A Study of the Invariant Subspace Decomposition Algorithm for Banded Symmetric Matrices

Christian Bischof[†] Xiaobai Sun[†] Anna Tsao[‡] Thomas Turnbull[‡]

Abstract

In this paper, we give an overview of the Invariant Subspace Decomposition Algorithm for banded symmetric matrices and describe a sequential implementation of this algorithm. Our implementation uses a specialized routine for performing banded matrix multiplication together with successive band reduction, yielding a sequential algorithm that is competitive for large problems with the LAPACK QR code in computing all of the eigenvalues and eigenvectors of a dense symmetric matrix. Performance results are given on a variety of machines.

1 Introduction

Computation of eigenvalues and eigenvectors is an essential kernel in many applications, and several promising parallel algorithms have been investigated [8, 11, 7]. The work presented in this paper is part of the PRISM (Parallel Research on Invariant Subspace Methods) Project, which involves researchers from Argonne National Laboratory, the Supercomputing Research Center, the University of California at Berkeley, and the University of Kentucky. The goal of the PRISM project is the development of algorithms and software for solving large-scale eigenvalue problems based on the invariant subspace decomposition approach originally suggested by Auslander and Tsao [1].

The algorithm described here is a promising variant of the Symmetric Invariant Subspace Decomposition Algorithm (SYISDA) [2] that requires significantly less overall work than SYISDA. Let A be a symmetric matrix. Recall that the SYISDA proceeds as follows:

Scaling: Compute bounds on the spectrum $\lambda(A)$ of A and use these bounds to compute α and β such that for $B = \alpha A + \beta I$, $\lambda(B) \subseteq [0, 1]$, with the mean eigenvalue of A being mapped to $\frac{1}{2}$.

Eigenvalue Smoothing: Let $p_i(x)$, $i = 1, 2, \ldots$ be polynomials such that in the limit all values are mapped to either 0 or 1. Iterate

$$C_0 = B, C_{i+1} = p_i(C_i), i = 0, 1, \ldots,$$

until $||C_{i+1} - C_i||$ is numerically negligible, at which point all the eigenvalues of the iterated matrix are near either 0 or 1. We denote the converged matrix by C_∞.

Invariant Subspace Computation: Find an orthogonal matrix $[U, V]$ such that the columns of U and V form orthonormal bases for the range space of C_∞ and its complementary orthogonal subspace, respectively. That is, $U^T U = I$, $V^T V = I, U^T V = 0$, and the range of $C_\infty U$ is U.

Decoupling: Update the original A with $[U, V]$, i.e., form

$$[U, V]^T A\, [U, V] = \begin{pmatrix} A_1 & \\ & A_2 \end{pmatrix}.$$

[†]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439.

[‡]Supercomputing Research Center, Bowie, MD 20715-4300.

*This paper is PRISM Working Note #16, available via anonymous ftp to `ftp.super.org` in pub/prism.

This work was partially supported by the Applied and Computational Mathematics Program, Advanced Research Projects Agency, under Contract DM28E04120, and by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

To compute the eigenvectors, the orthogonal matrices in the **Invariant Subspace Computation** step are used to update the eigenvector matrix. We can then apply the same approach in parallel to A_1 and A_2 until all subproblems have been solved.

In the next section, we describe a sequential implementation of a variant of SYISDA that is applicable to both dense and banded matrices. Throughout the remainder of this paper, we refer to this new algorithm as banded SYISDA. In the original SYISDA algorithm, dense matrix multiplications are performed in the **Eigenvalue Smoothing** step. Since a substantial number of these multiplications are required for each divide, we were interested in reducing the cost of this step. Reduction of the original matrix to narrow band would allow multiplication of banded rather than dense matrices. As seen in [12], banded matrix multiplication can be performed quite efficiently through the use of specialized routines. Rapid band growth normally occurs when multiplying random matrices together. However, as we shall see, the special properties of the polynomials in the **Eigenvalue Smoothing** step result in surprisingly slow band growth for the iterated matrices. In banded SYISDA, we have modified the **Eigenvalue Smoothing** step to perform periodic band reductions to allow the polynomial iteration to be performed with narrow banded matrices. The band reductions are performed using the successive band reduction (SBR) algorithm of Bischof and Sun [5]. In Section 3, we demonstrate that the run times for our implementation of banded SYISDA, when applied to dense matrices, are competitive with the symmetric QR algorithm. We present conclusions in Section 4.

2 Description of Sequential Algorithm

Banded SYISDA first reduces the original matrix to narrow band and then periodically reduces matrices in the **Eigenvalue Smoothing** step back to narrow band. Dense matrix multiplications are replaced by banded matrix multiplications plus a small number of band reductions during the polynomial iteration process. Multiplication of two $n \times n$ matrices of bandwidths b_1 and b_2 results in a matrix of bandwidth b_1+b_2, requiring only $O(b_1 b_2 n)$ work versus $O(n^3)$ for two dense matrices. Based on our studies of banded matrix multiplication [12], we decided to use the Madsen-Rodrigue-Karush (MRK) algorithm to perform the required matrix multiplications. Although blocked algorithms for banded matrix multiplication are about twice as fast as the MRK algorithm [12], we found that it is difficult to realize an actual time savings through the use of these algorithms in the overall context of banded SYISDA. First, the optimal block size varies with problem size. Second, if the blocksize used does not divide the order of the problem, then the packing and re-packing code is very complicated and time-consuming. In fact, we discovered that the use of blocked code did not result in an overall time savings for the banded SYISDA algorithm and could in fact yield worse running times. Additionally, depending on the blocksize, the packing scheme used for such algorithms could require significantly more memory than unblocked code. In general, for the complete eigenproblem, less than 10% of the total time is spent in our current implementation performing banded matrix multiplication. As a result, including the dense matrix multiplications required in the **Decoupling** step and the eigenvector updates, the total time spent in banded SYISDA doing matrix multiplication is usually less than half the total time.

In the dense SYISDA, there is no reason to expect that the use of higher order polynomials would lead to improved performance. But the slower than expected growth in "effective" bandwidth for banded SYISDA leads to improved performance if higher order polynomials are used initially. At some point, however, one must trade off memory for performance. In our implementation, the $\{p_i\}$ begin as the third incomplete Beta function, $\mathfrak{B}_3(x) = -20x^7 + 70x^6 - 84x^5 + 35x^4$, and then switch to the first incomplete Beta function, $\mathfrak{B}_1(x) = 3x^2 - 2x^3$.

The band growth properties of banded SYISDA are indeed its most surprising feature. Since $\mathfrak{B}_1$ and $\mathfrak{B}_3$ have degrees 3 and 7, respectively, one might expect that the bandwidth would increase by factors of 3 and 7, respectively, with each application. We show in Figure 1 that this is not the case. We show the actual band growth (solid line) compared to the band growth if the bandwidth actually increased by a factor of 3 or 7 (dashed line) for a 1000×1000 matrix where reductions to tridiagonal form are done whenever the bandwidth reaches 50. Let $||\cdot||$ denote the 2-norm. We have

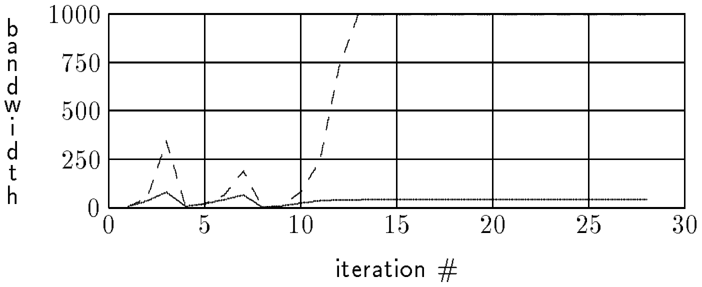

FIGURE 1. Band Growth for a 1000×1000 Matrix

Theorem. *Assume that X is a symmetric matrix of band b with all of its eigenvalues in $[0,1]$. Let $X_\infty = \lim_{i\to\infty} \mathfrak{B}_1(X)$ and $E = X - X_\infty$. Then*

$$\left\| \mathfrak{B}_1^{(k)}(X) - \mathfrak{B}_1^{(k-1)}(X) \right\| = O(\|E\|^{2^{(k-1)}}), \qquad k = 1, 2, 3, \ldots.$$

Proof. X_∞ only has eigenvalues 0, 1, and 1/2. Notice that $\|E\| \le 1/2$. Suppose that X_∞ has eigenvalues 0, 1, and 1/2 of multiplicities m_0, m_1, and m_2, respectively. Then

$$X = X_\infty + E = Q \begin{pmatrix} I_0 + E_0 & & \\ & \frac{1}{2} I_2 & \\ & & E_1 \end{pmatrix} Q^t,$$

where Q is an orthogonal matrix and for $j = 1, 2, 3$, I_j denotes the identity matrix of dimension m_j and E_j a matrix of dimension m_j. A straightforward calculation yields that $\mathfrak{B}_1(X) = X_\infty + O(E^2)$. Hence $\mathfrak{B}_1(X) = X - E + O(E^2)$ and the result follows by induction. □

This theorem basically says that subsequent bands generated by the application of $\mathfrak{B}_1$ die off exponentially, so that the outer bands quickly become negligible. Therefore, after a few applications of $\mathfrak{B}_1$, $\|E\|$ becomes small enough so that the growth in the effective bandwidth slows significantly. An analogous result holds for $\mathfrak{B}_3$.

The band reductions are performed using successive band reduction [5, 3, 6]. Instead of reducing a matrix from banded form directly to narrow band, SBR reduces a banded matrix to some intermediate band using a blocked algorithm and then reduces this intermediate matrix to the required band. The motivation for SBR is two-fold. First, we wanted to take advantage of the fact that the reduction of a narrow banded matrix requires significantly less work than reduction of a dense matrix. Second, SBR allows for effective blocking in both stages [5, 6], unlike previous algorithms that allow only limited blocking. In our current implementation, the first stage is blocked, but the second stage utilizes an improved, but as yet unblocked, version of Lang's algorithm [9, 6]. We hope to incorporate the block Householder approach of Bischof and Sun [6] soon. In order to minimize the time spent performing updates with the orthogonal matrices resulting from the band reductions performed, we accumulate these matrices and apply the resulting matrix where needed.

Instead of using QR factorization with column pivoting as in [10], we use the rank-revealing tridiagonalization strategy of Bischof and Sun [4] to compute the invariant subspaces of C_∞. The rank-revealing tridiagonalization technique exploits both symmetry and the special structure of matrices having only two eigenvalues. This special structure allows for a significant computational savings when performing tridiagonalization [4]. Since the converged matrix resulting from the polynomial iteration can have eigenvalues at 0, 1/2, and 1, we needed to modify our algorithm to ensure that only 0 and 1 eigenvalues are produced. Fortunately, we have discovered an inexpensive means of detecting this situation. Suppose the converged matrix, C_∞, has eigenvalues 0, 1/2, and 1 of multiplicities m_1, m_2, and m_3, respectively. It is easily shown that $m_2 = 4(\mathrm{trace}(C_\infty) - \mathrm{trace}(C_\infty^2))$. Both $\mathrm{trace}(C_\infty)$ and $\mathrm{trace}(C_\infty^2) = \|C_\infty\|_F^2$ are inexpensive computations. Here, $\|\cdot\|_F$ denotes the Froebenius norm. If $m_2 > 0$, we then apply either $3x - 2x^2$ or $2x^2 - x$ to C_∞. Both these quadratic functions map 0 and 1 to themselves; the pertinent difference between them is that one maps 1/2 to

1 and the other maps 1/2 to 0. The quadratic used is chosen to make the problem divide as evenly as possible. In practice, after the application of this quadratic, the clusters of eigenvalues at 0 and 1 require "tightening up." We accomplish this by a few further applications of the $\{p_i\}$.

In our current code, reducing matrices to tridiagonal form results in the fastest algorithm. Since the SBR strategy reduces a dense matrix to narrow band very quickly [2], one might expect that only reducing to narrow band might be superior. Unfortunately, when the band starts out larger, matrices of greater bandwidth must be multiplied, resulting in a significant increase in the matrix multiplication time, since the MRK algorithm is more efficient for smaller bandwidths. At the moment, the time savings from not reducing all the way to tridiagonal is lost in the multiplication time. We are therefore investigating methods of improving the efficiency of the banded matrix multiplication. If this primitive could be speeded up, the number of band reductions required could also potentially be reduced. In our current implementation, we typically perform as many as five band reductions for each divide. In the early band reductions, when the matrix has nonstructured eigenvalues, we use unblocked tridiagonalization. This is currently faster than SBR. We believe that the lack of blocking in our current implementation of the reduction from narrow band to tridiagonal is limiting our performance and that if we can perform this reduction step all in blocked code, the performance of SBR should be superior to the unblocked code. In the later band reductions, we use SBR because, even though the second stage is as yet unblocked, the skipping that results from even partial convergence of the eigenvalues still yields significant performance improvements.

3 Sequential Performance Results

In this section, we present some performance results in double precision on the IBM RS/6000 Model 580 and in single precision on the Cray-2 and C90. Test cases were generated using the test generation routine xLATMS from LAPACK. In the tables below, we omit the first letter indicating the precision of the computations. First, in Table 1, we compare the time required by our algorithm (xBSYISDA) to xSYEV (QR algorithm) and xSTEBZ (bisection) in finding all of the eigenvalues and eigenvectors of dense symmetric matrices with uniformly distributed eigenvalues ($\mathsf{MODE} = 6$ in xLATMS). Accuracy in the residuals and eigenvector orthogonality was comparable in all cases tested. In the cases shown, xBSYISDA is at most 2.3 times slower than xSYEV. On the RS/6000 (not shown) and Cray-2, relative performance of xBSYISDA to xYSEV improves with problem size; on the Cray-2, the difference is less than 10% by problem size 4000. Compared to xSTEBZ, xBSYISDA is less than a factor of 3 slower. We plan to study the relative asymptotic behavior of these algorithms in more detail. In these cases, roughly 1/2 to 2/3 of the time is spent in performing band reductions and the **Invariant Subspace Computation** step.

TABLE 1. Cray Timings for Complete Eigenproblem (single precision, $\mathsf{MODE} = 6$)

n	Cray-2 time (seconds)			C90 time (seconds)		
	SYEV	STEBZ	BSYISDA	SYEV	STEBZ	BSYISDA
1000	101.3	79.2	134.9	14.6	23.8	30.3
2000	645.6	381.9	725	101.5	111.4	191.0
3000	2006.5	991.2	2294.5	330.9	282.1	589.4
4000	4645.8	2104.7	5065.6	578.9	767.8	1335.9

TABLES 2 & 3. Timing Results for Partial Eigenproblem ($\mathsf{MODE} = 6$)

n	RS/6000 time (seconds)	
	STEBZ	BSYISDA
250	1.8	3.7
500	11.3	27.0
750	33.3	81.3

n	Cray-2 time (seconds)		C90 time (seconds)	
	STEBZ	BSYISDA	STEBZ	BSYISDA
1000	40.5	71.9	9.2	13.7
2000	212.6	419.2	51.7	120.6
3000	661.8	1340.7	153.7	374
4000	1468.1	3021.4	336.4	844.6

TABLE 4. RS/6000 Timings for Complete Eigensolution (double precision, MODE = 2)

n	time (seconds)		
	SYEV	STEBZ	BSYISDA
250	2.5	3.0	2.6
500	19.7	20.7	19.8
750	69.0	63.6	62.2

Tables 2 and 3 give timing comparisons for xSTEBZ and xBSYISDA in finding all the eigenvalues in a specified interval and the corresponding eigenvectors. In all cases, roughly 1/4 of the eigenvalues in the middle of the spectrum were found. xBSYISDA is roughly 1.5 to 2.5 times slower than xSTEBZ in these cases. In these cases, the percentage of time spent in the band reductions and **Invariant Subspace Computation** step is generally 80% or more.

Since we expect significant time improvements in SBR when our implementation incorporates blocking in the reduction from narrow band to tridiagonal, we expect our overall times to improve.

Matrices with clustered eigenvalues arise in many important applications, but are problematical for some algorithms. As described in [2], the SYISDA algorithms tend to isolate clusters of eigenvalues. Subproblems consisting of tightly clustered eigenvalues can often be detected using the heuristic described in [2] together with deflation-checking. In Table 4, we show relative timings for xSYEV, xSTEBZ, and xBSYISDA for test matrices generated by xLATMS having only three repeated eigenvalues (MODE = 2). These are, of course, very special matrices, and we plan more extensive testing on matrices with more interesting cluster structures.

4 Conclusions

In this paper, we have described a new eigensolver for banded symmetric matrices based on the Invariant Subspace Decomposition Algorithm that retains the property of using scalable primitives while reducing the computational complexity of previous versions. We have shown that, for large problems, this approach is competitive sequentially with the QR algorithm.

We plan continued investigations of this algorithm and expect the upcoming blocked version of SBR to significantly improve performance. Since the proportions of time spent in different primitives varies significantly between the dense and banded SYISDA, we plan to compare the behavior of parallel versions of both algorithms to each other and to other algorithms.

References

[1] Auslander, L. & A. Tsao, *On parallelizable eigensolvers*, Adv. Appl. Math. **13** (1992), 253–261.

[2] Bischof, C. H., S. Huss-Lederman, X. Sun, & A. Tsao, *The PRISM Project: Infrastructure and Algorithms for Parallel Eigensolvers*, Proceedings, Scalable Parallel Libraries Conference (Starksville, MS, Oct. 6–8, 1993), IEEE, 1993, (also PRISM Working Note #12).

[3] Bischof, C., M. Marques, & X. Sun, *Parallel bandreduction and tridiagonalization*, Proceedings, Sixth SIAM Conference on Parallel Processing for Scientific Computing (Norfolk, Virginia, March 22-24, 1993) (R. F. Sincovec, eds.), SIAM, Philadelphia, 1993, (also PRISM Working Note #8).

[4] Bischof, C. & X. Sun, *A divide-and-conquer method for tridiagonalizing symmetric matrices with repeated eigenvalues*, Preprint MCS-P286-0192, Argonne National Laboratory (1992), (also PRISM Working Note #1).

[5] Bischof, C. & X. Sun, *A framework for symmetric band reduction and tridiagonalization*, Preprint MCS-P298-0392, Argonne National Laboratory (1992), (also PRISM Working Note #3).

[6] Bischof, C. H. & X. Sun, *Parallel tridiagonalization through two-step band reduction*, Proceedings: Scalable High Performance Computing Conference '94, Knoxville, Tennessee, May 1994, IEEE Computer Society Press, 1994, (also PRISM Working Note #17) (to appear).

[7] Huo, Y. & R. Schreiber, *Efficient, massively parallel eigenvalue computation*, RIACS Technical Report 93.02, Research Institute for Advanced Computer Science (1993).

[8] Ipsen, I. & E. Jessup, *Solving the symmetric tridiagonal eigenvalue problem on the hypercube*, Tech. Rep. RR-548, Yale University (1987).

[9] Lang, B., *A parallel algorithm for reducing symmetric banded matrices to tridiagonal form*, SIAM J. Sci. Stat. Comp. **14** (1993), no. 6.

[10] Huss-Lederman, S., A. Tsao, & T. Turnbull, *A parallelizable eigensolver for real diagonalizable matrices with real eigenvalues*, Technical Report TR-91-042, Supercomputing Research Center (1991).

[11] Li, T.-Y., H. Zhang, & X.-H. Sun, *Parallel homotopy algorithm for symmetric tridiagonal eigenvalue problems*, SIAM J. Sci. Stat. Comput. **12** (1991), no. 3, 469–87.

[12] Tsao, A. & T. Turnbull, *A comparison of algorithms for banded matrix multiplication*, Technical Report SRC-TR-093-092, Supercomputing Research Center (1993), (also PRISM Working Note #6).