Parallel Studies of the Invariant Subspace Decomposition Approach for Banded Symmetric Matrices*

Christian Bischof† Steven Huss-Lederman‡ Xiaobai Sun†
Anna Tsao‡ Thomas Turnbull‡

Abstract

We present an overview of the banded Invariant Subspace Decomposition Algorithm for symmetric matrices and describe a parallel implementation of this algorithm. The algorithm described here is a promising variant of the Invariant Subspace Decomposition Algorithm for dense symmetric matrices (SYISDA) that retains the property of using scalable primitives, while requiring significantly less overall computation than SYISDA.

1 Introduction

Computation of eigenvalues and eigenvectors is an essential kernel in many applications, and several promising parallel algorithms have been investigated. 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 an eigensolver for finding all or some of the eigenvalue–eigenvector pairs of a banded real symmetric matrix. The algorithm is a variant of the Symmetric Invariant Subspace Decomposition Algorithm (SYISDA) [2] and like SYISDA, uses highly scalable primitives, while requiring significantly less overall computation than SYISDA. In the next section, we give an overview of the banded SYISDA. A parallel implementation of this algorithm is then presented in §3. Conclusions and future work are discussed in §4.

2 Description of Algorithm

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

*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.

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439.
‡Supercomputing Research Center, Bowie, MD 20715-4300.

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

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

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.

Banded SYISDA first reduces the original matrix to narrow band and then periodically reduces matrices in the **Eigenvalue Smoothing** step back to narrow band [7]. The dense matrix multiplications performed in dense SYISDA are replaced by banded matrix multiplications plus a small number of band reductions during the polynomial iteration process. The motivation for this approach is that 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. This fact, coupled with the slow band growth during the iteration process [7], dramatically reduces the cost of performing matrix multiplications over that of SYISDA, at the expense of a small number of band reductions.

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 periodic band reductions required are performed using *successive band reduction* [5]. As in dense SYISDA [9], the **Invariant Subspace Computation** step is performed using *rank-revealing tridiagonalization* [4, 9]. Sequential studies have demonstrated that banded SYISDA is quite promising [7].

3 Parallel Algorithm

In this section, we describe our parallel algorithm for banded SYISDA. Our original dense SYISDA code is being rewritten to use the Message Passing Interface (MPI), allowing for a simplified portable code, due to MPI features that facilitate the handling of subproblems produced by the divide and conquer, as well as collective operations. In addition, we shall see that the modular design of the PRISM software infrastructure leads to maximal reuse of previously developed software in the banded SYISDA implementation. Banded SYISDA can be applied to both dense and banded matrices. We begin with discussions of the two main kernels, followed by a description of the tradeoffs entailed in choosing an appropriate band reduction strategy, and then conclude with a discussion of the overall banded SYISDA algorithm.

3.1 Parallel Banded Matrix Multiplication

The initial data distribution used in banded SYISDA is identical to that used in our parallel implementation for dense SYISDA. That is, on a square number, p^2, of processors, we treat the processors as a $p \times p$ array of processors. The original symmetric matrix is distributed in torus wrap fashion in both dimensions. Note that if the initial matrix is banded and symmetric, then the submatrices on each processor arising from using 2D torus wrap are banded nonsymmetric matrices, as illustrated by the following example. We show in Figure 1 the initial data distribution for a 6×6 matrix of bandwidth 2 onto a 2×2 processor mesh.

processor $(0,0)$			processor $(0,1)$		
A_{00}	A_{02}	0	A_{01}	0	0
A_{20}	A_{22}	A_{24}	A_{21}	A_{23}	0
0	A_{42}	A_{44}	0	A_{43}	A_{45}
processor $(1,0)$			**processor $(1,0)$**		
A_{10}	A_{12}	0	A_{11}	A_{13}	0
0	A_{32}	A_{34}	A_{31}	A_{33}	A_{35}
0	0	A_{54}	0	A_{53}	A_{55}

Fig. 1. *Torus Wrapping of a 6×6 Matrix of Bandwidth 2 Onto a 2×2 Mesh*

Our parallel implementation of banded SYISDA will use a specialized algorithm we are developing for performing banded matrix multiplication. For our parallel implementation of dense SYISDA, we developed a distributed dense matrix multiplication routine, BiMMeR [8]. On square meshes, the same algorithm used in BiMMeR can be used to multiply banded matrices simply by replacing the dense matrix multiplication on each node by a banded matrix multiplication. In our implementation, these uniprocessor banded matrix multiplications are performed using efficient, specialized routines for multiplying banded matrices stored in packed format [12, 11]. The use of packed storage translates into decreased communication costs over using full storage.

Use of this strategy ensures that banded matrix multiplication will scale, provided that the matrix bandwidths grow proportionally to the matrix dimension and are large enough so that all processors have computation to do. Since we expect the bandwidths maintained to be small relative to the size of the original matrix, we expect finer grained computations on each processor than in the dense case, i.e., less efficient computation. Since banded SYISDA allows us free choice of both the maximum and minimum bandwidths allowed before and after a band reduction, we expect interesting performance tradeoffs between the allowed band growth and computational efficiency.

3.2 Successive Band Reduction and Rank-revealing Tridiagonalization

In banded SYISDA, symmetric band reductions play a pivotal role, since the band reductions performed during the **Eigenvalue Smoothing** step and rank-revealing tridiagonalization taken together account for the majority of the overall computation [7]. The required band reductions will be performed using the so-called *successive band reduction* [5]. The three kernels needed to fully realize the advantages of this strategy for both dense and banded matrices are the reduction of a dense matrix to narrow band, the reduction of a banded matrix to narrow band, and the reduction of a narrow banded matrix to tridiagonal. It turns out that each of these kernels can be performed almost entirely in block operations [6, 3].

To date, we have implemented the blocked reduction of a dense matrix to narrow band. We are currently developing a parallel realization of the blocked algorithms for reductions of a banded matrix to narrow band and of a narrow banded matrix to tridiagonal.

During the early iterations of the **Eigenvalue Smoothing** step, when the spectrum has no special structure, the band reductions are the most costly and are performed in either one or two stages, depending on the initial bandwidth of the matrix to be reduced and the final bandwidth desired. In contrast, for the matrix C_∞, the amount of work required is generally significantly reduced, because, as discussed in [4, 2], one generally expects a substantial number of the transformations required to reduce a symmetric matrix to tridiagonal to be "skipped" due to the special structure of its eigenvalues. In fact, even in intermediate band reductions, some "skipping" generally occurs as well, due to the convergent nature of the polynomial iteration. The amount of actual "skipping" depends on the eigenvalue distribution of the original matrix.

3.3 Reduction Scheme Tradeoffs

Determination of the "optimal" strategy in the **Eigenvalue Smoothing** step is quite complicated and depends on numerous factors, including the performance of the distributed banded matrix multiplication and successive band reduction in different bandwidth regimes, the band growth properties of the polynomial iteration over varying matrix size regimes and eigenvalue distributions, and the amount of skipping that occurs in intermediate band reductions. In the following table, we give some examples of the tradeoffs entailed from our sequential code for dense matrices with uniformly distributed eigenvalues. We denote by n the matrix dimension, by $b_{\max}$ the maximum bandwidth allowed, by $b_{\min}$ the bandwidth after each band reduction, and by t_{bmm}, t_{br}, and t_{total} the times spent performing banded matrix multiplication, band reduction, and the entire divide, respectively, during the first divide.

n	$b_{\max}$	$b_{\min}$	t_{bmm} (s)	t_{br} (s)	t_{total} (s)
1000	100	1	10.2	97.7	161.3
1000	200	1	38.5	125.7	222.5
1000	100	5	18.9	214.7	292.1
1000	200	5	50.1	226.8	323.1
1000	100	10	25.2	161.7	241.8
1000	200	10	111.2	158.9	333.0

TABLE 1. *Tradeoffs between Bandgrowth and Times Spent in Banded Matrix Multiplication and Band Reduction*

3.4 The Parallel SYISDA Algorithm

The previously-developed PRISM software developed for the orchestration of multiple subproblems is used essentially unchanged in banded SYISDA and consists of two separate stages. We briefly review the two stages of the divide and conquer strategy; more detail can be found in [10].

The first stage encompasses the early divides, where large subproblems are solved sequentially and the overall performance depends on the scaling properties of the banded matrix multiplication, successive band reductions, and rank-revealing tridiagonalization. Each divide produces two independent dense symmetric eigenvalue problems. Through the use of two-dimensional torus wrap, no data redistribution is required between divides. As in dense SYISDA, as the subproblem size decreases, the proportion of the total time required for communication during individual matrix multiplications increases and the granularity of local computation decreases. However, as discussed in [10], this approach guarantees near-perfect load balancing in the costly early stages.

As the subproblem size decreases, there is a point at which communications overhead makes it impractical to solve that subproblem over the entire mesh. The amount of work required to find the eigensolution of the remaining subproblem is very small, but the cost of the update of the eigenvector matrix Z, whose leading dimension is still the size of the original matrix A, is still substantial. The second stage, or end game, handles the small subproblems remaining, exploiting parallelism at two levels: first, by solving individual subproblems in parallel on single nodes and second, by performing distributed updates of the accumulated eigenvector matrix.

4 Future Work

We plan to compare the performance of different band reduction strategies on a variety of parallel machines, including the Intel Delta, Intel Paragon, and the IBM SP1. We will also compare the performance of banded SYISDA to that of dense SYISDA for dense problems.

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, pp. 123–131, (also PRISM Working Note #12).

[3] Bischof, C., B. Lang, & 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).

[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. and X. Sun, *On Orthogonal Block Elimination*, Technical Report MCS-P441-0594, Mathematics and Computer Science Division, Argonne National Laboratory (1994), (also PRISM Working Note #20).

[7] Bischof, C., X. Sun, A. Tsao, & T. Turnbull, *A study of the Invariant Subspace De-*

composition Algorithm for banded symmetric matrices, Proceedings: Fifth SIAM Conference on Applied Linear Algebra, Snowbird, UT, June, 1994 (John G. Lewis, eds.), SIAM, 1994, pp. 321–325, (also PRISM Working Note #16).

[8] Huss-Lederman, S., E. M. Jacobson, A. Tsao, & G. Zhang, *Matrix multiplication on the Intel Touchstone Delta*, Concurrency: Practice & Experience, (also PRISM Working Note #14) (to appear).

[9] Huss-Lederman, S., A. Tsao, & G. Zhang, *A parallel implementation of the Invariant Subspace Decomposition Algorithm for dense symmetric matrices*, Proceedings, Sixth SIAM Conference on Parallel Processing for Scientific Computing (Norfolk, Virginia, March 22-24, 1993) (R. F. Sincovec, eds.), SIAM, Philadelphia, 1993, pp. 367–374, (also PRISM Working Note #9, also appears as Technical Report SRC-TR-93-091, Supercomputing Research Center, 1993).

[10] Huss-Lederman, S., A. Tsao, & G. Zhang, *A parallel implementation of the Invariant Subspace Decomposition Algorithm for dense symmetric matrices*, Proceedings, Sixth SIAM Conference on Parallel Processing for Scientific Computing (Norfolk, Virginia, March 22-24, 1993) (R. F. Sincovec, eds.), SIAM, Philadelphia, 1993, pp. 367–374, (also PRISM Working Note #9).

[11] Quintana, G., X. Sun, A. Tsao, & T. Turnbull, *A comparison of algorithms for banded matrix multiplication*, in preparation.

[12] Tsao, A. & T. Turnbull, *A comparison of algorithms for banded matrix multiplication*, Technical Report SRC-TR-93-092, Supercomputing Research Center (1993), (also PRISM Working Note #6).