The arrowhead structure is extremely sparse and the associated eigenvalue problem can be solved by a highly parallelizable method.

4.1 Data Distribution

Data partitioning significantly affects the performance of a parallel system by determining the actual degree of concurrency of the processors. Matrices are partitioned along distinct processors so that the program exploits all the best possible data parallelism: The final distribution is well balanced, and most of the computational work can be performed without communication. These two conditions make the parallel program very suited for distributed memory architectures. Both computational workload and storage requirements are the same for all processors. Communication overhead is kept as low as possible. Matrix A is split into row blocks A^i, $i = 1, \ldots, N$, each one containing $\leq \lceil n/N \rceil$ rows of A. Thus processor i, $i = 1, \ldots, N$ stores A^i, the i^{th} row block of A. Matrices V_k and W_k are stored in the same fashion. This data distribution allow us to perform many of the matrix-vector computations in place.

The orthonormalization strategy is also an important aspect in parallel environments. Recall that the modified Gram Schmidt (MGS) algorithm will be applied to the extended matrix $[V_k, t_k]$ where the current basis V_k has been previously orthonormalized. This observation reduces the computational work by eliminating the outer loop from the two nested loops in the full MGS algorithm.

4.2 The Arrowhead Relationship Between Matrices S_k

As pointed in [2, 10], the relationship between S_k and S_{k-1} can be used to show that S_k is explicitly similar to an arrowhead matrix $\tilde{S}_k$ of the form

$$\tilde{S}_k = \begin{bmatrix} \Lambda_{k-1} & \tilde{s}_k \\ \tilde{s}_k^T & s_{kk} \end{bmatrix} , \tag{1}$$

where $\tilde{s}_k = Y_{k-1}^T V_{k-1}^T w_k$, $s_{kk} = v_k^T w_k$, and the diagonal matrix Λ_{k-1} corresponds to the orthonormal decomposition $S_{k-1} = Y_{k-1}\Lambda_{k-1}Y_{k-1}^T$. In practice, the matrix S_k does not need to be stored: only a vector for Λ_k and a matrix for Y_k are required from one iteration to the next. Thus, given the eigenvalues Λ_{k-1} and eigenvectors Y_{k-1} of S_{k-1}, matrix $\tilde{S}_k$ can be used to find the eigenvalues Λ_k of S_k. Arrowhead eigensolvers [8, 11] are highly parallelizable and typically perform $\mathcal{O}(k^2)$ operations, instead of the usual $\mathcal{O}(k^3)$ effort of algorithms for dense matrices S.

5 Numerical Results

In our numerical results we employ three kind of preconditiners: diagonal preconditioner (as in the original Davidson), multigrid and ADI. A preconditioner can be expressed as the matrix which solves $Ax = b$ by applying an iterative method to $MAx = Mb$ instead. In the case of a Diagonal preconditioner this would correspond to scaling the system and then solving. Multigrid and ADI preconditioners are more complex and for that we refer the reader to [16, 18, 19]. In our implementation level 1, 2 and 3 BLAS and the Message Passing Interface (MPI) library were used for easy portability.

The computational results in this section were obtained with a finite difference approximation for

$$-\Delta u + gu = f \tag{2}$$

on a unit square domain. Here g is null inside a 0.2×0.2 square on the center of the rectangle and $g = 100$ for the remaining domain.

To compare the performance delivered by distinct preconditioners we observe the total timing and number of iterations required for the sequential DSE for finding the ten smallest eigenpairs ($p = 10$) assuming convergence for residual norms less or equal to 10^{-7}. The restart indexes were $q = 10$ and $m = 15$. This corresponds to apply restarting every time that the projected matrix S_k achieves order 25, reducing its order to 15. Table 1 presents the actual running tIMINGS In a single processor of the Intel Paragon, running three grid sizes: 31×31, 63×63, and 127×127 (matrices of orders 961, 3969 and 16129, respectively.). It reflects the tradeoff between preconditioning strategies: although the diagonal preconditioner (DIAG) is the easiest and fastest to compute, it requires an increasing number of iterations for larger matrices. Multigrid preconditioners (MG) are more expensive than DIAG, but they turn to be more effective for larger matrices. Finally, the ADI method aggregate the advantages of the previous preconditioners in the sense that it is more effective and less expensive than

MG. More details about the preconditioners used here can be found in [2] and its references.

Table 1. Sequential times and number of iterations for three preconditioners.

	matrix order 961		matrix order 3969		matrix order 16129	
	iterations	time (sec)	iterations	time (sec)	iterations	time (sec)
ADI	29	4.7	26	16.2	27	80.7
MG	34	8.8	40	43.0	40	254.1
DIAG	174	8.6	319	43.2	700	386.0

The overall behavior of the DSE algorithm (with a multigrid preconditioner) is shown in Figure 1 for matrices sizes 3969 and 16129, as a function of the number of processors. Note that the estimated optimal number of processors is not far from the actual optimal. The model for our estimates is presented in [2].

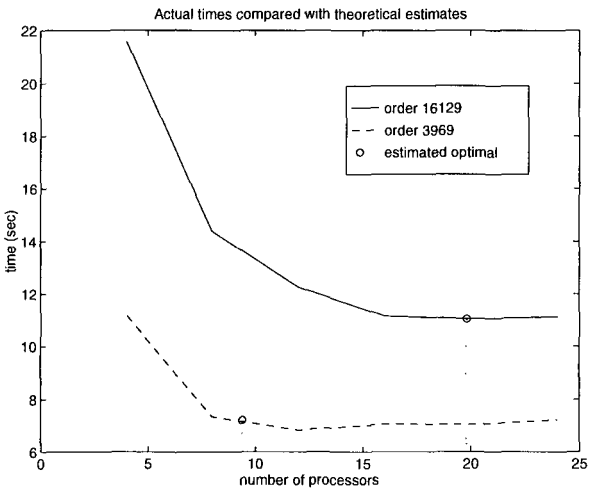

Fig. 1. Actual and estimated times for equation (2). Results for two different matrix sizes performance on the Paragon are shown.

To conclude, we compare the performance of the parallel DSE with PARPACK [7], a parallel implementation of ARPACK [1]. Figure 2 presents the total running times for both algorithms for the problem described above. For these runs, DSE used our parallel implementation of the ADI as its preconditioner.

[1] ARPACK implements an Implicitly Restarted Arnoldi Method (IRAM) which in the symmetric case corresponds to the Implicitly Restarted Lanczos algorithm. We used the regular mode when running PARPACK.

The problem was solved by using 4, 8, and 16 processors to obtain relative residuals $||Au-\lambda u||/||u||$ of order less than 10^{-5}. We show our theoretical analysis for the parallel algorithm in [2]. Other numerical results for the sequential DSE algorithm, including examples showing the behavior of the algorithm for eigenvalues with multiplicity greater than one, were presented in [10].

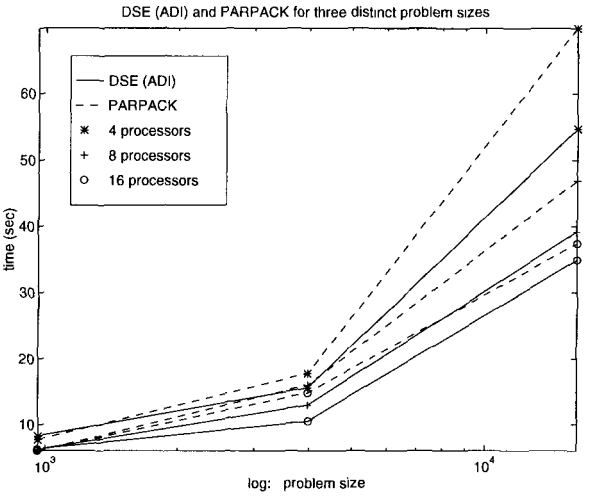

Fig. 2. Running times for DSE and PARPACK using 4, 8, and 16 processors on the Paragon.

References

1. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, 1992.
2. L. Borges and S. Oliveira. A parallel Davidson-type algorithm for several eigenvalues. *Journal of Computational Physics*. To appear.
3. G. Cisneros, M. Berrondo, and C. F. Brunge. DVDSON: A subroutine to evaluate selected sets of eigenvalues and eigenvectors of large symmetric matrices. *Compu. Chem.*, 10:281–291, 1986.
4. G. Cisneros and C. F. Brunge. An improved computer program for eigenvector and eigenvalues of large configuration iteraction matrices using the algorithm of Davidson. *Compu. Chem.*, 8:157–160, 1984.
5. M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.*, 15(1):62–76, 1994.
6. E. R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comp. Phys.*, 17:87–94, 1975.

7. K. Maschhoff and D. Sorensen. A portable implementation of ARPACK for distributed memory parallel architectures. In *Proceedings of Copper Mountain Conference on Iterative Methods*, April 9–13 1996.
8. D. P. O'Leary and G. W. Stewart. Computing the eigenvalues and eigenvectors of symmetric arrowhead matrices. *J. Comp. Phys.*, 90:497–505, 1990.
9. S. Oliveira. On the convergence rate of a preconditioned algorithm for eigenvalue problems. Submitted.
10. S. Oliveira. A convergence proof of an iterative subspace method for eigenvalues problem. In F. Cucker and M. Shub, editors, *Foundations of Computational Mathematics Selected Papers*, pages 316–325. Springer, January 1997. (selected).
11. S. Oliveira. A new parallel chasing algorithm for transforming arrowhead matrices to tridiagonal form. *Mathematics of Computation*, 67(221):221–235, January 1998.
12. B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix eigensystem routines: EISPACK guide.* Number 6 in Lecture Notes Comput. Sci. Springer–Verlag, Berlin, Heidelberg, New York, second edition, 1976.
13. A. Stathopoulos and C. F. Fischer. A Davidson program for finding a few selected extreme eigeinpairs of a large, sparse, real, symmetric matrix. *Comp. Phys. Comm.*, 79:268–290, 1994.
14. A. Stathopoulos, Y. Saad, and C. F. Fisher. Robust preconditioning of large, sparse, symmetric eigenvalue problems. *J. Comp. and Appl. Mathematics*, 64:197–215, 1995.
15. V. M. Umar and C. F. Fischer. Multitasking the Davidson algorithm for the large, sparse eigenvalue problem. *Int. J. Supercomput. Appl.*, 3:28–53, 1989.
16. R. S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, 1963.
17. J. Weber, R. Lacroix, and G. Wanner. The eigenvalue problem in configuration iteration calculations: A computer program based on a new derivation of the algorithm of Davidson. *Compu. Chem.*, 4:55–60, 1980.
18. J. R. Westlake. *A handbook of numerical matrix inversion and solution of linear equations.* Wiley, 1968.
19. D. M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, 1971.