LARGE SCALE SPARSE SINGULAR VALUE COMPUTATIONS

MICHAEL W. BERRY *

Abstract. In this paper, we present four numerical methods for computing the singular value decomposition (SVD) of large sparse matrices on a multiprocessor architecture. We particularly emphasize Lanczos and subspace iteration-based methods for determining several of the largest singular triplets (singular values and corresponding left- and right-singular vectors) for sparse matrices arising from two practical applications: information retrieval and seismic reflection tomography. The target architectures for our implementations of such methods are the Cray-2S/4-128 and Alliant FX/80. The sparse SVD problem is well motivated by recent information-retrieval techniques in which dominant singular values and their corresponding singular vectors of large sparse *term-document* matrices are desired, and by nonlinear inverse problems from seismic tomography applications in which approximate pseudo-inverses of large sparse Jacobian matrices are needed. It is hoped that this research will advance the development of future out-of-core sparse SVD methods, which can be used, for example, to handle extremely large sparse matrices ($\mathcal{O}(10^6)$ rows or columns) associated with extremely large databases in query-based information retrieval applications.

Key words. algorithms, applications, information, multiprocessor, retrieval, seismic, singular value decomposition, sparse, tomography

1. Introduction. The singular value decomposition (SVD) is commonly used in the solution of unconstrained linear least squares problems, matrix rank estimation, and canonical correlation analysis. In applications such as information retrieval, seismic reflection tomography, and real-time signal processing, the solution to these problems is needed in the shortest possible time. Given the growing availability of multiprocessor computer systems, there has been great interest in the development of parallel implementations of the singular value decomposition, in general. In applications such as information retrieval ([17], [12]), the data matrix whose SVD is sought is usually large and sparse. It is this particular case that motivates our research effort. Hence, we are primarily interested in SVD

* Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN, 37996-1301 (berry@cs.utk.edu). Part of this research was supported by the National Science Foundation under grants NSF CCR-8717492 and CCR-900000N (NCSA), the U.S. Department of Energy under grant DOE DE-FG02-85ER25001, the Air Force Office of Scientific Research under grant AFOSR-90-0044, and the National Aeronautics and Space Administration under grant NASA NCC 2-559, when the author was located at the Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign.

methods which can be used to determine singular values and singular vectors of large sparse matrices on multiprocessors. We particularly emphasize Lanczos, block-Lanczos, subspace iteration, and trace minimization methods for determining several of the largest singular values and corresponding singular vectors for *unstructured* sparse matrices arising from practical applications. The target multiprocessors for our implementations of such methods are the Alliant FX/80 and the Cray-2S/4-128. Before our discussion of multiprocessor algorithms for the sparse SVD, we make a few definitions, state the problem of interest, and review a few of the fundamental characterizations of the SVD.

Without loss of generality, suppose A is a sparse m by n $(m \gg n)$ matrix with $\mathrm{rank}(A) = r$. The singular value decomposition (SVD) of A can be defined as

$$A = U \Sigma V^T, \tag{1}$$

where $U^T U = V^T V = I_n$ and $\Sigma = diag(\sigma_1, ..., \sigma_n)$, $\sigma_i > 0$ for $1 \le i \le r$, $\sigma_i = 0$ for $i \ge r+1$. The first r columns of the orthogonal matrices U and V define the orthonormalized eigenvectors associated with the r nonzero eigenvalues of AA^T and $A^T A$, respectively. The singular values of A are defined as the diagonal elements of Σ which are the nonnegative square roots of the n eigenvalues of AA^T. The set $\{u_i, \sigma_i, v_i\}$ is called the i-th singular triplet. The singular vectors (triplets) corresponding to large (small) singular values are called large (small) singular vectors (triplets).

Problem. *Given the sparse $m \times n$ matrix A, determine the p-largest singular triplets of A as defined by* (1).

To illustrate ways in which the SVD can reveal important information about the structure of a matrix we state two well-known theorems:

THEOREM 1.1. *Let the SVD of A be given by (1) and*

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0 \ ,$$

and let $R(A)$ and $N(A)$ denote the range and null space of A, respectively, then

1. *Rank property:* $\mathrm{rank}(A) = r$, $\mathrm{N}(A) \equiv span\{v_{r+1}, \cdots, v_n\}$, *and* $\mathrm{R}(A) \equiv span\{u_1, \cdots, u_r\}$, *where* $U = [u_1 \ u_2 \ \cdots \ u_m]$ *and* $V = [v_1 \ v_2 \ \cdots \ v_n]$.
2. *Dyadic decomposition:* $A = \sum_{i=1}^{r} u_i \cdot \sigma_i \cdot v_i^T$.
3. *Norms:* $\|A\|_F^2 = \sigma_1^2 + \cdots + \sigma_r^2$, *and* $\|A\|_2^2 = \sigma_1$.

The rank property, perhaps one of the most valuable aspects of the SVD, allows us to use the singular values of A as quantitative measures of the qualitative notion of rank. The dyadic decomposition, which is the rationale for data reduction or compression in many applications, provides a canonical description of a matrix as a sum of r rank-one matrices of decreasing importance, as measured by the singular values. The three results in **Theorem 1.1** can be combined to yield the following quantification of matrix rank deficiency (see [21] for a proof):

THEOREM 1.2. [Eckart and Young] *Let the SVD of A be given by (1) with $r = rank(A) \leq p = min(m,n)$ and define:*

$$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^T \ \ with\ k < r \ ,$$

then

$$\min_{\mathrm{r}(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = \sigma_{k+1}^2 + \ \cdots \ + \sigma_p^2 \ .$$

This important result, which indicates that A_k is the best rank-k approximation (in a least squares sense) to the matrix A, is the basis for concepts such as *data reduction* and *image enhancement.* In fact, A_k is the best approximation to A for any unitarily invariant norm ([31]). Hence,

$$\min_{\mathrm{r}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}.$$

In the next section, we illustrate the applicability of **Theorem 1.2** to problems in information retrieval and seismic tomography which motivate the sparse SVD problem. In Section 3, we present two Lanczos-based methods and two subspace methods for solving equivalent sparse symmetric eigenvalue problems. Fundamental comparisons in complexity, parallelism, memory requirements, and parameter selection among our methods are given in Section 4. In Section 5, we discuss the performance of these methods on our target machines using a test suite of sparse matrices collected from researchers in information retrieval and seismic reflection tomography. With regard to speed of computation (in CPU time), we demonstrate that a single-vector Lanczos method (LASVD) is the fastest method for approximating several of the largest singular triplets for low or moderate accuracy. A performance summary of our sparse SVD *library* of methods is given in Section 6.

2. Applications for Sparse Singular Value Decomposition. Sparse linear least squares problems naturally arise in many *real*-world applications. The use of the sparse SVD to solve such problems is of current interest to researchers in fields such as query-based information retrieval and seismic reflection tomography, for example. In the following sections, we discuss the need for computing singular triplets of large sparse matrices as suggested by recent research in physical and social science.

2.1. Latent Semantic Indexing. In [12] and [17] a new approach to automatic indexing and retrieval is discussed. It is designed to overcome a fundamental problem that plagues existing information retrieval techniques that try to match words of queries with words of documents. The problem is that users want to retrieve on the basis of conceptual topic or meaning of a document. There are usually many ways to express a given concept (*synonymy*), so the literal terms in a user's query may not match those of a relevant document. In addition, most words have multiple meanings (*polysemy*), so terms in a user's query will literally match terms in irrelevant documents.

The proposed *latent semantic indexing* (LSI) approach tries to overcome the problems of word-based access by treating the observed word to text-object association data as an unreliable estimate of the true, larger pool of words that could have been associated with each object. It is assumed there is some underlying latent semantic structure[1] in word usage data that is partially obscured by the variability of word choice. Using the SVD defined in (1), we can estimate this latent structure and remove the obscuring *noise*.

Specifically, for an $m \times n$ *term-document* matrix A whose m rows and n columns ($m \gg n$) correspond to terms and documents, respectively, we seek the closest (in a least squares sense) rank-k ($k \ll n$) matrix

$$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^T \ \ with\ k < r\ , \tag{2}$$

given by **Theorem 1.2**. The idea is that the matrix A_k captures the major associational structure in the matrix and removes the noise. Since relatively few terms are used as referents to a given document, the rectangular matrix $A = [a_{ij}]$ is quite sparse. The matrix element a_{ij} indicates the frequency with which term i occurs in document j. Depending upon the size of the database from which the term-document is generated, the matrix A can have several thousand rows and slightly fewer columns. Table 1 lists a few statistics of four large sparse term-document matrices that have been generated[2]. We note that μ_r and μ_c are the average number of nonzeros per row and column, respectively. The *Density* of each sparse matrix listed in Table 1 is defined to be the ratio (Rows $\times$ Columns) / (Nonzeros).

Data	Columns	Rows	Nonzeros	Density	μ_c	μ_r
ADI	82	374	1343	4.38	16.0	4.0
CISI	1460	5143	66340	0.88	45.4	12.9
CRAN	1400	4997	78942	1.10	56.4	15.8
MED	1033	5831	52012	0.86	50.4	8.9
TIME	425	10337	80888	1.80	190.3	7.8
TECH	6535	16637	327244	0.30	50.0	20.0

Table 1

Sparse Term-Document Matrix Specifications .

By using the *reduced model* in (2), usually with $100 \leq k \leq 200$, minor differences in terminology are

[1] *Semantic structure* refers to the correlation structure in the way in which individual words appear in documents; *semantic* implies only the fact that terms in a document may be taken as referents to the document itself or to its topic.

[2] Special thanks to Sue Dumais from Bell Communications Research (Bellcore), Morristown, NJ for providing the various sparse matrices from Latent Semantic Indexing (LSI) studies.

virtually ignored. Moreover, the closeness of objects is determined by the overall pattern of term usage, so documents can be classified together regardless of the precise words that are used to describe them, and their description depends on a consensus of their term meanings, thus dampening the effects of polysemy. As a result, terms that do not actually appear in a document may still be used as referents, if that is consistent with the major patterns of association in the data. Position in the reduced space ($\mathrm{R}(A_k)$) then serves as a new kind of *semantic indexing*.

As discussed in [4] and [12], LSI using the sparse SVD can be more robust and economical than straight term overlap methods. However, in practice, one must compute at least 100-200 largest singular values and corresponding singular vectors of sparse matrices having similar characteristics to those matrices in Table 1. In addition, it is not necessarily the case that $\mathrm{rank}(A) = n$ for the $m \times n$ term-document matrix A, this is due to errors caused by term extraction, spelling, or duplication of documents. Regarding the numerical precision of the desired singular triplets for LSI, recent tests using a few of the databases listed in Table 1 have revealed that for the i-th singular triplet, $\{u_i, \sigma_i, v_i\}$,

$$10^{-6} \leq \|Av_i - \sigma_i u_i\| \ \leq 10^{-3}$$

will suffice. Finally, as the desire for using LSI on larger and larger databases or archives grows, fast algorithms for computing the sparse singular value decomposition will become of paramount importance.

2.2. Seismic Reflection Tomography. Seismic travel tomography is another application area in which the sparse SVD problem arises. In recent literature (see [41], [42], and [7]), great interest has been expressed in the ability to compute several of the largest singular values and corresponding singular vectors.

In this application, the sparse SVD problem arises from the solution of nonlinear inverse problems associated with the approximation of acoustic or elastic wavespeed from travel times. Specifically, the travel times (data) are related to the wavespeed (model parameters) via

$$t(r) = \int_{r(s)} s(x, y, z)\ dl\ , \tag{3}$$

where x, y, and z are spatial coordinates, dl is the distance (differential) along the ray r, and $s(x, y, z) = 1/\mu(x, y, z)$ is the slowness (reciprocal of velocity) at the point (x, y, z). For large 2D problems, the travel times ($t(r)$), extracted from the original seismograms, can involve up to $O(10^5)$ rays. The ray path depends on the slowness (unknown) and thus (3) must be linearized about some initial or reference slowness model. Discretization of the slowness by cells or finite elements, within which the slowness is assumed to be constant, allows the linearized integral to be approximated as a sum. The resulting overdetermined system of linear equations for the unknown slowness perturbation values is

$$D\Delta s = \Delta t\ , \tag{4}$$

where the components of Δt are the differences between the travel times computed for the model and observed times, the components of Δs are the differences between the initial and updated model, and D is the Jacobian matrix whose (i, j) element is the distance the i-th ray travels in the j-th cell. For 2D problems, the matrix D in (4) is generally large (up to order $O(10^5)$) and sparse. Table 2 lists the statistics for two such Jacobian matrices, D, arising from seismic reflection tomography research.

Data	Columns	Rows	Nonzeros	Density	μ_c	μ_r
AMOCO1	330	1436	35210	7.43	106.7	24.5
AMOCO2	8754	9855	1159116	1.34	132.4	117.6

TABLE 2

Sparse Jacobian Matrix Specifications from Seismic Tomography .

As discussed in [41], the linear least squares solution for (4) is usually determined using the pseudo-inverse

$$D^{\dagger} = V_k \Sigma_k^{-1} U_k^T ,$$

where $k = \text{rank}(D)$, $\Sigma_k = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_k)$, and $U_k^T U_k = V_k^T V_k = I_k$ so that

$$U_k^T D V_k = \Sigma_k$$

is the SVD of the matrix D. It has been observed ([41]), that perturbations or errors comparable to the smallest singular values may introduce *large* changes in the least squares solution, Δs_{ls}, which may become quite rough (non-smooth) for *noisy* singular inverse problems. Further, the singular vectors corresponding to the the smallest singular values may control the high-spatial-frequency aspects of the solution,

$$\Delta s_{ls} = D^{\dagger} \Delta t .$$

The poorly resolved features of the inverse problem are controlled by the smallest singular triplets since these features are in the approximate null space of the Jacobian matrix, $\text{N}(D)$. Since long-wavelength rather than short-wavelength trends are usually observed from seismic travel times, inversion for velocity trends alone using travel times, $t(r)$, will force the vectors of the smallest singular triplets to be high frequency. However, as discussed in [41] and [42], inversion for subsurface velocity ($\mu(x, y, z)$) and depths-to-reflectors from travel times involves a fundamental trade-off between velocity and reflector position: increasing (decreasing) the velocity above a reflector while simultaneously increasing (decreasing) position does not affect the travel times. In this case, the singular vectors corresponding to the smallest singular values control the velocity-reflector depth trade off. If these singular values are at or below the data *noise* level, the long-wavelength velocity-reflector depth trade-off is unresolvable. Consequently, researchers in seismic reflection tomography can assess trade-offs in model parameters using sparse SVD methods which approximate large numbers of singular triplets above specified quantities (noise level).

Figure 1 depicts typical nonzero patterns of the sparse matrices arising from information retrieval and seismic tomography applications in Figure 1, where each nonzero element is given by a single dot. The matrix in 1(a) is the `ADI` database matrix (374×82) listed in Table 1. The *nearly* dense rows reflect words such as *computer* which commonly occur in each document found in that particular database. The submatrix in 1(b) comprises the first 718 rows of the 1436×330 Jacobian matrix, `AMOCO1`, in Table 2. This particular matrix was supplied by Amoco Research Center[3].

3. Singular Value Decomposition of Sparse Matrices. Before presenting methods for computing the sparse singular value decomposition, we note that classical methods for determining the SVD of dense matrices: the Golub-Kahan-Reinsch method ([19], [21]) and Jacobi-like SVD methods ([5], [6], and [25]) are not optimal for large sparse matrices. Since these methods apply orthogonal transformations (Householder or Givens) directly to the sparse matrix A, they incur excessive fill-in and thereby require tremendous amounts of memory. Moreover, it would be necessary to provide enough computer storage for a full $m \times n$ matrix A (30 megabytes for `CISI` matrix in Table 1). Another drawback to these methods for computing the SVD of dense matrices is that they will compute all the singular triplets of A, and hence may be computationally wasteful when only a few of the largest singular triplets are desired.

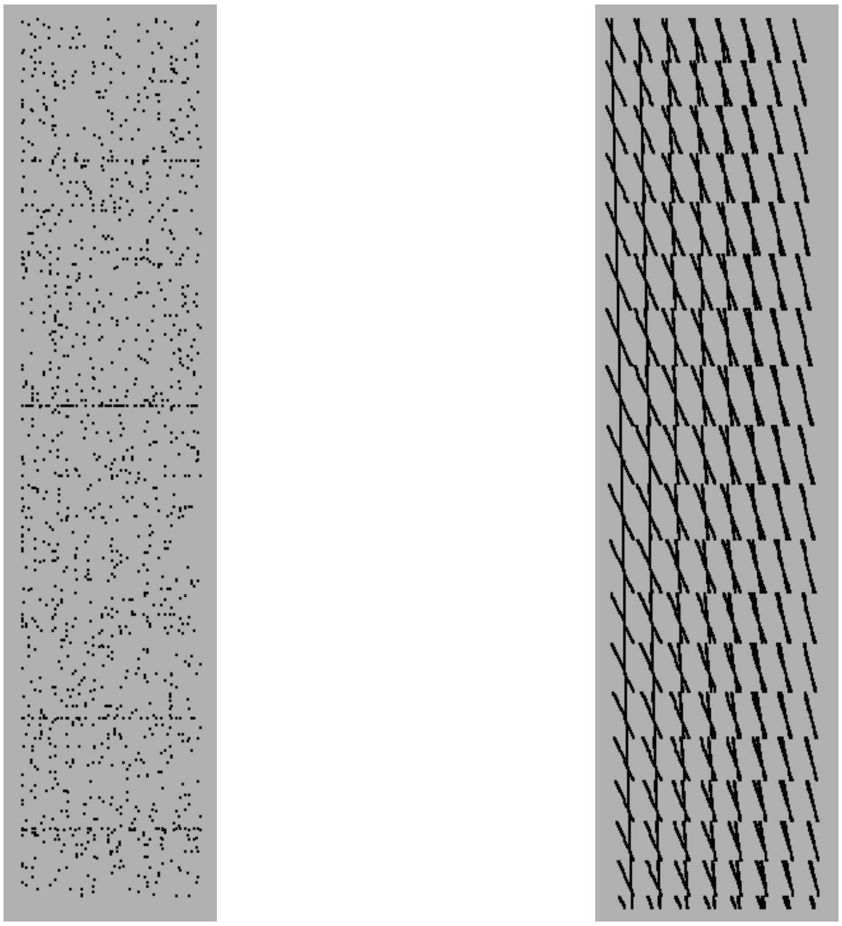
We now demonstrate how canonical sparse symmetric eigenvalue problems can be used to (indirectly) compute the sparse singular value decomposition. We then present several iterative methods which can be applied to these sparse symmetric eigenvalue problems.

3.1. Equivalent Eigenvalue Problems. Associated with an $m \times n$ ($m \geq n$) matrix A is the symmetric $(m+n) \times (m+n)$ matrix

$$B = \begin{pmatrix} O & A \\ A^T & O \end{pmatrix} . \tag{5}$$

If rank$(A) = n$, it can be easily shown that the eigenvalues of B are the n pairs, $\pm\sigma_i$, where σ_i is a singular value of A, with $(m-n)$ additional zero eigenvalues if $m > n$. The multiplicity of the zero eigenvalue of B is $m+n-2r$, where r=rank(A). The following Lemma (see [11] for proof) demonstrates how the SVD of A is generated from the eigenvalues and eigenvectors of the the matrix B in (5).

[3] Special thanks to John A. Scales from Amoco Research Center, Tulsa, OK for providing sample reflection data matrices.



(a) (b)

FIG. 1. *(a) Non-zero pattern of the* 374×82 *database matrix (*`ADI` *in Table 1) from an information retrieval application. (b) Non-zero pattern of the first* 718 *rows of a* 1436×330 *Jacobian matrix from a sample seismic travel tomography application in which subsurface velocities are needed.*

LEMMA 3.1. *Let A be an $m \times n$ $(m \geq n)$ matrix and B defined by (5).*

1. *For any positive eigenvalue, σ_i, of B let $(u_i, v_i)^T$ denote a corresponding eigenvector of norm $\sqrt{2}$. Then σ_i is a singular value of A and u_i, v_i are respectively, left and right singular vectors of A corresponding to σ_i.*
2. *For $\sigma_i = 0$, if B has corresponding orthogonal eigenvectors $(u_j, v_j)^T$ with $v_j \neq 0$ and $u_j \neq 0$ for $j = 1, ...t$ for some $t \geq 1$, then 0 is a singular value of the matrix A, and the corresponding left and right singular vectors can be obtained by orthogonalizing these u_j and v_j, respectively. Otherwise, A has full rank, i.e., rank(A) $= n$.*

The numerical accuracy of the i-th approximate singular triplet $(\tilde{u}_i, \tilde{\sigma}_i, \tilde{v}_i)$ as determined via the eigensystem of the *2-cyclic*[4] matrix B (provided $A \geq 0$) is then determined by the norm of the eigenpair residual vector r_i defined as

$$\|r_i\|_2 = \left[\|B(\tilde{u}_i, \tilde{v}_i)^T - \tilde{\sigma}_i(\ tildeu_i, \tilde{v}_i)^T\|_2\right] / \left[\|\tilde{u}_i\|_2^2 + \|\tilde{v}_i\|_2^2\right]^{\frac{1}{2}} ,$$

which can also be written as

$$\|r_i\|_2 = \left[(\|A\tilde{v}_i - \tilde{\sigma}_i\tilde{u}_i\|_2^2 + \|A^T\tilde{u}_i - \tilde{\sigma}_i\tilde{v}_i\|_2^2)^{\frac{1}{2}}\right] / \left[\|\tilde{u}_i\|_2^2 + \|\tilde{v}_i\|_2^2\right]^{\frac{1}{2}} . \tag{6}$$

We note that the interval

$$|\sigma_i - \tilde{\sigma}_i| \leq \|r_i\|_2$$

contains at least one singular value σ_i.

Alternatively, we may compute the SVD of A indirectly by the eigenpairs of either the $n \times n$ matrix A^TA or the $m \times m$ matrix AA^T. The following Lemma illustrates the fundamental relations between these symmetric eigenvalue problems and the SVD.

LEMMA 3.2. *Let A be an $m \times n$ $(m \geq n)$ matrix with rank(A) $= r$.*

1. *If $V = \{v_1, v_2, ..., v_r\}$ are linearly independent $n \times 1$ eigenvectors of A^TA so that $V^T(A^TA)V$ $=$ diag$(\sigma_1^2, \sigma_2^2, ..., \sigma_r^2, \underbrace{0, ..., 0}_{n-r})$, then σ_i is the i-th nonzero singular value of A corresponding to the right singular vector v_i. The corresponding left singular vector, u_i, is then obtained as $u_i = \frac{1}{\sigma_i}Av_i$.*
2. *If $U = \{u_1, u_2, ..., u_r\}$ are linearly independent $m \times 1$ eigenvectors of AA^T so that $U^T(AA^T)U$ $=$ diag$(\sigma_1^2, \sigma_2^2, ..., \sigma_r^2, \underbrace{0, ..., 0}_{m-r})$, then σ_i is the i-th nonzero singular value of A corresponding to the left singular vector u_i. The corresponding right singular vector, v_i, is then obtained as $v_i = \frac{1}{\sigma_i}A^Tu_i$.*

[4] A non-negative irreducible matrix B which is 2-cyclic has 2 eigenvalues of modulus $\rho(B)$, where $\rho(B)$ is the spectral radius of B. See Definition 2.2 on page 35 in [46].

Computing the SVD of A via the eigensystems of either A^TA or AA^T may be adequate for determining the largest singular triplets of A, but the loss of accuracy can be severe for the smallest singular triplets (see [11]). Whereas the smallest and largest singular values of A are the extremes of the spectrum of A^TA or AA^T, the smallest singular values of A lie at the center of the spectrum of B in (5). For computed eigenpairs of A^TA and AA^T, the norms of the i-th eigenpair residuals (corresponding to (6)) are given by

$$\|r_i\|_2 = \|A^TA\tilde{v}_i - \tilde{\sigma}_i^2\tilde{v}_i\|_2 \ / \ \|\tilde{v}_i\|_2$$

and

$$\|r_i\|_2 = \|AA^T\tilde{u}_i - \tilde{\sigma}_i^2\tilde{u}_i\|_2 \ / \ \|\tilde{u}_i\|_2 \ ,$$

respectively. Thus, extremely high precision in computed eigenpairs may be necessary to compute the smallest singular triplets of A. Difficulties in approximating the smallest singular values by any of the three equivalent symmetric eigenvalue problems are discussed in [4].

3.2. Subspace Iteration. Subspace iteration is perhaps one of the simplest algorithms used to solve large sparse eigenvalue problems. As discussed in [33], it can be viewed as a block generalization of the classical power method. The simplest version of subspace iteration was introduced by Bauer ([3]) and if adapted to the matrix B in (5) would involve forming the sequence

$$Z_k = B^k Z_0 \ ,$$

where $Z_0 = [z_1, z_2, \cdots, z_s]$ is an $(m+n) \times s$. If the column vectors, z_i, are normalized separately (as done in the power method), then these vectors will converge to the dominant eigenvector of B. Thus, the matrix Z_k will progressively lose the linear independence of its columns. In order to approximate the p-largest eigenpairs of B, Bauer demonstrated that linear independence among the z_i's could be maintained if they were orthogonalized at each step, say by a modified Gram-Schmidt procedure. However, the convergence rate of the z_i's to eigenvectors of B would only be linear. For example, suppose we define E_k as the linear space spanned by the k columns of Z_k so that

$$E_k = \{z | z = \tilde{B}x, \ x \in E_{k-1}\},$$

where $\tilde{B} = B + \gamma I_{m+n}$ and γ is chosen so that $\tilde{B}$ is (symmetric) positive definite. As illustrated in [36], the existence of

$$\bar{z} = \lim_{k \to \infty} E_k \ ,$$

is assured and the angle, $\Phi_j^{(k)}$, between the j-th eigenvector of $\tilde{B}$ and E_k would be

$$\mathcal{O}\left(\frac{\sigma_s + \gamma}{\sigma_j + \gamma}\right)^k \ , \tag{7}$$

where σ_j is the j-th largest singular value of the $m \times n$ matrix A. Hence, one should choose γ to be as small as possible.

The most sophisticated implementation of subspace iteration is that of Rutishauser's *ritzit* program (see [37]). This particular algorithm incorporates both a Rayleigh-Ritz procedure and acceleration via Chebyshev polynomials. The iteration which embodies the *ritzit* program is given in Table 3. The Rayleigh Quotient matrix, H_k, in step (3) is essentially the projection of $\tilde{B}^2$ onto the span(Z_{k-1}). The three-term recurrence in step (6) follows from the adaptation of the Chebyshev polynomial of degree q, say $T_q(x)$, to the interval $[-e, e]$, where e is chosen to be the smallest eigenvalue of H_k. We recall that the Chebyshev polynomial $T_q(x)$, of the first kind, can be defined as

$$T_q(x) = \begin{cases} cos(q\ arccos(x)), & \text{if } |x| \leq 1, \\ cosh(q\ arccosh(x)), & \text{otherwise} \end{cases} . \tag{8}$$

This use of Chebyshev polynomials has the desired effects of damping unwanted eigenvalues of $\tilde{B}$ (or B) and producing an improved rate of convergence

$$\mathcal{O}\left(\frac{T_q(\theta_s)}{T_q(\theta_1)}\right)$$

which is considerably smaller in magnitude than the original rate of convergence θ_s/θ_1 $(\theta_1 \geq \theta_2 \geq \cdots \geq \theta_s)$, where θ_i are the eigenvalues of H_k given by the square roots of the diagonal matrix Δ_k^2 in step (4) of Table 3. We note that if the matrix B is used to compute indirectly the singular values of A, we certainly forfeit the positive definiteness in the equivalent symmetric eigenvalue problem and must choose an initial subspace of dimension $s \geq 2p$ in order to approximate the p-largest singular values of A, σ_i, which occur as $\pm\sigma_i$ within the interval $[-\theta_s, \theta_s]$. To circumvent this problem in the *ritzit* program, one instead may compute the eigenvectors of the 2-cyclic matrix

$$\tilde{B} = \begin{pmatrix} \gamma I & A \\ A^T & \gamma I \end{pmatrix} , \tag{9}$$

where γ is an estimate for the largest singular value of A, or choose $\tilde{B} = A^T A$ and solve the equivalent n-th order symmetric positive definite eigenvalue problem (see Section 3.1). In practice, γ can be chosen as either $||A||_1$ or $||A||_\infty$ depending upon the sparse data structure used to store the nonzero elements (row or column-wise). Once, the eigenvectors of $\tilde{B}$ have been determined, one can easily recover the eigenvalues of B via Rayleigh quotients. Table 4 lists three possible Chebyshev intervals of the form $[-e, e]$ associated with operators of equivalent symmetric eigenvalue problems for the SVD of the matrix A. Although, any of these three intervals can be used for damping unwanted eigenvalues of B or $\tilde{B}$, the smallest interval associated with the 2-cyclic matrix B will certainly provide the most efficient damping (low degree Chebyshev polynomials $T_q(x)$). To compute the p-largest singular triplets of A via the eigensystem of B, one must then select $s \geq 2p$, and thus iterate within a subspace having twice the dimension of the eigenspace sought.

The primary cost of this iterative method lies in the total number of sparse matrix-vector multiplications required. If $s \geq p$ vectors, z_i, are used to approximate the p-largest eigenvectors of the $(m+n) \times (m+n)$ matrix $\tilde{B}$, the cost in floating-point operations per iteration would be

$$s \times [2(1+\mu_r)m \; + \; 2(1+\mu_c)n] \; , \tag{10}$$

where μ_r and μ_c are the average number of nonzeros per row and column, respectively. The number of sparse matrix-vector multiplications is one performance metric that we shall use in Section 5 to assess the complexity of our candidate iterative methods for computing the sparse SVD. The orthogonal

(1)	Compute C_k	$=$	$\tilde{B} Z_{k-1}$
(2)	Factor C_k	$=$	$Q_k R_k$
(3)	Form H_k	$=$	$R_k R_k^T$
(4)	Factor H_k	$=$	$P_k \Delta_k^2 P_k^T$
(5)	Form Z_k	$=$	$Q_k P_k$
(6)	Iterate Z_{k+j}	$=$	$\frac{2}{e} \tilde{B} Z_{k+j-1} - Z_{k+j-2}$
			$(j = 2, ..., q)$

Table 3

Subspace Iteration as implemented in Rutishauser's **ritzit** .

factorization in step(2) of Table 3 may be computed by a modified Gram-Schmidt procedure or by Householder transformations provided that the orthogonal matrix Q_k is explicitly available for the computation of Z_k in step (5). On multiprocessor architectures, especially those having hierarchical memories such as the Alliant FX/80 and Cray-2S, one may achieve high performance (with a slight increase in the total number of arithmetic operations) by using either a block Gram-Schmidt or block Householder orthogonalization method in step(2). As discussed in [18], significant improvements in the algorithmic performance of fundamental linear algebra kernels may be gained through the improved data locality associated with block-based methods. To improve upon the two-sided Jacobi algorithm originally suggested by Rutishauser ([36]) for the spectral decomposition (step (4)) in *ritzit*, one may employ a parallel two- or one-sided Jacobi method (see [6]) on a multiprocessor. In fact, the one-sided Jacobi scheme, when appropriately adapted for symmetric positive definite matrices (see [6]), is quite appropriate for step(4) provided the dimension of the current subspace, s, is not too large (e.g., $s \leq 100$). For larger subspaces, an optimized implementation of the classical EISPACK ([44]) pair, TRED2 and TQL2, or Cuppen's algorithm as parallelized by Dongarra and Sorensen ([15]) would suffice for step (4).

In the next section, we discuss an alternative subspace method which would appear to be more

B, $\tilde{B}$	e
$\begin{pmatrix} O & A \\ A^T & O \end{pmatrix}$	θ_s
$\begin{pmatrix} \gamma I & A \\ A^T & \gamma I \end{pmatrix}$	$\theta_s + \gamma$
$A^T A$	θ_s^2

TABLE 4

Intervals for Polynomial Acceleration within Subspace Iteration Method.

suitable for multiprocessors than subspace iteration in that the desired singular triplets are iterated upon (for the most part) in parallel.

3.3. Trace Minimization Method. Another candidate subspace method for the SVD of sparse matrices is based upon the trace minimization algorithm discussed in [40] and [49] for the generalized eigenvalue problem

$$Hx = \lambda G x \ , \tag{11}$$

where H and G are symmetric and G is also positive definite. In order to compute the SVD of an $m \times n$ matrix A, we initially replace H with $\tilde{B}$ as defined in (9) or set $H = A^T A$. Since we need only consider equivalent standard symmetric eigenvalue problems (see Section 3.1), we simply define $G = I_{m+n}$ (or I_n if $H = A^T A$). Accordingly, our appropriate trace minimization SVD scheme (TRSVD) is then based upon the following theorem which is a direct consequence of the Courant-Fischer theorem (see [47]). Without loss of generality, let us assume that $H = \tilde{B}$, $G = I_{m+n}$ and consider the associated symmetric eigensystem of order $m + n$.

THEOREM 3.3. *Let $\tilde{B}$ be as given in (9) and let $\mathcal{Y}$ be the set of all $(m+n) \times p$ matrices Y for which $Y^T Y = I_p$. Then*

$$\min_{Y \in \mathcal{Y}} \operatorname{trace}(Y^T \tilde{B} Y) = p\gamma - \sum_{i=1}^{p} \sigma_i \ ,$$

where σ_i is a singular value of A, $\lambda_i = \gamma \pm \sigma_i$ is an eigenvalue of $\tilde{B}$, and
$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p$. Given an $(m+n) \times p$ matrix Y which forms a *section* of the eigenvalue problem

$$\tilde{B} z = \lambda z \ , \tag{12}$$

i.e.,

$$Y^T \tilde{B} Y = \tilde{\Sigma} \ , Y^T Y = I_p \ , \tag{13}$$

$$\tilde{\Sigma} = \mathrm{diag}(\tilde{\sigma}_1, \tilde{\sigma}_2, \cdots, \tilde{\sigma}_p) \ ,$$

the TRSVD scheme finds a sequence of iterates $Y_{k+1} = F(Y_k)$, where both Y_k and Y_{k+1} form a section of (12), and have the property $\mathrm{trace}(Y_{k+1}^T \tilde{B} Y_{k+1}) < \mathrm{trace}(Y_k^T \tilde{B} Y_k)$. From Theorem 3.3, the matrix Y in (13) which minimizes $\mathrm{trace}(Y^T \tilde{B} Y)$ is the matrix of $\tilde{B}$-eigenvectors associated with the p-smallest eigenvalues of the problem (12). As discussed in [40] and [49], $F(Y)$ can be chosen so that global convergence is assured. Moreover, (12) can be regarded as the quadratic minimization problem

$$\text{minimize } \mathrm{trace}(Y^T \tilde{B} Y) \tag{14}$$

subject to the constraints

$$Y^T Y = I_p \ . \tag{15}$$

Using Lagrange multipliers, this quadratic minimization problem leads to solving the $(m+n+p) \times (m+n+p)$ system of linear equations

$$\begin{pmatrix} \tilde{B} & Y_k \\ Y_k^T & 0 \end{pmatrix} \begin{pmatrix} \Delta_k \\ L \end{pmatrix} = \begin{pmatrix} \tilde{B} Y_k \\ 0 \end{pmatrix} \ , \tag{16}$$

so that $Y_{k+1} \equiv Y_k - \Delta_k$ will be an optimal subspace iterate. We note that Peters and Wilkinson ([35]) also considered equations of this form in comparing Newton and inverse iterations.

Since the matrix $\tilde{B}$ is positive definite (by construction), one can alternatively consider the independent (parallel) subproblems

$$\text{minimize } \mathrm{trace}(\ (y_j^{(k)} - d_j^{(k)})^T \tilde{B} (y_j^{(k)} - d_j^{(k)}) \ ) \tag{17}$$

subject to the constraints

$$Y^T d_j^{(k)} = 0 \ , \ j = 1, 2, \ldots, p,$$

where $d_j^{(k)} = \Delta_k e_j$, e_j is a vector composed of all zeros except for the value 1 in the j-th component, and $Y_k = [y_1^{(k)}, y_2^{(k)}, \ldots, y_p^{(k)}]$. The corrections Δ_k in this case are selected to be orthogonal to the previous estimates Y_k (15), i.e., so that (see [28])

$$\Delta_k^T Y_k = 0.$$

We then recast (16) as

$$\begin{pmatrix} \tilde{B} & Y_k \\ Y_k^T & 0 \end{pmatrix} \begin{pmatrix} d_j^{(k)} \\ l \end{pmatrix} = \begin{pmatrix} \tilde{B} y_j^{(k)} \\ 0 \end{pmatrix} \ , j = 1, 2, \ldots, p, \tag{18}$$

where $2l$ is a vector of order p reflecting the Lagrange multipliers.

The solution of the p systems of linear equations in (18) can be done in parallel by either a direct or iterative solver. Since the original matrix A is assumed to be large, sparse, and without any particular sparsity structure (pattern of nonzeros) we have chosen an iterative method (conjugate gradient) for the systems in (18). Following [40] and dropping the subscripts in (18), let

$$Y_k = Q_k R = [Q_1, Q_2] R \ , \tag{19}$$

be the orthogonal factorization of Y_k so that $R^T = [\tilde{R}^T, 0]$, $\tilde{R}$ is $p \times p$ and upper triangular, $Q_k^T Q_k = I_{m+n}$, and Q_1 is $(m+n) \times p$. We may then rewrite (18) as

$$\begin{pmatrix} Q_k^T \tilde{B} Q_k & R_k \\ R_k^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{d}_j^{(k)} \\ l \end{pmatrix} = \begin{pmatrix} Q_k^T \tilde{B} y_j^{(k)} \\ 0 \end{pmatrix} \ , j = 1, 2, \ldots, p, \tag{20}$$

where $\tilde{d}_j^{(k)} = Q_k^T d_j^{(k)}$. If we partition $\tilde{d}_j^{(k)} = [g_j^{(k)}, h_j^{(k)}]$ so that $g_j^{(k)}$ is a vector of order p, then we must have $g_j^{(k)} = 0$ since $R^T \tilde{d}_j^{(k)} = 0$ and $\tilde{R}$ is nonsingular. Following [40], we may solve for $h_j^{(k)}$ in

$$Q_2^T \tilde{B} Q_2 h_j^{(k)} = Q_2^T \tilde{B} y_j^{(k)} \ , \tag{21}$$

via the conjugate gradient method, [23], and recover $d_j^{(k)}$ via

$$d_j^{(k)} = Q_k \tilde{d}_j^{(k)} = Q_2 h_j^{(k)}. \tag{22}$$

Using the fact that $I - Q_1 Q_1^T = Q_2 Q_2^T$ is the orthogonal projector onto the nullspace of Y_k and (22), the conjugate gradient method for solving the systems in (21) is simplified as shown in Table 5 where $P = I - Q_1 Q_1^T$. We note that although the orthogonal factorization (19) is necessary in early iterations of TRSVD, it may be avoided in the final iterations (ultimate convergence to eigenvectors of $\tilde{B}$).

Rather than iterating for the corrections, $d_j^{(k)}$, the conjugate gradient method in Table 5 produces iterates of the form $y_j^{(k)} - d_j^{(k)}$ so that

$$\lim_{l \to \infty} z_l = y_j^{(k+1)} \ .$$

The termination criteria in Step (3d) achieve two purposes. The requirement

$$\alpha_l \|p_l\|_2 \leq \mu \ ,$$

assures that increments to $y_j^{(k)}$ are within the machine-unit roundoff error, μ. On the other hand, the criterion

$$(\alpha_l \|r_{l+1}\|_2^2)/(\alpha_0 \|r_0\|_2^2) \leq \tau(y_j^{(k)})$$

is used to estimate an upper bound for the required CG error

$$(z_{l+1} - z_l)^T \tilde{B} (z_{l+1} - z_l) \approx \alpha_l \|r_l\|_2^2 \ .$$

(1)	$r_0 = P\tilde{B}z_0,\ z_0 = y_j^{(k)},$
(2)	$p_0 = r_0,$
(3)	For $l = 0, 1, 2, \ldots,$ do:
(3a)	$\alpha_l = r_l^T r_l / p_l^T \tilde{B} p_l,$
(3b)	$z_{l+1} = z_l - \alpha_l p_l,$
(3c)	$r_{l+1} = r_l - \alpha_l P \tilde{B} p_l,$
(3d)	If $(\alpha_l \|r_{l+1}\|_2^2)/(\alpha_0 \|r_0\|_2^2) \leq \tau(y_j^{(k)})$
	or $\alpha_l \|p_l\|_2 \leq \mu$, stop,
(3e)	$\beta_l = r_{l+1}^T r_{l+1} / r_l^T r_l,$
(3f)	$p_{l+1} = r_{l+1} + \beta_l p_l.$

TABLE 5

Conjugate gradient method for the quadratic minimization step of TRSVD.

In other words, we need not iterate in the CG algorithm beyond a certain factor $(\tau(y_j^{(k)}))$ of the initial CG error, $\alpha_0\|r_0\|_2^2$. Here $\tau(y_j^{(k)})$ is the convergence rate of $y_j^{(k)}$ to the k-th smallest eigenvector of $\tilde{B}$ (or the k-th largest left and right singular vectors of A). We note that the first m and last n components of $y_j^{(k)}$ approximate left and right singular vectors of A, respectively. Following the discussion of convergence rates for trace minimization in [40] when applied to (11), we have

$$\tau(y_j^{(k)}) = (\sigma_j/\sigma_p)^2 \ ,$$

which corresponds to the convergence rate of subspace iteration in (7). Hence, a major reason for our choice of the conjugate gradient (CG) method for the solution of (18) stems from the ability to terminate CG iterations early without obtaining fully- accurate corrections $d_j^{(k)}$ that are more accurate than warranted. In later stages, however, as Y_k converges to the desired set of eigenvectors of $\tilde{B}$, one needs full accuracy in computing the correction matrix Δ_k.

3.3.1. Polynomial Acceleration Technique for TRSVD. The Chebyshev acceleration strategy used within subspace iteration (see Section 3.2) can also be applied to TRSVD. However, to dampen unwanted singular values of A in this context we must solve the generalized eigenvalue problem (as opposed to (12))

$$x = \frac{1}{P_q(\lambda)} P_q(\tilde{B}) x \ , \tag{23}$$

where $P_q(x) = T_q(x) + \epsilon I_{m+n}$, $T_q(x)$ is the Chebyshev polynomial of degree q defined by (8), and ϵ is chosen so that $P_q(\tilde{B})$ is (symmetric) positive definite. The appropriate quadratic minimization problem similiar to (17) for (23) can be expressed as

$$\text{minimize trace}(\ (y_j^{(k)} - d_j^{(k)})^T (y_j^{(k)} - d_j^{(k)}) \ ) \tag{24}$$

subject to the constraints

$$Y^T P_q(\tilde{B}) d_j^{(k)} = 0 \ , \ j = 1, 2, \ldots, p.$$

In effect, we then approximate the smallest singular values of A (or eigenvalues of $\tilde{B}$) as the *largest* eigenvalues of the matrix $P_q(\tilde{B})$ whose gaps are considerably larger than those of the eigenvalues of $\tilde{B}$.

Although the additional number of sparse matrix-vector multiplications associated with the multiplication by $P_q(\tilde{B})$ will be significant for high degrees q, the system of equations via Lagrange multipliers in (18) becomes much easier to solve, i.e.,

$$\begin{pmatrix} I & P_q(\tilde{B}) Y_k \\ Y_k^T P_q(\tilde{B}) & 0 \end{pmatrix} \begin{pmatrix} d_j^{(k)} \\ l \end{pmatrix} = \begin{pmatrix} y_j^{(k)} \\ 0 \end{pmatrix} \ , j = 1, 2, \ldots, p. \tag{25}$$

It is easy to show that the updated eigenvector approximation, $y_j^{(k+1)}$, is determined by

$$y_j^{(k+1)} = y_j^{(k)} - d_j^{(k)} = P_q(\tilde{B}) Y_k \left[Y_k^T P_q^2(\tilde{B}) Y_k \right]^{-1} Y_k^T P_q(\tilde{B}) y_j^{(k)} \ .$$

Thus, we need not employ the use of an iterative solver for determining Y_{k+1} since the matrix $\left[Y_k^T P_q^2(\tilde{B}) Y_k \right]^{-1}$ is of order p and using the orthogonal factorization

$$P_q(\tilde{B}) Y_k = \hat{Q} \hat{R} \ ,$$

we have

$$\left[Y_k^T P_q^2(\tilde{B}) Y_k \right]^{-1} = \hat{R}^{-T} \hat{R}^{-1} \ .$$

The control of the polynomial degree, q, is determined by the strategy discussed in [37] and [4]. Table 6 lists three possible intervals for damping the unwanted singular values of A which correspond to the choice of H in (11). If we choose $H = \tilde{B}$, then for the TRSVD iteration in polynomial aceleration starts, we apply $P_q(B)$, where B is the 2-cyclic matrix in (5). This second shift by γ (to the left) will allow us to use the same interval suggested by Rutishauser for the *ritzit* program (see Section 3.2). If we desire the p-smallest singular values of A and select $H = A^T A$, then we must fix the right-hand endpoint to be γ^2, where $\gamma = \|A\|_1$ or $\|A\|_\infty$. Of course, if some knowledge of the spectrum of A

is available, say an estimate for the largest singular value, σ_{max}, then γ should be defined as any available *least* upper bound for σ_{max}. To approximate the p-largest singular values of A we may use the same interval but must choose $H = \gamma^2 I - A^T A$ so that $\gamma^2 - \sigma_i^2$, is the i-th smallest eigenvalue of H corresponding to the i-th largest singular value of A.

H	I
B	$[-\sigma_s, \sigma_s]$
$A^T A$	$[\sigma_s^2, \gamma^2]$
$\gamma^2 I - A^T A$	$[-\sigma_s^2, \sigma_s^2]$

TABLE 6

Intervals for Polynomial Acceleration within TRSVD.

3.3.2. Forming a Section. As with subspace iteration, our trace minimization-based method (TRSVD) will best approximate the p-largest singular triplets of the sparse matrix A if $s > p$ vectors (i.e., Y_k is $(m+n) \times s$ for $H = \tilde{B}$) are carried in the early iterations (prior to deflation). Let us assume that the TRSVD iterate Y_k is $(m+n) \times s$ and that we have yet to employ Chebyshev acceleration. To form a section of (12) as specified by (13), we must first orthogonalize the columns of Y_k, i.e., $y_j^{(k)}$, $j = 1, 2, \ldots, s$ as in (19). We can apply a (block) modified Gram-Schmidt orthogonalization procedure (see [18]) to orthogonalize the columns of Y_k, and then obtain (13) by determining the spectral decomposition of $F_k = Y_k^T \tilde{B} Y_k$,

$$W^T F_k W = \tilde{\Sigma} \ , \tag{26}$$

where W is an orthogonal matrix of order s. If we then define $Y_k \equiv Y_k W$, then

$$Y_k^T \tilde{B} Y_k = \tilde{\Sigma}, \ Y_k^T Y_k = I_p \ ,$$

where $\tilde{\Sigma}$ is given in (13). As with step (4) in Table 3 for subspace iteration, we may employ either a Jacobi method (one- or two-sided) or the classical tridiagonalization strategy in EISPACK to determine (26). Our current implementation of TRSVD employs an adaptation of a parallel one-sided Jacobi method (see [5], [6]) for symmetric positive definite eigensystems.

If Chebyshev acceleration (23) is used in iteration k of TRSVD, then we must generate $P_q(\tilde{B})$-orthogonal vectors $y_j^{(k)}$, i.e., we must have

$$Y_k^T P_q(\tilde{B}) Y_k = I_p \ , Y_k^T Y_k = \tilde{\Sigma} \ , \tag{27}$$

where the products $P_q(\tilde{B})y_i^{(k)}$, $i = l+1, l+2, \ldots, s$ can be formed in parallel.

3.3.3. Shifting Strategy for TRSVD. As discussed in [40], we can also accelerate the convergence of the Y_k to eigenvectors of $\tilde{B}$ (and hence singular vectors of A) by incorporating Ritz shifts (see [33]) into TRSVD. Specifically, we modify the symmetric eigenvalue problem in (12) as

$$(28) \qquad (\tilde{B} - \nu_j^{(k)} I) z_j = (\lambda_j - \nu_j^{(k)}) z_j \;, j = 1, 2, \ldots, s \;,$$

where $\nu_j^{(k)} = \tilde{\sigma}_j^{(k)}$ is the j-th approximate eigenvalue (13) from the k-th TRSVD iteration, and λ_j, z_j are an exact eigenpair of $\tilde{B}$. In other words, we simply use our most recent approximations to the eigenvalues of $\tilde{B}$ from our k-th section within TRSVD as Ritz shifts. As was shown by Wilkinson in [48], the Rayleigh quotient iteration associated with (28) will ultimately achieve cubic convergence to $\gamma - \sigma_j$, where σ_j is an exact singular value of A, provided $\nu_j^{(k)}$ is sufficiently close to $\gamma - \sigma_j$. However, since we have $\nu_j^{(k+1)} < \nu_j^{(k)}$ for all k (see Theorem 3.3), i.e., we approximate eigenvalues of $\tilde{B}$ from above, $\tilde{B} - \nu_j^{(k)} I$ will not be positive definite and thus we cannot guarantee the convergence of this shifted TRSVD method for any particular singular triplet j. However, the strategy outlined in [4] has been quite successful in maintaining global convergence with shifting.

The logic of the TRSVD method which appropriately utilizes polynomial (Chebyshev) acceleration prior to Ritz shifting is outlined in Table 7. It is important to note that once shifting has been invoked (Step (4)) we abandon the use of Chebyshev polynomials $P_q(\tilde{B})$ and solve shifted systems ($\tilde{B}$ replaced by $\tilde{B} - \nu_j^{(k)} I$) of the form (16) via (21) and the CG algorithm in Table 5. The *context switch* from either non-accelerated or polynomial-accelerated trace minimization iterations to trace minimization iterations with Ritz shifting is accomplished by monitoring the reduction of the residuals (6) for isolated eigenvalues ($r_j^{(k)}$) or clusters of eigenvalues ($R_j^{(k)}$). Measuring the reduction is facilitated by the use of Gershgorin disks in much the same way the block size for the block Lanczos method will be adjusted in Section 3.5. We simply compute Gershgorin bounds via F_k during the spectral decomposition phase (26) of forming a section. If D_j is a Gershgorin disk defined by

$$(29) \qquad D_j = \left\{ z \in \Re \;:\; |z - F_k[j,j]| \le \sum_{l=1}^{s} |F_k[j,l]| \right\} \;,$$

then $\tilde{\sigma}_j^{(k)}$ is an isolated eigenvalue approximation provided

$$D_j \bigcap D_l = \emptyset, \;\; \forall l \neq j \;.$$

Similarly, $\{\tilde{\sigma}_l^{(k)}, \tilde{\sigma}_{l+1}^{(k)}, \ldots, \tilde{\sigma}_{l+c}^{(k)}\}$ form a cluster of size c provided

$$\bigcap_{i=l}^{l+c} D_i \neq \emptyset \;.$$

For an isolated eigenvalue approximation $\tilde{\sigma}_j^{(k)}$, which is detected say after k_0 TRSVD iterations, we monitor succeeding iterations ($k > k_0$), and determine if the norm of the current residual ($\|r_j^{(k)}\|_2$)

is less than a chosen order of magnitude ($\eta = 10^{-t}$, for a small integer t) of $\|r_j^{(k_0)}\|_2$. Thus, the parameter η serves as our control for the context switch from polynomial-based acceleration to shift-based acceleraton in TRSVD. The value of η will naturally depend upon the desired accuracy of the singular triplets sought, since we would like to produce suitable shifts ($\nu_k^j \equiv \tilde{\sigma}_k^j$) for fast convergence of y_k^j to an eigenvector of $\tilde{B}$. However, for most problems we have considered (including Tables 1 and 2) we can obtain optimal convergence rates for $\eta = 10^{-1}, 10^0$.

(0) Set $k = 0$, choose an initial $n \times s$ subspace iterate $Y_0 = [y_1^{(0)}, y_2^{(0)}, \cdots, y_s^{(0)}]$.
and select polynomial acceleration (if desired).

(1) Form a section as in (13) or (27) if Chebyshev acceleration is used.

(2) Compute residuals: $r_j^{(k)} = \tilde{B}y_j^{(k)} - \tilde{\sigma}_j^{(k)} y_j^{(k)}$, where $(\tilde{\sigma}_j^{(k)},\, y_j^{(k)})$ is the j-th eigenpair approximation from TRSVD iteration k, and access accuracy of eigenpairs of $\tilde{B}$ (singular triplets of A).

(3) Analyze the current approximate spectrum using Gershgorin disks in (29) with $H_k \equiv F_k$, where F_k is defined in (26), and $bk \equiv s$.
(Monitor isolated and clustered eigenvalues.)

(4) Invoke Ritz shifting strategy ([4]):
For isolated eigenvalues:
if $\|r_j^{(k)}\|_2 \leq \eta \|r_j^{(k_0)}\|_2$, where $\eta \in [10^{-3}, 10^0]$ and $k_0 < k$ for some j .
For a cluster of eigenvalues (size c):
if $\|R_j^{(k)}\|_F \leq \eta \|R_j^{(k_0)}\|_F$, where $R_j^{(k)} \equiv \{r_j^{(k)}, \cdots, r_{j+c}^{(k)}\}$ and $k_0 < k$ for some j.
(Disable polynomial acceleration if shifting is selected.)

(5) Deflation: reduce subspace dimension, s, by number of approximate $\tilde{B}$-eigenpairs accepted for singular triplets of A.

(6) Adjust polynomial degree q (see [37] or [4]) for $P_q(\tilde{B})$ in iteration $k+1$ (if needed).

(7) Update subspace iterate $Y_{k+1} \equiv Y_k - \Delta_k$ via (16) or (25).

(8) Set $k = k + 1$ and go to Step (1).

TABLE 7

TRSVD *Algorithm with Chebyshev Acceleration and Ritz shifts .*

3.4. Single-Vector Lanczos Method. Other popular methods for solving large, sparse, symmetric eigenproblems originated from a method attributed to Lanczos (1950). This method generates a sequence of tridiagonal matrices T_j with the property that the extremal eigenvalues of the $j \times j$ matrix T_j are progressively better estimates of the original matrix B's extremal eigenvalues. Suppose we consider the $(m+n) \times (m+n)$ 2-cyclic matrix B given in (5), where A is the $m \times n$ matrix whose singular triplets are sought, and let v_1 be a randomly generated starting $(m+n) \times 1$ vector such that $\|v_1\|_2 = 1$. For $j = 1, 2, ..., l$ define the corresponding Lanczos matrices T_j using the following recursion ([32]). Define $\beta_1 \equiv 0$, and $v_0 \equiv 0$. Then for $i = 1, 2, ..., l$ define Lanczos vectors w_i and scalars α_i and β_{i+1} where

$$\beta_{i+1} w_{i+1} = B w_i - \alpha_i w_i - \beta_i w_{i-1} \ , \ \text{and} \tag{30}$$

$$\alpha_i = w_i^T (B w_i - \beta_i w_{i-1})$$

$$|\beta_{i+1}| = \|B w_i - \alpha_i w_i - \beta_i w_{i-1}\|_2 \ .$$

For each j, the corresponding Lanczos matrix T_j is defined as a real symmetric, tridiagonal matrix having diagonal entries α_i $(1 \leq i \leq j)$, and subdiagonal (superdiagonal) entries β_{i+1} $(1 \leq i \leq (j-1))$, i.e.,

$$T_j \equiv \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \beta_j \\ & & & & \beta_j & \alpha_j \end{pmatrix} . \tag{31}$$

By definition, the vectors $\alpha_i w_i$ and $\beta_i w_{i-1}$ in (30) are respectively, the orthogonal projections of $B w_i$ onto the most recent w_i and w_{i-1}. Hence for each i, the next Lanczos vector w_{i+1} is obtained by orthogonalizing $B w_i$ with respect to w_i and w_{i-1}. The resulting α_i, β_{i+1} obtained in these orthogonalizations define the corresponding Lanczos matrices. If we rewrite (30) in matrix form, then for each j we have

$$B W_j = W_j T_j + \beta_{j+1} w_{j+1} e_j^T \ , \tag{32}$$

where $W_j \equiv [w_1, w_2, ..., w_j]$ is the $n \times j$ matrix whose k-th column is the k-th Lanczos vector, and e_j^T is the j-th column of the $n \times n$ identity matrix. Thus, the Lanczos recursion (32) generates a family of real symmetric tridiagonal matrices related to both B and w_1. Table 8 outlines the basic Lanczos procedure for computing the eigenvalues and eigenvectors of the symmetric 2-cyclic matrix B.

As with subspace iteration, the matrix B is only referenced through matrix-vector multiplication in Table 8. At each iteration, the basic Lanczos recursion requires only the two most recently-generated vectors, although for finite-precision arithmetic modifications suggested by Grcar [24], Parlett and Scott [34], and Simon [43] require additional Lanczos vectors to be readily accessible via secondary storage. We note that on a multiprocessor architecture, step (2) in Table 8 may benefit from any

(1)	Use any variant of the Lanczos recursion (30) to generate a family of real symmetric tridiagonal matrices, T_j $(j = 1, 2, ..., q)$.
(2)	For some $k \leq q$, compute relevant eigenvalues of T_k.
(3)	Select some or all of these eigenvalues as approximations to the eigenvalues of the matrix B, and hence singular values of A.
(4)	For each eigenvalue λ compute a corresponding unit eigenvector z such that $T_k z = \lambda z$. Map such vectors into corresponding Ritz vectors $y \equiv W_q z$, which are then used as approximations to the desired eigenvectors (singular vectors) of the matrix B (A).

TABLE 8

Single-Vector Lanczos Recursion.

available optimized library routine that solves the symmetric tridiagonal eigenvalue problem (eg., multisectioning in [29] or divide and conquer in [15]).

Theoretically, it is easy to prove that for the basic Lanczos recursion that orthogonalization with respect to only the two most recently-generated Lanczos vectors is sufficient to guarantee that each succeeding Lanczos vector is orthogonal with respect to all previously generated Lanczos vectors (see [11] or [33]). In one sense, the Lanczos procedure can be viewed as the Gram-Schmidt orthogonalization of the set of Krylov vectors $w_1, Bw_1, ..., B^{k-1}w_1$. Alternatively, span $\{W_j\}$ is a Krylov subspace for the matrix B, and the Lanczos procedure is a mechanism for generating orthonormal bases for these Krylov subspaces and for computing the orthogonal projection of B onto these subspaces. Computing the eigenvalues of the T_j's is equivalent to computing the best approximations to the eigenvalues and eigenvectors of B restricted to the corresponding Krylov subspaces. The accuracy of these approximations have been studied in detail by Kaniel ([26]) and more recently by Saad ([38]). Saad improved the original error bounds of Kaniel, however, but these results still indicate deterioration of accuracy of the computed eigenvalues and of the corresponding Ritz vectors as we move to the interior of the spectrum of B.

In using finite-precision arithmetic, any practical Lanczos procedure must address problems created by losses in the orthogonality of the Lanczos vectors, w_i. Such problems include the occurrence of numerically-multiple eigenvalues of T_j (for large j) for simple eigenvalues of B, and the appearance of spurious eigenvalues among the computed eigenvalues for some T_j. Approaches to deal with these problems range from two different extremes. The total reorthogonalization of every Lanczos vector with respect to every previously-generated Lanczos vector is one extreme ([23]). The other approach accepts the loss in orthogonality and then deals with these problems directly. Total reorthogonalization is certainly one way of maintaining orthogonality, however, it may complicate the Lanczos recursion with regard to storage requirements and arithmetic operations. It requires that all of the previously-generated Lanczos vectors be available for the reorthogonalization of each Lanczos vector as it is generated, and therefore, the number of eigenvalues which can be computed is limited by the amount of secondary storage. On the Cray-2S/4-128 supercomputer, however, 128 megawords (1024 million bytes) of core memory may be sufficient for most Lanczos recursions requiring total reorthogonalization (for $m \times n$ matrices in which $mn \ll 10^7$). On the other hand, a Lanczos procedure with no reorthogonalization needs only the two most recently-generated Lanczos vectors at each stage, and hence has minimal computer storage requirements. One disadvantage of such a procedure is in the tracking ([11]) of spurious eigenvalues of B (singular values of A) associated with the loss of orthogonality in the Lanczos vectors, w_i.

We employ the most recent version of a single-vector Lanczos algorithm (30) equipped with a selective reorthogonalization strategy, LANSO (Version 1, April 1989), designed by Parlett and his colleagues at The University of California at Berkeley ([34], [43]). This particular method is primarily designed for the standard and generalized symmetric eigenvalue problem. We simply apply it to either $B = A^T A$ or the 2-cyclic matrix B defined in (5).

3.5. Block Lanczos Method. As subspace iteration is the block generalization of the classical power method for computing eigenpairs, we now consider a block analogue of the single vector Lanczos recursion given in (30). Exploiting the structure of the matrix B in (5), the following Lemma presents an alternative form for the Lanczos recursion (30).

LEMMA 3.4. *Let A be an $m \times n$ ($m \geq n$) matrix and B defined by (5). Apply the Lanczos recursion specified by (30) to B with a starting vector $\tilde{u} = (u, 0)^T$ such that $\|\tilde{u}\|_2 = 1$. Then the diagonal entries of the real symmetric tridiagonal Lanczos matrices generated are all identically zero, and the Lanczos recursion in (30) reduces to the following. Define $u_1 \equiv u$, $v_0 \equiv 0$, and $\beta_1 \equiv 0$. For $i = 1, 2, ..., k$ we then obtain the Lanczos recursions*

$$\begin{aligned} \beta_{2i} v_i &= A^T u_i - \beta_{2i-1} v_{i-1} \ , \\ \beta_{2i+1} u_{i+1} &= A v_i - \beta_{2i} u_i \ . \end{aligned} \tag{33}$$

The Lanczos recursion (33), however, can only compute the distinct singular values of an $m \times n$ matrix A and not their multiplicities. As shown in [11], these multiplicities can be recovered along with the suppression of zero eigenvalues of the matrix B caused by $m \neq n$.

Following the block Lanczos recursion for the sparse symmetric eigenvalue problem ([45], [22]), (33) may be represented in matrix form as

$$\begin{aligned} A^T \hat{U}_k &= \hat{V}_k J_k^T + Z_k \ , \\ A \hat{V}_k &= \hat{U}_k J_k + \tilde{Z}_k , \end{aligned} \tag{34}$$

where $\hat{U}_k = [u_1, \cdots, u_k]$, $\hat{V}_k = [v_1, \cdots, v_k]$, J_k is a $k \times k$ bidiagonal matrix with $J_k[j,j] = \beta_{2j}$ and $J_k[j,j+1] = \beta_{2j+1}$, and Z_k, $\tilde{Z}_k$ contain remainder terms. It is easy to show that the nonzero singular values of J_k are the same as the positive eigenvalues of

$$K_k \equiv \begin{pmatrix} O & J_k \\ J_k^T & O \end{pmatrix} . \tag{35}$$

For the block analogue of (34), we make the simple substitutions

$$u_i \leftrightarrow U_i \ , \ v_i \leftrightarrow V_i \ ,$$

where U_i is $m \times b$, V_i is $n \times b$, and b is the current block size. The matrix J_k is now a block upper bidiagonal matrix of order bk

$$J_k \equiv \begin{pmatrix} S_1 & R_1^T & & & \\ & S_2 & R_2^T & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & & \cdot & R_{k-1}^T \\ & & & & & S_k \end{pmatrix} , \tag{36}$$

where the S_i's and R_i's are $b \times b$ upper-triangular matrices. If U_i's and V_i's form mutually orthogonal sets of bk vectors so that $\hat{U}_k$ and $\hat{V}_k$ are orthonormal matrices, then the singular values of the matrix J_k will be identical to those of the original $m \times n$ matrix A. In essence, the 2-cyclic matrix K_k in (35) is the projection of the vectors defined by

$$\begin{pmatrix} \hat{U}_k & O \\ O & \hat{V}_k \end{pmatrix}$$

onto the Krylov subspace generated by the 2-cyclic matrix B in (5). Given the upper block bidiagonal matrix J_k, we approximate the singular triplets of A by first computing the singular triplets of J_k. To determine the left and right singular vectors of A from those of J_k, we must retain the Lanczos vectors in $\hat{U}_k$ and $\hat{V}_k$. Specifically, if $\{\sigma_i^{(k)}, y_i^{(k)}, z_i^{(k)}\}$ is the i-th singular triplet of J_k, then the approximation to the i-th singular triplet of A is given by $\{\sigma_i^{(k)}, \hat{U}_k y_i^{(k)}, \hat{V}_k z_i^{(k)}\}$, where $\hat{U}_k y_i^{(k)}$, $\hat{V}_k z_i^{(k)}$ are the left and right approximate singular vectors, respectively. The computation of singular triplets for J_k requires two phases. The first phase reduces J_k to bidiagonal form, say B_k, where

$$B_k \equiv \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & & \cdot & \beta_{bk-1} \\ & & & & & \alpha_{bk} \end{pmatrix} , \tag{37}$$

via a finite sequence of orthogonal transformations (thus preserving the singular values of J_k). The second phase reduces B_k to diagonal form by a modified QR algorithm. This diagonalization procedure is discussed in detail in [21]. The resulting diagonalized B_k will yield the approximate singular values of A and the corresponding left and right singular vectors are determined by products of all the left and right transformations (respectively) used in both phases of the SVD of J_k.

There are a few options for the reduction of J_k to the bidiagonal matrix, B_k. Golub, Luk, and Overton in [20] advocated the use of either band Householder or band Givens methods which in effect *chase off* (or zero) elements on the diagonals above the first super-diagonal of J_k. The method of choice in this case proved to be a band Givens algorithm using fast Givens rotations (see [23]) which required approximately $8(bk)^3/3$ multiplications to determine the bidiagonal matrix, B_k, from J_k with accumulation of orthogonal transformations. We note that the algorithm for diagonalizing B_k typically requires in excess of $8(bk)^3$ multiplications (with standard Givens rotations) and thus may dominate any savings in the reduction to bidiagonal form. In either reduction (bi-diagonalization or diagonalization), the computations are primarily sequential and offer limited data locality or parallelism for possible exploitation on a multiprocessor architecture such as the Cray-2S/4-128 or Alliant FX/80. For this reason, we adopt the single vector Lanczos bi-diagonalization recursion by (33) and (34) as our strategy for reducing the upper block bidiagonal matrix J_k to bidiagonal form (B_k), i.e.,

$$\begin{aligned} J_k^T \hat{Q} &= \hat{P} B_k^T \ , \\ J_k \hat{P} &= \hat{Q} B_k \ , \end{aligned} \tag{38}$$

or

$$\begin{aligned} J_k p_j &= \alpha_j q_j + \beta_{j-1} q_{j-1} \ , \\ J_k^T q_j &= \alpha_j p_j + \beta_j p_{j+1} \ , \end{aligned} \tag{39}$$

where $\hat{P} \equiv \{p_1, p_2, \ldots, p_{bk}\}$ and $\hat{Q} \equiv \{q_1, q_2, \ldots, q_{bk}\}$ are orthonormal matrices or order $bk \times bk$ and the α_i's and β_i's are defined by (37). The recursions in (39) require band matrix-vector multiplications which can be easily exploited by optimized level-2 BLAS routines ([14]) now resident in optimized mathematical libraries on most high-peformance computers. For orthogonalization of the outermost Lanczos vectors, $\{U_i\}$ and $\{V_i\}$, as well as the innermost Lanczos vectors, $\{p_i\}$ and $\{q_i\}$, we have chosen to apply a complete or total reorthogonalization ([23]) strategy to insure robustness in our triplet approximations for the matrix A. We note that while the Golub-Kahan-Reinsch method ([19], [21]) and the block-analogue of the Lanczos recursion (with or without reorthogonalization) in (33) by Golub, Luk, and Overton ([20]) compute the singular values of J_k, Cullum and Willoughby ([11]) must compute the eigenvalues of the order $2k$ Lanczos matrix (which by permutation is equivalent to K_k) to *weed out* spurious eigenvalues due to the lack of reorthogonalization in their particular Lanczos procedure. The Cullum-Willoughby algorithm, however, does not require the memory that re-orthogonalization schemes do, and hence may be more efficient on small-memory machines or those for which memory references are expensive compared to arithmetic operations. This hybrid Lanczos approach which incorporates inner iterations of single-vector Lanczos bidiagonalization within the outer iterations of a block Lanczos SVD recursion is outlined in Tables 9 and 10.

As an alternative to the outer recursion in Table 9, which is derived from the equivalent eigenvalue in the 2-cyclic matrix B, Table 11 depicts the simplified outer block Lanczos recursion for approximating the eigensystem of A^TA. Combining the equations in (34), we obtain

$$A^TA\hat{V}_k = \hat{V}_k H_k \ ,$$

where $H_k = J_k^T J_k$ is the $k \times k$ symmetric block tridiagonal matrix

$$H_k \equiv \begin{pmatrix} S_1 & R_1^T & & & & \\ R_1 & S_2 & R_2^T & & & \\ & R_2 & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & R_{k-1}^T \\ & & & & R_{k-1} & S_k \end{pmatrix} , \tag{40}$$

having block size b. We then apply the block Lanczos recursion ([23]) in Table 11 for computing the eigenpairs of the $n \times n$ symmetric positive definite matrix A^TA. The tridiagonalization of H_k via an inner Lanczos recursion follows from simple modifications to Table 10. Analogous to the diagonalization of B_k in (36), the computation of eigenpairs of the resulting tridiagonal matrix in this case can be performed via a Jacobi or QR-based symmetric eigensolver.

The conservation of computer memory for our iterative block Lanczos iterative SVD method is insured by enforcing an upper bound, c, for the order (bk) of any J_k constructed (see Table 9). This technique was suggested by Golub, Luk, and Overton in [20]. Given a block size b (usually $b \leq p$, where p is the desired number of triplets), the number of diagonal blocks, d, for J_k, is defined as $\lfloor c/b \rfloor$, where $\lfloor \cdot \rfloor$ denotes truncation of the mantissa. If $d < 2$, one may reset $b = c/2$ and then redefine $d = \lfloor c/b \rfloor$ so that J_k maintains the block upper bidiagonal form in (36).

As mentioned above, we may compute the SVD of the bidiagonal matrix B_k by a modified QR algorithm. Using (38) we may write

$$B_k = \bar{Q}\Sigma\bar{P}^T \ ,$$

so that

$$J_k = \hat{Q}\bar{Q}\Sigma\bar{P}^T\hat{P}^T \ ,$$

where $\Sigma = \text{diag}\ \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, and σ_i is an approximation to an exact singular value of the original $m \times n$ sparse matrix A. Hence, via (34) approximations to the i-th left and right singular vectors

(1)	[Formation of J_k]
	Choose V_1 ($n \times b$ and orthonormal) and $c = max\ \{bk\}$.
	Compute $W_1 = AV_1$. ($P_0 = U_0 = 0$ initially)
	Orthogonalize W_1 against U_0 (i.e., $W_1 = (I - U_0 U_0^T)W_1$).
	For $i = 2, 3, \ldots, k$ do: ($k = \lfloor c/b \rfloor$)
(1a)	Compute $Y_i = A^T U_{i-1} - V_{i-1} S_{i-1}$,
(1b)	Orthogonalize Y_i against $\{V_l\}_{l=0}^{i-1}$,
(1c)	Factor $Y_i = V_i R_{i-1}$,
(1d)	Compute $W_i = AV_i - U_{i-1} R_{i-1}^T$,
(1e)	Orthogonalize W_i against $\{U_l\}_{l=0}^{i-1}$,
(1f)	Factor $W_i = U_i S_i$.

TABLE 9

Hybrid Lanczos Outer Iteration.

corresponding to σ_i are given by

$$\begin{aligned} \bar{u}_i &= \hat{U}_k \hat{Q} \bar{q}_i \ , \\ \bar{v}_i &= \hat{V}_k \hat{P} \bar{p}_i \ , \end{aligned} \tag{41}$$

where $\bar{p}_i$, $\bar{q}_i$ are the i-th columns of $\bar{P}$, $\bar{Q}$, respectively. Suppose that before restarting the outer iteration in Table 9 we have determined that p_0 singular triplets are acceptable to a user-supplied tolerance for the residual error defined in (6). Then, we update the values of the block size (b), the maximum allowable order for J_k (c), the number of diagonal blocks for J_k (d), and the number of triplets yet to be found (p) as follows:

$$\begin{aligned} b_{new} &= b_{old} - p_0 \ , \text{ if } b \geq p_{old}, \\ &= min\ \{b_{old}\ ,\ p_{old} - p_0\} \ \text{ otherwise}, \\ c_{new} &= c_{old} - p_0 \ , \\ p_{new} &= p_{old} - p_0 \ , \\ d_{new} &= \lfloor c_{new}/b_{new} \rfloor \ . \end{aligned} \tag{42}$$

All converged left and right singular vector approximations are respectively stored in matrices U_0 and

(2)	[Bidiagonalization of J_k, Formation of B_k]
	Choose p_1 ($\|p_1\|_2 = 1$),
	Compute $t_1 = J_k p_1$, $\alpha_1 = \|t_1\|_2$,
	For $i = 1, 2, \ldots, \tilde{n}$ do: ($\tilde{n} = b \times k$)
	(while $\alpha_j \neq 0$) do:
	$q_j = t_j / \alpha_j$,
(2a)	Compute $z_j = J_k^T q_j - \alpha_j p_j$,
(2b)	Orthogonalize z_j against $\{p_l\}_{l=1}^{j}$,
	$\beta_j = \|z_j\|_2$,
	(while $\beta_j \neq 0$) do:
	$p_{j+1} = z_j / \beta_j$,
(2c)	Compute $t_{j+1} = J_k p_{j+1} - \beta_j q_j$,
(2d)	Orthogonalize t_{j+1} against $\{q_l\}_{l=1}^{j}$,
	$\alpha_{j+1} = \|t_{j+1}\|_2$.

TABLE 10

Hybrid Lanczos Inner Iteration.

V_0 so that

$$
\begin{aligned}
U_0 &\equiv (U_0 | \bar{u}_1, \bar{u}_2, \ldots, \bar{u}_{p_0}) \ , \\
V_0 &\equiv (V_0 | \bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{p_0}) \ ,
\end{aligned}
$$

where $U_0 = V_0 = 0$ initially (prior to any restart). For restarting the block Lanczos outer iteration in Table 9, we simply redefine V_1 to be the unconverged right singular vector approximations from the previous iteration, i.e.,

$$
V_1 = (\bar{v}_{i_1}, \bar{v}_{i_2}, \ldots, \bar{v}_{i_{bnew}}) \ ,
$$

where $i_j \in \{1, 2, \ldots, b_{old}\}$ is an index of any uncoverged triplet. To estimate the accuracy of our approximate singular triplets in say iteration l of our hybrid Lanczos method, we may conveniently

(1)	[Formation of symmetric block tridiagonal matrix H_k]
	Choose V_1 ($n \times b$ and orthonormal) and $c = max\ \{bk\}$.
	Compute $S_1 = V_1^T A^T A V_1$. ($V_0, R_0^T = 0$ initially)
	For $i = 2, 3 \ldots, k$ do: ($k = \lfloor c/b \rfloor$)
(1a)	Compute $Y_{i-1} = A^T A V_{i-1} - V_{i-1} S_{i-1} - V_{i-1} R_{i-2}^T$,
(1b)	Orthogonalize Y_{i-1} against $\{V_l\}_{l=0}^{i-1}$,
(1c)	Factor $Y_{i-1} = V_i R_{i-1}$,
(1d)	Compute $S_i = V_i^T A^T A V_i$.

TABLE 11

Hybrid Lanczos Outer Iteration for the Equivalent Symmetric Eigensystem of $A^T A$.

estimate the residual for some σ_k (see 6) by $\|y_k\|_2$ of Step (1a) in Table 9 for iteration $l+1$, where y_k is the k-th column of the $n \times b$ matrix Y_i. Hence, at the start of iteration $l+1$ we can determine the accuracy of our approximations from iteration l.

As with the previous iterative SVD methods, we access the sparse matrices A and A^T for this hybrid Lanczos method only through sparse matrix-vector multiplications. Some efficiency, however, is gained in the outer (block) Lanczos iterations by the multiplication of b vectors (Steps (1a), (1b) in Table 9) rather than by a single vector. These dense vectors may be stored in a fast local memory (cache) of a hierarchical memory-based archtitecture (Alliant FX/80, Cray-2S) and thus yield more effective data reuse. The total reorthogonalization strategy and deflation of converged singular vector approximations is accomplished in Steps (1b), (1e) in Table 9 and Steps (2b), (2d) in Table 10. A stable variant of Gram-Schmidt orthogonalization ([37]), which requires efficient dense matrix-vector muliplication (level-2 BLAS) routines ([14]), is used to produce the orthogonal projections of Y_i (i.e., R_{i-1}) and W_i (i.e., S_i) onto $\tilde{V}^\perp$ and $\tilde{U}^\perp$, respectively, where

$$\tilde{V} = (V_0, V_1, \ldots, V_{i-1}) \text{ and } \tilde{U} = (U_0, U_1, \ldots, U_{i-1}) \ .$$

The convergence of the block Lanczos recursion (34) in the approximation of the b-largest singular values of the matrix A is analyzed in [4]. Although the bound in [4] is somewhat tighter than that which was considered by Underwood in [45] for the symmetric eigenvalue problem, both results clearly indicate the desire for b (block size) to be chosen so that $\sigma_i - \sigma_{i+b}$ is as large as possible. Given this criterion, Underwood alluded to a block size control strategy in which the block size, b, is chosen to be at least as large as the number of eigenvalues (singular values in our case) in a given cluster. Obviously, the success of such a heuristic strategy for controlling the block size (irrespective of deflation) is spectrum-dependent and may not guarantee optimal convergence rates for a given matrix A. Nevertheless, conducting several experiments with various initial block sizes, say b_0, and detecting the most optimal deflation rate at a nominal number of sparse matrix-vector multiplications (with A and A^T) can be very expensive in computer access time and is not generally recommended when the SVD of several large sparse matrices A is required.

One plausible block size readjustment strategy that can be incorporated into our hybrid block Lanczos procedure (Tables 9 and 10) is to obtain spectral information for each projection matrix J_k. That is, identify clusters of approximate singular values of J_k or eigenvalues of $J_k^T J_k$ with an intent to increase the updated block size (b_{new}) so as to encompass the approximated cluster(s). To estimate the eigenvalue spectrum of $J_k^T J_k$ we can employ Gershgorin's theorem (see [23], p.341) after the formation of each upper block bidiagonal matrix, J_k. Note that although $H_k = J_k^T J_k$ is a $bk \times bk$ symmetric block tridiagonal matrix having block size b, we only need to store the bk inner products

$$(J_k^i)^T J_k^l \ , \quad i, l = 1, 2, \ldots, bk,$$

where J_k^i denotes the i-th column of J_k, and thereby avoid explicit formation of H_k. The success of one candidate block size readjustment strategy which relies upon the approximate spectrum of $J_k^T J_k$ is discussed in [4].

4. Comparison Of Methods. In this section, we compare the complexity, parallelism, memory requirements, and parameter selection of the four methods presented in the previous section. Although not discussed here, a comparison of the convergence properties of the four methods can be found in [4]. We will use the following naming conventions for our four sparse SVD methods throughout the remainder of the paper.

Method	Name
Single-Vector Lanczos SVD	LASVD
Block Lanczos SVD	BLSVD
Subspace Iteration SVD	SISVD
Trace Minimization SVD	TRSVD

4.1. Complexity. Perhaps the best way to assess the various computational tasks (or bottlenecks) associated with the four SVD methods presented in the previous section is to decompose (profile) them into their most time-consuming sub-algorithms as observed from actual runs on both a supercomputer, Cray-2S/4-128[5] and mini-supercomputer, Alliant FX/80[6]. The Cray-2S/4-128 supercomputer used in the experiments presented in Section 4 and 5 has 4 CPU's, where each CPU has a peak theoretical computational rate of 488 megaflops (millions of floating-point operations per second), and 128 megawords (64-bit words) of core memory. This particular Cray-2S uses the UNICOS 5.1 operating system, and supports *autotasking* (see [8]). Using autotasking, we allow the Cray-2S/4-128 Fortran pre-processor to generate multiple-CPU programs for each of our four SVD methods.

In contrast, the Alliant FX/80 used in our experiments is a multi-vector processor computer system having 8 vector processors (computational elements or CE's), where each processor has a peak computational rate in 64-bit arithmetic of 23.5 megaflops, and up to 10 megawords of shared physical memory. A modified version of the Concentrix 3.0 operating system is used on this particular Alliant FX/80.

In Table 12, we illustrate the dominant sub-algorithms or tasks associated with each of four sparse SVD methods when we determine the 100-largest singular triplets of the medical abstract database matrix, **MED**, from the Bellcore collection in Table 1 on the Cray-2S/4-128. To obtain these profiles, we invoke the *flowtrace* compiler option (see [9]) on only 1 CPU (profiling is not currently available for multiple-CPU programs). We terminate each method when residuals (6) less than or equal to 10^{-3} are achieved (no loss of accuracy associated with the use of the A^TA operator observed in this case).

To compute the 100-largest singular triplets (singular values given by Figure 4) of **MED**, we compute eigensystems of both B (defined in (5)) and A^TA for the Lanczos-based methods, LASVD and BLSVD. For SISVD and TRSVD, we determine eigensystems of A^TA and the shifted cyclic matrix, $\tilde{B}$, from (9). Table 12 illustrates the typical dominance (in percentage of total CPU time[7]) of sparse matrix-vector multiplications (SPMXV) in each of the four iterative methods under consideration. The percentage of CPU time required for these multiplications ranges from 42% for BLSVD with the cyclic operator, B, to 88% for either SISVD with the A^TA operator or TRSVD with the shifted cyclic operator, $\tilde{B}$. We note that the SPMXV includes the number of multiplications by both A and A^T. The cost of these multiplications (10) certainly depends upon the sparsity of A, i.e., μ_r and μ_c from (10), and the storage scheme used for the nonzero elements of matrix A. In our experiments, we employ the column-wise storage format associated with the Harwell/Boeing Sparse Matrix Collection ([16]). A detailed discussion of efficient kernels for sparse matrix-vector multiplication on Cray and Alliant computer systems can be found in [39].

Table 12 reveals that the selective re-orthogonalization (ORTHG) strategy has minimal impact

[5] Cray-2S/4-128 supported by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign.

[6] Alliant FX/80 located at the Center for Supercomputing Research and Development (CSRD) at the University of Illinois at Urbana-Champaign.

[7] We report only those sub-algorithms or kernels requiring more than 1% of the total CPU time.

(2-4% of the total CPU time) on the execution of LASVD on 1 CPU of the Cray-2S/4-128. It is also interesting to note that the computation of eigenvalues and accumulation of orthogonal transformations (for recovering eigenvectors) of the symmetric tridiagonal matrix T_j in (31) as determined by the implicit QL method (see [30]) and implemented by an optimized version of the EISPACK routine, IMTQL2, requires approximately 12% of the total CPU time when the eigensystem of the order 1033 matrix A^TA is determined but as much as 34% of the total CPU time if we are computing the eigensystem of the order 6864 matrix B. Since the singular values of A, σ, occur as $\pm$ eigenvalue pairs of B, we must obtain for $p = 200$ eigenpairs (largest) of B with LASVD and hence require, for this particular matrix A, as many as 590 steps (maximum dimension of Krylov subspace) before we have acceptable residuals for the 100-largest singular triplets of A. In contrast, only 292 Lanczos steps are required for the eigensystem of A^TA and thus the discrepancy in the impact of IMTQL2.

The difference in the profiles of BLSVD in Table 12 can be attributed to the simpler outer block Lanczos recursion for the eigensystem of A^TA (see Table 11). The reduction in the amount of CPU time spent in total re-orthogonalization (ORTHG) by an optimized modified Gram-Schmidt procedure ([37]) from 43% to 12% for the recursion on A^TA follows from iterating with only one set of Lanczos vectors V_l in Table 11.

The 1-CPU Cray-2S/4-128 profiles of SISVD and TRSVD depicted in Table 12 are the most similar among the four methods. The usage of polynomial acceleration in both methods and the use of the conjugate gradient (CG) method by TRSVD for solving the reduced system of equations in (21) will insure the dominance of sparse matrix-vector multiplications (SPMXV) over other kernels such as orthogonalization (ORTHG) by a modified Gram-Schmidt procedure. For these profiles of SISVD and TRSVD on the Cray-2S/4-128, subspaces of order $s = 120$ were used to approximate the 100-largest singular triplets of the `MED` matrix. Eigensystems of both A^TA and the shifted cyclic matrix $\tilde{B}$ were successfully used to obtain the desired residuals (6) in both methods. Whereas the context switch parameter, $\eta = 10^{-1}$, for TRSVD limits the extent of polynomial acceleration (maximum polynomial degree of 2 for this test case), Chebyshev polynomials of degrees up to 10 and 28, are required by SISVD for eigensystems of A^TA and $\tilde{B}$, respectively. If one chooses to employ SISVD with the unshifted 2-cyclic operator, B, in (5), then as with LASVD the subspace necessary to approximate the 100-largest singular triplets of `MED` should contain at least 200 (6864×1) vectors.

Table 13 indicates the profiles of our four methods on 8 processors of the Alliant FX/80. Again, we seek the 100-largest singular triplets of the 5831×1033 `MED` matrix from Table 1 with residuals (6) less than or equal to 10^{-3}. For each of the methods, we compute singular triplets via eigenpairs of A^TA only. Memory limitations on the Alliant FX/80 prevented us from computing eigensystems of the shifted and unshifted 2-cyclic matrices, $\tilde{B}$ and B, associated with the larger matrices from Tables 1 and 2. Comparisons in memory requirements for each of the methods is discussed in the next section.

For LASVD on the Alliant FX/80, the percentage of CPU time spent for sparse matrix-vector

Algorithm	Percentage of Total CPU Time							
	LASVD		BLSVD		SISVD		TRSVD	
	B	A^TA	B	A^TA	$\tilde{B}$	A^TA	$\tilde{B}$	A^TA
SPMXV	54	72	42	77	86	88	88	85
ORTHG	4	2	43	12	11	7	1	2
(IM)TQL2	34	12	–	–	–	–	–	–
QR	–	–	5	–	–	–	–	–
CG (BLAS1)	–	–	–	–	–	–	4	3

TABLE 12

Profile of the four methods for computing the 100*-largest singular triplets of the* 5831×1033 MED *matrix in Table 1 on the Cray-2S/4-128. Eigensystems of the original or modified* ($\tilde{B}$) 2*-cyclic matrix* B*, and the matrix* A^TA *are approximated by each method.*

multiplications (SPMXV) is a fraction of that consumed by LASVD on the Cray-2S/4-128. However, the amount of time spent in level-1 BLAS kernels (DCOPY, DAXPY, DDOT) totals to just under 40% of the CPU time. On the Cray-2S/4-128, the level-1 BLAS operations (used within the conjugate gradient (CG) algorithm) consumed no more than 5% of the total CPU time. Such vector operations (see [13]) must be synchronized across the 8 processors of the Alliant FX/80, and large numbers of memory fetches for vectors of length 1033 easily stress the memory hierarchy of the Alliant FX/80 which supports a 4-way interleaved 128 kilobyte cache between main memory and the 8 processors (see [1]). The assembly language level-1 BLAS kernels used by LASVD on the Alliant FX/80 were supplied by the Alliant FX/Series Scientific Library ([2]). On the Cray-2S/4-128, optimized Cray Assembly Language (CAL) implementations of the level-1 BLAS kernels (see [10]) required less than 1% of the total CPU time for each method, and hence do not appear in the profiles presented in Table 12.

Table 13 also indicates that a significant proportion of time (24% of total CPU time) is spent in the level-2 (matrix-vector) and level-3 (matrix-matrix) BLAS kernels. The outer block Lanczos recursion for A^TA (see Table 11), as with the outer recursion in Table 9, primarily consists of these higher-level BLAS kernels (also supplied by the Alliant FX/Series Scientific Library) which are designed for execution on all 8 processors of the Alliant FX/80. The modified Gram-Schmidt procedure we employ for re-orthogonalization is also driven by the higher-level BLAS kernels. Although it did not appear in the profile of BLSVD on the Cray-2S/4-128, multiplication by the Krylov projection matrix, J_k, which is a symmetric block tridiagonal matrix (see Section 3.5) for the recursion in A^TA, requires just under 10% of the total CPU time on the Alliant FX/80. The computation of the eigenpairs of the resulting symmetric tridiagonal matrix (B_k in inner Lanczos iteration from Table 10) by the EISPACK routine, TQL2, is not as demanding in BLSVD (8%) as opposed to LASVD (14%) since we incorporate a bound on the dimension of the Krylov subspace generated by each outer iteration in our hybrid block Lanczos strategy. In this particular experiment, we selected an initial block size of $b = 30$ and maintained a maximum dimension of $c = 150$ for the Krylov subspace in each outer iteration. As in Table 12, we again note the similarity in profiles of SISVD and TRSVD on the Alliant FX/80 in Table 13. Both methods are clearly dominated by sparse matrix-vector multiplications (SPMXV).

Another important algorithmic profile we can use to discern differences among the four methods is the effective use of multiple processors during execution time. In the next section, we assess the degree of parallelism exhibited by each of the four sparse iterative methods on 8 processors of the Alliant FX/80.

	Percentage of Total CPU Time			
Algorithm	LASVD	BLSVD	SISVD	TRSVD
SPMXV	27	46	63	69
ORTHG	–	–	–	1
BLAS2(3)	–	24	8	14
DSBMV	–	9	–	–
(IM)TQL2	14	8	3	–
TRED2	–	–	3	–
DAXPY	17	–	14	–
DCOPY	20	–	–	–
DDOT	2	–	–	–
DNRM2	–	–	–	1

TABLE 13

Profile of the four methods for computing the 100*-largest singular triplets of the* 5831×1033 MED *matrix in Table 1 on the Alliant FX/80. Only eigensystems of* $A^T A$ *are approximated.*

4.2. Parallelism. In order to assess the degree of effective parallelism achieved by our four methods for the sparse SVD on the Alliant FX/80, we record the actual CPU time when *exactly* j processors are active, where $j = 1, 2, \ldots, 8$. Suppose that t_j is the total user CPU time (in seconds) for which j processors were active during our experiment. Then the average concurrency, μ_{co}, achieved for our sparse SVD method would be given by

$$\mu_{co} = (\sum_{j=1}^{8} j t_j)/t_{tot} \ ,$$

where t_{tot} is the total user CPU time spent. For $t_k = 0,\ k = 1, 2, \ldots, 7$, we would naturally have a completely parallel method and $\mu_{co} = 8$. This situation, however, is difficult to obtain since overhead in scheduling, start-up, and synchronization for even the most parallelized instruction will have some of the 8 processors inactive before, during, or after its execution. Given μ_{co}, we also define the concurrency efficiency, E_c, of a particular method by

$$E_c = \frac{\mu_{co}}{8} \; .$$

In Table 14, we indicate the *breakdown* in percentage of total (user) CPU time spent on exactly j processors by each of the four sparse SVD methods when the 100-largest triplets of the `MED` matrix in Table 1 are approximated via eigenpairs of $A^T A$. Essentially, this data reveals the effective *distribution* of parallelism (or utilization of multiple processors) among the four methods across the 8 processors of the Alliant FX/80. By inspection, we can see that TRSVD is by far the most parallel of the four methods since as much as 64% of the total CPU time is spent when all 8 processors are active. This is somewhat expected, since TRSVD ultimately determines each $h_j^{(k)}$ in (21) concurrently via independent conjugate gradient (CG)iterations. The second-most parallel method would be SISVD in that approximately 58% of the total CPU time is spent on 8 processors. BLSVD is a close third with 56%, and LASVD is the least parallel with only 37%. This ranking is not too surprising since the subspace methods, TRSVD and SISVD, have only a few dominant sub-algorithms (SPMXV and BLAS2[3] from Table 13) which can be easily parallelized on the Alliant FX/80. It is important to note, however, that although LASVD with the $A^T A$ operator may be the least parallel of the four methods in this case, it can still be the fastest method on the Alliant FX/80 (see [4]). Extensive comparisons of the CPU times required by each of the four methods in computing the 100-largest singular triplets of the matrices in Tables 1 and 2 are presented in Section 5.

The average concurrency, μ_{co}, and concurrency efficiency, E_c, for each method is listed in Table 15. BLSVD achieves a higher concurrency efficiency than LASVD given its greater dependence (see Table 13) on highly optimized BLAS2[3] kernels which are specifically-designed for the 8 processors of the Alliant FX/80. These concurrency profiles exemplify the parallelism (processor utilization) observed for our four methods when we approximate as many as 100 of the largest singular triplets of the sparse matrices in Tables 1 and 2 on the Alliant FX/80.

Finally, in Table 16 we report the overall speedups for our four methods along with individual speedups for the sub-algorithms in Table 13. Here, we define speedup as

$$t_1/t_8,$$

where t_1 and t_8 are the (user) CPU times consumed by a program (or code section) executing on 1 and 8 processors of the Alliant FX/80, respectively. The speedup for sparse-matrix multiplication (SPMXV), as would be somewhat expected, essentially determines the overall speedup for all four methods. The efficiency of the level-2 and level-3 BLAS modules is evidenced by the speedups ranging from 5.0 to 5.5 on BLSVD, SISVD, and TRSVD. The level-1 BLAS operations (e.g., DAXPY, DDOT, DNRM2) as well as the (block) modified Gram-Schmidt orthogonalization (ORTHG) also achieved significant

	Percentage of Total CPU Time (on Exactly j Processors)			
j	LASVD	BLSVD	SISVD	TRSVD
1	57.82	33.35	32.47	18.43
2	1.43	1.69	1.80	1.83
3	0.74	1.10	1.09	1.61
4	0.69	1.02	1.04	1.88
5	0.50	1.07	1.18	1.80
6	0.67	1.49	1.09	1.93
7	1.21	4.65	3.77	8.21
8	36.95	55.62	57.57	64.31

TABLE 14

Percentage of CPU time used by the four methods on 1 *through* 8 *processors of the Alliant FX/80 when computing the* 100*-largest singular triplets of the* 5831×1033 `MED` *matrix in Table 1.*

speedups. We note that the rank-ordering of average concurrency (μ_{co}) in Table 15 correlates well with the rank-ordering of speedups in Table 16.

Method	μ_{co}	E_c
TRSVD	6.3	78
SISVD	5.4	68
BLSVD	5.4	67
LASVD	3.7	47

TABLE 15

Average concurrency (μ_{co}) and efficiency (E_c) of sparse SVD methods on Alliant FX/80 when computing the 100*-largest triplets of the* 5831×1033 `MED` *matrix in Table 1 via the eigensystem of* $A^T A$.

4.3. Memory. Although the effective CPU time required by our methods for the spare SVD is certainly an essential metric in measuring their performance within applications such as information retrieval and seismic tomography, it is also important to assess the amount of computer memory that is required. We estimate the amount of memory in 64-bit words (8 bytes) which is required by the four methods when equivalent eigensystems of the unshifted/shifted 2-cyclic matrices, B and $\tilde{B}$, given by (5) and (9), respectively, and $A^T A$ are used to determine the p-largest singular triplets of an $m \times n$ sparse matrix A. The estimates are based on our implementations of each method on both the Cray-2S/4-128

Algorithm	LASVD	BLSVD	SISVD	TRSVD
SPMXV	3.0	3.0	3.1	3.6
ORTHG	—	—	—	4.8
BLAS2(3)	—	5.0	5.3	5.5
DSBMV	—	2.0	—	—
TQL2	—	2.8	3.5	—
IMTQL2	4.3	—	—	—
TRED2	3.3	—	—	—
DAXPY	5.0	—	4.4	—
DCOPY	3.6	—	—	—
DDOT	7.7	—	—	—
DNRM2	—	—	—	5.5
Overall Speedup	3.0	3.2	3.4	4.0

TABLE 16

Speedups of the four methods (and their sub-algorithms) for computing the 100*-largest singular triplets of the* 5831×1033 `MED` *matrix in Table 1 on the Alliant FX/80. Speedups indicated are the ratio of (user) CPU time on* 1 *processor to CPU time on* 8 *processors.*

(static memory allocation only) and the Alliant FX/80 (static and dynamic memory allocation). We exclude the memory needed to store the matrix A in the Harwell/Boeing sparse matrix format in these counts. Also, we exclude the memory (mp words) needed to compute the left singular vectors as in Lemma 3.2 from the eigensystem of A^TA. The parameters for BLSVD (see Table 9 in Section 3.5) include the initial block size, b, an upper bound on the dimension of the Krylov subspace, c. For LASVD, we also include a similar upper bound, q, for the order of the symmetric tridiagonal matrix T_j in (31). The parameter s for SISVD and TRSVD defines the number of vectors ($s \geq p$) for the initial subspace.

Table 17 lists the required megabytes ($8 \times 10^{-6} \times$ (number of words)) for the four sparse SVD methods when we require $p = 100$ of the largest singular triplets of the sparse matrices in Tables 1 and 2 (excluding `ADI`). The other parameters used for these memory counts are

$$b = 30, \ c = 150, \ s = 120, \ q = \min\{\tilde{n}, 600\} \ ,$$

where $\tilde{n} = n$ when approximating eigensystems of A^TA, and $\tilde{n} = m + n$ when approximating eigensystems of B or $\tilde{B}$. If we compute eigensystems of B or $\tilde{B}$, the Bellcore matrix `TECH` will pose the greatest memory constraint, while the Amoco matrix `AMOCO2` will require the largest amount of memory if we approximate the eigensystem of A^TA. In Table 17, we include the memory requirements for SISVD when the eigensystem of the indefinite 2-cyclic matrix B is determined. Since we must have $s \geq 2p$ for SISVD and LASVD when approximating eigenpairs of B (see Section 3.2), the entries in Table 17 reflect the selection of $s = 200$ for SISVD and $p = 200$ for LASVD for this case.

With regard to memory consumption, Table 17 reveals that SISVD is the most *conservative* of the four methods while LASVD can be considered the most *liberal*. Whereas BLSVD incorporates bounds (c) on the order of J_k in (36), LASVD must simply have an upper bound (q) for the dimension of the Krylov subspace that will be required to approximate all of the 100-largest singular triplets. Of course, the order of the equivalent eigensystem used ($m+n$ for B or $\tilde{B}$ and n for $A^T A$) is one possible choice for q but the associated memory requirement may prove to be too stringent for various computer systems (e.g., Alliant FX/80). The memory requirement of LASVD is nearly 5 times that of SISVD across most of the Bellcore and Amoco matrices. When the eigensystem of $A^T A$ is approximated, however, the factor is approximately 3. For most of the matrices in Tables 1 and 2 the savings in memory for all four methods ranges from a factor of 1.5 to 5 when we approximate eigenpairs of $A^T A$ rather than eigenpairs of the order $m + n$ matrices B and $\tilde{B}$. However, for suitably rectangular matrices ($m \gg n$) such as the 10337×425 Bellcore matrix `TIME`, the savings can be enormous. Specifically, LASVD and TRSVD each require a factor 22 fewer megabytes whereas the factors for SISVD and BLSVD are 17 and 19, respectively. Hence, we can have an important trade-off in memory versus accuracy when we choose to determine singular triplets of the matrix A via the eigensystem of $A^T A$. In approximating the p-largest triplets of A, the effect of scaling for the left singular vector in Lemma 3.2 on the residual in (6) will determine whether or not the less-stringent memory requirement associated with $A^T A$ is a mixed blessing.

4.4. Parameter Estimation. In the preceding sections we have revealed a few issues regarding the choices of parameters for the methods for sparse SVD under consideration. It is important to note the difficulties (if any) in selecting *good* parameters for these iterative methods. In the subspace methods, SISVD and TRSVD, we seek the smallest initial subspace dimension s, where $s \geq p$, that can approximate the desired p-largest singular triplets of A. For TRSVD we must also choose an appropriate context switch parameter, η, to switch from polynomial acceleration to shifting by Ritz values. We have been fairly successful with the choices $\eta = 10^0, 10^{-1}$, however, for the matrices in Tables 1 and 2.

As mentioned in Section 3.2, applying SISVD with the indefinite 2-cyclic operator, B, will force us to choose s to be at least twice the size ($2p$) of desired singular triplets. The same situation holds for LASVD when we approximate the eigensystem of B. The consequence of doubling p in terms of memory is discussed in Section 4.3.

As mentioned in the preceding section, BLSVD requires an initial block size b, where $b \leq p$, and the bound c on the Krylov subspace generated within the outer block Lanczos recursion given in Table 9. The choice of b can be difficult, and as mentioned in Section 3.5 we have made some gains

Matrix	Memory in Megabytes								
	LASVD		BLSVD		SISVD			TRSVD	
	B	A^TA	B	A^TA	B	$\tilde{B}$	A^TA	$\tilde{B}$	A^TA
AMOCO1	17	2	9	2	6	3	1	8	2
TIME	55	4	31	3	21	10	1	27	2
MED	56	10	32	6	22	11	2	28	4
CISI	58	12	35	8	22	11	3	29	6
CRAN	90	12	57	8	35	17	3	45	6
TECH	153	45	79	37	60	30	14	79	28
AMOCO2	190	60	109	49	75	37	19	98	37

TABLE 17

Memory (in megabytes) required by sparse SVD methods when computing the 100-largest triplets of matrices in Tables 1 and 2 via shifted or unshifted 2-cyclic eigensystems (B or $\tilde{B}$) and eigensystems of A^TA.

in removing this ambiguity by incorporating spectral estimation via Gershgorin disks (see [4]). This block size adjustment strategy, however, is not necessarily *robust* and may warrant further refinement. The choice of c, however, is perhaps one parameter whose value is more or less determined by the computer system rather than the programmer. As with the Krylov bound, q, for LASVD mentioned in Section 3.4, we can certainly choose c to be the order of the eigensystem used to approximate the desired singular triplets ($m+n$ for B or $\tilde{B}$, n for A^TA), but the available memory (in 64-bit words) is perhaps the best choice for c.

5. Performance Of Methods. In this section, we discuss the performance of the methods for the sparse SVD presented in Section 3 on both a supercomputer, Cray-2S/4-128, and a mini-supercomputer, Alliant FX/80. Model SVD problems using matrices from Tables 1 and 2 are presented, and a brief summary of our findings is given in Section 6. In order to clarify which eigensystems are used on the Alliant FX/80 and the Cray-2S/4-128 by all four methods, Table 18 lists the equivalent eigensystems considered for our model problems. For clarification purposes, SISVD and SISVD2 denote subspace iterati on using eigensystems of $\tilde{B}$ and B, respectively, where $\tilde{B}$ is given by (9) and B is the 2-cyclic matrix defined by (5). Due to memory limitations in working with B or $\tilde{B}$ (see Table 17), we only consider eigensystems of either A^TA or $\gamma^2 I_n - A^TA$ for each method on the Alliant FX/80.

5.1. Model Problems. In the following experiments, we determine the $p = 100$ largest singular triplets of the matrices (A) in Tables 1 and 2 (excluding ADI). We determine these triplets to residual norms (6) of order 10^{-3} or less via eigensystems of either A^TA or the 2-cyclic matrix B (see 5) and its

Largest Singular Triplets				
Machine	LASVD	BLSVD	SISVD	TRSVD
Cray-2S/4-128	B, A^TA	B, A^TA	B, $\tilde{B}$, A^TA	$\tilde{B}$, $\gamma^2 I_n - A^TA$
Alliant FX/80	A^TA	A^TA	A^TA	$\gamma^2 I_n - A^TA$

TABLE 18

Eigensystems computed by the four methods for the sparse SVD on the Alliant FX/80 and Cray-2S/4-128. B and $\tilde{B}$ are the appropriate unshifted and shifted cyclic matrices, respectively.

positive definite counterpart $\tilde{B}$ (see 9). For singular triplet approximations via eigenpairs of A^TA, the left singular vectors, u_i, were obtained according to Lemma 3.2. For reference purposes, we include plots of the desired singular values of the matrices in Figures 2 through 5.

As discussed in Section 4.3, with LASVD we must request $k = 2p = 200$ singular triplets of the $m \times n$ sparse matrix A if we are to determine the largest p singular triplets by considering the eigensystem of the matrix B. The same situation is certainly true for SISVD, although we can also compute eigensystems of the positive definite matrix $\tilde{B}$, where $\gamma = \|A\|_1$. As discussed in Section 3.3.1, TRSVD requires either eigensystems of $\tilde{B}$ or $\gamma^2 I_n - A^TA$ for this problem. The values of $\gamma = \|A\|_1$ along with the largest singular value (σ_{max}) in our suite of matrices is given in Table 19.

	ADI	AMOCO1	AMOCO2	CISI	CRAN	MED	TECH	TIME
$\gamma = \|A\|_1$	78	51	1401	230	5700	160	640	110
σ_{max}	20.7	23.8	553.9	33.8	1693.3	35.9	54.2	58.1

TABLE 19

Matrix 1-Norms for Bellcore and Amoco Matrices.

For our Lanczos-based methods, LASVD and BLSVD, we limit the dimension of Krylov subspaces by selecting

$$c = 150, \text{ and } q = \min\{n, 600\} ,$$

where c is the maximum order of the matrices J_k in (36) and H_k in (40) for BLSVD, and q is the bound on the number of Lanczos steps, j, in LASVD (or dimension of T_j in (31)). Our initial block size for BLSVD is $b = 30$ and the initial subspace dimension for SISVD and TRSVD is $s = 120$. For TRSVD, we choose $\eta = 10^{-1}$ as our residual reduction factor for the context switch from polynomial acceleration to shifting by Ritz values.

In presenting the performance of our candidate methods for this problem, we traverse the two rows of Table 18 as our guide for comparisons. We begin by comparing the two Lanczos-based methods,

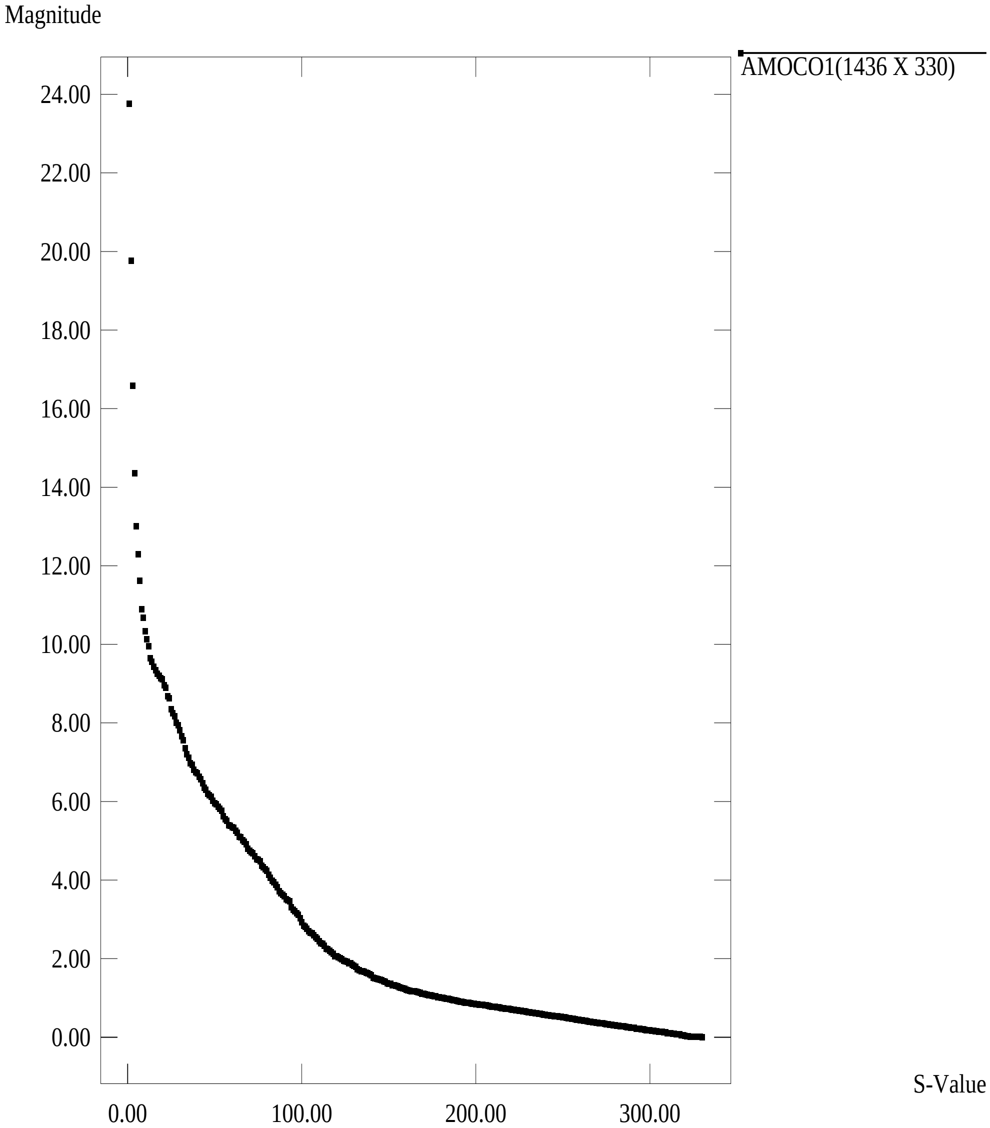


FIG. 2. *Full spectrum of the* 1436×330 `AMOCO1` *matrix in Table* 2.

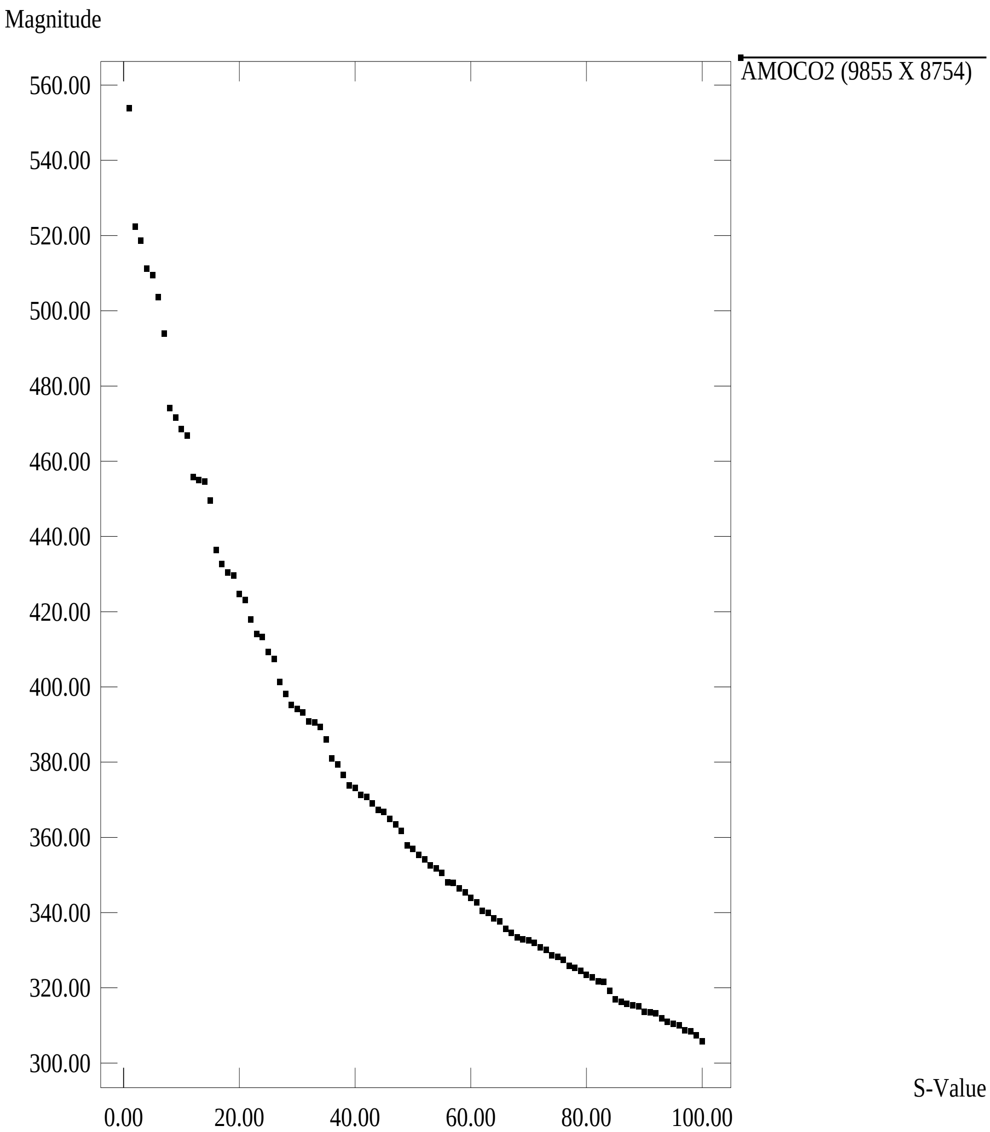


FIG. 3. *The 100-largest singular values of the* 9855×8754 `AMOCO2` *matrix in Table* 2.

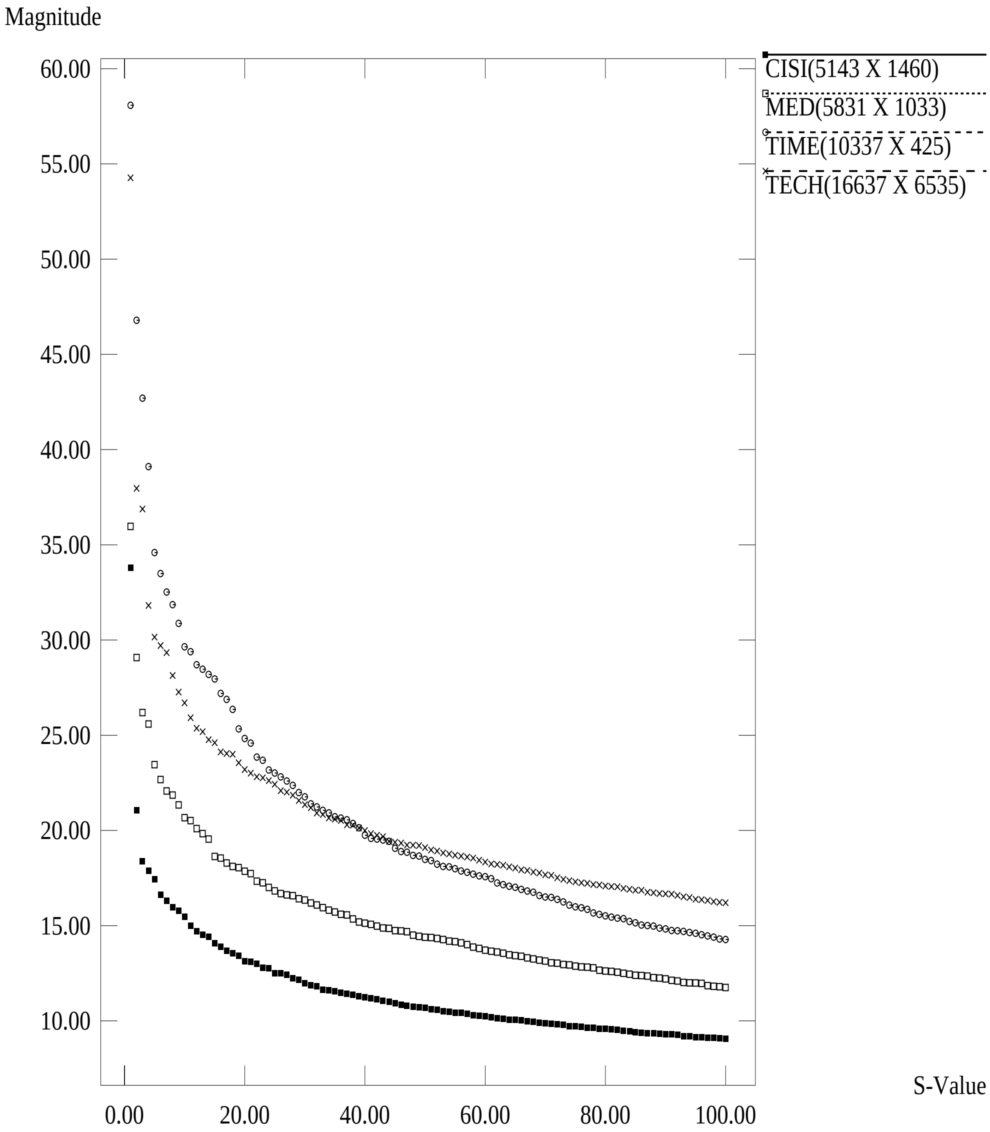


FIG. 4. *The 100-largest singular values of the matrices:* `CISI`, `MED`, `TECH`, *and* `TIME` *in Table* 1.

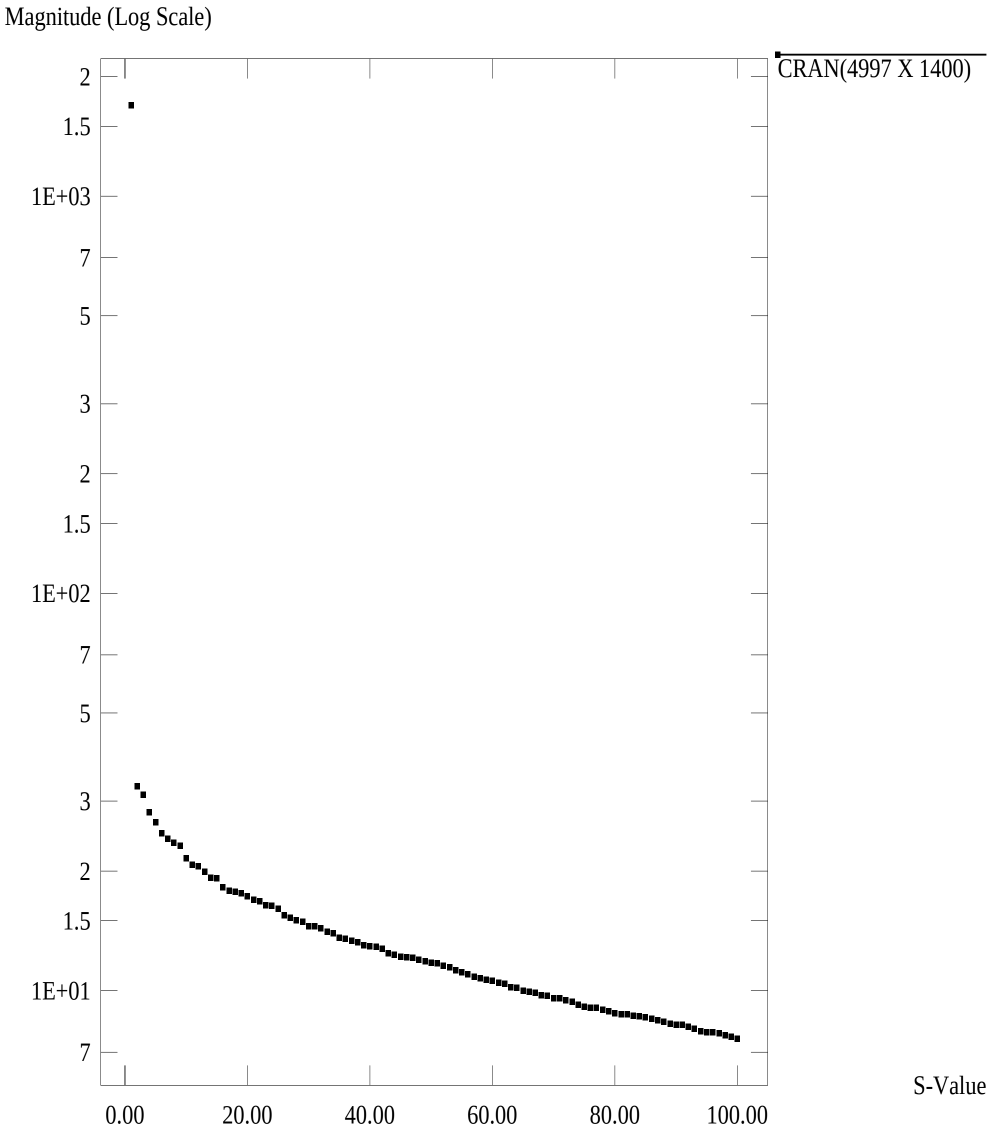


FIG. 5. *The* 100*-largest singular values of the* 4997×1400 *matrix* `CRAN` *in Table* 1.

LASVD and BLSVD, when eigensystems of the 2-cyclic matrix B are approximated. Table 20 illustrates the CPU time on the 4-CPU Cray-2/4-128. We observe that BLSVD can be between 1.2 to 1.8 times faster than LASVD for most problems (the exceptions being **AMOCO1** and **TECH** for which both times are comparable). Requiring $k = 2p = 200$ singular triplets to be approximated by LASVD practically *doubles* the number of Lanczos steps normally required to approximate $k = 100$ eigenvalues. This requirement, of course, is not necessary if eigensystems of $A^T A$ can be used to approximate these triplets. We especially note the superiority of LASVD over BLSVD when we approximate the 100-largest eigenpairs of $A^T A$ as opposed to those of B. Specfically, LASVD ranges between 2.6 to 6.7 times faster than BLSVD across our suite of matrices. Hence, when approximating eigensystems of B and $A^T A$ (see Table 20), we see that using LASVD to approximate the eigenpairs of $A^T A$ is the Lanczos scheme of choice for obtaining the largest singular triplets. To assess the specific gains in performance (time reduction) associated with the eigenvalue problem of order n, we list the speed improvements for LASVD and BLSVD (and the other candidate methods) when eigensystems of $A^T A$ are approximated in Table 21. The limited improvement for BLSVD, in this case, stems from the fact that although the less time is spent in re-orthogonalization (see Table 12) the number of outer iteration steps for the $A^T A$-based recursion (see Table 11) can be as much as 1.5 times greater than the number of outer iterations for the cyclic-based hybrid recursion (Tables 9 and 10). Hence, the deflation associated with *larger* gaps among the $p = 100$ eigenvalues of $A^T A$ is not quite as efficient. Also, the number of sparse matrix-vector multiplications per outer iteration step is 4 for the $A^T A$ implementation rather than 2 for the cyclic-based recursion.

Next, we compare the performance of the two versions of SISVD with that of TRSVD when eigensystems of B and $\tilde{B}$ are approximated. As mentioned above, SISVD2 denotes the computation of eigenpairs of B rather than those of $\tilde{B}$. The effective speed improvement of SISVD2 over SISVD ranges from 1.08 to 3.98. This, in effect, is due to the length of the Chebyshev intervals given in Table 4, which certainly determines the polynomial of minimial degree needed to accelerate convergence. We note that Chebyshev polynomials of degrees as high as 44 are required by SISVD, while TRSVD and SISVD2 require polynomials of maximum degree 2 and 22, respectively. As a result, TRSVD ranges between 2.2 to 4.96 times faster than SISVD2 for our test suite. Since, compiler-generated (*auto-tasking*) parallelism for TRSVD, yielded an average of 1.68 active CPU's in a *non-dedicated* environment, it is reasonable to expect even higher speed improvements for a dedicated machine. When approximating eigenpairs of $A^T A$ on the Cray-2S/4-128, we find SISVD to be a bit more competitive with TRSVD being approximately 1.5 times faster across the suite of matrices. As indicated in Table 21, SISVD is much more effective in approximating singular triplets of A via eigenpairs of $A^T A$. Excluding **AMOCO1**, SISVD only required Chebyshev polynomials of degree 10 for each matrix in our test suite (polynomial of degree 4 was sufficient for **AMOCO1**). TRSVD, on the other hand, required polynomials of degree 2 for each matrix.

With the exception of BLSVD, we note that the best performance on the Cray-2S/4-128 for our candidate methods is obtained when eigensystems of A^TA or $\gamma^2 I_n - A^TA$ are approximated. In Table 20, we compare the Cray-2S/4-128 CPU times of the best Lanczos method (LASVD) with that of the best subspace method (TRSVD) across the 7 matrices in our suite. Although more competitive than BLSVD, TRSVD cannot match the performance of LASVD. We note that LASVD ranges between 2.6 to 6.7 times faster than BLSVD, and 1.9 to 4.3 times faster than TRSVD.

SVD via eigensystems of B, $\tilde{B}$					
Matrix	LASVD	BLSVD	SISVD	SISVD2	TRSVD
AMOCO1	27	32	102	94	43
MED	139	103	1269	333	136
CISI	143	120	1276	515	187
TIME	147	127	1616	634	300
CRAN	167	117	1105	874	176
TECH	479	486	5598	1405	636
AMOCO2	654	360	4250	1160	508
SVD via eigensystems of A^TA					
Matrix	LASVD	BLSVD	SISVD		TRSVD
AMOCO1	10	26	23		19
MED	16	86	88		52
CISI	23	120	137		76
TIME	13	87	93		56
CRAN	22	117	120		77
TECH	89	490	605		292
AMOCO2	48	360	801		384

TABLE 20

Cray-2S/4-128 CPU times (in seconds) for determining the 100*-largest singular triplets of matrices in Tables 1 and 2.*

Having completed our comparisons for the first row of Table 18, we proceed down the second row and compare the performance of our methods on the Alliant FX/80 (matrices TECH and AMOCO2 omitted due to memory limitations). The A^TA implementation of LASVD is on average 2.7 times faster than BLSVD, and 2.4 times faster than TRSVD (via eigensystems of $\gamma^2 I_n - A^TA$) on the Alliant FX/80. The effective parallelization of TRSVD and SISVD (see Table 15) tends to produce comparable times for both methods (see Table 22) for the moderate-order matrices. The concurrency efficiency (see Table 15) is consistently high for TRSVD, and the parallel conjugate gradient (CG) iterations for solving (21)

Matrix	LASVD	BLSVD	SISVD	TRSVD
AMOCO1	2.7	1.2	4.4	2.3
AMOCO2	8.6	1.2	14.4	2.6
CISI	6.2	1.1	9.3	2.5
CRAN	11.3	1.5	17.3	5.4
MED	7.5	1.1	9.2	2.3
TECH	5.3	1.0	9.2	2.2
TIME	13.6	1.0	5.3	1.3

TABLE 21

Speed improvements for approximating the largest $p = 100$ singular triplets via eigensystems of $A^T A$ or $\gamma^2 I_n - A^T A$ as opposed to those of B or $\tilde{B}$ (cyclic matrices) on the Cray-2S/4-128.

can sustain an optimal performance rate (lower total execution time) with each processor handling one independent system.

SVD via eigensystems of $A^T A$				
Matrix	LASVD	BLSVD	SISVD	TRSVD
AMOCO1	110	66	49	63
MED	71	247	197	189
CISI	110	319	364	266
TIME	43	194	172	165
CRAN	95	250	315	277

TABLE 22

Alliant FX/80 CPU times (in seconds) for determining the 100-largest singular triplets of matrices in Tables 1 and 2.

Another interesting performance comparison that we can draw from Tables 20 and 22 is the speed improvement for each method (with $A^T A$ eigensystem operators) on the Cray-2S/4-128 relative to the Alliant FX/80. Across the test suite, we find that the average ratios, μ_t = (CPU time on the Alliant FX/80) / (CPU time on the Cray-2S/4-128) are:

Method	LASVD	BLSVD	SISVD	TRSVD
μ_t	4.2	2.5	2.3	3.4

Given, the fact that the Cray-2S/4-128 is (theoretically) capable of peak computational rate 10 times that of an Alliant FX/80, one can argue from a purely cost-performance perspective that our implementations on an Alliant FX/80 mini-supercomputer are quite competitive with those on the Cray-2S/4-128 supercomputer (assuming no memory limitations, of course).

6. Summary. Certainly one important by-product of this research effort is the portable Fortran software library of methods for the sparse SVD. The effective use of this library certainly depends on a good understanding of the merits of each algorithm composing the library as well as the applicability of the library itself. Essentially, we have constructed a *library* of sparse SVD algorithms which can be used for solving sparse linear least squares problems and for spectral estimation, in general. The performance of the methods comprising our library are further analyzed according to the criteria in Section 4 in [4]. Our efforts in solving relevant sparse SVD problems from information retrieval and seismic reflection tomography have revealed that LASVD (via A^TA eigensystems) is the fastest method for approximating several of the largest singular triplets on the Cray-2S/4-128 and Alliant FX/80 to low to moderate accuracies. The most conservative method with regard to memory requirements is SISVD, and TRSVD would be highly recommended for machines which can exploit a high degree of coarse-grain parallelism. For problems in which equivalent eigensystems of A^TA may not be suitable (ill-conditioned), BLSVD for cyclic-based eigensystems is recommended.

7. Acknowledgements. The author would like to thank Susan Dumais at Bell Communications Research (Bellcore) in Morristown, NJ, and John Scales at the Amoco Research Center in Tulsa, OK for their help in constructing the test suite of sparse matrices from information retrieval and seismic tomography applications. Also, the comments and suggestions supplied by the referees were most helpful.

REFERENCES

[1] FX/Series Architecture Manual. Alliant Computer Systems Corporation, Littleton, MA, September 1987.

[2] FX/Series Scientific Library. Alliant Computer Systems Corporation, Littleton, MA, September 1987.

[3] F. L. Bauer. Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischer Eigenwertprobleme. *ZAMP*, 8:214–235, 1957.

[4] M. W. Berry. Multiprocessor sparse SVD algorithms and applications. PhD thesis, The University of Illinois at Urbana-Champaign, Urbana, IL, October 1990.

[5] M. Berry and A. Sameh. Multiprocessor Jacobi schemes for dense symmetric eigenvalue and singular value decompositions. *Proceedings of ICPP 86*, St. Charles, IL, 433–440, 1986.

[6] M. Berry and A. Sameh. An overview of parallel algorithms for the singular value and dense symmetric eigenvalue problems. *Journal of Computational and Applied Mathematics*, 27:191–213, 1989.

[7] R. Bording, A. Gertsztenkorn, L. Lines, J. Scales, and S. Treitel. Applications of seismic traveltime tomography. *Geophys. J. R. astr. Soc.*, 90(2):285–304, 1987.

[8] Cray-2 Multitasking Programmers' Manual. Publication SN-2026C, Cray Research Inc., Mendota Heights, MN, March 1989.

[9] UNICOS Fortran Library Reference Manual. Publication SR-2079 5.0, Cray Research Inc., Mendota Heights, MN, February 1989.

[10] UNICOS Math and Scientific Library Reference Manual. Publication SR-2081 5.0, Cray Research Inc., Mendota Heights, MN, March 1989.

[11] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithm for Large Symmetric Eigenvalue Computations.* Volume 1 Theory, Birkhäuser, Boston, 1985.

[12] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[13] J. Dongarra, J. Bunch, C. Moler, and G. W. Stewart. *LINPACK Users' Guide.* SIAM, Philadelphia, 1979.

[14] J. Dongarra, J. Du Croz, S. Hammarling, and R. Hanson. An extended set of Fortran basic linear algebra subprograms. *ACM Trans. on Math. Software*, 14(1):1–17, 1988.

[15] J. Dongarra and D. Sorensen. A fast algorithm for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8(2):s139–s154, 1987.

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

[17] S. T. Dumais, G. W. Furnas, and T. K. Landauer. Using latent semantic analysis to improve access to textual information. In *Proceedings of Computer Human Interaction '88*, 1988.

[18] K. Gallivan, W. Jalby, and U. Meier. The use of BLAS3 in linear algebra on a parallel processor with a hierarchical memory. *SIAM J. Sci. Stat. Comput.*, 18(6):1079–1084, 1987.

[19] G. Golub and W. Kahan. Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal.*, 2(3):205–224, 1965.

[20] G. H. Golub, F. T. Luk, and M. L. Overton. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software*, 7(2):149–169, 1981.

[21] G. Golub and C. Reinsch. Singular value decomposition and least squares solutions. In *Handbook for Automatic Computation II, Linear Algebra*, Springer-Verlag, New York, 1971.

[22] G. H. Golub, and R. R. Underwood. The block Lanczos method for computing eigenvalues. In *Mathematical Software III*, Academic Press, New York, 361–377, 1977.

[23] G. Golub and C. Van Loan. *Matrix Computations*, Second Edition, Johns Hopkins, Baltimore, 1989.

[24] J. Grcar. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1981.

[25] M. R. Hestenes. Inversion of matrices by biorthogonalization and related results. *J. Soc. Indust. Appl. Math.*, 6:51–90, 1958.

[26] S. Kaniel. Estimates for some computational techniques in linear algebra. *Math. Comp.*, 20:369–378, 1966.

[27] C. Lanczos. *Linear Differential Operators.* Van Nostrand, New York, 1961.

[28] D. G. Luenberger. *Introduction to Linear and Nonliner Programming.* Addison-Wesley, Reading, MA, 1973.

[29] S. Lo, B. Phillipe, and A. Sameh. A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8(2):s155–s165, 1987.

[30] R. S. Martin and J. H. Wilkinson. The implicit QL Algorithm. *Num. Math.*, 12:377–383, 1968.

[31] L. Mirsky. Symmetric gage functions and unitarilly invariant norms. *Quart.J. Math*, 11:50–59, 1960.

[32] C. C. Paige. Error analysis of the Lanczos algorithms for tridiagonalizing a symmetric matrix. *J. Inst. Math. Appl.*, 18:341–349, 1976.

[33] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, NJ, 1980.

[34] B. N. Parlett and D. S. Scott. The Lanczos algorithm with selective reorthogonalization. *Math. Comp.*, 33:217–238, 1979.

[35] G. Peters and J. H. Wilkinson. Inverse iteration, ill-conditioned equations and Newton's method. *SIAM Review*, 21(3):339–360, 1979.

[36] H. Rutishauser. Computational Aspects of F.L. Bauer's Simultaneous Iteration Method. *Numer. Math.*, 13:4–13, 1969.

[37] H. Rutishauser. Simultaneous iteration method for symmetric matrices. *Numer. Math.*, 16:205–223, 1970.

[38] Y. Saad. On the rates of convergence of the Lanczos and the block-Lanczos methods. *SIAM J. Numer. Anal.*, 17:687–706, 1980.

[39] Y. Saad. Krylov subspace methods on supercomputers. *SIAM J. Sci. Stat. Comput.*, 10(6):1200–1232, 1989.

[40] A. H. Sameh and J. A. Wisniewski. A trace minimization algorithm for the generalized eigenvalue problem. *SIAM J. Numer. Anal.*, 19(6):1243–1259, 1982.

[41] J. A. Scales, P. Dochery, and A. Gerszternkorn. Regularization of nonlinear inverse problems: imaging the near-surface weathering layer. *Inverse Problems*, 6(1):115–131, 1990.

[42] J. A. Scales and A. Gerszternkorn. Robust methods in inverse theory. *Inverse Problems*, 4:1071–1091, 1988.

[43] H. Simon. Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Lin. Alg. and Its Appl.*, 61:101–131, 1984.

[44] B. Smith, J. Boyce, J. Dongarra, B. Garbow, Y. Ikebe, V. Klema, and C. Moler. *Matrix Eigensystem Routines - EISPACK Guide.* Springer, Berlin, 2nd ed., 1976.

[45] R. R. Underwood. An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems. PhD thesis, Stanford University, May 1975.

[46] R. S. Varga. *Matrix Iterative Analysis.* Prentice Hall, Englewood Cliffs, NJ, 1962.

[47] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[48] J. H. Wilkinson. *Inverse iteration in theory and in practice.* in Symposia Mathematica X, Academic Press, London, 361–379, 1972.

[49] J. A. Wisniewski. On solving the large sparse generalized eigenvalue problem. PhD thesis, The University of Illinois at Urbana-Champaign, Urbana, IL, February 1981.