SIAM J. MATRIX ANAL. APPL.
Vol. 15, No. 1, pp. 000–000, January 1994
© 1994 Society for Industrial and Applied Mathematics

A SHIFTED BLOCK LANCZOS ALGORITHM FOR SOLVING SPARSE SYMMETRIC GENERALIZED EIGENPROBLEMS*

ROGER G. GRIMES†, JOHN G. LEWIS†, AND HORST D. SIMON‡

Abstract. An "industrial strength" algorithm for solving sparse symmetric generalized eigenproblems is described. The algorithm has its foundations in known techniques in solving sparse symmetric eigenproblems, notably the spectral transformation of Ericsson and Ruhe and the block Lanczos algorithm. However, the combination of these two techniques is not trivial; there are many pitfalls awaiting the unwary implementor. The focus of this paper is on identifying those pitfalls and avoiding them, leading to a "bomb-proof" algorithm that can live as a black box eigensolver inside a large applications code. The code that results comprises a robust shift selection strategy and a block Lanczos algorithm that is a novel combination of new techniques and extensions of old techniques.

Key words. Lanczos algorithm, sparse eigenvalue problems, structural analysis, symmetric generalized eigenvalue problem, orthogonalization methods

AMS subject classifications. 65F15, 15A18, 65F50, 73K99

1. Introduction.

The Lanczos algorithm [22] is widely appreciated in the numerical analysis community [6]–[9], [14], [15], [17], [23], [29], [30], [32], [35], [37] as a very powerful tool for extracting some of the extreme eigenvalues of a real symmetric matrix H, i.e., to find the largest and/or smallest eigenvalues and vectors of the symmetric eigenvalue problem

$$Hx = \lambda x.$$

It is often believed that the algorithm can be used directly to find the eigenvalues at both ends of the spectrum (both largest and smallest in value). In fact, many applications result in eigenvalue distributions that only allow effectively extracting the eigenvalues at one end of the spectrum. Typical eigenvalue distributions in structural engineering vibration problems have small eigenvalues of order unity with separations $|\lambda_{i+1}-\lambda_i|$ also of order unity, apparently well separated. However, for physical reasons the largest eigenvalues of these problems are very large, say, $\mathcal{O}(10^{10})$. The convergence rates for the eigenvalues is determined by the relative separation $\frac{|\lambda_{i+1}-\lambda_i|}{|\lambda_n-\lambda_1|}$, $\mathcal{O}(10^{-10})$ for the smallest eigenvalues. We expect and find very slow convergence to the small eigenvalues, which are the eigenvalues of interest. The dependence of convergence on *relative* separation between eigenvalues is often ignored.

It is also often believed that the Lanczos algorithm can be applied to the generalized symmetric problem

$$Hx = \lambda Mx$$

by using the naive reduction to standard form [16], [32]: factor M into its Cholesky decomposition $M = LL^T$ and then solve the ordinary eigenproblem $L^{-1}HL^{-T}y =$

* Received by the editors May 23, 1988; accepted for publication (in revised form) March 18, 1992.

† Mathematics and Engineering Analysis, Research and Technology Division, Boeing Computer Services, Mail Stop 7L-22, P.O. Box 24346, Seattle, Washington 98124-0346 (rgrimes@espresso.rt.cs.boeing.com, jglewis@espresso.rt.cs.boeing.com).

‡ Numerical Aerodynamic Simulation (NAS) Systems Division, National Aeronautics and Space Administration Ames Research Center, Mail Stop TO45-1, Moffett Field, California 94035 (simon@nas.nasa.gov). The author is an employee of Computer Sciences Corporation. This work was funded in part through National Aeronautics and Space Administration contract NAS 2-12961.

λy. Suppose that we applied this algorithm to the *vibration* problem of structural engineering,

$$Kx = \lambda Mx, \tag{1}$$

where K is the stiffness matrix and M is the mass matrix. We would fail abysmally for three separate reasons:

- M is very often semidefinite—it may admit no Cholesky factorization.
- Even when M can be factored, the eigenvalues that are desired are often very badly separated.
- The eigenvectors x must be computed by a back transformation $x = L^{-T}y$. When it exists, L is usually poorly conditioned, which can lead to considerable numerical error in the back transformation.

When K is positive definite, the vibration problem can be addressed by applying the usual reduction to the reciprocal problem:

$$Kx = \lambda Mx \Leftrightarrow Mx = \frac{1}{\lambda}Kx \Leftrightarrow L^{-1}ML^{-T}y = \mu y, \tag{2}$$

where L is the Cholesky factor of K and $\mu = \frac{1}{\lambda}$. Often this is sufficient as a cure for the first two problems in (1), because the reciprocals of the eigenvalues are well separated. Eigenanalysis codes in structural engineering packages [24], [27] have been built upon this transformation. But this transformation is still inadequate when:

- the model has rigid body modes—K is positive semidefinite and has no Cholesky decomposition.
- a considerable number of eigenvalues are desired.
- the eigenvalues wanted are not the smallest eigenvalues.

Applications with these characteristics do arise. The stiffness matrix in aerospace applications often has a six-dimensional nullspace of rigid body modes. Detailed analyses of structures may require more than just a few eigenvalues and vectors. One of our test problems is an analysis of a nuclear reactor containment floor, where more than 200 eigenpairs were needed to adequately model the response of the structure to a simulated earthquake. Another problem we analyzed was a model of a large industrial ventilating fan mounted on a large concrete platform, for which we needed good approximations to the eigenvalues near the fan's rotational rate, eigenvalues that are in the interior of the spectrum.

There is a more elaborate transformation of the problem, the *spectral transformation* of Ericsson and Ruhe [14], which treats all of these difficulties. The spectral transformation is discussed in detail in §2, where we discuss an extension of the standard algorithm to buckling as well as to vibration problems. The general idea behind the spectral transformation comes from considering the shifted problem $(K-\sigma M)x = (\lambda-\sigma)Mx$. If we invert $(K-\sigma M)$, we transform the eigenvalues nearest the *shift* σ into the largest and well separated eigenvalues of the reciprocal problem. Normally we need only to choose a shift σ near the eigenvalues we want. When the number of eigenvalues is large, the reduced convergence rate of the eigenvalues farthest from σ makes it worthwhile to choose additional shifts (and factorizations) in order to search through the spectrum.

Formally we cannot shift at an eigenvalue of the problem, because the shifted operator is singular. In fact, avoiding even near-singularity is an issue for the choice of shifts, particularly the very first shift, because shifts very close to eigenvalues are useful only for computing isolated clusters of eigenvalues.

In general, a well chosen shift allows us to compute tens of eigenvalues with a single Lanczos run. There is a complicated tradeoff between the cost of a Lanczos run, which increases nonlinearly with increasing numbers of steps, and the cost of computing a new shift and its concomitant factorization. As an example, we consider the oceanography model (matrix PLAT1919 in the Harwell/Boeing sparse matrix collection [11]), with four different paradigms for choosing shifts:

- the heuristic described in this paper;
- a conservative modification of this heuristic;
- an aggressive modification of this heuristic;
- a fixed shift—compute all 200 eigenvalues with a single factorization.

All of these analyses begin with a Lanczos run using the factors of $A - .0001I$ to find the eigenvalues of $(A - .0001I)^{-1}$. Table 1 contains the salient results for these choices, demonstrating the complexity of the tradeoffs and, dramatically, the value of shifting.

TABLE 1
Computing the 200 *lowest eigenvalues in* [.0001,.24] *of* PLAT1919.

Choice of shift	Number of Lanczos runs	Total number of Lanczos steps	Execution cost
normal	8	192	208.1
conservative	13	243	257.4
aggressive	8	209	225.5
fixed shift	1	318	5382.2

(These results were obtained on a Sun 4/690 workstation. The code used a blocksize of three. Execution cost is the sum of central processor (cpu) and input/output (i/o) processor seconds.)

Shifting can provide reliability as well as efficiency. Each factorization provides eigenvalue location information in the form of *matrix inertias* (see §3.1). The collected inertias from a series of well chosen shifts can provide an independent guarantee on the success of the eigenvalue computation and can be used to drive the choice of further shifts and Lanczos runs to ensure that all of the desired eigenvalues have been computed. Our heuristic strategy for choosing shifts is discussed in §3.

Our goal is a code that can serve as a "black-box" eigenextraction routine in large applications codes. Eigenvalues cannot be assumed to be simple, so our shifting strategy is prepared to continue looking at a small piece of the spectrum until it has determined the full multiplicity of the eigenvalues therein. The shifting scheme and the Lanczos algorithm interact to ensure that we find an orthogonal basis for the invariant subspace for each cluster (see §4.3.3). Most importantly, we use a *block* version of the Lanczos algorithm. The Lanczos algorithm usually will compute the full multiplicities of each cluster without any intervention from the shifting strategy, provided that we have been able to choose a blocksize as large as the largest multiplicity of any cluster we will encounter.

The block Lanczos algorithm also confronts the problem that applications codes often use general representations for their data, even when particular machine architectures would allow or favor alternatives. It is still common for general applications codes to represent their matrices as "out-of-core." The block Lanczos code substitutes, almost on a one-for-one basis, matrix-block multiplies and block solves for matrix-vector products and simple solves. This decreases the i/o cost essentially by the blocksize.

Our production eigenextraction code is a synthesis of the ideas of the spectral transformation and the block Lanczos algorithm. In §2 we begin to address the effects of the generalized problem on the recurrence. We explain what modifications to the Lanczos recurrence result from the use of shifted and inverted operators. With the exception of the development of a spectral transformation for buckling problems, our presentation is quite standard and is provided for the reader not already familiar with these results.

We present our heuristic shifting strategy in §3. There are eight subsections: a discussion of *trust intervals* and matrix inertias, our basic tools for robustness; our heuristic for choosing a shift in a generic case; the idea of *sentinels*, a tool for ensuring orthogonality of invariant subspaces; heuristics for choosing an initial shift; heuristics for determining how to expand the primary trust interval; analysis of a specified finite interval; treatment of various special and pathological cases; and, last, the modifications needed for the buckling problem.

The special characteristics of our block Lanczos algorithm are discussed in §4. This considers the effects due to the spectral transformation. One major problem is that vectors must be orthonormalized with respect to an inner product defined by a positive definite matrix M. We discuss the issues associated with implementing M-orthonormalization of vectors in the basic block Lanczos algorithm, including the further precautions needed to allow cases where M induces only a seminorm, in §4.1.

The block Lanczos recurrence by itself produces only a block tridiagonal matrix T. In §4.2 we describe how to compute eigenvalue and vector approximations, and error bounds on these approximations, from T and the Lanczos vectors. Section 4.3 contains our approach for dealing with the loss of orthogonality in the Lanczos vectors, with a novel combination of various reorthogonalization schemes that work effectively with the unusual distributions of eigenvalues that result from the spectral transformation. Section 4.4 concludes with discussions of when to end and how to start the recurrence. The integration of all of these techniques is a block Lanczos recurrence that will effectively find a limited number of eigenvalues and corresponding eigenvectors of a spectrally transformed operator.

We close with numerical experiments solving a small set of eigenproblems obtained from applications codes.

2. The spectral transformation block Lanczos algorithm. The eigenvalue problem in vibration analysis is given as

$$Kx = \lambda Mx, \tag{3}$$

where K and M are symmetric matrices, and M is positive semidefinite. Usually only the smallest eigenvalues of (3) are wanted, but they typically have very poor relative separation, rarely better than $\mathcal{O}(10^{-6})$. A priori estimates for the rate of convergence predict very slow convergence at the desired end of the spectrum. We can obtain rapid convergence to the desired eigenvalues by using the *spectral transformation* [14], [27] of (3).

2.1. The spectral transformation for vibration problems. Consider the problem

$$M(K - \sigma M)^{-1}Mx = \mu Mx, \tag{4}$$

where σ, the shift, is a real parameter. Assume for the moment that M is positive definite. It is easy to verify that (λ, x) is an eigenpair of (3) if and only if $(\frac{1}{\lambda-\sigma}, x)$ is

an eigenpair of (4). Hence, the transformation of the eigenvalue problem from (3) to (4) does not change the eigenvectors, and the eigenvalues are related by

$$\mu = \frac{1}{\lambda - \sigma}. \tag{5}$$

The form of the spectral transformation is dictated by our need to be able to apply the Lanczos algorithm even when M is semidefinite. Other advantages of this form are well documented in [38].

The main advantage of applying the Lanczos algorithm to (4) instead of to (3) becomes clear when the effect of the spectral transformation on the spectrum is considered. The results in Table 2 demonstrate this in detail. These are the values obtained using the initial shift described in §3.4; the generalized eigenproblem is the model of a nuclear reactor containment floor, given by the stiffness and mass matrices BCSSTK26 and BCSSTM26, respectively, from the Harwell–Boeing sparse matrix collection [11]. (We denote the generalized eigenproblem by BCSST_26.)

Relative separation is affected dramatically by the spectral transformation. The smallest eigenvalues are transformed into eigenvalues with good relative separation, even though their absolute separation is decreased. In addition, eigenvalues far from the shift are transformed to poorly separated values near zero. This spread of the eigenvalues ensures rapid convergence to the eigenvalues near σ. This example clearly demonstrates that the shift does not have to be very close in an absolute sense to work well.

TABLE 2
Vibration spectral transformation of BCSST_26, $\sigma_1 = 385.3$.

			Original		Transformed	
i	λ_i	μ_i	gap	relative gap	gap	relative gap
1	4.6×10^{3}	2.4×10^{-4}	6.4×10^{3}	1.2×10^{-11}	1.4×10^{-4}	6.0×10^{-1}
2	1.1×10^{4}	9.4×10^{-5}	2.5×10^{2}	4.6×10^{-13}	2.2×10^{-6}	9.2×10^{-3}
3	1.1×10^{4}	9.2×10^{-5}	2.5×10^{2}	4.6×10^{-13}	2.2×10^{-6}	9.2×10^{-3}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1920	3.0×10^{14}	3.3×10^{-15}	3.6×10^{11}	6.7×10^{-4}	3.9×10^{-18}	1.7×10^{-14}
1921	3.1×10^{14}	3.3×10^{-15}	3.6×10^{11}	6.7×10^{-4}	3.9×10^{-18}	1.7×10^{-14}
1922	5.4×10^{14}	1.8×10^{-15}	2.4×10^{14}	4.4×10^{-1}	1.4×10^{-15}	6.0×10^{-12}

The primary price for this rapid convergence is the cost of a factorization of $K - \sigma M$. The transformation $M(K - \sigma M)^{-1}M$ is realized implicitly as a sequence of operations in which we compute MQ for a block of vectors Q or solve the linear systems $(K - \sigma M)X = Q$. These operations are usually realized by independent subroutines, which allow tuning the matrix factorization and multiplication routines to the class of problem under consideration.

We must generalize the Lanczos algorithm itself to solve the transformed generalized symmetric eigenproblem. We make this generalization in three steps. We will first consider the ordinary block Lanczos algorithm for a symmetric matrix H. Next we consider a direct generalization of the Lanczos algorithm for an arbitrary generalized symmetric eigenproblem $Hx = \lambda Mx$, where we assume temporarily that M is positive definite. In these first two steps the issue of shifting disappears for the moment. In a third step we consider the much more effective form that results when H is a spectral transformation operator.

2.2. Basic block Lanczos algorithm. Consider first the ordinary eigenvalue problem

$$Hx = \lambda x,$$

where H is a real symmetric linear operator. An important characteristic of the Lanczos algorithm is that H is not required explicitly. All that is required is a subroutine that computes Hy for a given vector y. The block Lanczos iteration with *blocksize* p for an $n \times n$ matrix H is given in Fig. 1.

Initialization:

Set $Q_0 = 0$
Set $B_1 = 0$
Choose R_1 and orthonormalize the columns of R_1 to obtain Q_1

Lanczos Loop:

For $j = 1, 2, 3 \ldots$ do
Set $U_j = HQ_j - Q_{j-1}B_j^T$
Set $A_j = Q_j^T U_j$
Set $R_{j+1} = U_j - Q_j A_j$, the residual
Compute the orthogonal factorization $Q_{j+1}B_{j+1} = R_{j+1}$,
where B_{j+1} is upper triangular and Q_{j+1} is orthogonal
End loop

FIG. 1. *Basic block Lanczos algorithm.*

The matrices Q_j, U_j, R_j for $j = 1, 2, \ldots$ are $n \times p$, whereas A_j and B_j are $p \times p$, with A_j symmetric.

This formulation of the Lanczos loop is the one least susceptible to roundoff errors [31] and is the form that should be used in computation. In exact arithmetic, however, U_j and R_{j+1} can be eliminated from the Lanczos loop and the recurrence becomes

$$Q_{j+1}B_{j+1} = HQ_j - Q_j A_j - Q_{j-1}B_j^T. \tag{6}$$

This three-term recurrence simplifies theoretical discussion. It is shown in [6], [17] that the combined column vectors of the matrices $Q_1, Q_2, \ldots, Q_j$, the so-called *Lanczos vectors*, form an orthonormal set. The computational efficiency of the Lanczos algorithm rests on the fact that these vectors can be computed simply, with a fixed amount of work per iteration step.

The blocks of Lanczos vectors collectively form an $n \times jp$ matrix $\mathcal{Q}_j$, where

$$\mathcal{Q}_j = [Q_1, Q_2, Q_3, \ldots, Q_j].$$

The algorithm also defines a $jp \times jp$ block tridiagonal matrix T_j:

$$T_j = \begin{pmatrix} A_1 & B_2^T & 0 & \ldots & 0 \\ B_2 & A_2 & B_3^T & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & B_{j-1} & A_{j-1} & B_j^T \\ 0 & \ldots & 0 & B_j & A_j \end{pmatrix}.$$

Since the matrices B_j are upper triangular, T_j is a band matrix with half-bandwidth $p+1$ (rather than $2p$, if the B_j were full). The first j instances of formula (6) can be combined into a single formula:

$$H\mathcal{Q}_j = \mathcal{Q}_j T_j + Q_{j+1} B_{j+1} E_j^T. \tag{7}$$

Here E_j is an $n \times p$ matrix whose last $p \times p$ block is the $p \times p$ identity matrix and which is zero otherwise.

By premultiplying (7) by $\mathcal{Q}_j^T$ and using the orthogonality of the Lanczos vectors, we see that $\mathcal{Q}_j^T H \mathcal{Q}_j = T_j$. Hence T_j is the orthogonal projection of H onto the subspace $\mathrm{span}(\mathcal{Q}_j)$ spanned by the columns of $\mathcal{Q}_j$. It can be shown by induction that $\mathrm{span}(\mathcal{Q}_j) = \mathrm{span}(Q_1, HQ_1, H^2Q_1, \ldots, H^{j-1}Q_1)$. From a different perspective, the (block) Lanczos algorithm is a method for constructing an orthonormal basis for the (block) Krylov subspace determined by H and Q_1. The orthogonal projection of H onto the (block) Krylov subspace is T_j. Hence the eigenvalues of T_j are the Rayleigh–Ritz approximations from $\mathrm{span}(\mathcal{Q}_j)$ to the eigenvalues of H. In addition, if s is an eigenvector of T_j, the vector $y = \mathcal{Q}_j s$ is an approximate eigenvector of H. Viewed in this form, the Lanczos algorithm replaces a large and difficult eigenvalue problem involving H by a small and easy eigenvalue problem involving the block tridiagonal matrix T_j.

How good are the approximations obtained by solving the block tridiagonal eigenvalue problem involving the matrix T_j? An a posteriori bound on the residual is given by Underwood [17]: Let θ, s be an eigenpair for T_j, i.e., $T_j s = s\theta$, and let $y = \mathcal{Q}_j s$, then

$$\|Hy - y\theta\|_2 = \|B_{j+1} s_j\|_2, \tag{8}$$

where s_j are the last p components of the eigenvector s. The quantity $\|B_{j+1} s_j\|_2$ can be computed without computing the approximate eigenvector y. Hence, with some modifications described in §4.2, (8) provides an inexpensive a posteriori error bound.

Formula (8), however, does not guarantee that good approximations to eigenpairs will appear quickly. Such a priori estimates are provided by the Kaniel–Paige–Saad theory. Parlett [32] gives the most detailed discussion for the single vector case ($p = 1$). The generalizations to the block case were originally derived by Underwood [17]. Extensions to both of these presentations can be found in [36].

2.3. The spectral transformation block Lanczos algorithm. The next step is to consider the generalized symmetric eigenproblem $Hx = \lambda M x$. Were we to reduce the problem to standard form by factoring M, the three-term recurrence (6) would become

$$Q_{j+1} B_{j+1} = M^{-1/2} H M^{-1/2} Q_j - Q_j A_j - Q_{j-1} B_j^T. \tag{9}$$

If we premultiply (9) by $M^{1/2}$ and make the transformation of variables $\hat{Q}_j = M^{-1/2} Q_j$, (9) becomes

$$\begin{aligned} M\hat{Q}_{j+1} B_{j+1} &= M^{1/2} M^{-1/2} H \hat{Q}_j - M\hat{Q}_j A_j - M\hat{Q}_{j-1} B_j^T \\ &= H\hat{Q}_j - M\hat{Q}_j A_j - M\hat{Q}_{j-1} B_j^T. \end{aligned} \tag{10}$$

The matrices $\hat{Q}_j$ are now M-orthogonal, since $Q_j^T Q_j = I$ implies $\hat{Q}_j^T M \hat{Q}_j = I$. This is also a property of the eigenvectors X of this generalized eigenproblem. The approximate eigenvectors will eventually be computed in the subspace $\mathrm{span}(\hat{\mathcal{Q}})$, regardless

For $j = 1, 2, 3 \ldots$ do
  Set U_j $= H\hat{Q}_j - M\hat{Q}_{j-1}B_j^T$
  Set A_j $= \hat{Q}_j^T M U_j$
  Set W_{j+1} $= U_j - M\hat{Q}_j A_j$
  Solve $MR_{j+1} = W_{j+1}$
  Compute the M-orthogonal factorization $\hat{Q}_{j+1}B_{j+1} = R_{j+1}$
End loop

FIG. 2. *Inner loop of generalized symmetric block Lanczos algorithm.*

of the form used for the Lanczos recurrence. The inner loop of Lanczos recurrence in this subspace is given in Fig. 2.

The matrix M appears in several instances to assure the M-orthogonality of the Lanczos vectors. In particular, the last step requires computing the M-orthogonal factorization of R_{j+1}. Standard derivations of the orthogonality of the Lanczos vectors easily generalize to show that these vectors are M-orthonormal. It appears that $M^{-1/2}$ has disappeared from the standard recurrence, only to reappear at the penultimate step in disguise as a solution operation. Indeed, (10) applied to the original problem $Kx = \lambda Mx$ is merely an implicit form of the explicit reduction to standard form. This is not the case when H is taken as the operator in the spectral transformation. Substituting $M(K - \sigma M)^{-1}M$ for H gives:

$$M\hat{Q}_{j+1}B_{j+1} = M(K - \sigma M)^{-1}M\hat{Q}_j - M\hat{Q}_j A_j - M\hat{Q}_{j-1}B_j^T. \tag{11}$$

M now appears in *all* of the terms in the recurrence. Formally we can premultiply (11) by M^{-1} to obtain a recurrence

$$\hat{Q}_{j+1}B_{j+1} = (K - \sigma M)^{-1}M\hat{Q}_j - \hat{Q}_j A_j - \hat{Q}_{j-1}B_j^T \tag{12}$$

in which M^{-1} does not appear. This allows us to apply the same recurrence even when M is semidefinite. The justification for doing so appears later in §2.4.

At this point we shall no longer put "hats" on the matrices. The actual Lanczos recurrence for solving (4) is given in Fig. 3.

Assuming the matrix MQ_{j+1} is actually stored (at least temporarily), the algorithm as written requires only one multiplication by M per step and no factorization of M is required. The last step of the Lanczos loop, the *M-orthogonalization* of a set of p vectors, is discussed in §4.1.

Our next goal is to generalize the standard eigenvalue approximation results to the spectral transformation block Lanczos algorithm. As before, combining all j instances of (12) into one equation yields

$$(K - \sigma M)^{-1}M\mathcal{Q}_j = \mathcal{Q}_j T_j + Q_{j+1}B_{j+1}E_j^T, \tag{13}$$

where $\mathcal{Q}_j$, T_j, and E_j are defined as in (7). Premultiplying (13) by $\mathcal{Q}_j^T M$ and using the M-orthogonality of the Lanczos vectors, it follows that

$$\mathcal{Q}_j^T M(K - \sigma M)^{-1}M\mathcal{Q}_j = T_j.$$

Hence, T_j is the M-orthogonal projection of $(K - \sigma M)^{-1}$ onto the block Krylov subspace spanned by the columns of $\mathcal{Q}_j$. The eigenvalues of T_j will approximate the

Initialization:

Set $Q_0 = 0$
Set $B_1 = 0$
Choose R_1 and orthonormalize the columns of R_1 to obtain Q_1
with $Q_1^T(MQ_1) = I_p$

Lanczos Loop:

For $j = 1, 2, 3 \ldots$ do
Set $U_j = (K - \sigma M)^{-1}(MQ_j) - Q_{j-1}B_j^T$
Set $A_j = U_j^T(MQ_j)$
Set $R_{j+1} = U_j - Q_jA_j$
Compute Q_{j+1} and (MQ_{j+1}) such that
a) $Q_{j+1}B_{j+1} = R_{j+1}$
b) $Q_{j+1}^T(MQ_{j+1}) = I_p$
End loop

FIG. 3. *Block Lanczos algorithm for the vibration problem.*

eigenvalues of (4). If (s, θ) is an eigenpair of T_j, i.e., $T_j s = s\theta$, then $(y = \mathcal{Q}_j s, \nu = \sigma + \frac{1}{\theta})$ will be an approximate eigenpair of (3).

The generalization of the a posteriori residual bound (8) is

$$(K - \sigma M)^{-1}My - y\theta = Q_{j+1}B_{j+1}E_j^T s. \tag{14}$$

For $\theta \neq 0$ it follows that

$$(K - \nu M)y = -\frac{1}{\theta}(K - \sigma M)Q_{j+1}B_{j+1}E_j^T s.$$

The quantity on the right is computable without explicitly computing the eigenvector y, but only at the cost of a multiplication by $K - \sigma M$, which is not desirable. In §4.2 we present a better way to obtain a residual bound. (Note that $\theta = 0$ corresponds to an infinite eigenvalue of (3), which should not appear in T, as discussed below. Very small θ's correspond to eigenvalues far from the shift. These converge slowly—the division by θ in the residual bounds reflects their relative inaccuracy.)

2.4. Semidefiniteness in the matrix M. Throughout the discussion above, we assumed that M was a positive definite matrix. The formulation of the block Lanczos algorithm for the vibration problem does not require the factorization of M. Hence the spectral transformation Lanczos algorithm can be applied formally when M is semidefinite without further modifications. However, the eigenproblem (3) has infinite eigenvalues. Fortunately, we need only to make the obvious block modification of the analysis in [29] to remove the infinite eigenpairs from the recurrence. Following Nour-Omid et al., the starting block for the Lanczos algorithm should be computed as in Fig. 4.

The eigenvectors of $Kx = \lambda Mx$ corresponding to finite eigenvalues consist of a component orthogonal to the null vectors of M and a component in the nullspace of M. Ericsson [13] shows that the second, nullspace component is determined by an algebraic constraint from the non-nullspace component. The constraint expresses the

Choose	$\tilde{R}_1$
Compute	$R_1 = (K - \sigma M)^{-1} M \tilde{R}_1$
M-orthogonalize	$R_1 = Q_1 B_0$

FIG. 4. *Computation of the starting block.*

fact that all of these eigenvectors lie in the range of $(K - \sigma M)^{-1}M$. It is shown in [13], [29] that all of the Lanczos vectors lie in this subspace when the starting vectors are chosen in this subspace, as above. With this choice of starting block, infinite eigenvalues have no influence on the block Lanczos algorithm in exact arithmetic. In §4.2 we add a final postprocessing step to purge the approximate eigenvectors of components not satisfying the constraint in finite precision arithmetic.

2.5. A spectral transformation for buckling problems. The final point of this section is the spectral transformation for the buckling problem

$$Kx = \lambda K_\delta x, \tag{15}$$

where K is the symmetric positive semidefinite stiffness matrix and K_δ is the symmetric differential or geometric stiffness matrix. Typically only a few eigenvalues closest to zero are wanted. A simple approach would be to interchange the roles of K and K_δ and to compute the largest eigenvalues of the problem

$$K_\delta x = \mu K x, \tag{16}$$

with $\mu = \frac{1}{\lambda}$ by applying the simple Lanczos algorithm without shifts [21]. This reciprocal approach has the same drawbacks as (2). However, it is often effective when K is positive definite because the number of eigenvalues sought is rarely large.

Shifting and particularly the semidefinite K case require an alternative form of the spectral transformation [19]. The shifted and inverted problem

$$K(K - \sigma K_\delta)^{-1} K x = \mu K x \tag{17}$$

is solved instead of the original problem (15). The Lanczos recurrence is carried out using K-orthogonality among the Lanczos vectors. Each multiplication by the mass matrix M in the vibration case is replaced with a multiplication by the stiffness matrix K in the buckling case; the rest of the recurrence remains the same.

In the buckling spectral transformation (λ, x) is an eigenpair of (15) if and only if $(\frac{\lambda}{\lambda - \sigma}, x)$ is an eigenpair of (17). Hence the *buckling spectral transformation* does not change the eigenvectors, and the eigenvalues are related by $\mu = \frac{\lambda}{\lambda - \sigma}$. These results can be obtained directly, or by applying the vibration spectral transformation with reciprocated shifts to the reciprocal problem (16).

The advantages of the buckling spectral transformation are essentially the same as those of the vibration spectral transformation. Large eigenvalues of the buckling problem are transformed to a cluster of eigenvalues near unity. Eigenvalues near the shift σ are transformed into well separated eigenvalues, which are easily computed by the Lanczos algorithm. The major difference is that a shift at $\sigma = 0$ is not allowed, since all eigenvalues would be transformed to one. This singularity in the transformation also affects shifts close to zero; very small shifts should not be taken in this form of the transformation. Table 3 gives details for the eigenproblem BCSST_28, treated as if it were a buckling problem. The initial shift is negative because we ordinarily

expect the first negative eigenvalue to be the eigenvalue of most interest in buckling problems (see §3.8). Just as in the case of the vibration spectral transformation, we see that the shift does not need to be close to the desired eigenvalues in any absolute sense. Indeed, in this case the shift is on the wrong side of the origin and yet still has the desired effect on relative separation.

TABLE 3
Buckling spectral transformation of BCSST_26, $\sigma_1 = -385.3$.

			Original		Transformed	
i	λ_i	μ_i	gap	relative gap	gap	relative gap
1	4.6×10^{3}	9.23×10^{-1}	6.4×10^{3}	1.2×10^{-11}	4.3×10^{-2}	5.6×10^{-1}
2	1.1×10^{4}	9.66×10^{-1}	2.5×10^{2}	4.6×10^{-13}	7.3×10^{-4}	9.5×10^{-3}
3	1.1×10^{4}	9.67×10^{-1}	2.5×10^{2}	4.6×10^{-13}	7.3×10^{-4}	9.5×10^{-3}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1920	3.0×10^{14}	1.00×10^{0}	3.6×10^{11}	6.7×10^{-4}	1.6×10^{-15}	2.0×10^{-14}
1921	3.1×10^{14}	1.00×10^{0}	3.6×10^{11}	6.7×10^{-4}	1.6×10^{-15}	2.0×10^{-14}
1922	5.4×10^{14}	1.00×10^{0}	2.4×10^{14}	4.4×10^{-1}	5.5×10^{-13}	7.1×10^{-12}

Except for the different role of the stiffness matrix K, all implementation details are the same for vibration and buckling analysis. Issues involving the M-orthogonality of the Lanczos vectors apply equally to the K-orthogonal Lanczos vectors in the buckling case. Since the stiffness matrix K is used in the initialization phase in the same way as M in the vibration case, the sequence of Lanczos vectors will be orthogonal to the space spanned by the eigenvectors corresponding to zero eigenvalues of K. Hence T_j will contain no approximations to the exactly zero eigenvalues of K, which are also zero eigenvalues of (15), which is desirable.

The eigenvalues of T_j approximate the eigenvalues of (17). Hence, if (s, θ) is an eigenpair of T_j, that is, $T_j s = s\theta$, then $(\frac{\sigma\theta}{\theta - 1}, Q_j s)$ is an approximate eigenpair of (15). The approximate eigenvectors form a K-orthonormal set. Bounds on the residuals of approximate eigenpairs are derived in §4.2.

3. A strategy for choosing shifts. Let us try to find some of the eigenvalues and eigenvectors of $KX = MX\Lambda$ or $KX = K_\delta X\Lambda$. We emphasize the fact that we want some, not all, of the eigenvalues, because the eigenvector matrix X is almost always dense. The problem can be written in its general form as:

- find the p eigenvalues of smallest magnitude in $[a, b]$ and their eigenvectors; or
- find the p eigenvalues of largest magnitude in $[a, b]$ and their eigenvectors; or
- find the p eigenvalues in $[a, b]$ closest to ξ and their eigenvectors; or
- find all eigenvalues and eigenvectors in $[a, b]$.

Here $[a, b]$ is the *computational interval*, which can be finite (both a and b finite), semi-infinite (only one of a and b finite), or infinite (no restrictions at all). Note that the problem of finding the algebraically least eigenvalues in an interval can be transformed into one of finding the eigenvalues of smallest magnitude by a suitable shift of origin.

The purpose of the spectral transformation is to transform the original problem into one whose dominant eigenvalues represent some of the desired eigenvalues. The dominant eigenvalues of the transformed problem correspond to the eigenvalues of the original problem nearest σ. There are two major goals that drive our strategy for choosing shifts. One is efficiency—we would like to choose a sequence of shifts $\sigma_1, \sigma_2, \ldots, \sigma_s$ so that the total cost, including the cost of the s factorizations and

the costs of the individual Lanczos runs, is minimized. Our heuristic approach to measuring and reducing cost is described in §§3.2 and 4.4. The second goal of our shift selection is robustness. A paramount objective for our design was a code that would be able to compute all of the desired eigenpairs accurately, except under extreme, pathological conditions. Furthermore, we wanted a code that could diagnose and report any failures. The tools we use to create robustness, trust intervals, and matrix inertias, are an appropriate place to begin the detailed discussion of our choices of shifts.

3.1. Trust intervals, matrix factorizations, and inertias. Suppose that during the course of eigenanalysis, we have computed a set of eigenvalues lying between two shifts σ_1 and σ_2. We would like to confirm that these are, in fact, all the eigenvalues in this interval.

Suppose that C is a real symmetric matrix, which has been decomposed as $C = LDL^T$, where D is diagonal. The *inertia* of C is the triple (π, ν, ζ) of integers, where π is the number of positive eigenvalues, ν the number of negative eigenvalues, and ζ the number of zero eigenvalues. Sylvester's Inertia Theorem [32, p. 10] states that the inertia of F^TCF is the same as that of C. Sylvester's theorem with $F = L^{-T}$ implies that the number of negative entries in D is the number of negative eigenvalues from C. The number of negative terms in D from the LDL^T decomposition of $C - \sigma I$ gives the number of eigenvalues smaller than σ. Frequently $\nu(C - \sigma I)$ is called the Sturm sequence number in engineering references.

It is easy to see that $\nu(C - \sigma_2 I) - \nu(C - \sigma_1 I)$ is the number of eigenvalues in the interval $[\sigma_1, \sigma_2]$ (assuming $\sigma_1 < \sigma_2$ and the two factorizations are nonsingular). When the number of eigenvalues expected in the interval agree with the number actually computed, we say that the interval $[\sigma_1, \sigma_2]$ is a *trust interval*. We want our shifting strategy to establish a trust interval around all of the desired eigenvalues.

However, applying these Sturm sequence results to generalized eigenproblems requires a transformation from the ordinary eigenvalue problem $CX = X\Lambda$ to the generalized problem $KX = MX\Lambda$. In order to guarantee that the generalized eigenvalue problems have real solutions, we assume that the pencils are definite; a positive definite linear combination of K and M must exist. In our code we assume that M or K is positive semidefinite. We compute $K - \sigma M = LDL^T$ (or $K - \sigma K_\delta = LDL^T$), and we want to draw conclusions from $\nu(LDL^T)$. The interpretation of $\nu(LDL^T)$ is given in Table 4; proofs are found in Appendix A. The major surprise in this table of the appearance of the null space dimension $\dim(\mathcal{N}(\cdot))$ when the matrix used as a norm is only a seminorm. This term corresponds to an assignment of signs to the infinite eigenvalues in the vibration case and the zero eigenvalues in the buckling case. We note that in most common vibration cases the term $\dim(\mathcal{N}(M))$ does not appear, because K is positive semidefinite. When it does appear, it is because the infinite eigenvalues have negative signs, which adds a serious complication to the problem of finding the algebraically smallest eigenvalues (the infinite eigenvalues are the algebraically smallest, but cannot be computed by the recurrence as written). However, the problem of finding the eigenvalues of smallest magnitude is only slightly more difficult in this case.

Semidefiniteness in buckling analysis is more significant, because the usual problem is to find the eigenvalues of smallest magnitude and the zero eigenvalues cannot be computed directly. The problem still can be solved if $\dim(\mathcal{N}(K))$ is known, either adventitiously or by a partial eigenanalysis of K. The problem of finding the eigenvalues of smallest magnitude in an interval bounded away from zero is still

TABLE 4
Interpretation of $\nu(K - \sigma M)$ or $\nu(K - \sigma K_\delta)$.

Vibration analysis:	
M positive definite	# of eigenvalues $< \sigma$
M positive semidefinite	(# of eigenvalues $< \sigma$) $+ \gamma$
	$\gamma = \begin{cases} 0 & \text{some cases} \\ \dim(\mathcal{N}(M)) & \text{other cases} \end{cases}$

Buckling analysis:	
K positive definite	# of eigenvalues in $(0, \sigma)$ or $(\sigma, 0)$
K positive semidefinite	(# of eigenvalues in $(0, \sigma)$ or $(\sigma, 0)$) $+ \gamma$
	$\gamma = \begin{cases} 0 & \sigma \text{ of one sign} \\ \dim(\mathcal{N}(K)) & \sigma \text{ of other sign} \end{cases}$

well posed.

The result of a successful eigenextraction is a trust interval containing all of the desired eigenvalues. This goal drives our selection of shifts. We create, as soon as possible, a trust interval containing some of the desired modes; thereafter, we extend the trust interval to contain more, and eventually all, of the desired modes. The process begins with an initial shift at some point σ_1. The factorization is followed by a Lanczos run with the shifted operator $(K - \sigma_1 M)^{-1} M$ (or its counterpart in buckling analysis). We will always compute a second factorization, if only to provide the inertia to close a trust interval. If only some of the desired eigenvalues were computed during the first Lanczos run, we would like to make the factorization at σ_2 serve both as a basis for an inertia computation and as the factorization for a new Lanczos run. Ideally we would choose σ_2 close enough to σ_1 that the second Lanczos run finds all the remaining eigenvalues in the interval; at the same time, we would like σ_2 to be far enough away from σ_1 so that the second Lanczos run stops, for efficiency reasons, exactly when it has computed all the missing eigenvalues. Thus, a simple description of our shift selection is that we choose each new shift to *maximally* extend an existing trust interval.

3.2. Shifting to extend a trust interval. In selecting each new shift, we try to use as much information as we have, including any computed eigenvalues, other knowledge about the existing trust interval, and additional information from the previous Lanczos runs. In general, each Lanczos run creates a set of approximations to eigenvalues, which provide a general picture of the spectrum. Fig. 5 gives an illustration of the general situation, in which the last Lanczos run was at a shift σ_i that forms the right endpoint of a trust interval. The tall, thin lines denote approximations that we accept as eigenvalues. The lines of medium height and width are approximations that are not yet acceptable as eigenvalues, though they do have accuracy estimates good enough to know that at least one significant digit is correct. We call these *Ritz values*. (All of the Lanczos approximations are Ritz values, but we abuse the mathematical term to describe only those approximations that are not good enough to be accepted, and not bad enough to be meaningless.) The short, broad lines denote approximations whose accuracy estimates are larger than their values, which we ignore.

The shift selection assumes that the inverted spectrum as viewed from σ_{i+1} will be similar to the inverted spectrum as viewed from σ_i. One view of this similarity of inverted spectra is that if the Lanczos run from σ_i computed k eigenvalues to the right of σ_i efficiently, we expect that an efficient run at any σ_{i+1} should compute

k eigenvalues to its left. We use the first k Ritz values to estimate the missing eigenvalues and place the new shift σ_{i+1} between the kth and $(k+1)$st Ritz values. The choice of the bisector is intended to avoid a choice extremely close to an eigenvalue. Furthermore, we use a relaxed tolerance to detect "clusters" of eigenvalues and bisect clusters rather than Ritz values.

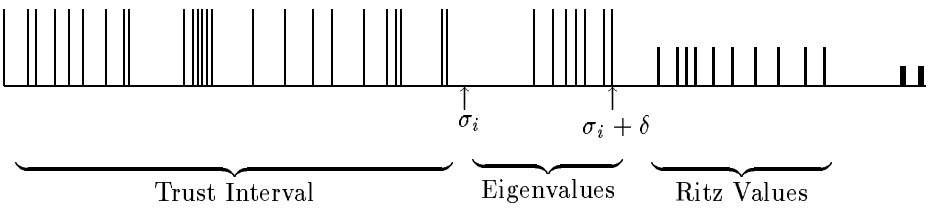


FIG. 5. *Trust intervals.*

If there are fewer than k Ritz values available to the right of σ_i, we use a second view of the inverted spectra based on the assumption that the "radius of convergence" should be about the same for each shift. We define δ to be the maximum of its previous value and the distance between the right endpoint of the trust interval and the rightmost computed eigenvalue (see Fig. 5). Initially, δ is set to the problem scale (see §3.4). Then a second choice for the next shift is $\sigma_{i+1} = \sigma_i + 2 * \delta$. We take the more aggressive choice, the maximum of the two possibilities, in the case where we still need to compute more eigenvalues than we have knowledge of Ritz values. If more Ritz values are available than there are eigenvalues left to compute, we choose the next shift based solely on the Ritz values, ignoring the shift based on δ.

Table 1 shows some results for normal, conservative, aggressive, and fixed shifting. For this table, we used a $1 * k$, $1 * \delta$ rule for conservative shifting and a $3 * k$, $3 * \delta$ rule for aggressive shifting.

We have described the general rule for choosing σ_{i+1} when σ_{i+1} is taken to the right of σ_i. Of course, we obtain two similar views of the spectra to the left of σ_i, which give another alternative for the next shift. In general we do not know in which direction the next shift should be taken. Indeed, when finding eigenvalues nearest to an interior point we first move in one direction from σ_i and then in the other direction. At the completion of each Lanczos run in which we attempted to extend a trust interval, we compute, and save, the next shift that would extend the new trust interval further in the same direction. The first shift, unless it is at a finite endpoint of the computational interval, is treated as extending the null trust interval both to the left and to the right. The Ritz values are then discarded.

These two views of the inverted spectra, albeit simplistic, have proven to be effective. A model based on convergence rates of the eigenvalues [36] is far too pessimistic to be of any use here.

3.3. Sentinels. There are several aspects of our eigenanalysis code where the shift selection mechanism and the implementation of the Lanczos algorithm are closely tied together. For example, we do not want to recompute at later shifts eigenpairs that have been computed from earlier shifts. Any computation spent recomputing known eigenpairs is wasted. Even allowing accidental recomputation creates a difficult situation in which we must determine the correct multiplicity of a computed eigenvalue

for which several eigenvectors have been computed. We choose never to allow this situation to arise.

In theory there is a very simple fix. If the starting block for the Lanczos recurrence is chosen to be M-orthogonal to all previously computed eigenvectors, the recurrence should remain M-orthogonal to all previously computed eigenvectors. This is not sufficient in practice, as rounding errors introduce components of the excluded eigenvectors. We reorthogonalize the recurrence to specific eigenvectors only when necessary using *external selective orthogonalization* (see §4.3.3). This mechanism dramatically reduces the cost of preventing the reappearance of excluded eigenvectors.

A second mechanism for reducing this cost is in the purview of the shifting code. A common situation is depicted in Fig. 6. The new shift, σ_{i+1}, has been chosen; the nearest previous shift, σ_j, forms the end of a trust interval. (Fig. 6 depicts the initial case where the trust interval including σ_j is trivial.) Between the two shifts lie a set of eigenvalues and Ritz values computed during the run at σ_j. Because the convergence rate for the eigenvalues in the Lanczos algorithm decreases as the distance from the shift increases, the usual pattern is that the accepted eigenvalues are those closest to σ_j and the Ritz values are those farther out with little or no interlacing of the two sets.

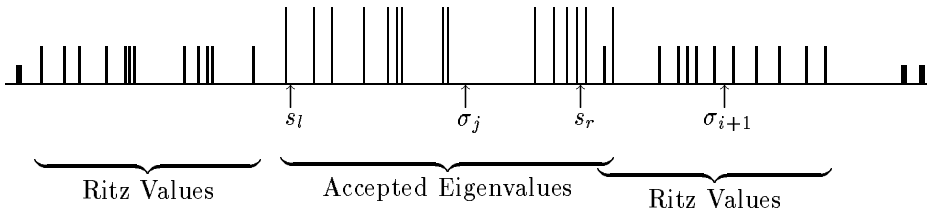


FIG. 6. *Sentinels.*

Consider in each direction the eigenvalue farthest from σ_j such that between it and σ_j no (unaccepted) Ritz values are found. There is such an eigenvalue to the right of a shift and similarly to the left, each being the last eigenvalue before a Ritz value is found. We call these two eigenvalues λ_r^* and λ_l^*. In normal circumstances we assume that there are no eigenvalues missing between σ_j and λ_r^* or λ_l^*.

We define the *right sentinel* s_r as the left endpoint of the interval of uncertainty for λ_r^*, based on the required accuracy tolerance. Thus the true value of λ_r^* lies to the right of the sentinel s_r. A *left sentinel* is defined similarly. Assume that $\sigma_{i+1} > \sigma_j$. The eigenvectors corresponding to λ_r^* and to any other eigenvalues found between s_r and σ_{i+1} are prevented from reappearing by use of external selective orthogonalization. We allow the recurrence to recompute eigenvalues which lie to the left of s_r, but these are discarded immediately. This technique allows us to trust any eigenpairs that are computed in the region in which we expect new eigenpairs to appear, without incurring the cost of extensive reorthogonalization. The reorthogonalization with respect to λ_r^*'s eigenvector removes any doubt that could exist about the exact location of this eigenvalue in the shifted and inverted spectrum for the new shift. At the same time, the eigenvector(s) most likely to reappear are suppressed.

We generalize the notion of sentinels slightly to handle clusters of eigenvalues. Should the sentinel s_r lie to the left of λ_{r-1}, we move the sentinel back to the endpoint

of the uncertainty interval for λ^*_{r-1}. We continue this process until the sentinel lies between the intervals of uncertainty for two eigenvalues, or until the shift itself is used as the sentinel.

3.4. The initial shift. The most difficult task is usually the first: getting started. The selection of the first shift must be made with no information about the spectrum other than the specification of the desired eigenvalues. We use any location information in the specification to make an initial choice for the first shift, σ_1,

$$\sigma_1 = \begin{cases} a \text{ if } |a| \le |b| \\ b \text{ if } |a| > |b| \end{cases} \Big\} \text{ if lowest modes or all modes wanted and } \min|a|,|b| < \infty,$$

$$\begin{cases} a \text{ if } |a| \ge |b| \\ b \text{ if } |a| < |b| \end{cases} \Big\} \text{ if highest modes wanted } (a \text{ and } b \text{ must both be finite}),$$

$$\xi \quad \text{if modes nearest } \xi \text{ wanted},$$

$$0 \quad \text{otherwise.}$$

This choice of σ_1 gives a reference point in the spectrum as to which eigenvalues are important to the user. In cases where ξ is not specified by the user, we define ξ to be σ_1 as defined above. We note that 0 is a natural choice when we have no location information—in that common case we want the eigenvalues of least magnitude, i.e., closest to 0.

Unfortunately, a choice of $\sigma_1 = 0$ is fraught with difficulties. A shift at zero is not allowed in the buckling transformation and yields a singular operator in vibration analysis when K is semidefinite. If a shift at zero were taken in the latter case, it is unlikely that the singularity of the operator would be detected. It is more likely that only the zero eigenvalues would be computed and no other useful information could be extracted from the run. (The near-singularity of the operator would cause the Lanczos recurrence to break down after computing the invariant subspace of the zero eigenvalues.) This would leave us little better off than we were when we began, with no information as to where the nonzero eigenvalues are located. A better initial shift would be a shift somewhere in the vicinity of the first few nonzero eigenvalues. Such a shift would allow computing both the zero, rigid body modes and a number of the nonzero modes as well.

The difficulty is in getting some idea of the scale of the first nonzero eigenvalues. We have adopted a heuristic strategy recommended by Louis Komzsik of The MacNeal–Schwendler Corporation. This heuristic computes the geometric mean of the centers of the Gershgorin circles while excluding the centers smaller than 10^{-4}. This heuristic usually computes a reasonable *problem scale* χ. Specifically,

$$\chi = \frac{1}{l * \sum \frac{|m_{ii}|}{|k_{ii}|}},$$

where the summation is taken over all terms with $k_{ii} \neq 0$, $\frac{|m_{ii}|}{|k_{ii}|} < 10^4$; l is the number of entries included in the sum. Table 5 gives an idea of the reliability of this heuristic.

We use χ to correct the initial selection of σ_1 whenever $|\sigma_1| < \chi$. In either the vibration problem or ordinary eigenvalue problem we adjust σ_1 as

$$\sigma_1 = \begin{cases} \chi & \text{if } a \le \chi \le b, \\ -\chi & \text{otherwise, if } a \le -\chi \le b, \\ \max(|a|,|b|) & \text{otherwise.} \end{cases}$$

We adjust the initial shift in a similar fashion for the buckling problem. However, we try $\sigma_1 = -\chi$ first and then $\sigma_1 = \chi$ second, because the most common buckling analysis in structural analysis is computation of the smallest negative eigenvalue.

TABLE 5
Comparison of problem scale χ and lowest eigenvalues.

Matrix	χ	Lowest eigenvalue	Closest to eigenvalue
BCSST_08	1.8×10^{-2}	6.9×10^{0}	1
BCSST_09	1.3×10^{7}	2.9×10^{7}	1
BCSST_10	3.1×10^{-3}	7.9×10^{-2}	1
BCSST_11	3.0×10^{2}	1.1×10^{1}	12
BCSST_12	1.5×10^{3}	3.5×10^{3}	1
BCSST_13	1.2×10^{2}	1.5×10^{3}	1
BCSST_19	6.6×10^{0}	2.1×10^{0}	3
BCSST_20	5.5×10^{2}	6.6×10^{0}	7
LUND	2.1×10^{1}	2.1×10^{2}	1
PLAT1919	2.1×10^{-6}	1.1×10^{-13}	315

3.5. Choosing a direction in which to expand a trust interval. The majority of vibration analyses result in a simple, monotonic expansion of the trust interval from lowest to higher values. In these cases we know that there are no additional eigenvalues of interest to the left of the trust interval; extending the interval to the right is the only reasonable action. Cases in which we need to choose a direction arise when a shift is taken in the interior of the spectrum by accident or by design. For example, ξ is a very reasonable initial shift when we want to find eigenvalues nearest ξ. In general, finding the eigenvalues of smallest magnitude for an ordinary eigenproblem or for buckling analysis is also such a case.

We use the reference value ξ, either as set in the problem description or from the initial shift (see §3.4), to determine the direction in which to move the shift. If multiple trust intervals exist, the trust interval including or closest to ξ is *primary*; §3.7.1 describes how multiple trust intervals can exist and the logic for determining a new shift in that case. In the most typical case we have only a single trust interval, which we attempt to extend.

We distinguish two subcases, when the trust interval includes an endpoint of the computational interval and when it does not. In the first case the trust interval can only be extended in one direction without moving outside the computational interval, so the choice of direction is trivial. When the trust interval includes neither endpoint, we further distinguish between cases where ξ is or is not in the trust interval. If the trust interval does not include ξ, we shift in the direction of ξ, because that is where the eigenvalues of most importance to the user lie.

The only remaining case is of a single trust interval that contains ξ, but neither endpoint of the computational interval. In this case we compute the interval $[z_l, z_r]$ that includes the entire trust interval and all computed eigenvalues, even those outside of the trust interval. We define $r = \min(\xi - z_l, z_r - \xi)$ to be the radius of a symmetric *umbrella* about ξ where we have some degree of confidence that we have computed all the eigenvalues in the umbrella. Note that this confidence may not be confirmed by inertia values. We try to enlarge this umbrella enough to include all of the eigenvalues that the user has requested or until one end of the umbrella is an endpoint of the computational interval. We move in whichever direction increases r. Ties are broken by shifting to the left.

3.6. Analysis in a finite interval. Frequently the user of the sparse eigensolver will specify a computational interval with finite endpoints. The number of eigenvalues in the interval is usually valuable information to the user and the eigenanalysis code, even when not all of these eigenvalues are actually computed. We obtain this information at the beginning of the analysis by computing the factorization for each endpoint. If these factorizations can be used in the eigenanalysis itself, the cost of gaining this information would be nominal. (Note that both factorizations will be required in any case when all eigenvalues in the interval are requested.) We save both factorizations off-line and use them whenever it appears to be appropriate.

As discussed in the previous section, we often choose the initial shift to be one of the endpoints. If so, one of the factorizations will be used immediately. We modify the shift strategy slightly in order to take advantage of the second factorization. When the natural choice of a shift would be near an otherwise unselected finite endpoint, and when a shift at the finite endpoint would not cause a large number of extra eigenvalues to be computed, we choose the endpoint as the shift. This may result in some additional work during the Lanczos iteration, but it will save the cost of a factorization. There are cases where we can extend a trust interval to a finite endpoint without making a Lanczos run at the endpoint. These occur when the analysis at another shift results in computation of all of the eigenvalues between the shift and the endpoint.

3.7. Special cases. Robustness is one of our goals. It is naive to expect that the heuristics described above will work for all problems. Here we describe a number of special cases that can and do arise in practice and our approaches for handling them smoothly.

3.7.1. Filling gaps. The shift selection is designed to extend the trust interval obtained from previous Lanczos runs. Strange, asymmetric distributions of eigenvalues or very high multiplicities may create situations in which the shift σ_{i+1} to extend the trust interval is taken too far from σ_i to allow computation of all the eigenvalues in (σ_i, σ_{i+1}) with a single run. The inertias from σ_i and σ_{i+1} will indicate that some eigenvalues between the two shifts have not been computed.

Our goal is to maintain a trust interval, so we find the missing eigenvalues *before* we attempt to extend our knowledge beyond σ_{i+1}. We attempt to fill the *gap* between the two *active* shifts σ_i and σ_{i+1}, before proceeding. We assume that the missing eigenvalues lie between the right sentinel s_i for the shift σ_i at the left and the left sentinel s_{i+1} for the shift σ_{i+1} at the right, that is, in $[s_i, s_{i+1}]$. If the sentinel values overlap we use $[\sigma_i, \sigma_{i+1}]$ instead. In either case we have an interval $[c, d]$ in which we want to chose a shift. We choose σ_{i+2} as

$$\sigma_{i+2} = \begin{cases} \sqrt{cd} & \text{if } 0 < 2c < d, \\ -\sqrt{cd} & \text{if } c < 2d < 0, \\ \frac{c+d}{2} & \text{otherwise.} \end{cases}$$

The gap between two trust intervals is not always filled on the first attempt. The shifting strategy will continue recursively, computing missing eigenvalues, until the primary trust interval has grown large enough to contain the requested eigenvalues or when all trust intervals have been merged into one.

3.7.2. Restart at the same shift. Economizing on the number of factorizations is also a goal. In two cases a single Lanczos run will not find all the desired eigenvalues near a given shift. These occur when eigenvalues with multiplicity greater

than the blocksize exist or when a shift has been taken very close to a eigenvalue. If we suspect either case we make an additional Lanczos run at the same shift. During this run we perform external selective reorthogonalization against all newly computed eigenvectors and any other eigenvectors in the interval between this shift and any neighboring shifts. We discard any use of sentinels because the assumption behind them has probably broken down.

3.7.3. Hole in the spectrum. A particularly difficult spectrum for our selection of shifts is one with a very large disparity in the magnitudes of the desired eigenvalues. In such cases our notion of a reasonable distance may be faulty and yet we may have no Ritz value information to help us choose a new shift.

Our code treats as special a situation in which no new information is obtained at consecutive shifts. That is, we compute no meaningful Ritz values and the inertias at the two shifts σ_i and σ_{i+1} are identical. We suspect that there is a "hole" in the spectrum, that the remaining eigenvalues are farther away than our notion of a reasonable distance. We expand the notion of a reasonable distance in an attempt to cross the hole. If the computational interval $[a, b]$ has a finite endpoint that has not been used previously as a shift (see §3.6), the shift strategy will select the new shift at that endpoint. Otherwise, assuming that we are expanding a trust interval to the right, we take the new shift $\sigma_{i+2} = \sigma_{i+1} + 10\delta$ (see §3.2 for a description of δ). If this Lanczos run still provides no new information, we take $\sigma_{i+3} = \sigma_{i+2} + 100\delta$. If we still obtain no new information, we make a final attempt to cross the gap with a shift $\sigma_{i+4} = \sigma_{i+3} + 1000\delta$. If this run still provides no new information, we terminate on the assumption that the remaining eigenvalues are infinite. We return the eigenvalues already computed, together with an appropriate warning.

3.7.4. Treatment of δ in no-Ritz value cases. The setting of the "reasonable distance" value, δ, must be made carefully in cases in which the Lanczos algorithm terminates abnormally. This value is not updated if no new information is available for the next shift.

3.7.5. Overly aggressive shifts. Unusual distributions of eigenvalues or unusual convergence patterns may cause situations in which a shift is selected much farther out than required for the desired eigenvalues. We determine that the shift is too far from the current trust interval if a run at this shift will have to compute more than 30 eigenvalues before computing eigenvalues of interest to us. (The number 30 is a heuristic estimate of the number of eigenvalues we can profitably find with a single run.) In such a case we record the current shift, to keep another shift from going out too far in that direction, and select a new shift. We choose the new shift by linear interpolation between the end of the trust interval, σ_t, and the shift we reject, σ_r. The new shift is:

$$\sigma = \sigma_t + \frac{q}{[\nu(K - \sigma_r M) - \nu(K - \sigma_t M)]}(\sigma_r - \sigma_t).$$

3.8. Modifications for buckling problems. The spectral transformation used in the buckling problem for the Lanczos iteration is ill posed for shifts at or near zero. The shift strategy for the buckling problem is similar to the vibration strategy except that shifts at zero are not allowed. A shift at zero is replaced by one half the minimum of the problem scale χ, the absolute value of the shift nearest to zero, and the absolute value of the computed eigenvalue nearest to zero.

4. Implementation of the block Lanczos algorithm. The underpinning of our eigenanalysis code is the block Lanczos algorithm, as specialized for the spectral transformations (§§2.3 and 2.5). The use of the block Lanczos algorithm in the context of the spectral transformation and within applications code necessitates careful attention to a series of details: the implications of M-orthogonality of blocks; block generalizations of single vector orthogonalization schemes; effect of the spectral transformation on orthogonality loss; and interactions between the Lanczos algorithm and the shifting strategy. The success of the algorithm hinges on all of these issues.

4.1. The M-orthogonal QR factorization. Each step of the block Lanczos recurrence generates an $n \times p$ matrix R, whose column vectors are to be orthogonalized with respect to an inner product defined by a positive definite matrix, which we will call M.

Given R, we must compute its orthogonal decomposition QB such that

- $R = QB$,
- $Q^T M Q = I$,
- Q is $n \times p$,
- B is $p \times p$ and upper triangular.

When M is not the identity, the number of good choices for computing an orthogonal factorization appear to be limited. In addition, we want to avoid repeated matrix-vector multiplications with M, because we expect M, though sparse, not to be stored in main memory; each multiplication by M may require accessing secondary storage. We have developed a generalization of the modified Gram–Schmidt process that requires only matrix-block products, never matrix-vector products. We save a set of p auxiliary vectors that represent the product MQ throughout the process. This matrix is initialized to MR when the matrix that will hold Q is initialized to R; thereafter, updates made to vectors in Q are shadowed by identical updates in MQ. As a result, M is used explicitly only in the initialization.

This way of enforcing M-orthogonality certainly suggests questions of numerical stability. Following [10], we repeat the orthogonalization process up to $2p$ times, another repetition being required whenever the norm of any of the q_j vectors is less than η times its norm at the beginning of the iteration. When another repetition is required we recompute the matrix MQ by an explicit multiplication by M. The choice of $\eta = \sqrt{2}/2$ from [10] guarantees that the final set of vectors is orthonormal.

In our algorithm for computing the M-orthogonal factorization (Fig. 7), the vectors w_j are the auxiliary vectors that represent the vectors Mq_j. The matrix $\hat{B}$ is the triangular matrix computed in one iteration of the algorithm; the M-orthogonal triangular factor B is the product of all of the individual triangular matrices $\hat{B}$.

It should be noted that this algorithm may encounter a rank deficient set of vectors q_j and identically zero vectors are possible. Further details can be found in our discussion on when to terminate a Lanczos run (§4.4).

We have assumed in the discussion above that M is positive definite. In the case of M positive semidefinite, the recurrence, when properly started, generates a sequence of blocks, all of whose columns lie in the range of $(K - \sigma M)^{-1} M$. This is the subspace from which the eigenvectors corresponding to finite eigenvalues must be drawn [13]. Clearly, the orthogonalization algorithm preserves this subspace. Further, this subspace has only the trivial intersection with the nullspace of M [13], [29]. Thus, the appearance of a nontrivial column with zero M-norm represents a breakdown equivalent to rank deficiency, since such a vector cannot lie in the range of $(K - \sigma M)^{-1} M$.

Initialization:

$Q = R$
$B = I$

Factorization:

Repeat
W = MQ

For $i = 1, 2, \ldots p$ do
$\hat{b}_{ii} = \sqrt{q_i^T w_i}$
$q_i = q_i / \hat{b}_{ii}$
$w_i = w_i / \hat{b}_{ii}$
For $j = i + 1, \ldots p$ do
$\hat{b}_{ij} = q_i^T w_j$
$q_j = q_j - \hat{b}_{ij} q_i$
$w_j = w_j - \hat{b}_{ij} w_i$
End
End

$B = \hat{B} B$

Until convergence or iteration limit exceeded

FIG. 7. *M-orthogonal modified Gram–Schmidt orthogonalization.*

4.2. Analysis of the block tridiagonal matrix T_j. The original eigenvalue problem is reduced by the block Lanczos algorithm to an eigenvalue problem of the form $T_j s = s\theta$, where T_j is a block tridiagonal matrix. In §2.2 we noted the standard result by which bounds on the accuracy of the computed eigenvalues can be computed without explicit computation of the eigenvectors. These bounds are used to determine whether to terminate the Lanczos recurrence and to evaluate which eigenpairs are accurate enough to be considered to have converged. The results in §2.2 generalize to provide a bound on the accuracy of the approximate eigenvalues of the spectrally transformed problem. However, our real interest is in the accuracy of our approximations to the original, untransformed problem. We need to determine which eigenpairs of the original problem have converged, and we need accuracy estimates for all of the Ritz values for use in the shift selection process. To get these estimates we need to unravel the effects of the spectral transformation. Throughout we must account for possibly multiple eigenvalues.

Recall that the following relation (14) holds for vibration analysis:

$$(K - \sigma M)^{-1} M y - y\theta = Q_{j+1} B_{j+1} E_j^T s.$$

Therefore, because Q_{j+1} is M-orthogonal,

$$\begin{aligned} \|M(K - \sigma M)^{-1} M - My\theta\|_{M^{-1}} &= \|M Q_{j+1} B_{j+1} E_j^T s\|_{M^{-1}} \\ &= \|B_{j+1} E_j^T s\|_2 \equiv \beta_j . \end{aligned}$$

For each eigenvector s the corresponding β_j is the Euclidean norm of the product of the upper triangular matrix B_{j+1} with the last p components of s. We apply a theorem on the error in eigenvalue approximations for the generalized eigenproblem from [32, p. 318] to obtain:

$$\left|\frac{1}{\lambda-\sigma}-\theta\right| \leq \frac{\|M(K-\sigma M)^{-1}My-My\theta\|_{M^{-1}}}{\|My\|_{M^{-1}}} = \beta_j. \tag{18}$$

Thus, as in the ordinary eigenproblem, β_j is a bound on how well the eigenvalue of T_j approximates an eigenvalue of the operator to which the Lanczos algorithm is applied. We extend this to find a bound on the error $|\lambda-\nu|$.

Ericsson and Ruhe [14] show that

$$|\lambda-\nu| \leq \frac{\beta_j}{\theta^2}. \tag{19}$$

This shows how the accuracy requirements are modified by the spectral transformation. When λ is close to the shift σ we need only a moderately small β_j to guarantee a good approximate eigenvalue ν because θ is large. Conversely, eigenvalues far from the shift are transformed to small values of θ, requiring smaller values of β_j than would otherwise be expected.

The bound (19) can be improved for well separated eigenvalues. Define the gap γ as:

$$\gamma \equiv \min_{\lambda_i \neq \lambda} \left|\frac{1}{\lambda_i-\sigma}-\frac{1}{\lambda-\sigma}\right|,$$

The gap bound theorem from [32, p. 222] then results in

$$|\lambda-\nu| \leq \frac{\beta_j^2}{\theta^2\gamma}. \tag{20}$$

Both bounds (19) and (20) are valid. In general, the first is smaller than the second for clustered eigenvalues and larger for well separated eigenvalues. In our implementation we use whichever bound is smaller:

$$|\lambda-\nu| \leq \min\left\{\frac{\beta_j}{\theta^2}, \frac{\beta_j^2}{\theta^2\gamma}\right\}. \tag{21}$$

The definition of γ should be modified to account for clusters of eigenvalues; the gap between sets of multiple eigenvalues is used. In practice we have only an approximation to γ, which we derive from the shifted and inverted eigenvalues of T_j.

Similar error bounds can be derived for buckling analysis. Let (ν, y) be a computed eigenpair of (K, K_δ). Then $\theta = \frac{\nu}{\nu-\sigma}$ and $|\frac{\lambda}{\lambda-\sigma}-\theta| \leq \beta_j$. From the fact that $\nu = \frac{\sigma\theta}{\theta-1}$, it follows that

$$\begin{aligned}
\lambda-\nu &= \lambda - \frac{\sigma\theta}{\theta-1} \\
&= \left(\frac{1}{\theta-1}\right)(\lambda(\theta-1)-\sigma\theta) \\
&= \left(\frac{1}{\theta-1}\right)(\lambda-\sigma)\left(\theta-\frac{\lambda}{\lambda-\sigma}\right) \\
&= \left(\frac{\sigma}{\theta-1}\right)\left(\frac{\lambda-\sigma}{\sigma}\right)\left(\theta-\frac{\lambda}{\lambda-\sigma}\right).
\end{aligned}$$

The Lanczos algorithm approximates θ by a projection onto a subspace. When the inversion of the operator is taken into account, the computed eigenvalues of the transformed problem are always closer to one than the true eigenvalues of the transformed problem. Therefore,

$$\left|\frac{\lambda-\sigma}{\sigma}\right| = \frac{1}{|\frac{\lambda}{\lambda-\sigma}-1|} \leq \frac{1}{|\theta-1|}.$$

The resulting simple error bound for buckling analyses is

$$|\lambda-\nu| \leq \frac{|\sigma|}{(\theta-1)^2}\beta_j.$$

The analogous refined gap error bound is

$$|\lambda-\nu| \leq \frac{|\sigma|\beta_j^2}{(\theta-1)^2}\gamma_b,$$

where γ_b is defined by

$$\gamma_b \equiv \min_{\lambda_i \neq \lambda}\left|\frac{\lambda}{\lambda_i-\sigma}-\frac{\lambda}{\lambda-\sigma}\right|,$$

As in the vibration case, the lesser of the two bounds

$$|\lambda-\nu| \leq \min\left\{\frac{|\sigma|}{(\theta-1)^2}\beta_j, \frac{|\sigma|\beta_j^2}{(\theta-1)^2}\gamma_b\right\} \tag{22}$$

is chosen, with the definition of γ_b modified in the presence of multiple eigenvalues.

The spectral transformation preserves the eigenvectors, so there is no need to account for the transformation *vis à vis* the approximate eigenvectors. However, Ericsson and Ruhe [14] introduced a correction term that results in improved eigenvector approximations for the untransformed problem. This was later discovered to have the additional benefit [29] of ensuring that the computed eigenvectors lie in the proper subspace in cases where the metric matrix is semidefinite.

Let $\nu = \sigma + \frac{1}{\theta}$ be the computed eigenvalue. The correction step is formally one step of inverse iteration with the computed eigenvector y; $\tilde{z}$ is computed to satisfy $(K-\sigma M)\tilde{z} = My$. By (14)

$$\tilde{z} = (K-\sigma M)^{-1}My = y\theta + Q_{j+1}B_{j+1}E_j^T s.$$

The vector

$$z = \frac{1}{\theta}\tilde{z} = y + \frac{1}{\theta}Q_{j+1}B_{j+1}E_j^T s$$

can be obtained cheaply by adding a linear combination of the vectors in the next block of Lanczos vectors to y. This gives a better approximation to the eigenvector of the vibration problem and ensures that the approximate eigenvectors are uncontaminated by the effects of a semidefinite M. The corresponding correction for a semidefinite K in buckling analysis is given by

$$z = y + \frac{1}{\theta-1}Q_{j+1}B_{j+1}E_j^T s.$$

During the course of the Lanczos algorithm we need an estimate of the residual bounds. The bounds in (21) or (22) require most of the eigenvalues and the corresponding entries in the bottom block row of the matrix of eigenvectors. Parlett and Nour-Omid [33] have a very efficient algorithm for the single vector Lanczos algorithm. Block generalizations have yet to be found, so we use a more straightforward approach. The eigenvalue problem for T_j is solved by reducing the band matrix T_j to tridiagonal form and then by applying the tridiagonal QL algorithm. We use subroutines from EISPACK [16], [41], with slight modifications, to obtain only the bottom p entries of the eigenvectors of T_j. These modifications reduce considerably both computation and storage requirements for each Lanczos step. Only $p^2 j$ words are needed as opposed to $(pj)^2$ for the full eigenvector matrix. We use the corresponding unmodified routines to obtain the full eigenvectors at the conclusion of a Lanczos run, at which time temporary space used during the recurrence is available to store the entirety of the eigenvector matrix for T.

4.3. Global loss of orthogonality and reorthogonalization. Up to this point our discussion of the block Lanczos algorithm has assumed exact arithmetic, but the various error bounds hold in finite precision as well. It is well known that there is a global loss of orthogonality among the computed Lanczos vectors in inexact arithmetic. A reasonable correction is to perform *limited* reorthogonalization to keep $\mathcal{Q}_j$ sufficiently close to orthogonal. Our approach is twofold—we identify mechanisms whereby orthogonality is lost and then apply a model of the loss of orthogonality to determine when to correct the situation. In the context of the block shifted Lanczos recurrence, orthogonality is lost in three different ways. First, there is a loss of orthogonality between adjacent blocks in $\mathcal{Q}_j$, the blocks the recurrence should make orthogonal. This is corrected by use of *local reorthogonalization*. Second, the recurrence suffers a global loss of orthogonality with respect to the blocks of $\mathcal{Q}_j$ not explicitly involved in the reorthogonalization. We correct for this with a block version of *partial reorthogonalization*. Lastly, it is important that a Lanczos run at some shift not recompute eigenvectors computed as a result of a previous Lanczos run. We present a new reorthogonalization scheme, *external selective reorthogonalization*, to ensure that this does not occur. Throughout the process our goal is to apply a minimal amount of extra work, particularly as it requires accessing the entirety of $\mathcal{Q}_j$, to maintain at least $\mathcal{O}(\sqrt{\epsilon})$-orthogonality in $\mathcal{Q}_j$.

The fundamental approach is to model the Lanczos recurrence in finite precision. The following recurrence is our model of what really happens:

$$Q_{j+1}B_{j+1} = (K - \sigma M)^{-1} M Q_j - Q_j A_j - Q_{j-1} B_j^T + F_j, \tag{23}$$

where F_j represents the roundoff error introduced at step j. Then,

$$\begin{aligned} Q_k^T M Q_{j+1} B_{j+1} = {} & Q_k^T M (K - \sigma M)^{-1} M Q_j - Q_k^T M Q_j A_j \\ & - Q_k^T M Q_{j-1} B_j^T + Q_k^T M F_j. \end{aligned}$$

For convenience we define $W_{j,k} \equiv Q_k^T M Q_j$, with which the previous equation becomes

$$W_{j+1,k} B_{j+1} = Q_k^T M (K - \sigma M)^{-1} M Q_j - W_{j,k} A_j - W_{j-1,k} B_j^T + Q_k^T M F_j. \tag{24}$$

This equation is nearly sufficient for our computational purposes. We can easily find norms for the blocks A_j and B_j during the recurrence, and we will compute bounds for all the other terms except for the first term on the right side of (24).

We eliminate $Q_k^T M(K-\sigma M)^{-1} M Q_j$ from (24) by obtaining an expression for its transpose by premultiplying the occurrence of (23) with $j=k$ by $Q_j^T M$:

$$Q_j^T M(K-\sigma M)^{-1} M Q_k = W_{j,k+1}B_{k+1} + W_{k,j}A_j + W_{k-1,j}B_k^T + Q_j^T M F_k.$$

The obvious substitution then results in

$$\begin{aligned} W_{j+1,k}B_{j+1} = {} & B_{k+1}^T W_{j,k+1} + A_k W_{j,k} + B_k W_{j,k-1} \\ & -W_{j,k}A_j - W_{j-1,k}B_j^T + G_{j,k}. \end{aligned} \tag{25}$$

Here $G_{j,k} \equiv Q_k^T M F_j - F_k^T M Q_j$ represents the local roundoff error. Formula (25) explains the global loss of orthogonality. We will use this model to estimate and bound the loss of orthogonality among the Lanczos vectors and thereby determine how to correct the loss of orthogonality.

4.3.1. Monitoring the loss of orthogonality. The development of our modeling procedure has two parts, both based on the bounds available by taking norms of (25):

$$\begin{aligned} \|W_{j+1,k}\|_2 \le {} & \|B_{j+1}^{-1}\|_2 (\|B_{k+1}\|_2 \|W_{j,k+1}\|_2 \\ & + \|B_k\|_2 \|W_{j,k-1}\|_2 + \|B_j\|_2 \|W_{j-1,k}\|_2 \\ & + (\|A_j\|_2 + \|A_k\|_2)\|W_{j,k}\|_2 + \|G_{j,k}\|_2). \end{aligned}$$

We use this equation to compute a bound $\omega_{j,k}$ on $\|W_{j,k}\|_2$ at each step.

The first part of our development addresses the bounds $\omega_{j+1,k}$ for $k \le j-1$, that is, for blocks that are not explicitly involved in the orthogonalization of the Lanczos vectors within the recurrence itself. For these blocks the loss of orthogonalization depends on the loss already incurred at previous steps. Bounds on that loss of orthogonality will be available to us from previous steps of the simulation given in Fig. 8.

Initialize:

$\epsilon_s \equiv \epsilon p\sqrt{n}$, where $\epsilon \equiv$ roundoff unit, p is the blocksize
and $n =$ number of degrees of freedom
$\omega_{2,1} = \epsilon_s$

Loop:

For $j = 2, 3, 4, \ldots$ do
$\omega_{j+1,j} = \epsilon_s$
$\omega_{j+1,j-1} = \tilde{\beta}_{j+1}(2\beta_j\epsilon_s + (\alpha_j + \alpha_{j-1})\epsilon_s + \beta_{j-1}\omega_{j,j-2})$
For $k = 1, \ldots j-2$ do
$\omega_{j+1,k} = \tilde{\beta}_{j+1}(\beta_{k+1}\omega_{j,k+1} + \beta_k\omega_{j,k-1} + \beta_j\omega_{j-1,k} + (\alpha_j + \alpha_k)\omega_{j,k})$
End
End

FIG. 8. *Simulation of loss of orthogonality (ω-recurrence).*

The following quantities from the Lanczos recurrence are required for the simulation:

$$\alpha_k \equiv \|A_k\|_2,$$

$\beta_k \equiv \|B_k\|_2$,
$\tilde{\beta}_k \equiv 1/\sigma_p(B_k)$, where $\sigma_p(B_k)$ is the smallest singular value of B_k.

In addition, we follow [32], [37], [39] in making a standard assumption on a bound for the error term: $\|G_{j,k}\|_2 \leq \epsilon_s = \epsilon p\sqrt{n}$. We have left unstated the origin of the two initializing terms, $\omega_{j+1,j}$ and $\omega_{j+1,j-1}$. In examining them we will uncover a particular artifact of the *block* Lanczos algorithm. By (25),

$$\begin{aligned} W_{j+1,j-1}B_{j+1} &= B_j^T W_{j,j} + A_{j-1}W_{j,j-1} + B_{j-1}W_{j,j-2} \\ &\quad -W_{j,j-1}A_j - W_{j-1,j-1}B_j^T + G_{j,j-1} \\ &= (B_j^T W_{j,j} - W_{j-1,j-1}B_j^T) \\ &\quad +(A_{j-1}W_{j,j-1} - W_{j,j-1}A_j) \\ &\quad +B_{j-1}W_{j,j-2} + G_{j,j-1}. \end{aligned}$$

By reason of the care with which we compute the QR factorization, we assume that $Q_j^T M Q_j = I+E$, where I is the identity matrix and $\|E\|_2 \leq \epsilon_s$. For reasons discussed below, we can assume that $\|W_{j,j-1}\|_2 \leq \epsilon_s$. From this it follows that

$$\|W_{j+1,j-1}\|_2 \leq \tilde{\beta}_{j+1}(2\beta_j\epsilon_s + (\alpha_j + \alpha_{j-1})\epsilon_s + \beta_{j-1}\omega_{j,j-2}). \tag{26}$$

Notice from (26) that $\omega_{j+1,j-1} > \tilde{\beta}_{j+1}\beta_j\epsilon_s$. At the next step this term will appear as $\tilde{\beta}_{j+2}\beta_j\omega_{j+1,j-1}$; in the following step it will be one of the contributions to $\tilde{\beta}_{j+3}\beta_{j+1}\omega_{j+2,j}$. Both $\tilde{\beta}_{j+1}$ and β_{j+1} appear in this last product. The growth of the bound occurs as fast as $\kappa(B_j) = \tilde{\beta}_{j+1}\beta_{j+1}$, the condition number of B_j. The analysis of the ordinary Lanczos algorithm has unity corresponding to the term $\kappa(B_j)$, because the condition number of a nonzero 1×1 matrix is always one. The loss of orthogonality occurs more rapidly in the block Lanczos algorithm, particularly when $\kappa(B_j)$ is significantly larger than one, but also in general.

A different, but related, analysis can be used to show that the term $\kappa(B_j)$ appears in the bound for $\omega_{j+1,j}$. This was first observed in [23], where this growth was also actually observed in the Lanczos recurrence. An inexpensive correction is needed to make the recurrence useful: at each step a *local reorthogonalization* between Q_{j+1} and Q_j is performed. Because the Lanczos recurrence is itself just a special form of Gram–Schmidt orthogonalization, local reorthogonalization can be seen as a simple generalization of the reorthogonalization required in computing the M-orthogonal factorization of a single block. Local reorthogonalization ensures that ϵ_s-orthogonality holds between successive blocks of Lanczos vectors. Note that a local orthogonalization step is also performed on completion of a partial reorthogonalization. If storage is not an issue, a local reorthogonalization between Q_{j+1} and Q_{j-1} should also be performed, in which the obvious modification should be made to the algorithm for computing the ω-recurrence.

4.3.2. Partial reorthogonalization. The global loss of orthogonality modeled by the ω-recurrence can be corrected by two different schemes. These are the selective orthogonalization scheme of Parlett and Scott [35] and the partial reorthogonalization scheme of Simon [40]. Selective orthogonalization takes advantage of the fact that orthogonality is lost exactly in the direction of eigenvectors that have become well represented in $\mathcal{Q}_j$. Selective orthogonalization is implemented in two steps. In the first, the Lanczos recurrence is "interrupted" when an eigenvector converges. The

eigenvector is computed, which requires access to all previous blocks in $\mathcal{Q}_j$. The second step occurs whenever the model indicates orthogonality is lost again in the direction of the eigenvector. The second step requires that the latest two Lanczos blocks be reorthogonalized against the computed eigenvector, but does not require access to preceding blocks of $\mathcal{Q}_j$.

Partial reorthogonalization interrupts the recurrence to reorthogonalize Q_j and Q_{j+1} against all preceding blocks whenever the simulation indicates too great a loss of orthogonality. Each reorthogonalization step requires access to all of $\mathcal{Q}_j$. For this reason partial reorthogonalization has previously been recommended for situations in which the eigenvectors were not of any interest (as in solving sparse linear equations [40]). The extra cost in an application of partial reorthogonalization does have an extra payoff; orthogonality is restored against all converged and nearly converged eigenvectors simultaneously.

TABLE 6
Comparison of partial and selective reorthogonalization.

Matrix	Eigen-values	Block steps	Partial reorthog. steps	Selective orthog. steps
BCSST_26[a]	211	181	51	98
PLAT1919[b]	636	579	143	291

[a] blocksize 3, lowest 200 modes.
[b] blocksize 3, all modes in [.000025, .24].

Shifting and the block recurrence each accelerate the convergence of eigenpairs; together they cause eigenpairs to converge very rapidly. Frequently one or more eigenpairs converge at *each* block step, once the recurrence is established. In this circumstance selective orthogonalization has possibly greater requirements for accessing $\mathcal{Q}_j$ than does partial reorthogonalization. Selective orthogonalization will require an eigenvector computation at almost each step; partial reorthogonalization will occur only every three to four steps in typical problems. It would be possible to combine the two schemes—to carry out partial reorthogonalization during the computation of an eigenvector for selective orthogonalization, but it is not clear that the combination would be more effective than partial reorthogonalization alone. (See [34] for a discussion of these issues for the ordinary Lanczos recurrence.) Table 6 summarizes the reorthogonalization requirements of two extensive eigencomputations. The number of selective orthogonalization steps given in this table is the number of block steps at which one or more eigenvalues converge; the number of partial reorthogonalization steps is the number of block steps at which partial reorthogonalization was performed.

Our implementation of the Lanczos recurrence uses the block generalization of partial reorthogonalization, based on the block ω-recurrence presented above. The single vector version of this simulation has been shown previously [40] to provide a good order of magnitude estimate of growth of the loss of orthogonality, as well as a bound. We use the block version to estimate the loss of orthogonality to determine when reorthogonalization is necessary. Previous work [32], [35], [39] indicates that reorthogonalization is needed whenever

$$\max_k \omega_{j+1,k} \geq \sqrt{\epsilon}.$$

The reorthogonalization should be carried out with both of the last two block of

vectors Q_j and Q_{j+1}, in order that the next block generated by the recurrence, Q_{j+2}, be strictly orthogonal to all of its predecessors. This leads to the following *partial reorthogonalization* [40] algorithm (Fig. 9) for maintaining orthogonality:

```
At each Lanczos step, after computing Q_{j+1} and B_{j+1}, do:

    Update the ω-recurrence as above

    ω_max ≡ max_k ω_{j+1,k}
    If ω_max ≥ √ε then
        For k = 1, ..., j − 1 do
            Orthogonalize Q_j against Q_k
            Orthogonalize Q_{j+1} against Q_k
        End
        Orthogonalize Q_{j+1} against Q_j
        Reinitialize ω-recurrence:
            ω_{j+1,k} = ω_{j,k} = ε_s, k = 1, ..., j
    End if
```

FIG. 9. *Partial reorthogonalization.*

Note that the orthogonalization of Q_j and Q_{j+1} involves M-inner products. This requires the storage of both the Lanczos vectors and their product with M in secondary storage, or, alternatively, reapplying M to the Lanczos vectors. The appropriate form depends on cost.

4.3.3. External selective orthogonalization. A different type of loss of orthogonality occurs in the context of the shifted and inverted Lanczos algorithm. It is possible that, after computing some eigenvalues with shift σ_1, the same eigenvalues and vectors are computed again with shift σ_2. External selective orthogonalization is an efficient way of keeping the current sequence of Lanczos vectors orthogonal to previously computed eigenvectors, and thereby avoiding the recomputation of eigenvalues that are already known. External selective orthogonalization is motivated by the classical selective orthogonalization algorithm [35], but the development here is entirely new.

In theory it would be sufficient to orthogonalize the starting block against known eigenvectors, because all subsequent Lanczos vectors would be orthogonal as well. Of course, this does not hold in practice. A global loss of orthogonality occurs, similar to the one among the Lanczos vectors themselves; in addition, the computed eigenvector is not exact. The contribution of both sources of error to the recomputation of eigenvalues and vectors is analyzed below.

Let (ν, y) be an approximate eigenpair of (K, M). For clarity, denote the current shift as σ_{new}. The relationship between the eigenvector y and the Lanczos vectors obtained with the shift σ_{new} is found by premultiplying the finite precision recurrence (23) by $y^T M$ to obtain

$$
\begin{aligned}
y^T M Q_{j+1} B_{j+1} = {} & y^T M (K - \sigma_{\text{new}} M)^{-1} M Q_j - y^T M Q_j A_j \\
& - y^T M Q_{j-1} B_j^T + y^T M F_j.
\end{aligned} \tag{27}
$$

We assume that B_{j+1} is nonsingular. Then we can obtain a bound on the loss of

orthogonality between y and Q_j by taking norms of both sides of (27):

$$\|y^T M Q_{j+1}\|_2 \le \|B_{j+1}^{-1}\|_2(\|y^T M(K-\sigma_{\rm new}M)^{-1}MQ_j - y^T M Q_j A_j\|_2 + \|y^T M Q_{j-1}\|_2\|B_j^T\|_2 + \|y^T M F_j\|_2).$$

As with partial reorthogonalization, we can define a recurrence relation for a quantity τ_j to bound the loss of orthogonality between y and the Lanczos vectors. Assuming that $\tau_i \ge \|y^T M Q_i\|_2$ for $i = 1, \ldots, j$, we obtain

$$\|B_{j+1}^{-1}\|_2(\|y^T M(K-\sigma_{\rm new}M)^{-1}MQ_j - y^T M Q_j A_j\|_2 + \tau_{j-1}\|B_j^T\|_2 + \|y^T M F_j\|_2)$$

as a bound for the right-hand side of the $j+1$st step. Of the three terms on the right-hand side of this equation, the second is easily computed and we have a standard assumption for a bound on the third: $\|F_j\|_2 \le \epsilon p\sqrt{n}$. We need then only to bound the first term $\|y^T M(K-\sigma_{\rm new}M)^{-1}MQ_j - y^T M Q_j A_j\|_2$. The spectral transformations preserve eigenvectors, so y is also an approximate eigenvector of the spectrally transformed problem. Define the transformed residual vector $z_{\rm new}$ by

$$(K-\sigma_{\rm new}M)^{-1}My - \frac{1}{\nu-\sigma_{\rm new}}y = z_{\rm new}.$$

Then

$$y^T M(K-\sigma_{\rm new}M)^{-1}MQ_j = \frac{1}{\nu-\sigma_{\rm new}}y^T MQ_j + z_{\rm new}^T MQ_j,$$

from which it follows that

$$\|y^T M(K-\sigma_{\rm new}M)^{-1}MQ_j - y^T MQ_jA_j\|_2 \le \left\|\left(\frac{1}{\nu-\sigma_{\rm new}}I - A_j\right)\right\|_2 \tau_j + \|z_{\rm new}^T MQ_j\|_2.$$

But

$$\begin{aligned}\|z_{\rm new}^T MQ_j\|_2 &= \|(z_{\rm new}^T M^{1/2})(M^{1/2}Q_j)\|_2 \\ &\le \|z_{\rm new}^T M^{1/2}\|_2\|M^{1/2}Q_j\|_2 = \|z_{\rm new}^T\|_M\|Q_j\|_M = \|z_{\rm new}^T\|_M.\end{aligned}$$

Thus, the following simple recurrence for τ gives a bound for the loss of orthogonality observed in (27):

(28)
$$\tau_{j+1} = \|B_{j+1}^{-1}\|_2\left(\tau_j\left\|\left(\frac{1}{\nu-\sigma_{\rm new}}I - A_j\right)\right\|_2 + \tau_{j-1}\|B_j^T\|_2 + \|z_{\rm new}\|_M + \epsilon p\sqrt{n}\right).$$

The same analysis applies to the buckling spectral transformation, where the eigenvector orthogonality error (27) becomes:

$$\begin{aligned}y^T KQ_{j+1}B_{j+1} = {}& y^T K(K-\sigma_{\rm new}K_\delta)^{-1}KQ_j - y^T KQ_jA_j \\ & - y^T KQ_{j-1}B_j^T + y^T KF_j.\end{aligned}$$

The transformed residual vector $z_{\rm new}$ is

$$(K-\sigma_{\rm new}K_\delta)^{-1}Ky - \frac{\nu}{\nu-\sigma_{\rm new}}y = z_{\rm new}.$$

By the same analysis as above, the recurrence for τ in the buckling context is

(29)
$$\tau_{j+1} = \|B_{j+1}^{-1}\|_2 \left(\tau_j \left\| \left(\frac{\nu}{\nu - \sigma_{\text{new}}} I - A_j \right) \right\|_2 + \tau_{j-1} \|B_j^T\|_2 + \|z_{\text{new}}\|_K + \epsilon p \sqrt{n} \right).$$

The recurrences in (28) and (29) provide a mechanism for estimating the loss of orthogonality to externally computed eigenvectors, regardless of the source. Each requires computing the transformed residual vector, z_{new}, and its norm, but the recurrence applies to situations where eigenvectors are known adventitiously. For example, in the vibration analysis of structures where K is singular, the so-called rigid body modes, the zero eigenvalues and vectors, often can be computed at much less cost than a factorization. Typically, the cost of computing the residual norms for all of the vectors involved in external selective orthogonalization is less than the cost of one additional step of the Lanczos recurrence.

In the context of a Lanczos code within a larger shifting strategy, it would be attractive to use the information from the Lanczos recurrence to bound the errors in the computed eigenvectors and thereby avoid having to compute $\|z_{\text{new}}\|_M$. In [20] we provide an analysis for the case where the approximate eigenpair (ν, y) was computed by the Lanczos code at a previous shift σ_{old}. However, we use the more general form exclusively in our code.

As with partial reorthogonalization, we define a recurrence relation for a quantity τ_j that estimates the loss of orthogonality of the Lanczos vectors with respect to y. In the recurrence, τ_j is defined be:

(30)
$$\tau_{j+1} = \tilde{\beta}_{j+1}(\alpha_{\nu\sigma j}\tau_j + \beta_j \tau_{j-1} + \|z_{\text{new}}\|_M),$$

which we initialize with $\tau_0 \equiv 0$ and $\tau_1 = \epsilon p \sqrt{n}$. The terms β_j and $\tilde{\beta}_{j+1}$ are defined as in the ω-recurrence. The term $\alpha_{\nu\sigma j} \equiv \|(\nu - \sigma)^{-1} I - A_j\|_2$.

An external selective orthogonalization is performed whenever $\tau_{j+1} \geq \sqrt{\epsilon}$. A relatively large residual for the computed eigenvector will cause frequent reorthogonalization, but, as noted below, usually only a very small number of vectors are actually involved. External selective orthogonalization is implemented as in Fig. 10.

```
Before the Lanczos iteration:
    Determine the set of SO-vectors (eigenvectors for selective orthogonalization)
    Orthogonalize Q_1 against the SO-vectors.
    Orthogonalize Q_2 against the SO-vectors.

At each Lanczos step j = 3, 4, ... do:
    Update the τ-recurrence according to (30) for each SO-vector;
    If (τ_j has been greater than √ε or τ_{j+1} ≥ √ε ) then
        Orthogonalize Q_{j+1} against y
        Set τ_{j+1} = ε_s
    End if
```

FIG. 10. *External selective orthogonalization.*

It is unnecessary to perform external selective orthogonalization against *all* previously computed eigenvectors. From (28) and (29) it is evident that one of the main

driving forces in the loss of orthogonality is $(\nu - \sigma)^{-1}$. It would appear that loss of orthogonality should mostly occur in the direction of eigenvectors corresponding to eigenvalues close to the new shift. Furthermore, as discussed in §3.3, only a few eigenvectors, again usually those close to the new shift, need be considered in order to avoid confusing new eigenvectors with the old. In our implementation, we use sentinels to reduce the cost of maintaining orthogonality. The set of eigenvectors used for external selective orthogonalization is usually the eigenvectors corresponding to any known eigenvalues closer to the shift than the sentinels. Eigenvalues beyond the sentinels are discarded in the analysis of the block tridiagonal system.

The effect of using sentinels on the work required for external selective orthogonalization is more dramatic than is suggested by the analysis above. Although proximity to the shift is the driving force in the growth of τ, neither recurrence (28) nor (29) begins at ϵ. The term $\|z_{\text{new}}\|_M$ is usually near $\sqrt{\epsilon}$. The eigenvalues themselves are only good to the convergence tolerance (usually $\epsilon^{2/3}$ in our code). Furthermore, the spectral transformations preserve eigenvectors, but do not preserve the property of being the best minimizers for approximate eigenvalues (see [14] for a discussion of the need to modify the approximate eigenvectors). As a result, external selective orthogonalization happens more often than we might expect, often at every step for the eigenpairs nearest the sentinels, which frequently are simultaneously least accurate and nearest the new shift.

Experimental results are shown for two examples in Table 7. The results shown as "with sentinels" refers to the selection described in §3.3; the results shown as "without sentinels" uses as SO-vectors all eigenvectors in the intervals between the current shift and any neighboring trust intervals. The figure given as "cpu cost" includes both cpu time and i/o processor time. The difference between the costs for the two variations gives only a rough idea of the added cost for complete selective orthogonalization because the difference in cost affects the termination decision for each run and thereby changes the choice of shifts.

TABLE 7
External selective orthogonalization.

Matrix	With sentinels			Without sentinels		
	Average number of S.O. Vectors	Total number of S.O. Steps	cpu cost	Average number of S.O. Vectors	Total number of S.O. Steps	cpu cost
BCSST_26[a]	2.1	313	174.2	15.7	2265	222.7
PLAT1919[b]	6.4	2776	668.2	28.1	8692	801.5

[a] blocksize 3, lowest 200 modes.
[b] blocksize 3, all modes in [.000025, .24].

The orthogonalizations involve again both y and My. In order to avoid the repeated computation of My, all selective orthogonalization vectors are premultiplied by M and the result is stored on the same random access file as the eigenvectors y. This computation is performed before the actual Lanczos run begins.

4.3.4. Summary of reorthogonalization schemes. We now present in summary form the reorthogonalized block Lanczos algorithm we use in our production code. Our scheme consists of applying, in turn, external selective, partial, and local reorthogonalization to the result of a single block Lanczos step. The first two schemes are applied only when the respective model signals a need; each should be applied

before orthogonality is lost badly enough that repeated orthogonalizations are needed. The local reorthogonalization is applied at each step. It may be applied repeatedly, but this normally occurs only when the recurrence has broken down, which will cause termination. The integration of these is indicated in Fig. 11.

Initialization:

Set $Q_0 = 0$
Set $B_1 = 0$
Choose R_1 and orthonormalize the columns of R_1 to obtain Q_1
with $Q_1^T(MQ_1) = I_p$.

Lanczos Loop:

For $j = 1, 2, 3 \ldots$ do

Set $U_j = (K - \sigma M)^{-1}(MQ_j) - Q_{j-1}B_j^T$
Set $A_j = U_j^T(MQ_j)$
Set $R_{j+1} = U_j - Q_jA_j$
Compute Q_{j+1} and (MQ_{j+1}) such that
a) $Q_{j+1}B_{j+1} = R_{j+1}$
b) $Q_{j+1}^T(MQ_{j+1}) = I_p$

Evaluate γ-recurrence for each SO vector and perform selective orthogonalization if necessary
Evaluate ω-recurrence and perform partial reorthogonalization if necessary

Repeat up to $2p$ times:
Reorthogonalize Q_{j+1} to Q_j
Recompute M-orthogonal factor of Q_{j+1}
Until orthogonal factorization occurs in one step

End loop

FIG. 11. *Spectral transformation block Lanczos algorithm preserving semi-orthogonality.*

4.4. Cost analysis and termination of a Lanczos run. The block Lanczos algorithm exists as part of a larger code, in which each Lanczos run solves only a subproblem. In this environment there are three ways in which a given Lanczos run can terminate:

1. All eigenvalues required for this subproblem have converged.
2. The B_{j+1}-block is ill conditioned or singular. In this case a continuation of the Lanczos run is either numerically difficult or impossible. Singular or ill conditioned B_{j+1}-blocks can be encountered for the following reasons:
   - The shift is very close to an eigenvalue.
   - The effective space of Lanczos vectors is exhausted—we cannot compute more orthogonal vectors than the problem has finite eigenvalues.

- Dependencies within the starting block cause a singular B_{j+1} at some later stage.

3. Eigenvalues farther from the shift appear to be converging slowly. The estimated cost for computing them in the current Lanczos run is great enough that a new shift should be chosen.

The first of these is easy to detect in most cases. There is a minor complication when we want the eigenvalues closest to some specified value because we do not know in advance how many eigenvalues are required on each side of ξ. At a given shift our conservative code looks for as many eigenvalues as are required to complete the total, even if these may represent more than what is needed on one side of ξ. As a result, we may not terminate as early as we might with hindsight.

Breakdown in the recurrence is perhaps more likely than might otherwise be expected. The first of the causes we try to avoid during the shift selection process; the second occurs primarily during user evaluation of the code, when it is not uncommon to be faced with problems like finding all of the eigenvalues of 7×7 matrices using a blocksize of 5. The third we have never seen. Breakdown is detected by one of two mechanisms—the norm of the residual block is very small compared to the norm of the diagonal block or the off-diagonal block is ill conditioned and presumed rank-deficient. We use a relative norm of $1/\sqrt{\epsilon}$ for the first case. For the second we compute, at each step, the extreme singular values of the off-diagonal block B_i; we terminate if the condition number of $B_i \geq \frac{1}{\epsilon}$. We really want only the condition number of B_i, but the cost of a singular value decomposition of a $p \times p$ matrix is trivial compared to the cost of an $n \times n$ sparse block solve.

The most common reason for termination is that computing more eigenvalues in the current run is inefficient. Normally, eigenvalues far from the shift converge slowly and require a large number of steps. Usually the fastest convergence occurs early, with the number of eigenvalues converging per step tapering off as the length of the run increases. Initially the cost *per eigenvalue* decreases rapidly, as the cost of the factorization is amortized over several eigenvalues. Later, as the convergence rate slows and the other costs increase, the average cost also increases. Our goal is to stop at the minimum average cost.

The cost of a Lanczos run depends on a number of parameters, each a function of the number of steps taken. The factorization typically represents the largest single cost, but it occurs once. There is a large constant cost per step, comprising the matrix-block solve and multiplication and other operations in the recurrence. The cost of the eigenanalysis of T_j increases quadratically in the number of block steps. Inasmuch as the eigenvalue nearest the shift is usually the first to converge, and dominates the reappearance of banished subspaces, the frequency with which partial reorthogonalization is needed is generally independent of the number of eigenvalues that have converged and so represents another quadratic term. Terminating the run by computing the converged eigenvectors from the Lanczos vectors is a cubic term.

We determine when to terminate a given Lanczos run by modeling the cost of continuing the recurrence beyond the current step. The residual bounds estimating the accuracy of yet unconverged eigenvalues are monitored step by step; the observed changes are used to estimate future convergence. We attempt to locate a point in an individual run where the average cost per eigenvalue is minimized. This is itself a heuristic attempt to minimize the average cost for all eigenvalues. The effectiveness of the heuristic is demonstrated for a particular example in Table 8.

TABLE 8
Comparison of variations on termination model.

Matrix	Standard Strategy			Termination Early			Termination Late		
	Shifts	Block steps	cpu cost	Shifts	Block steps	cpu cost	Shifts	Block steps	cpu cost
BCSST_26[a]	9	181	177.2	10	209	200.5	7	182	206.8
PLAT1919[b]	21	595	706.5	33	735	736.1	19	696	956.3

[a] blocksize 3, lowest 200 modes.
[b] blocksize 3, all modes in [.000025, .24].

We assume that a measure of the real user cost, including i/o, is available. We use this in a cubic model of cost, from which we obtain a least squares fit to the real cost over the preceding ten steps. From this model we predict the real cost over the next few steps. The cost of the final step, computing the eigenvectors, is estimated from measurements of components of the computation as they appear elsewhere in the recurrence. To start the process, we require that a certain minimum number of steps be taken. The number required is a function of blocksize and the type of problem, as indicated in Table 9. (The values in Table 9 are heuristic values derived from extensive empirical testing.)

The rate of convergence for the as yet unconverged eigenvalues is estimated by taking a weighted geometric average of the change in accuracy of the first unconverged Ritz value over the previous five steps. From this, we extrapolate to estimate the accuracy of the unconverged eigenvalues over a small number of additional steps. The number of extrapolated steps is also a function of blocksize and the type of problem; the actual values used are given in Table 9. We continue the Lanczos run if the estimated average cost per eigenvalue decreases for any of the steps over which we extrapolate convergence. In addition, if we predict that all of the eigenvalues remaining to be computed will converge in the steps corresponding to twice the number of steps given in Table 9, we continue the recurrence to avoid computing another factorization.

TABLE 9
Steps to initialize cost model and over which convergence is extrapolated.

	Vibration		Buckling		
			$\leq$ 10 modes	$>$ 10 modes	
Blocksize	Initial steps	Extrapolation steps	Initial steps	Initial steps	Extrapolation steps
1	35	6	15	30	6
2	20	4	15	25	6
3	20	2	10	25	4
$\geq$ 4	15	2	10	10	4

Our experience with this scheme is that the cost curve is relatively flat near its minimum, making the choice of where to stop appear to be flexible. This is misleading; the global minimum is quite sensitive to the local choice. To demonstrate the value of a well tuned dynamic scheme for evaluating cost, we include some simple experiments here. We modified our standard scheme to make it terminate early and to force it to run ten steps beyond where it would normally stop. The results are given in Table 8 and show some sensitivity to small changes in the stopping procedure.

4.5. Choice of blocksize and starting block. The two largest benefits of the block algorithm are in i/o cost reduction and in treating multiple eigenvalues. How-

ever, the costs of obtaining the M-orthogonal factorization and of the eigenanalysis of T_j increase quadratically with the blocksize p. In general, it is best to choose a blocksize as large as the largest expected multiplicity if eigenvalues of moderate multiplicities are expected. This is particularly important if many clusters of eigenvalues are expected (Table 12). A blocksize of 6 or 7 works well in problems with rigid body modes. We rarely find that $p > 10$ is cost-effective.

The effect of input/output cost is considerable. Within the MacNeal–Schwendler NASTRAN product, which runs on a variety of commercial systems, extensive testing resulted in a default blocksize of 7 on all systems. Input and output is particularly expensive within NASTRAN. In an environment in which input/output cost is less costly, a blocksize of 3 was found to be more effective. We provide our results on a small number of experiments in §5; it is likely that the optimal blocksize would change on other systems.

One would like to start the Lanczos algorithm with a good guess at a solution. We begin the first Lanczos run with a randomly generated starting block. Thereafter, the approximate eigenvectors (Ritz vectors) from unconverged Ritz values are available as estimates of the next eigenvectors to be found. At the time that the eigenvectors of T are available, we do not know where the next shift will be taken. Therefore, we take a starting block built from all of these Ritz vectors. If t vectors are available, each column in the starting block is taken to be the sum of t/p Ritz vectors. We fill the block out randomly when $t < p$. We adopted this approach after extensive experiments comparing various choices of starting blocks, including mixtures of Ritz vectors and random components. We did not find a significant change in the overall cost of the eigensolution with any of the approaches.

5. Experimental results. The algorithm described in the paper was developed as a general purpose eigensolver for the MacNeal–Schwendler Corporation's structural engineering package NASTRAN [18]. One of the goals in the software design was to make the eigensolver independent of the form of the sparse matrix operations representing the matrices involved: the matrix-block products, triangular block solves, and sparse factorizations. The eigensolver has been used in MSC NASTRAN with two different approaches to the sparse linear equations involved, a profile and a sparse multifrontal factorization. In both cases the factorization and solve modules are the standard operations of MSC NASTRAN, used directly by the eigensolver. The code has also been incorporated in four other structural engineering packages and in mathematics libraries supplied by Boeing Computer Services (BCSLIB-EXT)[1] [1] and Convex Computer Corporation (Veclib). In all of these implementations the sparse linear equations are solved with vectorized multifrontal codes based on the work in [2]–[4]. The multifrontal code computes a stable symmetric indefinite factorization, as described in [26].

In this section we report on experiments using our standard eigensolver from BCSLIB-EXT. The experiments were all performed on a Sun 4/690 workstation with 64 megabytes of main memory. The codes are all written in Fortran 77, and were run with the "-O" optimization option of the Sun Fortran compiler, which is quite effective with the inner loops of the numerical operations. We note that our code is always a block code, even when run with blocksize 1. This results in greater costs for the analysis of the tridiagonal system, where the results of Parlett and Nour-Omid would be available [33]. However, the cost of the tridiagonal analysis is less than 1%

[1] BCSLIB-EXT is available at no cost on all Cray Research, Inc. computers.

in general.

The test problems are drawn from the symmetric eigenproblems from the Harwell–Boeing test collection [11]. Our code has been used to solve eigenproblems with more than a million degrees of freedom, but the largest problem in the current test collection is of order 15,439 and most of the problems are much smaller. As a result, the order independent costs of the Lanczos algorithm, primarily the analysis of the block tridiagonal systems, are more important than they would be in large production problems. For most of the examples, we report the costs of the required eigenanalysis as a function of blocksize. For the largest problem we also report the breakdown of the cost in terms of the functional operations of the code.

5.1. Some empirical examples. Throughout this paper we have used some of the problems from the Harwell–Boeing test collection [11] to demonstrate particular aspects of our algorithms. We close by using a small subset to illustrate some of the global behavior of the code, particularly as it concerns aspects over which the user exercises control. We chose four test problems, listed in Table 10, which were collected from actual industrial or research applications.

TABLE 10
Test problems.

Matrix	order	Nonzeros in K	Nonzeros in M	Description
BCSST_08	1074	7017	1074	television station
BCSST_25	15439	133840	15439	76-story skyscraper
BCSST_26	1992	16129	1922	nuclear reactor containment floor
PLAT1919	1919	17159	—[a]	Atlantic and Indian Oceans

[a] ordinary eigenvalue problem.

Two of the problems have been used as the primary examples in this paper. They are BCSST_26, a model of a nuclear reactor containment floor used for seismic analysis, and PLAT1919, a finite difference model of tidal currents. These models were included in the test collection because of the large number of eigenpairs that were required of each. In both cases the number of modes is large because the analysis depended on knowing *all* of the modes in specified intervals.

Details of the eigenanalysis of the nuclear reactor containment floor problem, as a function of blocksize, are given in Table 11. These results exhibit a pattern common to all of the problems: The number of factorizations and Lanczos runs decrease rapidly as the blocksize increases; the cost of the eigenanalysis initially decreases as well, but then increases. This reflects the fact that as the blocksize increases, the length of the Lanczos runs increase in terms of the dimension of $\mathcal{Q}_j$. Longer runs involve greater costs, particularly for maintaining semi-orthogonality and for the back transformation of the eigenvectors. For these relatively small matrices, the costs of longer runs begin rather early to dominate the costs of factoring and applying the matrix operators. For reference, an analysis with a single Lanczos run with a blocksize of 3 had a cost of 543.4 for this problem, nearly three times the cost of the analysis with shifting.

The desired eigenvalues in the oceanography problem are very much in the interior of the spectrum. There are 818 eigenvalues above and 465 eigenvalues below the values we want. This problem was analyzed without the use of the spectral transformation in [5], [23]. Without shifting, it was barely possible to compute the eigenvalues in the interval [.0001, .24]; the eigenvalues in [.000025, .0001] were also of interest, but

TABLE 11
Computation of 200 *eigenvalues from* BCSST_26 *(shift statistics).*

Block-size	cpu cost	Factor-izations	Runs	Block	
				steps	solves
1	131.1	12	12	440	475
2	143.3	11	10	254	283
3	188.5	9	8	181	204
4	272.8	8	8	182	205
5	346.3	7	7	162	182
6	301.2	4	4	93	104

were impossible to compute. Secondly, all the eigenvalues, except a singleton at zero, are positive and occur in pairs. These multiple eigenvalues can play havoc with an ordinary, point Lanczos algorithm. With either a blocksize of 1 or 2, it is difficult for a code to be sure that it has exhibited the full multiplicities of the eigenvalues—the shifting strategy must be prepared to assist. Even with shifting, the single vector code of Ericsson and Ruhe [12], [15] was unable to cope with the multiplicities of the eigenvalues [25].

Table 12 shows the difficulty that arises with rank determination when the block-size is the same as the multiplicity of the eigenvalues. When we use a blocksize of 2, we cannot distinguish between doubletons that are truly doubletons and those that are only two of a larger number of copies of a multiple eigenvalue. As a result, our code makes a large number of reruns to ensure that we have the full multiplicities of eigenvalues. This is shown by the discrepancy between the number of factorizations and the number of runs. Although the reruns incur no new cost for factorizations, they do require more extensive use of external selective orthogonalization than would an ordinary run. Surprisingly, the point version of the code is able to cope well with this problem. As expected, blocksize larger than the multiplicity of 2 have no difficulties.

TABLE 12
Computation of 636 *eigenvalues from* PLAT1919 *(shift statistics).*

Block-size	cpu cost	Factor-izations	Runs	Block	
				steps	solves
1	659.6	33	33	1461	1526
2	1101.9	19	35	1068	1137
3	696.0	21	22	595	638
4	825.2	16	16	427	458
5	953.8	15	14	362	389
6	1043.6	12	12	291	314

BCSST_08 is a model of a building housing a television studio. Its claim to fame is the presence of isolated double and near triple eigenvalues. The lowest 24 eigenvalues are given in Table 13. The close eigenvalues cause relatively slow convergence, which causes our code to make more runs than we might expect. This problem can be solved easily enough with a single run, but at increased cost. We note that the multiple eigenvalues provide some challenges for blocksizes of 1 or 2. Details are given in Table 14.

BCSST_25 is an incomplete seismic model of the Columbia Center, a 76-story skyscraper in Seattle, Washington. The spectrum of this model is pathologically difficult—the lowest 132 eigenvalues are listed in Table 15. For reference, the largest eigenvalue of this structure is 1.51×10^8.

TABLE 13
Lowest 26 *eigenvalues of* BCSST_08.

i	λ_i
1	6.900
2	18.14206
3	18.1423664462086
4	18.1423664462086
5	84.78615
6	84.7864335537914
7	84.7864335537914
8	85.54
9	91.05
10	93.45
11	130.9
12	131.5
13	132.9
14	136.2
15	137.2
16	138.4
17	138.7
18	139.6
19	140.6
20	141.1
21	141.566
22	141.638
23	142.19
24	142.642

TABLE 14
Computation of lowest 20 *eigenvalues from* BCSST_08 *(shift statistics).*

Block-size	cpu cost	Factor-izations	Runs	Block steps	Block solves
1	37.6	5	5	179	193
2	26.0	4	3	63	71
3	22.2	2	2	39	44
4	34.3	4	3	46	54
5	33.4	3	2	31	36
6	37.9	2	2	29	34

TABLE 15
Lowest 132 *eigenvalues of* BCSST_25.

i	λ_i
1	9.6140×10^{-4}
2	9.7948×10^{-4}
3	9.7961×10^{-4}
4	9.8380×10^{-4}
5	9.85801×10^{-4}
⋮	⋮
68	9.85803×10^{-4}
69	9.86240×10^{-4}
⋮	⋮
132	9.86243×10^{-4}

The smallest eigenvalues are nearly negligible when compared to the largest eigenvalue and they are very close to one another. Our code determines clusters of eigenvalues based on its accuracy tolerance, which defaults to 2.31×10^{-11} in IEEE arithmetic. We apply this tolerance to the transformed eigenvalues, which are *not* close enough to be treated as a cluster or even as two clusters and four isolated values. (Note that if we applied the tolerance to the untransformed eigenvalues, all of these values would be a cluster, which is not appropriate.) As a result, this problem counters our usual shifting strategy—in this case we must take a shift very close to the eigenvalues in order to overcome the very poor separation and slow convergence. This distribution, eigenvalues almost, but not quite, in a cluster represents a worst case. Table 16 documents the performance of our code on this problem. We see that for this problem the costs of larger blocksizes are more than offset by the additional power they provide in attacking the very close and large clusters of eigenvalues. In Table 17 we present the breakdown of cost by function within the algorithm for this, the largest of our test problems. This breakdown is typical of larger problems in that neither the cost of analyzing T nor of choosing shifts is significant. It is atypical in that the startup cost is high, a result of there being a large number of vectors involved in external selective orthogonalization.

5.2. Summary. The results in the previous section illustrate some of the characteristics of the shifted block Lanczos algorithm. Only BCSST_25 is large enough to begin to demonstrate the behavior of the algorithm on large problems. For larger

TABLE 16
Computation of 132 *eigenvalues from* BCSST_25 *(shift statistics).*

Block-size	cpu cost	Factor-izations	Runs	Block	
				steps	solves
1	6372.8	7	7	586	606
2	5451.8	9	9	293	320
3	3683.3	5	5	158	172
4	3935.4	5	5	126	140
5	4063.3	5	5	108	122
6	2743.3	2	2	56	61

TABLE 17
Computation of lowest 132 *eigenvalues from* BCSST_25 *(cost breakdown).*

Percent of Cost							
Block-size	Recur-rence	Factor-ization	Re-orthog.	Block tridiag.	Eigen-vector	Start up	Shift select.
1	25	15	44	0	4	12	0
2	28	22	30	0	5	15	0
3	33	18	33	1	10	4	0
4	33	16	35	1	10	4	0
5	34	16	35	1	10	5	1
6	32	10	40	1	16	1	1

problems we expect to see the cost of the factorization and linear equation solutions to increase faster than linearly. Assuming that the eigenvalue distributions do not change, the cost of reorthogonalization, of generating the starting block, and of the eigenvector computation will increase linearly with the change in problem size. The block tridiagonal eigenanalysis and the shift selection should remain constant and their contributions to cost will become even smaller. We note that the cost of the necessary reorthogonalizations is an important fraction of the cost—this is a strong argument for preserving only *semi-orthogonality* rather than complete orthogonality. We remind the reader that our cost measures include a factor for i/o traffic, an essential ingredient in preserving semi-orthogonality.

The reader will see the difficulty in making an a priori choice of blocksize. The advantages and disadvantages of the block algorithm are clearly demonstrated, but we see no optimal choice for blocksize. A choice of three is always good on these problems on our Sun workstation, but is likely to be less than optimal for a vibration problem with six rigid body modes. Systems that impose higher costs for i/o will make higher blocksizes more effective, particularly when the problems are large enough that the factored matrices must reside on secondary storage.

These issues should be kept in the perspective of the power of the spectral transformation. None of the problems described here is solvable in any practical sense using the naive reduction to standard form. For example, the oceanography problem, PLAT1919, was analyzed in [5], [23] without any transformation—the desired eigenvalues were not close to appearing after N steps. (In unreported experiments, $3N$ steps had resulted in little improvement.) Although it is possible to solve some of the simpler problems by inverting the problem, as in (2), this is clearly not sufficient for all of the problems. The oceanography problem, PLAT1919, is singular, so some nontrivial shift is required. Even with a shift at the lower endpoint, .000025, a single Lanczos run to compute the lowest 200 eigenvalues above this point had a cost of 5382 for blocksize 3. In contrast, our standard shifted code with the same blocksize had

a cost of 696 for computing *all* 636 desired eigenvalues. The Columbia Center model has the same characteristics. The naive reduction would result in a problem with separations of 10^{-13} for the "well separated" eigenvalues; the simple reciprocal transformation would be clearly inadequate to begin to solve this problem. It is only with the combined power of the block Lanczos algorithm and the spectral transformation that we can solve these problems in a reasonable amount of time.

A. Matrix inertias. We need to interpret the number of negative eigenvalues of $K - \sigma M$ and $K - \sigma K_\delta$ in terms of the eigenvalues of the original vibration or buckling problems. The result we want to prove follows in Table 18. We use this result to conclude that

$$\nu(K - \sigma_2 M) - \nu(K - \sigma_1 M) = \text{number of eigenvalues in } (\sigma_1, \sigma_2),$$

where we assume that $\sigma_2 > \sigma_1$. In the case of buckling analyses we further assume that both σ_1 and σ_2 have the same sign.

TABLE 18
Interpretation of $\nu(K - \sigma M)$ or $\nu(K - \sigma K_\delta)$.

Vibration analysis:	
M positive definite	# of eigenvalues $< \sigma$
M positive semidefinite	(# of eigenvalues $< \sigma$) + γ
	$\gamma = \begin{cases} 0 & \text{some cases} \\ \dim(\mathcal{N}(M)) & \text{other cases} \end{cases}$
Buckling analysis:	
K positive definite	# of eigenvalues in $(0, \sigma)$ or $(\sigma, 0)$
K positive semidefinite	(# of eigenvalues in $(0, \sigma)$ or $(\sigma, 0)$) + γ
	$\gamma = \begin{cases} 0 & \sigma \text{ of one sign} \\ \dim(\mathcal{N}(K)) & \sigma \text{ of other sign} \end{cases}$

There are four cases, which will be considered in pairs. In all cases we assume that the problem is a definite generalized symmetric eigenproblem, i.e., that there exists some linear combination $\alpha K + \beta M$ that is positive definite.

A.1. $Kx = \lambda Mx$ **with** M **positive definite.** We can apply the obvious reduction to standard form. The eigenvalues of $Kx = \lambda Mx$ are the same as the eigenvalues of $C = L_M^{-1} K L_M^{-T}$, where L_M is the Cholesky factor of M. It follows that the number of eigenvalues of C less than σ is the same as the number of eigenvalues of $Kx = \lambda Mx$ less than σ. But $C - \sigma I$ is congruent to $L_M(C - \sigma I)L_M^T$ and this is simply $K - \sigma M$. Thus, the decomposition of $K - \sigma M$ gives the number of eigenvalues less than σ. Obviously, the interpretation of the inertia has the same meaning here as in the ordinary eigenvalue problem.

A.2. $Kx = \lambda Mx$ **with** M **positive semidefinite.** Signs must be assigned to the infinite eigenvalues when M is singular. Assume that M is positive semidefinite, with p zero eigenvalues. Then there exists a nonsingular matrix W so that WMW^T is the two-by-two block-partitioned matrix

$$WMW^T = \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix},$$

where I is an $(n-p) \times (n-p)$ identity matrix. Partition $WKW^T = C$ conformally as

$$WKW^T = \begin{pmatrix} C_{11} & C_{21}^T \\ C_{21} & C_{22} \end{pmatrix}.$$

Some linear combination $\alpha K + \beta M$ is positive definite, from which it follows that $\alpha(WKW^T) + \beta(WMW^T)$ is positive definite. But a positive definite matrix has positive definite principal minors, which implies that αC_{11} is positive definite. Let y satisfy $(WKW^T)y = \lambda(WMW^T)y$ and partition y conformally as $[y_1, y_2]^T$. Then

$$C_{11}y_1 + C_{21}^T y_2 = 0 \tag{31}$$

and

$$C_{21}y_1 + C_{22}y_2 = \lambda y_2. \tag{32}$$

Equation (31) then implies that

$$y_1 = -C_{11}^{-1}C_{21}^T y_2.$$

Substituting in (32), we obtain

$$(C_{22} - C_{21}C_{11}^{-1}C_{21}^T)y_2 = \lambda y_2.$$

Thus, the finite eigenvalues of $Kx = \lambda Mx$ are the eigenvalues of the Schur complement of C_{11} in C.

By Sylvester's theorem the inertia of $(K - \sigma M)$ is the same as the inertia of $WKW^T - \sigma WMW^T$. But the partitioned form of the LDL^T decomposition of $WKW^T - \sigma WMW^T$ has as its (1,1) block the decomposition of C_{11}, and as its (2,2) block the decomposition of $(C_{22} - C_{21}C_{11}^{-1}C_{21}^T) - \sigma I$. The inertia for the entire matrix is offset by the inertia of the (1,1) block. The offset is constant—it describes the sign given to the infinite eigenvalues. That all of the infinite eigenvalues have the same sign is due to the fact that a positive definite linear combination of K and M exists, that is, that the problem is a definite generalized symmetric eigenproblem [13]. The difference between $\nu(K - \sigma_1 M)$ and $\nu(K - \sigma_2 M)$ will still be the number of eigenvalues in $[\sigma_1, \sigma_2)$, since the constant term cancels.

Furthermore, in vibration analysis, we know that both K and M are positive semidefinite. It follows that both α and β will be positive when M is only semidefinite. The positive semidefiniteness of K then implies that C_{11} is a positive definite matrix, so $\nu(C_{11}) = 0$. Thus, the inertia of the factored matrix retains exactly the same meaning for the positive semidefinite vibration case as for the positive definite case.

A.3. $Kx = \lambda K_\delta x$ **with** K **positive definite.** In buckling analysis, only K has any definiteness properties. We can invert the problem when K is positive definite. Thus

$$Kx = \lambda K_\delta x$$

implies

$$K_\delta x = \frac{1}{\lambda} Kx = \mu Kx,$$

and all the eigenvalues μ in the second equation are finite. This transformed problem is in the standard (K_δ, K) form in which the right-hand side matrix, K, is positive definite. We will determine the number of eigenvalues of (K_δ, K) that lie in the image of the interval of interest in the original problem. Thus, to determine the number of eigenvalues of $Kx = \lambda K_\delta x$ less than σ, we find the number of eigenvalues of the

inverted problem (K_δ, K) in the interval(s) in the variable $\mu = \frac{1}{\lambda}$ that corresponds to the interval $(-\infty, \sigma)$ in the variable λ.

There are three subcases that must be considered. The first is the case $\sigma = 0$. The interval $(-\infty, 0)$ for λ is mapped to the interval $(-\infty, 0)$ in $\frac{1}{\lambda}$. Thus, the number of negative eigenvalues of $Kx = \lambda K_\delta x$ is the same as the number of eigenvalues of (K_δ, K) less than 0. This is simply the number of negative eigenvalues of K_δ, $\nu(K_\delta)$.

The second case is the case $\sigma < 0$. The transformation from λ to $\frac{1}{\lambda}$ transforms σ to $\frac{1}{\sigma}$. The number of eigenvalues in $(-\infty, \sigma)$ is the same as the number of eigenvalues of (K_δ, K) in the interval $(\frac{1}{\sigma}, 0)$. This is

$$\nu(K_\delta) - \nu\left(K_\delta - \frac{1}{\sigma}K\right),$$

which, because σ is negative, is the same as

$$\nu(K_\delta) - \nu(K - \sigma K_\delta).$$

Note that the number of eigenvalues between σ and 0 is simply $\nu(K - \sigma K_\delta)$.

The third case is $\sigma > 0$. In this case, the interval $(-\infty, \sigma)$ in λ must be treated as the union of the interval $(-\infty, 0)$ and the interval $[0, \sigma)$. There are $\nu(K_\delta)$ eigenvalues in the first subinterval. The second subinterval is transformed into $(\frac{1}{\sigma}, +\infty)$. The union has

$$\nu(K_\delta) + \pi\left(K_\delta - \frac{1}{\sigma}K\right)$$

or

$$\nu(K_\delta) + \nu(K - \sigma K_\delta)$$

eigenvalues. Even in this case $\nu(K - \sigma K_\delta)$ is the number of eigenvalues between 0 and σ.

The buckling problem will have infinite eigenvalues if K_δ is singular. However, the signs of these eigenvalues are irrelevant to the interpretation of the inertias because the interpretation always considers only finite subintervals.

A.4. $Kx = \lambda K_\delta x$ **with** K **positive semidefinite.** The most general case we consider is a buckling analysis in which K is only positive semidefinite. We combine the analysis for the semidefinite vibration case with the positive definite buckling case to assign signs to the zero eigenvalues.

We assume K is semidefinite, with P zero eigenvalues. As before, there exists a nonsingular matrix $\widehat{W}$ such that

$$\widehat{W}K\widehat{W}^T = \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix}.$$

Partition $\widehat{W}K_\delta\widehat{W}^T = E$ conformally as

$$\begin{pmatrix} E_{11} & E_{21}^T \\ E_{21} & E_{22} \end{pmatrix}.$$

The eigenvalues of $KX = K_\delta X\Lambda$ are those of $\widehat{W}K\widehat{W}^TY = \widehat{W}K_\delta\widehat{W}^TY\Lambda$. Let y be an eigenvector, partitioned conformally as $[y_1, y_2]^T$. Then

$$E_{11}y_1 + E_{21}^Ty_2 = 0 \tag{33}$$

and

$$\lambda(E_{21}y_1 + E_{22})y_2 = y_2.$$

As before the (1,1) block of the transformed linear combination, βE_{11}, is a positive definite matrix. Equation (33) then implies that

$$y_1 = -E_{11}^{-1}E_{21}^T y_2,$$

or

$$\lambda(E_{22} - E_{21}E_{11}^{-1}E_{21}^T)y_2 = y_2.$$

Thus, the finite, nonzero eigenvalues of $Kx = \lambda K_\delta x$ are the reciprocals of the nonzero eigenvalues of the Schur complement of E_{11} in E.

The partitioned form of the LDL^T decomposition of $\widehat{W}K\widehat{W}^T - \sigma\widehat{W}K_\delta\widehat{W}^T$ has as its (1,1) block the decomposition of $-\sigma E_{11}$, and as its (2,2) block the decomposition of $I - \sigma(E_{22} - E_{21}E_{11}^{-1}E_{21}^T)$. The Schur complement block is in the form of §A.3, taking the identity matrix as M. Again, the inertia of the full matrix is the inertia of $I - \sigma(E_{22} - E_{21}E_{11}^{-1}E_{21}^T)$ offset by the inertia of the (1,1) block. Notice that the offset depends on the sign of the shift—it describes the signs of the eigenvalues of $-\sigma E_{11}$. Because E_{11} is definite, either all the eigenvalues of $-\sigma E_{11}$ are positive or all are negative. Thus, the offset will be zero for shifts of one sign and nonzero for shifts of the other sign. Still, the difference between $\nu(K - \sigma_1 K_\delta)$ and $\nu(K - \sigma_2 K_\delta)$ will still be the number of eigenvalues in $[\sigma_1, \sigma_2)$, as long as both shifts have the same sign. The dimension of the nullspace of K, $\nu(E_{11})$, is often known adventitiously; if not, it can be estimated by factoring $K - \rho I$, where ρ is chosen smaller than the least nonzero eigenvalue of K, but large enough so that the factorization is stable.

Acknowledgments. The authors would like to thank David Scott for his participation in the initial design of this code and Thomas Ericsson and Bahram Nour-Omid for many enlightening discussions during the preparation of this code. We particularly thank Beresford Parlett for his interest and intellectual support, which led to important changes, even after we thought our work was done. We also are grateful to Louis Komzsik and The MacNeal–Schwendler Corporation for their support of this work. Finally, Linda Kaufman provided a fine example of careful and thorough editing.

REFERENCES

[1] *The Boeing Extended Mathematical Subprogram Library*, Boeing Computer Services, P.O. Box 24346, Seattle, WA 98124-0346, 1989.

[2] C. C. Ashcraft, *A vector implementation of the multifrontal method for large sparse symmetric positive linear systems*, Tech. Report ETA-TR-51, Boeing Computer Services, P.O. Box 24346, Seattle, WA 98124-0346, 1987.

[3] C. C. Ashcraft and R. G. Grimes, *The influence of relaxed supernode partitions on the multifrontal method*, ACM Trans. Math. Software, 15 (1989), pp. 291–309.

[4] C. C. Ashcraft, R. G. Grimes, J. G. Lewis, B. W. Peyton, and H. D. Simon, *Progress in sparse matrix methods for large linear systems on vector supercomputers*, Internat. J. Supercomput. Appl., 1 (1987), pp. 10–30.

[5] A. Cline, G. Golub, and G. Platzman, *Calculations of normal modes of oceans using a Lanczos method*, in Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, New York, 1976, pp. 409–426.

[6] J. Cullum and W. E. Donath, *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large sparse real symmetric matrices*, Proc. 1974 IEEE Conference on Decision and Control, pp. 505-509, IEEE Computer Society, 1974.

[7] J. Cullum and R. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol.* 1 *Theory*, Birkhäuser, Boston, 1985.

[8] ———, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol.* 2 *Users Guide*, Birkhäuser, Boston, 1985.

[9] ———, *Computing eigenvalues of large matrices, some Lanczos algorithms and a shift and invert strategy*, in Advances in Numerical Partial Differential Equations and Optimization: Proc. 5th Mexico–United States Workshop, S. Gomez, J. P. Hennart, and R. A. Tapia, eds., Proc. in Applied Mathematics, Vol. 47, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[10] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, *Reorthogonalizations and stable algorithms for updating the Gram–Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

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

[12] T. Ericsson, *User Guide for STLM*, Tech. Report UMINF-96.82, University of Umea, Umea, Sweden, 1982.

[13] ———, *A generalized eigenvalue problem and the Lanczos algorithm*, in Large Scale Eigenvalue Problems, J. Cullum and R. Willoughby, eds., North–Holland, Amsterdam, 1986.

[14] T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method*, Math. Comp., 34 (1980), pp. 1251–1268.

[15] ———, *STLM-A Software Package for the Spectral Transformation Lanczos Algorithm*, Tech. Report UMINF-101.82, University of Umea, Umea, Sweden, 1982.

[16] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler, *Matrix Eigensystem Routines—EISPACK Guide Extension*, Vol. 51 of Lecture Notes in Computer Sciences, Springer-Verlag, Berlin, 1977.

[17] G. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical Software III, J. Rice, ed., Academic Press, New York, 1977.

[18] R. Grimes, J. Lewis, L. Komzsik, D. Scott, and H. Simon, *Shifted block Lanczos algorithm in MSC/NASTRAN*, in Proc. MSC/NASTRAN User's Conference, Los Angeles, 1985.

[19] R. Grimes, J. Lewis, and H. Simon, *Eigenvalue Problems and Algorithms in Structural Engineering*, in Large Scale Eigenvalue Problems, J. Cullum and R. Willoughby, eds., North–Holland, Amsterdam, 1986.

[20] ———, *A Shifted Block Lanczos Algorithm for Solving Sparse Symmetric Generalized Eigenproblems*, Tech. Report AMS-TR-166, Boeing Computer Services, P.O. Box 24346, Seattle, WA 98124-0346, 1991.

[21] M. T. Jones and M. L. Patrick, *LANZ: Software Solving the Large Sparse Symmetric Generalized Eigenproblem*, Tech. Report NAS1-18605, ICASE, NASA Langley Research Center, Hampton, VA, August 1990.

[22] C. Lanczos, *An iteration method for the solution of eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Stand., 45 (1950), p. 255.

[23] J. Lewis, *Algorithms for Sparse Matrix Eigenvalue Problems*, Ph.D. thesis, Dept. of Computer Science, Stanford University, Stanford, CA, 1977.

[24] J. Lewis and R. Grimes, *Practical Lanczos algorithms for solving structural engineering eigenvalue problems*, in Sparse Matrices and Their Uses, I. Duff, ed., Academic Press, London, 1981.

[25] J. Lewis and H. Simon, *Numerical Experience with the Spectral Transformation Lanczos Method*, Tech. Report ETA-TR-16, Boeing Computer Services, P.O. Box 24346, Seattle, WA 98124-0346, 1983.

[26] J. W.-H. Liu, *A partial pivoting strategy for sparse symmetric matrix decomposition*, ACM Trans. Math. Software, 13 (1987), pp. 173–182.

[27] M. Newman and P. Flanagan, *Eigenvalue Extraction in NASTRAN by the Tridiagonal Reduction (FEER) Method—Real Eigenvalue Analysis*, NASA Contractor Report CR-2731, NASA, 1976.

[28] B. Nour-Omid, *The Lanczos Algorithm for Solution of Large Generalized Eigenproblems*, in The Finite Element Method, T. Hughes, ed., Prentice-Hall, Englewood Cliffs, NJ, 1987.

[29] B. Nour-Omid, B. N. Parlett, T. Ericsson, and P. S. Jensen, *How to implement the spectral transformation*, Math. Comp., 48 (1987), pp. 663–673.

[30] C. C. Paige, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, Ph.D. thesis, University of London, London, UK, 1971.

[31] ———, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, J. Inst. Math. Appl., 18 (1976), pp. 341–349.

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

[33] B. N. Parlett and B. Nour-Omid, *The use of refined error bounds when updating eigenvalues of a growing symmetric tridiagonal matrix*, Linear Algebra Appl., 68 (1985), pp. 179–219.

[34] B. Parlett, B. Nour-Omid, and Z. A. Liu, *How to Maintain Semi-Orthogonality Among Lanczos Vectors*, Tech. Report PAM-420, Center for Pure and Applied Mathematics, University of California, Berkeley, CA, 1988.

[35] B. Parlett and D. Scott, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 217–238.

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

[37] D. Scott, *Block Lanczos Software for Symmetric Eigenvalue Problems*, Tech. Report ORNL/CSD 48, Oak Ridge National Laboratory, Oak Ridge, TN, 1979.

[38] ———, *The advantages of inverted operators in Rayleigh–Ritz approximations*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 68–75.

[39] H. Simon, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods*, Linear Algebra Appl., 61 (1984), pp. 101–131.

[40] ———, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–136.

[41] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines—EISPACK Guide*, Lecture Notes in Computer Sciences, Vol. 6, Springer-Verlag, Berlin, 1976.