LBNL-43178

An Evaluation of the Parallel Shift-and-Invert Lanczos Method[†]

Kesheng Wu[‡] and Horst Simon[‡]

Abstract *When the Lanczos method is used to compute eigenvalues, it is often restarted or used with the shift-and-invert scheme. The restarted scheme usually uses less memory but the shift-and-invert scheme is more robust. In addition, the shift-and-invert Lanczos method requires accurate solutions of a series of linear systems. Parallel software packages suitable for these linear systems are only started to become available. In this talk, we will present our evaluation of two such packages and briefly exam when it is necessary to use the shift-and-invert scheme.*

Keywords: eigenvalue, Lanczos algorithm, parallel direct method

1 Introduction

This talk is on how to compute eigenvalues and eigenvectors of large sparse symmetric matrices on massively parallel machines. One of the most commonly used algorithms for this task is the Lanczos method which projects the large eigenvalue problem onto a low dimensional Krylov subspace [3, 10]. In most cases, the Krylov subspace basis is built through a series of matrix-vector multiplications, the whole Lanczos method can be implemented with a matrix-vector multiplication routine and a few simple vector operations. This algorithm is highly efficient on parallel machines and it is effective for computing extreme and well separated eigenvalues.

To compute the interior or not well-separated eigenvalues, the shift-and-invert Lanczos method is one of the most effective methods. Since the eigenvalues of a matrix A are related to the eigenvalues of $(A-\sigma I)^{-1}$ by a simple relation ($\lambda(A) = \sigma + 1/\lambda((A-\sigma I)^{-1})$) and the corresponding eigenvectors of A and $(A-\sigma I)^{-1}$ are identical, the shift-and-invert scheme computes the extreme eigenvalues of $(A-\sigma I)^{-1}$ and deduce the corresponding eigenvalues of A. With appropriate choice of σ, the extreme eigenvalues of $(A-\sigma I)^{-1}$ are well separated and can be easily computed. To build a Krylov subspace basis of $(A-\sigma I)^{-1}$, the shift-and-invert Lanczos method solves a series of linear systems with the coefficient matrix $(A-\sigma I)$. The linear systems need to be solved accurately and the only reliable means to accomplish this is by using a direct method [4]. Because it is difficult to implement sparse direct solvers on parallel machines, efficient parallel implementation has not been widely available until recently [1, 7]. This talk will present our study of the parallel efficiency of the shift-and-invert Lanczos method using these newly available direct solvers.

When the shift-and-invert Lanczos method cannot be used, the restarted Lanczos method is used. In recent years, there have been significantly improvements to restarting which enhance the overall effectiveness [2, 13]. In this talk we will also present a small number of comparisons between the restarted Lanczos method and the shift-and-invert Lanczos method to determine when to use each method.

[†]This work was supported in part by the Director, Office of Energy Research, Office of Laboratory Policy and Infrastructure Management, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Energy Research of the U.S. Department of Energy.

[‡]Lawrence Berkeley National Laboratory/NERSC, Berkeley, CA 94720. Email: {kwu, hdsimon}@lbl.gov.

2 The software packages

Before giving the comparison data, this section briefly describes the software packages used. The four packages consists of two parallel direct method packages and two versions of Lanczos method.

SPOOLES This parallel direct method package implements factorization procedures for symmetric and nonsymmetric, real and complex matrices. The basic algorithm is based on the fundamental supernode tree [1]. In our study, this package is only used to solve simple symmetric linear systems. In this case, it computes a diagonal matrix D and an upper triangular matrix U such that $U^T DU = A$. It performs pivoting if diagonal element is found to be small. This is the main package to be evaluated because it is designed to be used in the shift-and-invert Lanczos method.

PSPASES This parallel direct solver software package implements a multifrontal Cholesky factorization for symmetric positive definite matrices, i.e., it computes an upper triangular matrix U such that $U^T U = A$ [7]. Because it requires the input matrix to be positive definite, there is no need to perform pivoting. In practice, this also prevents it from factoring some ill-conditioned matrices. Compared to SPOOLES, PSPASES accepts limited type of matrices and provides less functionality. However, because of these limitations, it is able to perform its tasks more effectively in some cases. We use this package as a reference to judge the performance of SPOOLES. To use this package we limit the test matrices to be symmetric positive definite ones.

PLANSO This is a parallel version of the Lanczos method with partial reorthogonalization [12]. It implements the standard non-restarted Lanczos algorithm for symmetric generalized eigenvalue problems. The partial reorthogonalization scheme monitors the loss of orthogonality among the Lanczos vectors and maintains a minimal orthogonality level that is necessary to compute the Ritz values accurately. This software package is used as the basis for the shift-and-invert Lanczos method with either PSPASES or SPOOLES as the linear system solver.

TRLan This software package implements a version of the thick-restart Lanczos method [13]. In theory, the thick-restart Lanczos algorithm is equivalent to the implicitly restarted Lanczos algorithm [2]. However, the thick-restart Lanczos algorithm avoid some numerical instabilities of the implicit restarting scheme and TRLan also implements more sophisticated restarting strategies.

3 Performance characteristics

Since there are general purpose direct solver packages for distributed parallel machines, the shift-and-invert Lanczos method is an available option if the matrix is explicitly generated and the factors can be stored. To answer the question of whether it is an effective option, this section will show its performance characteristics by using PLANSO with SPOOLES and PSPASES. Most of the algorithmic issues have been carefully studied before [6].

In the shift-and-invert Lanczos method, the most time-consuming operation is the factorization of the matrix and the solution of the linear systems. The factorization step is done once for each matrix and the triangular solution step is invoked at each Lanczos iteration. Typically, it needs about 3 – 5 Lanczos iterations to compute one eigenvalue. Unless a large number of eigenpairs are wanted, the factorization time dominates the whole computation. For these reasons, we will discuss the factorization time and solution time separately.

The first set of tests shown in Tables 2 and 3 uses SPOOLES to solve a set of linear systems with 3-dimensional 27-point stencil matrices, Table 2 shows the factorization time and Table 3 shows the solution time. The tests are conducted using a Cray T3E 900 massively

Table 2: Time (seconds) used to factor a series of 3-D 27-point stencil matrices.

n	# of PE									
	1	2	4	8	16	32	64	128	256	512
28	25.2	14.6	8.2	2.4	3.7	3.2	2.8	2.8	3.2	3.3
34	74.4	41.5	23.5	14.3	9.9	8.3	6.8	6.5	6.8	6.9
40		100.9	56.1	32.5	21.8	16.7	14.1	12.9	13.1	13.4
48			152.3	87.1	55.5	40.3	32.4	29.2	28.1	
56				204.0	125.1	86.6	66.1	58.2		

Table 3: Time (seconds) used to solve four linear systems with triangular factors of the 3-D 27-point stencil matrices.

n	# of PE									
	1	2	4	8	16	32	64	128	256	512
28	1.1	0.6	0.4	0.3	0.1	0.1	0.1	0.1	0.1	0.1
34	2.3	1.2	0.6	0.6	0.3	0.3	0.3	0.2	0.2	0.3
40		2.2	1.2	0.9	0.5	0.5	0.4	0.4	0.4	0.4
48			2.4	1.4	1.3	0.8	0.8	0.7	0.7	
56				3.5	2.3	1.7	1.3	1.2		

Table 1: The sizes of 3-D 27-point stencil matrices (A) and their triangular factors (U).

n	$N(\equiv n^3)$	NNZ(A)	NNZ(U)
28	2.2×10^4	5.5×10^5	5.7×10^6
34	3.9×10^4	1.0×10^6	1.3×10^7
40	6.4×10^4	1.6×10^6	2.5×10^7
48	1.1×10^5	2.9×10^6	5.4×10^7
56	1.8×10^5	4.6×10^6	1.0×10^8

parallel computer. The test matrices are generated on a uniform $n \times n \times n$ grid. The blank cells in the lower left corner are due to memory limitations of the processors. Each processor of this T3E has only 256 MB(MegaBytes) of memory. The blank cells in the lower right corner are due to some limitations in the current release of the SPOOLES software. From Table 1, it is clear that the factorized form has many more nonzero entries than the original matrix (NNZ(A) $\propto n^3$, NNZ(U) $\propto n^4$). All the test matrices shown here can be stored on one processor of the T3E, however, at least eight processors are need to store the triangular factor of the matrix based on the $56 \times 56 \times 56$ grid. When a sparse matrix is stored in memory, only a very small amount of additional memory is needed to perform a distributed matrix-vector multiplication efficiently. Thus, the eigenvalues of a much larger matrix can be computed if only matrix-vector multiplication is used.

From the data in Table 2 and 3 we see that if the minimum number of processors needed to perform the factorization is p_0, the parallel efficiency of using twice as many processors is roughly 80%. As more and more processors are used, the execution time quickly approaches a minimal value and the parallel efficiency approaches p_0/p where p is the number of processors used. For these grid matrices, the execution time almost reaches the minimum when using $16p_0$ processors. Using more processors does not generate meaningful additional reduction in computer time. In fact, the time may

Table 4: Information about the Harwell-Boeing test matrices.

name	N	NNZ	description
CT20	52329	1375396	an engine block
NASASRB	54870	2677324	shuttle rocket booster

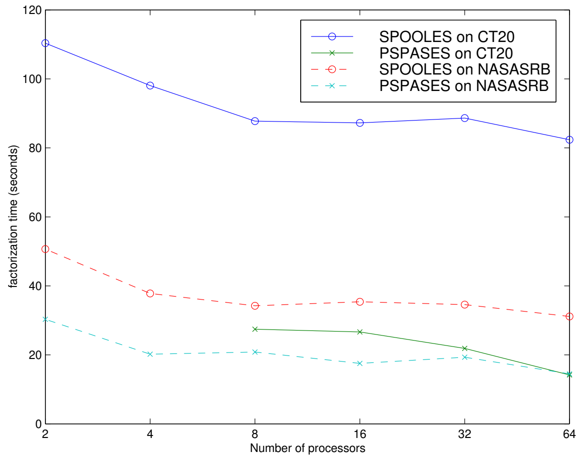

Figure 1: Time to factor the two Harwell-Boeing matrices.

actually increase.

The second set of test matrices are two large symmetric matrices in Harwell-Boeing format [5]. Information about the matrices are listed in Table 4. The time used to factor these two test matrices by the SPOOLES and PSPASES are shown in Figure 1. It is clear that as more processors are employed for the factorization, the time does not proportionally decrease. In fact, as the number of processors change from 2 to 64, the time decrease about 20 seconds in each of the four test cases (speedup: 1.3 – 2.1). Because these two matrices has more complicated nonzero patterns than the simple grid matrices tested previously, the parallel efficiency of the factorization procedures are worse than before.

Some additional performance information about the factorizations are shown in Table 5. These data are collected using 8 processors. They are aggregate data, where NNZ(U) is the total number of nonzero entries in the triangular factor, OPS is the total number of floating-point operations used during the factorization and MFLOPS is the aggregate speed of all 8 processors. PSPASES generates about 30-40 percent more nonzero entries in U and uses about twice as many floating-point operations as SPOOLES. These differences are mainly due to different reordering strategies used. PSPASES employs parMETIS to perform reordering; SPOOLES performs many independent multi-section ordering and minimal degree ordering, then chooses the best one. The ordering algorithm in SPOOLES takes more time. On the two test matrices, the ordering generated by SPOOLES leads to smaller triangular factors, but the arithmetic operations in PSPASES have more opportunity to use dense BLAS functions which can compute at higher speed. In addition, because PSPASES does not perform pivoting, its internal data structure and data communication pattern can be completely determined during the symbolic factorization phase which also reduces the time needed for the numerical factorization phase. On the two test problems, the speed of PSPASES is considerably higher than that of SPOOLES.

The timing results of solving linear systems using the triangular factors are shown in Figure 2. As the number of processors changes from 2 to 64, the solution time decreases by a factor of 4. The triangular solution stages of the two packages are more efficient than the factorization stages. Because of differences in ordering, PSPASES also uses less time to perform the triangular solution than SPOOLES on the two test problems. Similar to the factorization time, the parallel efficiencies for solving these two matrices are lower than solving the grid matrices test problems.

Figure 3 shows the total time used by

Table 5: Performance information about factorizations on 8 processors.

	CT20		NASASRB	
	SPOOLES	PSPASES	SPOOLES	PSPASES
NNZ(U)	1.0×10^7	1.4×10^7	1.0×10^7	1.3×10^7
OPS	5.2×10^9	1.3×10^{10}	2.8×10^9	5.9×10^9
MFLOPS	59.4	461	81.1	283

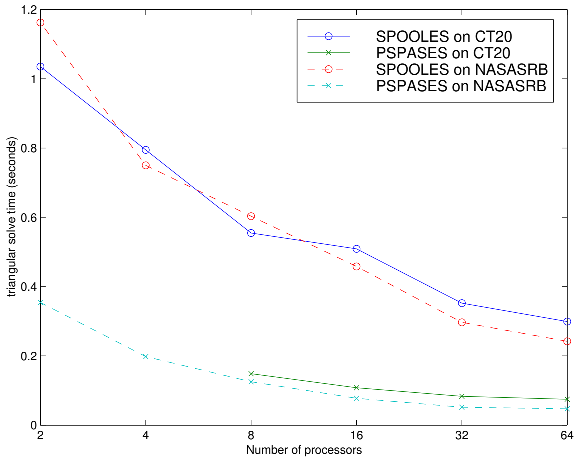

Figure 2: Time to solve with the triangular factors of two Harwell-Boeing matrices.

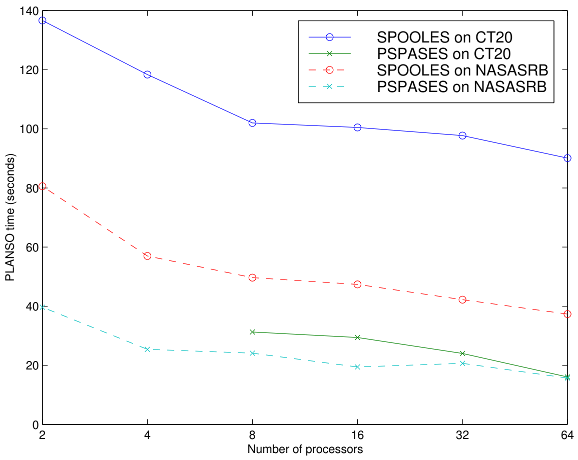

Figure 3: Time to run PLANSO for 25 steps.

PLANSO to take 25 Lanczos steps. The time shown here include both the factorization, the triangular solution and other operations needed by the Lanczos method such as the orthogonalization, the Rayleigh-Ritz projection, etc.. For both test matrices, 25 Lanczos steps is sufficient to compute 3 smallest eigenvalues and the corresponding eigenvectors. Comparing Figure 1 and 3, it is clear that the factorization time dominates the overall execution time. The total time used by PLANSO does not decrease significantly as the number of processors increases.

When the shift-and-invert scheme is not used, the most time-consuming operation in the Lanczos method is multiplying the matrix with a vector. Many researchers have shown that this operation can be parallelized effectively [9, 11]. Thus, the Lanczos method without shift-and-invert can be run efficiently on parallel machines [12]. In particular, when using the uniform grid as the test problem, if the number of grid points is scaled as the number of processors increases, the perfect parallel efficiency can be achieved. Clearly, the direct methods are yet to achieve the same level of parallel efficiency.

4 Shift-and-invert or restart

This section tries to answer the question: when is the shift-and-invert Lanczos method preferred over the restarted method. As long as an appropriate shift can be determined and the factorization can be computed, the shift-and-invert Lanczos method should be able to compute any eigenvalue and the associated

Table 6: Number of Lanczos steps to compute five eigenvalues of three diagonal matrices using TRLan(25).

N	A_l	A_q	A_c
100	286	297	527
1000	1250	7720	> 10000
10000	13691	> 10000	> 10000

Table 7: The relative gap ratios.

N	A_q	A_c	A_l
100	3×10^{-4}	8×10^{-6}	2.5×10^{-3}
1000	3×10^{-6}	8×10^{-9}	1.6×10^{-4}
10000	3×10^{-8}	8×10^{-12}	1.2×10^{-5}

eigenvector. If the goal is to compute some eigenvalues in the minimal number of Lanczos steps, the shift-and-invert scheme is usually preferred. However, if the goal is to compute some eigenvalues in the least amount of time, the restarted Lanczos method may be preferred. This section will give some concrete examples to demonstrate the point.

When trying to compute the largest eigenvalues of the NASASRB matrix, 5 seconds are needed using the Lanczos method on 2 processors. Clearly, the shift-and-invert scheme is not appropriate here because the factorization takes about 30 seconds on 2 processors. On the other hand, as little as 21 seconds are needed to compute the smallest eigenvalues of NASASRB with shift-and-invert Lanczos method on 32 processors, but 2587 seconds are needed for PLANSO on the same number of processors without shift-and-invert. In this case, 11619 Lanczos steps are taken and 11619 Lanczos vectors are stored. Even if restarting is used, we can reduce the memory but not time. For example, if 1000 Lanczos vectors are stored, TRLan needs 8546 seconds to compute the smallest eigenvalue of NASASRB on 8 processors. Clearly, the shift-and-invert Lanczos method is preferred in this case.

To get a better understanding of what type of eigenvalue problems can be solved without shift-and-invert, we apply the restarted Lanczos method to three simple matrices of varying difficulties. They are:

$$
\begin{aligned}
A_q &= \mathrm{diag}(1^2, 2^2, 3^2, \ldots), \\
A_c &= \mathrm{diag}(1^3, 2^3, 3^3, \ldots), \\
A_l &= \mathrm{diag}(\log 2, \log 3, \log 4, \ldots).
\end{aligned}
$$

The next set of tests tries to compute the five smallest eigenvalues of A_q and A_c, and the five largest eigenvalues of A_l using TRLan [13]. Table 6 shows the number of Lanczos steps needed when the basis size is 25. If the shift-and-invert Lanczos method were used, the wanted eigenvalues are computed within 20 Lanczos steps.

From analysis, we know that the converge rate of the restarted Lanczos method is related to the relative gap ratio [8]. The relative gap ratios of the test problems are shown in Table 7. Let $\lambda_1, \lambda_2, \ldots, \lambda_N$ denote the eigenvalues of the test matrices in ascending order, the relative gap ratio shown in Table 7 are $(\lambda_2 - \lambda_1)/(\lambda_N - \lambda_1)$ for A_q and A_c, $(\lambda_N - \lambda_{N-1})/(\lambda_N - \lambda_1)$ for A_l. From this set of tests, we see that when the relative gap is less than 10^{-9}, TRLan cannot compute the eigenvalues in less than 10000 steps. The smallest eigenvalue of NASASRB has a relative gap ratio of 10^{-10}. The restarted Lanczos method needs more than 50000 steps with a basis of 1000 vectors. This indicates that it is possible for TRLan to compute very difficult eigenvalues, however it needs to keep a large number of Lanczos vectors and it may take a large number of iterations.

The comparison here is only on extreme eigenvalues, if an interior eigenvalue is wanted, the spectrum needs to be transformed so that the wanted eigenvector corresponds to an extreme eigenvalue. There are different ways to achieve this, but often the shift-and-invert scheme is the most effective one.

5 Summary

From the tests conducted, we see that if the relative gap ratio of the wanted eigenvalue is smaller than 10^{-9} the shift-and-invert scheme is likely necessary. At this time, the two parallel direct methods package tested does not exhibit the same level of parallel efficiency as sparse matrix-vector multiplications. However, it is still an effective choice if the LU factors can be stored. Between the two direct solvers tested, PSPASES often uses less time than SPOOLES. However, SPOOLES is more robust for more general shift-and-invert schemes.

References

[1] C. C. Ashcraft, R. G. Grimes, D. J. Pierce, and D. K. Wah. *The user manual for SPOOLES, Release 2.0: An Object Oriented Software Library for solving sparse linear systems of equations*, 1998. The latest version of the software is available from NETLIB at `http://www.netlib.org/-linalg/spooles/spooles.html`.

[2] D. Calvetti, L. Reichel, and D. Sorensen. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2:1–21, 1994.

[3] J. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Theory*, volume 3 of *Progress in Scientific Computing*. Birkhauser, Boston, 1985.

[4] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse matrices*. Oxford University Press, Oxford, OX2 6DP, 1986.

[5] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Soft.*, 15:1–14, 1989.

[6] 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.*, 15(1):228–272, 1994.

[7] M. Joshi, G. Karypic, and V. Kumar. *PSPASES: Scalable parallel director solver library for sparse symmetric positive definite linear systems*, 1998. The latest version of the software package is available at `http://www-users.cs.umn.edu/-~mjoshi/pspases/`.

[8] Ronald B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Mathematics of Computation*, 65(215):1213–1230, July 1996.

[9] Y. Saad, K. Wu, and S. Petiton. Sparse matrix computations on the CM-5. In R. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, editors, *Proceedings of Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pages 414–420, Philadelphia, 1993. SIAM.

[10] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, 1993.

[11] R. S. Tuminaro, J. N. Shadid, and S. A. Hutchinson. Parallel sparse matrix-vector multiply software for matrices with data locality. Technical Report SAND95-1540J, Sandia National Laboratories, Albuquerque, NM, 1995.

[12] Kesheng Wu and Horst Simon. A parallel Lanczos method for symmetric generalized eigenvalue problems. Technical Report 41284, Lawrence Berkeley National Laboratory, 1997. The source code is available from `http://www.nersc.gov/-research/SIMON/planso.html`.

[13] Kesheng Wu and Horst Simon. Thick-restart Lanczos method for symmetric eigenvalue problems. Technical Report 41412, Lawrence Berkeley National Laboratory, 1998.