

Time-Memory Trade-offs Using Sparse Matrix Methods For Large-Scale Eigenvalue Problems ^{*}

Keita Teranishi¹, Padma Raghavan¹, and Chao Yang²

¹ Department of Computer Science and Engineering, The Pennsylvania State University, 220 Pond Lab, University Park, PA 16802-6106. E-mail

{`teranish,raghavan`}@`cse.psu.edu`.

² Lawrence Berkeley National Laboratory Cyclotron Rd, Berkeley, CA 94720. E-mail `cyang@lbl.gov`.

Abstract. Iterative methods such as Lanczos and Jacobi-Davidson are typically used to compute a small number of eigenvalues and eigenvectors of a sparse matrix. However, these methods are not effective in certain large-scale applications, for example, “global tight binding molecular dynamics.” Such applications require all the eigenvectors of a large sparse matrix; the eigenvectors can be computed a few at a time and discarded after a simple update step in the modeling process. We show that by using sparse matrix methods, a direct-iterative hybrid scheme can significantly reduce memory requirements while requiring less computational time than a banded direct scheme. Our method also allows a more scalable parallel formulation for eigenvector computation through spectrum slicing. We discuss our method and provide empirical results for a wide variety of sparse matrix test problems.

1 Introduction

Consider the solution of the standard eigenvalue problem $Ax = \lambda v$ where A is sparse and symmetric positive definite. Iterative methods such as Lanczos and Jacobi-Davidson are typically used to compute a small number of eigenvalues and eigenvectors efficiently [2, 6, 10, 11, 24, 25]. However, certain molecular dynamics applications using a *Global Tight Binding* model [17, 18], require all the eigenvalues and eigenvectors of a sparse matrix A whose dimension (N) is proportional to the number of atoms in the simulation. For models of interest with several thousand atoms, the eigenvector computations are main limitation. Using pure iterative Lanczos type methods is not feasible (due to poor convergence) in such applications. The computations are typically performed using dense direct methods [2] which have $O(N^3)$ time and $O(N^2)$ space complexity. The latter can be a limiting factor since all eigenvectors are not needed all at once.

In general, direct methods start by converting the original matrix into tridiagonal form. The eigenvalues and eigenvectors of the tridiagonal form are then

^{*} This work was supported in part by the National Science Foundation through grants ACI-0102537, EIA-022191, and DMR-0205232.

computed using very efficient techniques. The eigenvalues of the tridiagonal matrix are the eigenvalues of the original matrix. The eigenvectors of the original matrix are computed by multiplying the eigenvectors of the tridiagonal matrix by the orthogonal matrix of transforms used in the first step. For a detailed discussion and overview of eigensolution techniques, two excellent references are the books by Demmel and Parlett [6, 22]. Good serial and parallel implementations exist in the form of packages LAPACK [1] and ScaLAPACK [4]. In such a direct method, the eigenvector computation is the computationally expensive part; the eigenvalue computations typically require only a small fraction of the time and memory.

In this paper, we consider a hybrid of direct and iterative methods for reducing the total time and memory requirements for computing all the eigenvalues and eigenvectors of sparse matrices. Our method uses a direct band method to compute all the eigenvalues; now the dense orthogonal factor is not computed and stored. Instead, the eigenvalues are used in a shift-and-invert Lanczos method with the original sparse matrix A to compute the eigenvectors. This step is based on *spectrum slicing* and has the potential to provide a scalable parallel implementation.

This paper is organized as follows. We provide a brief overview of direct methods and we introduce a primitive form of our scheme in Section 2. We describe our hybrid approach with shift-invert Lanczos [2, 25] in Section 3 and report on its performance on a large suite of sparse matrices in Section 4. We end with some concluding remarks in Section 5.

2 Overview Of Computational Schemes

In this section we provide a brief overview of direct dense and band methods and discuss how eigenvectors can be computed using inverse iteration on the original sparse matrix.

Dense Methods. Direct dense methods are typically used when a relatively large number of eigenpairs are to be computed. In general, the solution comprises the following three main steps.

1. Compute a tridiagonal matrix T from A using orthogonal transformations. The orthogonal factor, Q , is also computed where $A = QTQ^T$.
2. Compute all eigenvalues of T ; these are the eigenvalues of A .
3. Compute eigenvectors of T ; use Q to compute eigenvectors of A .

Sophisticated schemes exist for eigenvalue computations of the tridiagonal matrix T ; for example, $O(N^2)$ schemes to compute all eigenvalues and eigenvectors [7, 12]. The tridiagonalization of A is done through Givens or Householder transformations [2, 6, 10] at a cost of $O(N^3)$. There are four different methods available in LAPACK [1] and currently, the ‘‘Relatively Robust Representation’’ is regarded as the best method among these [1, 2, 6, 7, 22].

Our schemes concern the last part of Step 3 which requires $O(N^2)$ space and $O(N^3)$ time to compute the eigenvectors of A using Q .

Band-Tridiagonalization Methods. When the matrix is sparse, the tridiagonalization step, using for example, Householder transformations, destroys the sparsity of the original matrix. To overcome this problem, various researchers have proposed band-tridiagonalization after reordering the original sparse matrix to band form using a profile-minimization scheme such as Reverse Cuthill-McKee (RCM) scheme [8]. To preserve the band form, the off-tridiagonal elements are reduced element-wise using Givens transformation [14], or block-wise by Householder transformation [3] (with some extra memory overheads). These methods incrementally reduce the bandwidth until the matrix becomes tridiagonal. The benefits of the band methods are their lower memory requirements and computational costs. However, methods like element-wise tridiagonalization are usually slower than the dense method because of poor utilization of the cache hierarchy. The block-wise method addresses this issue but nonetheless, a band method must necessarily be less efficient than a dense method with respect to cache-handling.

Computing Eigenvectors Using Sparse Inverse Iteration. If only eigenvalues are required using either the band or the dense method, then the amount of computation is substantially reduced because there is no longer the need to compute and apply the orthogonal matrix Q . Another observation is that if the eigenvalues are available, then eigenvectors can be computed with the original matrix. Although this process is fraught with problems of reorthogonalization of eigenvectors, it benefits from reduced storage because Q is not computed. Sparse inverse iteration proceeds by using the computed eigenvalues with the sparse LDL^T factors of the original matrix A ; the latter can be computed using standard sparse matrix methods [8, 9]. Figure 1 describes a basic algorithm using this approach; to reduce memory requirement for eigenvalue calculations, we use band-tridiagonalization algorithm.

1. Use a band ordering to construct $B = PAP^T$
2. Tridiagonalize B to obtain T (do not construct Q)
3. Compute eigenvalues of A (T)
4. Find eigenvectors by sparse inverse iteration using the sparse factorization $(A - \lambda_i I) = LDL^T$. This factorization utilizes better fill-reducing orderings than a band scheme; for example, a multiple minimum degree or nested dissection orderings [13, 15, 20].

Fig. 1. Computing eigenvalues and eigenvectors using band tridiagonalization and sparse inverse iteration.

Ideally, inverse iteration requires only repeated linear system solutions of the form $(A - \lambda_i I)x = y$. However, for large matrices, re-orthogonalization for each eigenvector is usually required. This can dramatically affect the performance of sparse inverse iteration. To reduce reorthogonalization, we use the scheme proposed by J. L. Barlow [5]. The inverse iteration is restarted using a suitable canonical vector e_j . A vector s is computed using $(A - \lambda_i I)s = r$ with a random

r . The component j corresponds to the location of the largest element of s and inverse iterations continue with $(A - \lambda_i I)x = e_j$. We found that this technique improves the performance substantially except in the instances when matrix is very highly ill-conditioned. This is the scheme we implement and report on.

3 A Direct-Iterative Hybrid Using Shift-and-Invert Lanczos

The simple inverse iteration scheme suffers from the fact that a new LDL^T factorization has to be computed for each eigenvalue. Even for sparse matrices with factorization costs that grow as $O(N^{1.5})$, the cost of computing all N eigenvectors would be $O(N^{2.5})$. This is less expensive than the dense method requiring $O(N^4)$ operations but more expensive than for a tridiagonal matrix (cost $O(N^2)$). The naive inverse iteration approach can be improved by using the shift-and-invert Lanczos method [25] to compute eigenvectors corresponding to many eigenvalues near the shift. This would dramatically reduce the number of sparse LDL^T decompositions. Our hybrid scheme utilizes this approach and is shown in Figure 2. The main problem is one of selecting the shift points; since all the eigenvalues are known, we typically pick small clusters of 10–100 eigenvalues and use the mean as the shift.

1. Use a band ordering to construct $B = PAP^T$
2. Tridiagonalize B to obtain T (do not construct Q)
3. Compute eigenvalues of A (T)
4. Group eigenvalues into small clusters, and determine a shift value for each cluster
5. Compute eigenvectors for each cluster by shift-and-invert Lanczos, with an appropriate sparse LDL^T factorization

Fig. 2. Computing eigenvalues using a direct band method and eigenvectors using sparse shift-and-invert Lanczos.

4 Empirical Results

We study the performance of our methods using four classes of sparse matrices. The test suite has six matrices from each class (a total of 24) ranging in dimension from 814 to 6000. The `bcsstk` collection represents six finite element matrices from Matrix Market [19], the `struc` collection is from structural mechanics, `image` contains six image analysis matrices, and `xerox` represents matrices from colloidal analysis. Results over the entire collection are labeled `all` in subsequent figures.

We use band-tridiagonalization routines in LAPACK [1], and the Lanczos method in ARPACK [16], and sparse LDL^T factorization in DSCPACK [23].

To establish the base performance of the direct method, we measure the performance of DSYEVR and its band version, which implement the RRR method [7, 22]. The performance of our methods are compared with the performance of these methods (which are considered to be the best). All the methods were tested with 500 MHz Intel Pentium III workstations.

Figure 3 contains two subplots, each containing five bars, one for each type of matrix and one over the entire collection. The plot at left shows the average percentage of time spent for tridiagonalization, computing eigenvalues, and eigenvectors using the dense scheme. The plot at right shows the same three quantities for the band scheme. However, the values are now normalized with respect to the dense scheme (set at 1). Observe that the band methods require approximately 2.2 times the time required for the dense method.

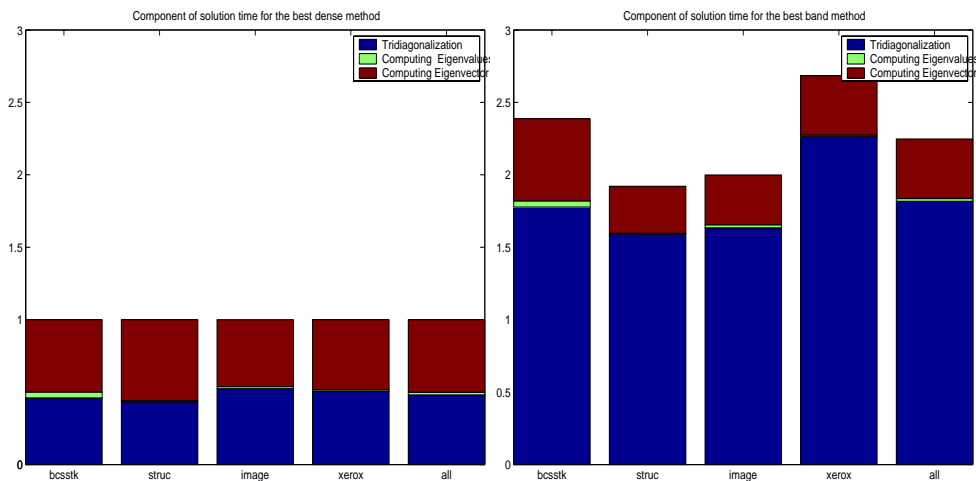


Fig. 3. Component-wise execution times of the best dense and band methods; the latter are normalized relative to the dense method (set at 1).

Figure 4 shows the performance of the sparse scheme of Figure 1 in detail. The subplot at left shows the execution time relative to both dense and band methods. The sparse scheme requires on average 2.55 times the time required by the dense method, and it is on average 13% slower than the band scheme. The plot at right shows the memory requirements relative to both the dense and band schemes; on average, the sparse method requires only 10% (9%) of the memory required by the dense scheme (band scheme).

The performance of our hybrid scheme is shown in Figure 5, once again in the form of two subplots. The subplot at left shows the execution time relative to both dense and band methods. Our hybrid scheme requires on average 1.37 times the time required by the dense method. That is, average, it is only 37% slower than the dense scheme. Additionally, it is (on average) 39% faster than

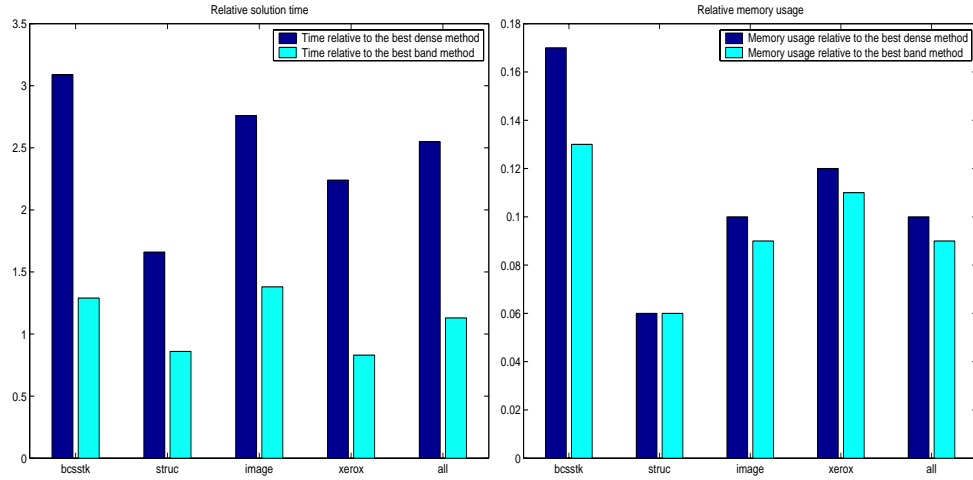


Fig. 4. The performance of the sparse inverse iteration using LDL^T factors of A relative to the best dense and band methods.

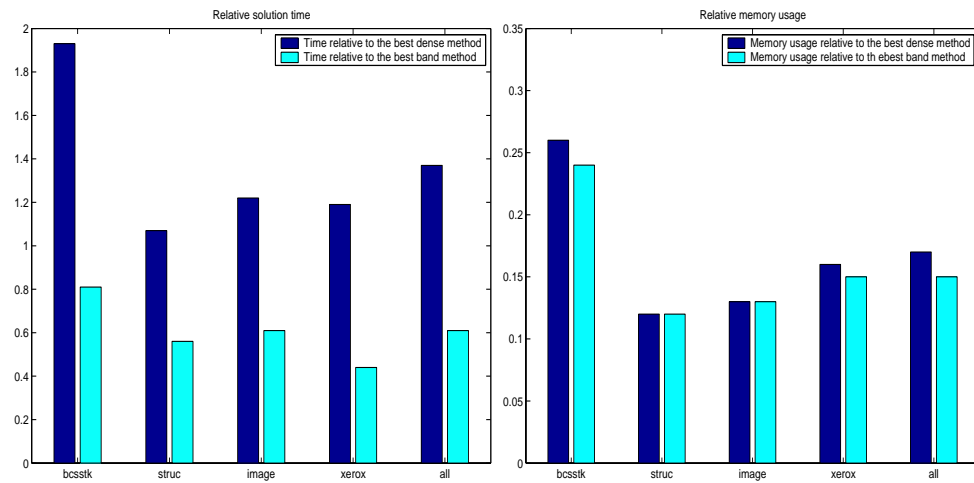


Fig. 5. The performance of our hybrid method relative to the best dense and band methods.

the band scheme. Its memory requirements are substantially lower too, at 17% of the dense scheme and 15% of the band scheme. The slight increase in memory requirements compared to the simple sparse scheme comes from the workspace requirements of the shift-invert Lanczos scheme (for reorthogonalization). These results indicate that our method is feasible for our applications of interest; it allows the solution of larger problems at execution times competitive with the best dense schemes.

5 Conclusions

Our direct-iterative hybrid approach combines the strengths of both classes of methods for the type of eigenvector computations required in large-scale molecular dynamics models. On average, for problems in our test-suite, the execution time of our method is slower only by 37% when compared to the dense method. It is quite surprising that the penalty is indeed so small. Although our method utilizes the cache-hierarchy effectively through sparse factorization, the underlying dense submatrices are relatively small preventing it from achieving the execution rates of the fully dense method. Interestingly enough, our method on average, requires only 41% of the time required by the band method. The main improvements relate to memory requirements; on average, our method requires less than 17% of the memory requirements of either band or dense methods.

Another benefit of our scheme lies in the natural parallelization of the eigenvector computation; the latter could proceed independently and in parallel for disjoint ranges (clusters) of eigenvalues. This could potentially reduce execution time significantly, because the dense parallel scheme incurs substantial inter-processor communication overheads for reorthogonalization during eigenvector computation. We plan to provide parallel execution times for large systems from our molecular dynamics applications [17, 18].

6 Acknowledgments

We gratefully acknowledge several useful discussions with J. L. Barlow at the Pennsylvania State University and R. C. Ward and W. Gansterer at the University of Tennessee.

References

1. E. ANDERSON, Z. BAI, C. BISCHOF, S. BLACKFORD, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORENSEN, *LAPACK User's Guide, third edition*. SIAM, Philadelphia, 1999.
2. Z. BAI, J. DEMMEL, J. DONGARRA, A. RUHE, AND H. VAN DER VORST, EDITORS. *Templates for the Solution of Algebraic Eigenvalue Problems: A practical Guide* SIAM, Philadelphia, 2000.
3. C. H. BISCHOF, B. LANG AND X. SUN, *A framework for symmetric band reduction*. ACM Transactions on Mathematical Software Vol. 26 no. 4 pp. 581–601, 2000.

4. L. S. BLACKFORD, J. CHOI, A. CLEARY, E. D'AZEVEDO, J. DEMMEL, I. DHILLON, J. DONGARRA, S. HAMMARLING, G. HENRY, A. PETITET, K. STANLEY, D. WALKER, AND R. C. WHALEY, *ScaLAPACK User's Guide* SIAM, Philadelphia, 1997.
5. J. L. BARLOW, *Personal Communication*
6. J. W. DEMMEL, *Applied Numerical Linear Algebra* SIAM, Philadelphia, 1997.
7. I. S. DHILLON, *A New $O(N^2)$ Algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem*. PhD. Thesis, University of California, Berkeley, 1997.
8. I. S. DUFF, A. M. ERISMAN, AND J. K. RAID, *Direct Methods for sparse Matrices*, Clarendon Press, Oxford, 1986.
9. J. A. GEORGE, AND J. W-H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall Inc. , Englewood Cliffs, NJ, 1981.
10. G. GOLUB AND C. F. VAN LOAN, *Matrix Computations, third edition*, The Johns Hopkins University Press, Baltimore, MD, 1996.
11. R. G. GRIMES, J. G. LEWIS, AND H. .D. SIMON, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*. SIAM J. Matrix Anal. Appl., Vol. 15, No. 1, pp. 228-272, 1994.
12. M. GU AND S. C. EISENSTAT, *A Divide-and-Conquer algorithm for the symmetric tridiagonal eigenproblem* SIAM J. Matrix Anal. Appl. Vol. 16, No. 1 pp. 172–191 1995.
13. B. HENDRICKSON AND E. ROTHBERG, *Improving the runtime and quality of nested dissection ordering*, tech. rep., Sandia National Laboratories, Albuquerque, NM 87185, 1996.
14. L. KAUFMAN, *Banded Eigenvalue Solvers on Vector Machines* ACM Trans. Math. Software, Vol. 10, No. 1, pp. 73–86, 1984.
15. G. KARYPIS AND V. KUMAR, *METIS: Unstructured graph partitioning and sparse matrix ordering system*, tech. rep., Department of Computer Science, University of Minnesota, Minneapolis, MN, 1995.
16. R. B. LEHOUCQ, D. C. SORENSEN, AND C. YANG, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*
17. M. MENON AND K.R. SUBBASWAMY, *A Transferable Nonorthogonal Tight-Binding Scheme for Silicon*, Phys. Rev. B50, 11577 (1994).
18. M. MENON, R. RICHTER, P. RAGHAVAN AND K. TERANISHI, *Large Scale Quantum Mechanical Simulations of Carbon Wires, Superlattices and Microstructures*, Vol. 27, No. 5/6, pp. 577–581, 2000.
19. *Matrix Market* <http://math.nist.gov/MatrixMarket/>
20. E. G.-Y. NG AND P. RAGHAVAN, *Performance of greedy ordering heuristics for sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., Vol. 20, No. 4, pp. 902–914, 1997.
21. B. PARLETT., *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, (1980).
22. B. N. PARLETT AND I. S. DHILLON, *Relatively Robust Representations for Symmetric Tridiagonals*, Linear Algebra and its Applications, 309, pp. 121–151, 2000.
23. P. RAGHAVAN, *DSCPACK: Domain-Separator Codes for the parallel solution of sparse linear systems* Technical Report CSE-02-004, Department of Computer Science and Engineering, The Pennsylvania State University, 2002.
24. S. L. G. SLEJPEN AND H. A. VAN DER VORST, *A Jacobi-Davidson iteration method for linear eigenvalue problems*. SIAM J. Matrix Anal. Appl., Vol. 17, pp. 401–425, 1996.

25. C. YANG, *Accelerating the Arnoldi Iteration: Theory and Practice* Ph.D Thesis, Department of Computational and Applied Mathematics, Rice University, Houston, Jan. 1998.