(2)

NASA Contractor Report 181914
ICASE Report No. 89-69

ICASE

THE USE OF LANCZOS'S METHOD TO SOLVE THE LARGE GENERALIZED SYMMETRIC DEFINITE EIGENVALUE PROBLEM

Mark T. Jones
Merrell L. Patrick

Contract No. NAS1-18605
September 1989

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665-5225

Operated by the Universities Space Research Association

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

Recently, ICASE has begun differentiating between reports with a mathematical or applied science theme and reports whose main emphasis is some aspect of computer science by producing the computer science reports with a yellow cover. The blue cover reports will now emphasize mathematical research. In all other aspects the reports will remain the same; in particular, they will continue to be submitted to the appropriate journals or conferences for formal publication.

The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem

Mark T. Jones*and Merrell L. Patrick*†

Abstract

The generalized eigenvalue problem, $Kx = \lambda Mx$, is of significant practical importance, especially in structural engineering where it arises as the vibration and buckling problems. A new algorithm, **LANZ**, based on Lanczos's method is developed. **LANZ** uses a technique called dynamic shifting to improve the efficiency and reliability of the Lanczos algorithm. A new algorithm for solving the tridiagonal matrices that arise when using Lanczos's method is described. A modification of Parlett and Scott's selective orthogonalization algorithm is proposed. Results from an implementation of **LANZ** on a Convex C-220 show it to be superior to a subspace iteration code.

*Department of Computer Science, Duke University, Durham, NC 27706

†This research was supported by the National Aeronautics and Space Administration under NASA contract No. NAS1-18605 and the Air Force Office of Scientific Research under AFOSR grant No. 88-0117 while the authors were in residence at ICASE. Additional support was provided by NASA grant No. NAG-1-466.

1 Introduction

The solution of the symmetric generalized eigenvalue problem,

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

where K and M are real, symmetric matrices, and either K or M is positive semi-definite, is of significant practical importance, especially in structural engineering as the vibration problem and the buckling problem [BH87]. The matrices K and M are either banded or sparse. Usually $p << n$ of the smallest eigenvalues of Equation 1 are sought, where n is the order of the system. The method of Lanczos, suitably altered for the generalized eigenvalue problem, is shown to be useful for the efficient solution of Equation 1. [Lan50] [NOPEJ87].

A sophisticated algorithm, based on the simple Lanczos algorithm, is developed in this paper. The algorithm, called **LANZ**, has been implemented on supercomputer architectures at NASA Langley Research Center and results from the implementation are discussed. Two applications from structural engineering are described in Section 2. The properties of the generalized eigenvalue problem and solution methods are given in Section 3. The simple Lanczos algorithm is presented in Section 4. The use of Lanczos's method for the generalized eigenvalue problem and the **LANZ** algorithm are discussed in Section 5. An execution time cost analysis of **LANZ** is developed in Section 6. The solution of the tridiagonal matrices that arise when using Lanczos's method is considered in Section 7. Methods for solving the symmetric indefinite linear systems that arise in **LANZ** are described in Section 8. In Section 9, the performance of the **LANZ** algorithm is analyzed and compared to the performance of subspace iteration, the most prevalent method for solving this class of problems in structural engineering.

2 Problems of Interest

Two important applications of **LANZ** which arise in structural engineering are the vibration problem and the buckling problem. Practical problems from these applications will be used to test the performance of the program.

In the free vibration problem, the vibration frequencies, ω, and mode shape vectors, x, of a structure are sought. The equation

$$Kx = \omega^2 Mx, \tag{2}$$

is solved, where K is the positive definite stiffness matrix and M is the semi-positive definite mass matrix. The mass matrix, M, can be either diagonal (in which case it is referred to as a diagonal mass matrix) or can have approximately the same structure as K (in which case it is referred to as a consistent mass matrix). Because K is positive definite and M is semi-positive definite, the eigenvalues of 2 are non-negative. When a dynamic load is applied to a structure, the structure begins to vibrate. If the vibration frequency is close to a natural frequency of the structure, ω, then the resulting resonance can lead to structural failure. Engineers want to ensure that a structure has no natural frequencies near the range of frequencies of expected dynamic loads. Engineers are often interested in a few of the lowest frequencies of the structure to ensure that these natural frequencies are well above the frequencies of expected dynamic loads. [Jon89]. In some situations, all the frequencies in a particular range may be sought. [Kni89].

In the buckling problem, the smallest load at which a structure will buckle is sought. These buckling loads of a structure, λ, are the eigenvalues of the system,

$$Kx = -\lambda K_G x, \tag{3}$$

where K is the positive definite stiffness matrix, K_G is the geometric stiffness matrix, and x is the buckling mode shape vector [Mah87]. The K_G matrix has approximately the same structure as K and can be indefinite and singular. Because K_G is indefinite, the eigenvalues of Equation 3 can be negative, although in most practical problems they are positive. Equation 3 can have up to n distinct eigenvalues; although only the smallest eigenvalue is of physical importance since any load greater than the smallest buckling load will cause buckling [Jon89]. Designers may, however, be interested in the six to ten lowest eigenvalues in order to observe the spacing of the buckling loads.

If the lowest buckling loads are greatly separated from other buckling loads, the designer may seek to change the structure in order to cause the lowest buckling loads to become closer to the other buckling loads of the structure. [Kni89]. Because the K_G matrix can be indefinite and singular, the buckling problem is more difficult to solve numerically than the vibration problem.

3 The Generalized Eigenvalue Problem

3.1 Properties

Equation 1 yields n eigenvalues if and only if the rank of M is equal to n, which may not be the case in the problems under consideration [GL83]. In fact, if M is rank deficient, then the number of eigenvalues of Equation 1 may be finite, empty, or infinite [GL83]. In practical problems, however, the set of eigenvalues is finite and non-empty. When K and M are symmetric and one or both of them is positive definite, the eigenvalues of Equation 1 are real (without sacrificing generality, it will be assumed that M is positive definite). An orthogonal matrix, R, exists such that

$$R^T M R = diag(\beta_i^2) = D^2, \tag{4}$$

where the β_i^2 are the eigenvalues of M and are therefore positive real. If M is expanded to RD^2R^T, then Equation 1 becomes

$$(K - \lambda M)x = (K - \lambda RDDR^T)x. \tag{5}$$

Equation 5 can be rearranged to become

$$(K - \lambda M)x = (RD(D^{-1}R^T K R D^{-1} - \lambda I)DR^T)x. \tag{6}$$

Taking the determinant of each side yields

$$det(K - \lambda M) = (det(R))^2 (det(D))^2 det(P - \lambda I), \tag{7}$$

where $P = D^{-1}R^T K R D^{-1}$. Because R is orthogonal $det(R) = 1$. $Det(D)$ is the product of the eigenvalues of M, which are all positive real. Therefore, the roots of $det(K - \lambda M)$ are those of $det(P - \lambda I)$. Because P is a symmetric matrix, its eigenvalues are all real and therefore those of Equation 1 are all real [Wil65].

When K is positive definite and M is positive semi-definite, as they are in the vibration problem, the eigenvalues of Equation 1 are all non-negative. To show that the eigenvalues are non-negative, multiply each side of Equation 1 by x^T to get

$$x^T K x = \lambda x^T M x. \tag{8}$$

Because K is positive definite, $x^T Kx$ is positive, and because M is positive semi-definite, $x^T Mx$ is non-negative; therefore, λ must be non-negative [Wil65].

The eigenvectors of Equation 1 satisfy M-orthogonality ($x^T My = 0$, if x and y are M-orthogonal). The matrix, P, has a complete set of eigenvectors, z_i, and the same eigenvalues as Equation 1. Thus,

$$Pz_i = \lambda_i z_i, \tag{9}$$

and,

$$D^{-1}R^T KRD^{-1}z_i = \lambda_i z_i. \tag{10}$$

Then, the sequence of transformations in Equation 11 show that $x = RD^{-1}z$,

$$KRD^{-1}z_i = \lambda_i RDz_i = \lambda_i RD(DR^T RD^{-1})z_i = \lambda_i M(RD^{-1}z_i) \tag{11}$$

Because the z_i are known to be orthogonal,

$$0 = z_i^T z_j = (DR^T x_i)^T DR^T x_j = x_i Mx_j. \tag{12}$$

The methods and algorithms discussed in this paper rely on the eigenvalues being real and the eigenvectors being M-orthogonal. In addition, **LANZ** makes use of the property that the eigenvalues in the vibration problem are non-negative.

3.2 Solution Methods

Several methods for solving the large symmetric generalized eigenvalue problem exist. A popular method, especially in structural engineering, is subspace iteration [Bat82]. Nour-Omid, Parlett and Taylor provide convincing theoretical and experimental evidence that Lanczos's method is superior to subspace iteration on a sequential machine [NOPT83]. The parallelization of subspace iteration was investigated by Mahajan and it remains an open question as to whether Lanczos's method will be superior to subspace iteration on parallel machines [Mah87]. Another solution method for the generalized eigenvalue problem is a two-level iterative method proposed by Szyld [Szy83]. He uses a combination of the inverse iteration and Rayleigh quotient methods at the outer level, and an iterative solver for indefinite systems at the inner level. Szyld assumes, however, that M is non-singular, which is not

always the case for the applications under examination. A promising method for parallel machines based on the Sturm sequence property is proposed by Ma [Ma88]. He uses multi-sectioning or bi-sectioning to obtain the eigenvalues, and then uses inverse iteration to recover the eigenvectors. Again, the assumption is made that M is non-singular. Schwarz proposes a method which minimizes the Rayleigh quotient by means of the conjugate gradient method. The method uses a partial Choleski decomposition as a preconditioner to speed convergence [Sch89]. SOR-based methods have also been proposed for the generalized eigenvalue problem, but these suffer from two flaws: 1) the methods have difficulty when the eigenvalues are clustered, and 2) the methods require that M be positive definite [Ruh74]. Other methods have also been proposed [SW82] [Sch74]. Block Lanczos methods have been developed but are more complicated than the simple Lanczos process. Block methods are limited in that the user of the method must choose the size of the blocks [NOC85]. One significant advantage of the block methods is that they can easily reveal the presence of multiple eigenvalues [GU77]. To the best of the authors' knowledge, no satisfactory method of choosing this block size to best determine the multiplicity of eigenvalues has been proposed. The block size is usually chosen to take advantage of a particular computing architecture [Ruh89].

4 Lanczos's Method

4.1 Execution in Exact Arithmetic

In order to understand Lanczos's method when applied to the generalized eigenvalue problem, it is first necessary to examine the method when applied to

$$Ax = \lambda x, \tag{13}$$

where A is an $n \times n$ real symmetric matrix. In Lanczos's method, the matrix A is transformed into a tridiagonal matrix, T, in n steps in exact arithmetic. However, roundoff errors make the simple use of Lanczos's method as a direct method for tridiagonalizing A impractical [Sim84]. Paige suggests using Lanczos's method as an iterative method for finding the extremal eigenvalues of a matrix [Pai71]. At each step, j, of the Lanczos algorithm, a tridiagonal matrix, T_j, is generated. The extremal eigenvalues of T_j approximate those of A, and as j grows, the approximations become increasingly good [GL83]. Both Kaniel and Saad have examined the convergence properties of Lanczos in exact arithmetic [Kan66] [Saa80]. The convergence results that they derive imply that the speed of convergence of the eigenvalues of the T matrices to eigenvalues of A depends on the distribution of the eigenvalues of A; if an extreme eigenvalue of A is well separated from its neighbors, convergence to this eigenvalue will be fast. However, clustered eigenvalues, or eigenvalues that are in the middle of the spectrum of A, will not be converged to as quickly as the extremal eigenvalues.

In Lanczos's method, a series of orthonormal vectors, $q_1 \ldots q_n$, is generated which satisfy:

$$T = Q^T A Q, \tag{14}$$

where the vectors q_i are the columns of Q. If each side of Equation 14 is multiplied by Q to yield,

$$QT = AQ, \tag{15}$$

and the columns on each side are set equal, then

$$\beta_{j+1} q_{j+1} + q_j \alpha_j + q_{j-1} \beta_j = A q_j. \tag{16}$$

where the α's and β's are the diagonal and subdiagonal, respectively, of T,

the tridiagonal matrix in Equation 17 [GL83].

$$T_j = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & \cdots & \cdots & & \\ & & & \beta_{j-1} & \alpha_{j-1} & \beta_j \\ & & & & \beta_j & \alpha_j \end{pmatrix} \tag{17}$$

Equation 16 can be rearranged to become the three-term recurrence relation

$$\beta_{j+1} q_{j+1} = A q_j - q_j \alpha_j - q_{j-1} \beta_j. \tag{18}$$

From this recurrence relation at step j:

$$A Q_j = Q_j T_j + r_j e_j^T, \tag{19}$$

where $r_j = q_{j+1}/\beta_{j+1}$ and e_j is the jth column of the $j \times j$ identity matrix [GL83]. These q vectors, also called Lanczos vectors, are the key to the algorithm. They are generated as shown in Equation 18. That this recurrence relation generates a set of Lanczos vectors, $q_1 \ldots q_j$, which belong to the Krylov subspace, $\kappa(A, q_1, j)$, can be shown by construction. That these Lanczos vectors are also orthonormal, and therefore span the Krylov subspace, $\kappa(A, q_1, j)$, is shown by induction in [GL83]. If the Lanczos vectors are not orthonormal, then this three-term recurrence relation does not generate the correct T matrix.

Two of the major benefits of the simple Lanczos algorithm are: 1) the structure of the matrix A is not important for the Lanczos algorithm; the only access to A that the algorithm requires is a routine that returns the product of A times a vector, and 2) at step j, only two of the Lanczos vectors, q_{j-1} and q_j, are required to be stored in core memory, the rest can be stored in secondary memory until needed for the computation of the eigenvectors. The simple Lanczos algorithm is shown in Figure 1 [GL83].

The tridiagonal eigenvalue problem generated by Lanczos's method can be written in the form

$$T_j s = \theta s, \tag{20}$$

where the θ's are sometimes called Ritz values. The eigenvalues of T_j will approximate those of A, especially the extremal ones. Approximations to the

$r_0 = starting\ vector$
$\beta_1 = \| r_0 \|$
$q_0 = 0$
$j = 1$
while $(\beta_j \neq 0)$
$q_j = r_{j-1}/\beta_j$
$\alpha_j = q_j^T A q_j$
$r_j = Aq_j - \alpha_j q_j - \beta_j q_{j-1}$
$\beta_{j+1} = \| r_j \|$
$j = j + 1$

Figure 1: The simple Lanczos algorithm

eigenvectors of A, called Ritz vectors, can be calculated using the equation

$$y_i = Q_j s_i, \tag{21}$$

where y_i is the Ritz vector corresponding to s_i [PNO85]. These Ritz vectors satisfy: [NOPEJ87]

$$Ay_i = y_i\theta_i + r_j s_i(j). \tag{22}$$

4.2 Effect of Roundoff Errors

The algorithm and equations in Subsection 4.1 hold only in exact arithmetic; they degenerate in the presence of roundoff errors, and, therefore, the Lanczos vectors can no longer be assumed to be orthonormal. When roundoff errors are taken into account, Equation 19 becomes

$$AQ_j = Q_j T_j + r_j e_j^T + F_j, \tag{23}$$

where F_j is used to represent roundoff error. Then,

$$\| Ay_i - y_i\theta_i \| \leq \beta_{ji} + \| F_j \|, \tag{24}$$

can be used to bound the error in the Ritz pair (θ_i, y_i). The norm of F can be bounded by $n^{1/2}\epsilon \| A \|$, where ϵ is a constant based on the floating point

precision of the computer, and $\beta_{ji} = \beta_{j+1} \mid s_i(j) \mid$ [PNO85]. The bound on $\| F(j) \|$ is small and can be disregarded. The most important factor then becomes β_{ji}. From [Pai71]

$$\mid y_i^T q_{j+1} \mid = \gamma_i / \beta_{ji}, \tag{25}$$

where γ is a roundoff term approximately equal to $\epsilon \| A \|$. From Equation 25, the conclusion can be drawn that as β_{ji} becomes small (and hence the error in the Ritz pair, (θ_i, y_i), also becomes small), the q vectors lose orthogonality [Par80]. Equation 25 implies that convergence of a Ritz pair to an eigenpair of A results in a lack of orthogonality among the Lanczos vectors. More specifically, significant components of y_i, the converged Ritz vector, creep into subsequent q_j's, causing spurious copies of Ritz pairs to be generated by the Lanczos process.

Several remedies for the loss of orthogonality have been proposed. Paige suggests full reorthogonalization, in which the current Lanczos vector, q_j, is orthogonalized against all previous Lanczos vectors [Pai71]. Full reorthogonalization becomes increasingly expensive as j grows. Cullum and Willoughby advocate a method in which the lack of orthogonality is ignored and sophisticated heuristics are used to detect the eigenvalues that are being sought among the many spurious eigenvalues that are generated [CW85]. Simon proposes a scheme called partial reorthogonalization in which estimates for the orthogonality of the current Lanczos vector, q_j, against all previous Lanczos vectors are inexpensively computed. Based on the estimates, a small set of the previous Lanczos vectors are orthogonalized against q_j [Sim84]. Partial reorthogonalization maintains semi-orthogonality among the Lanczos vectors. For all q vectors, semi-orthogonality is defined by:

$$q_j^T q_i \leq \epsilon^{1/2} \quad i \neq j. \tag{26}$$

"Semi-orthogonality" among the q vectors guarantees that the T matrix generated by the Lanczos algorithm executed with roundoff errors will be the same up to working precision as the T matrix generated using exact arithmetic [PNO85]. Selective orthogonalization is used by **LANZ** in an adapted form to maintain semi-orthogonality among the Lanczos vectors [PS79]. The strategy of selective reorthogonalization, as proposed by Parlett and Scott, orthogonalizes the current residual vector, r_j, and the last Lanczos vector, q_{j-1}, at the beginning of each step in the Lanczos algorithm against "good"

Ritz vectors (more details of how this occurs will be given in Section 5). "Good" Ritz vectors are those which correspond to Ritz values for which the value of β_{ji} is below $\epsilon^{1/2} \parallel A \parallel$. A low β_{ji} value suggests that the Ritz value is converging and, therefore, from Equation 25, $\mid y_i^T q_{j+1} \mid$ is increasing. The value of $\epsilon^{1/2} \parallel A \parallel$ is used to ensure that the quantity $\mid y_i^T q_{j+1} \mid$ never rises above $\epsilon^{1/2}$. As a result, semi-orthogonality, as defined in Equation 26, is maintained.

5 The LANZ Algorithm

5.1 The Spectral Transformation

In order to use Lanczos's method to find the lowest eigenvalues or the eigenvalues closest to some value, σ, of $Kx = \lambda Mx$, a transformation of the problem must be made. The Lanczos algorithm described in Section 4 is applicable only to $Ax = \lambda x$. Two transformations are available when K and M are symmetric and M is positive semi-definite. Each transformation in this section will be represented by a capital letter that has no other meaning. Transformation A, proposed by Ericsson and Ruhe, replaces A with $W^T(K - \sigma M)^{-1}W$ to yield,

$$W^T(K - \sigma M)^{-1}Wy = \nu y, \tag{27}$$

where $M = WW^T$, $y = W^Tx$, and $\lambda = \sigma + 1/\nu$ [ER80]. Transformation B, suggested by Nour-Omid, Parlett, Ericsson and Jensen, replaces A with $(K - \sigma M)^{-1}M$ and uses the M-inner product, because the operator is no longer symmetric [NOPEJ87]. Note that M does not form a true inner product because M is not positive definite. This semi-inner product is acceptable, however, because the only situation in this algorithm in which $x^TMx = 0$, for a non-trivial x, is when $\beta_{j+1} = 0$, which indicates exact convergence in the Lanczos algorithm. (Hereafter, the semi-inner product will be referred to as just an inner product). Equation 1 becomes

$$(K - \sigma M)^{-1}Mx = \nu x, \tag{28}$$

where the eigenvalues of the original system can be recovered via

$$\lambda = \sigma + 1/\nu. \tag{29}$$

Transformation B is a shifted inverted version of Equation 1 by virtue of the following steps. Substituting Equation 29 in Equation 1 yields

$$Kx - \sigma Mx = 1/\nu Mx. \tag{30}$$

Then, solving for x and multiplying by ν gives

$$\nu x = (K - \sigma M)^{-1}Mx. \tag{31}$$

1) Choose an initial vector, *guess*
2) $r_0 = (K - \sigma M)^{-1} M guess$ (Purge $n(B)$ from *guess*)
3) $p_1 = M r_0$
4) $\beta_1 = (r_0^T p_1)^{1/2}$
5) For $j = 1$, maximum number of iterations
6) Reorthogonalization phase
7) $q_j = r_{j-1}/\beta_j$
8) $p_j = p_j/\beta_j$
9) $(K - \sigma M) r_j = p_j$ (symmetric indefinite solve)
10) $r_j = r_j - q_{j-1}\beta_j$
11) $\alpha_j = r_j^T p_j$
12) $r_j = r_j - q_j \alpha_j$
13) $p_{j+1} = M r_j$
14) $\beta_{j+1} = (r_j^T p_{j+1})^{1/2}$
15) Compute the eigenvalues of T_j and the corresponding error bounds
16) Compute any converged Ritz vector
17) Halt if enough eigenvalues have been found
18) End of Loop

Figure 2: The Lanczos Algorithm Using Transformation B

Transformation B is superior to transformation A in both storage and operation count requirements [NOPEJ87]. Transformation B requires some modifications to the original algorithm, including the solution of the system $(K - \sigma M)x = y$ for x, and the use of the M-inner product. The vector p has also been added to the original Lanczos algorithm to hold the intermediate quantity Mr_j. In the initialization step, the initial guess vector is multiplied by B to ensure that r_0 is contained in $r(B)$, where $B = (K - \sigma M)^{-1} M$. The efficient implementation of these operations is described in later sections. The modified algorithm is shown in Figure 2 [NOPEJ87]. Reorthogonalization in step 6 is much the same as that described in Section 4, with the exception that the reorthogonalization is done in the M-inner product [NOPEJ87].

If the matrix M is singular, then $n(B)$ might not be null. The eigenvectors of Equation 1 have no components in $n(M)$ (also, $n(M) = n(B)$)

[NOPEJ87]. In exact arithmetic, if the starting vector, q_0, of the Lanczos process is restricted to $r(B)$, then all subsequent Lanczos vectors will be restricted to $r(B)$, because they are computed by multiplying by B. However, in finite precision arithmetic, roundoff error allows components of subsequent Lanczos vectors to be in $n(B)$; therefore, the Ritz vectors calculated from them will have components in $n(B)$ [NOPEJ87]. Purifying these Lanczos vectors is an expensive process, but a method exists that instead will inexpensively purify the calculated Ritz vectors. The vector w_i is computed, where w_i is a $j+1$-length vector whose first j components are calculated as,

$$w_i = (1/\theta_i)(T_j s_i), \tag{32}$$

and whose last component is

$$(s_i(j)\beta_{j+1})/\theta_i. \tag{33}$$

Equation 21 then becomes:

$$y_i = Q_{j+1} w_i, \tag{34}$$

where w_i has replaced s_i.

5.2 Transformations for Buckling

Transformations A and B are not applicable to the buckling problem because K_G can be indefinite. Transformation A fails because it requires the Choleski factorization of K_G. Transformation B fails because it requires the use of a K_G-inner product which would be an indefinite inner product and introduce complex values into the calculations. Transformation C, suggested for the buckling problem in [Ste89b], uses the operator $(K + \sigma K_G)^{-1} K$. The transformation $(K - \sigma K_G)$ is actually suggested in [Ste89b], but $(K + \sigma K_G)$ is preferred because it yields the correct inertia for the buckling problem (note that the inertia referred to here is not the inertia of a physical body, it is the definition of inertia used in Sylvester's inertia theorem relating to the number of positive, negative, and zero eigenvalues of a matrix). The inertia of $(K + \sigma K_G)$ reveals how many eigenvalues of Equation 3 are less than σ. Transformation C can be derived from Equation 3 by substituting $\sigma\nu/(\nu-1)$

for λ in Equation 3, multiplying each side by $(\nu - 1)$, rearranging terms, and finally, multiplying each side by $(K + \sigma K_G)^{-1}$ to yield

$$\nu x = (K + \sigma K_G)^{-1} K x. \tag{35}$$

To recover λ, use Equation 36.

$$\lambda = \sigma \nu / (\nu - 1). \tag{36}$$

Transformation C requires that σ be non-zero. The factorization of $(K + \sigma K_G)$ and the use of the K-inner product are necessary for transformation C.

In exact arithmetic, the convergence rate of the Lanczos algorithm is the same for transformations B and C when σ is fixed (B performs the same transformation on the spectrum as A). The Kaniel-Page-Saad theory is used to explain the convergence of eigenvalues when using the Lanczos algorithm and is now introduced to allow a comparison of the transformations and to show the effects of moving σ on the convergence rate [Saa80]. Three definitions that will be useful in this explanation are:

$$K_i^j = \prod_{m=1}^{i-1} (\theta_m^j - \nu_{inf}) / (\nu_m^j - \nu_i), \tag{37}$$

$$\gamma_i = 1 + 2(\nu_i - \nu_{i+1}) / (\nu_{i+1} - \nu_{inf}), \tag{38}$$

and the Chebyshev polynomial,

$$C_n(x) = 1/2((x + (x^2 - 1)^{1/2})^n + (x - (x^2 - 1)^{1/2})^n). \tag{39}$$

The bound on the difference between an eigenvalue of the T_j matrix, θ_i, and an eigenvalue of the transformed system, ν_i, at the jth step of the Lanczos algorithm is

$$0 \leq \nu_i - \theta_i^j \leq (\nu_i - \nu_{inf})(K_i^j tan\ \omega(x_i, r_0) / C_{j-i}(\gamma_i))^2, \quad \nu_i > \nu_{i+1}. \tag{40}$$

The $tan\ \omega(x_i, r_0)$ is determined by the angle between the eigenvector, x_i, associated with ν_i and the starting Lanczos vector, r_0. Because the angle between x_i and r_0 does not change during the Lanczos algorithm and because K_i^j does not vary greatly, the term that governs the rate of the convergence is $C_{j-i}(\gamma_i)$. As j increases, $C_{j-i}(\gamma_i)$ grows more quickly for large ϕ_i than for

Eigenvalue	order	$\sigma = 0$	$\sigma = 10$	$\sigma = 25$	$\sigma = 25.9$
		ϕ_1	ϕ_1	ϕ_1	ϕ_1
26	λ_1	0.2199	0.3214	4.2857	42.3442
30	λ_2	-	-	-	-
100	λ_{inf}	-	-	-	-

Figure 3: Effects of transformation on eigenvalue separation

small ϕ_i (ϕ_i is a term, $\mid (\nu_i - \nu_{i+1})/(\nu_{i+1} - \nu_{inf}) \mid$, obtained from the definition of γ). The ϕ_i reflect the separation of individual eigenvalues from their neighbors relative to the remaining width of the spectrum. Transformations are used to increase ϕ_i for the desired eigenvalues by transforming the spectrum such that the desired eigenvalues are well-separated from other eigenvalues. It can be shown that, if used with the same σ, transformations B and C have the same effect on the ϕ_i. However, moving σ closer to a desired eigenvalue, λ_i, increases the corresponding ϕ_i (and therefore speeds convergence of θ_1 to λ_i). The increase in ϕ_1 as σ is moved closer to λ_1 is shown in Figure 3. Thus, as σ is moved closer to λ_1 the convergence of θ_1 to λ_1 is speeded up.

Transformations B and C have the same effect on the convergence rates of the Lanczos process and C can be used in both the buckling and vibration problems, so the question arises "Why not use transformation C for both the buckling and vibration problems?" Although B and C have the same effect in exact arithmetic, they each yield different ν's for the same σ. In finite precision arithmetic, transformation C is inferior to transformation B when σ is small relative to the desired λ's. Although each transformation requires the solution of a linear system and the multiplication of a matrix by a vector, the distribution of the ν's for small σ in transformation C leads to large errors in the computation of the λ's. For small σ, the ν's of transformation C become close to 1 while the same effect is not seen when transformation B is used (note that the ϕ_i's in each case are identical). The ν's that result when using transformation C become increasingly close to 1 as σ is moved from 1.0 to 0.01, whereas the ν's that result when using transformation B show little change (this trend is shown in Figure 4). Because the Lanczos algorithm consists of the same calculations for each transformation, in finite precision arithmetic the algorithm computes perturbed values assumed to be

Eigen-value	order	$\sigma = 1.0$ ν for Trans. B	$\sigma = 1.0$ ν for Trans. C	$\sigma = 0.01$ ν for Trans. B	σ=0.01 ν for Trans. C
26	λ_1	0.04000	1.04000	0.03848	1.0003848
26	λ_1	0.04000	1.04000	0.03848	1.0003848
30	λ_2	0.03448	1.03448	0.03334	1.0003334
100	λ_{inf}	0.01010	1.01010	0.01000	1.0001000

Figure 4: Effects of transformation on eigenvalues

of the form, $(1+\epsilon)\nu$, instead of an exact ν. The effect of this perturbation on the computed λ's is the difference between the two transformations. Recall that in transformation B, $\lambda = \sigma + 1/\nu_B$, and that in transformation C, $\lambda = \sigma + \sigma/(\nu_C - 1)$ (the subscript on ν is introduced because the ν's are different in each transformation and these values are being compared). If ν_C is solved for in terms of ν_B, then

$$\nu_C = \sigma\nu_B + 1. \tag{41}$$

Let $\delta\lambda_B$ and $\delta\lambda_C$ denote the difference between the true λ and the λ computed using transformations B and C, respectively. These $\delta\lambda$'s are expressed in terms of perturbed ν's in the following equations:

$$\lambda + \delta\lambda_B = \sigma + 1/(1+\epsilon)\nu_B, \tag{42}$$

$$\lambda + \delta\lambda_C = \sigma + \sigma/((1+\epsilon)\nu_C - 1). \tag{43}$$

If Equation 41 is substituted into Equation 43, then

$$\lambda + \delta\lambda_C = \sigma + \sigma/(\sigma\nu_B + \epsilon\sigma\nu_B + \epsilon). \tag{44}$$

If the true λ is subtracted from each side of Equations 42 and 44, then

$$\delta\lambda_B = 1/(\nu_B + \epsilon\nu_B) - 1/\nu_B. \tag{45}$$

$$\delta\lambda_C = 1/(\nu_B + \epsilon\nu_B + \epsilon/\sigma) - 1/\nu_B. \tag{46}$$

Thus, the error in the computed λ for transformation C increases sharply as σ decreases. To show the increase in error for transformation C, the

errors in the two transformations (from Equations 45 and 46) are plotted in Figure 5 for $\lambda = 10$ and $\epsilon = 0.0001$. From this derivation and the graph of the functions, it is clear that transformation C should be avoided when σ is small compared to the desired λ.

From the previous discussion the conclusion can be drawn that transformation B is preferred to C *whenever possible.* However, as was pointed out previously, transformation B is not applicable to the buckling problem. Therefore, a new transformation, D, which transforms the eigenvalues in the same fashion as transformation B (when $\sigma = 0$) is introduced. Transformation D can be used with an indefinite K_G matrix but can only be used when σ is 0. Transformation D is derived from Equation 3 in the following steps [CW85]: first, substitute $1/\nu$ for λ and then multiply each side by $K^{-1}\nu$ to yield

$$\nu x = K^{-1}K_G x; \tag{47}$$

next, expand the implicit identity matrix in each side as $I = C^{-T}C^T$, where $K = CC^T$, let $y = C^T x$, and, finally, multiply each side by C^T to yield

$$\nu y = C^{-1}K_G C^{-T} y. \tag{48}$$

The operator for transformation D is $C^{-1}K_G C^{-T}$. This transformation requires the Choleski factorization of K and uses the standard inner product. The eigenvectors, x, must be recovered via the solution of a triangular linear system, using the foregoing equation for computing y. When an initial nonzero guess for σ exists, the method used in **LANZ** for solving the buckling problem uses transformation C exclusively; when an initial guess for σ isn't available, the method used begins by using transformation D with σ at 0, and then switches to transformation C when a shift of σ is needed (the use of shifting will be described in the next subsection). Thus, the use of transformation C with small σ is avoided, and, yet, the advantage resulting from shifting is maintained.

5.3 The Use of Shifts

An efficient algorithm for computing several eigenvalues requires that the shift, σ, be moved as close as possible to the eigenvalues that are being computed. The closer that σ is to an eigenvalue being computed, the faster the convergence to that eigenvalue. Ericcson and Ruhe describe a method for

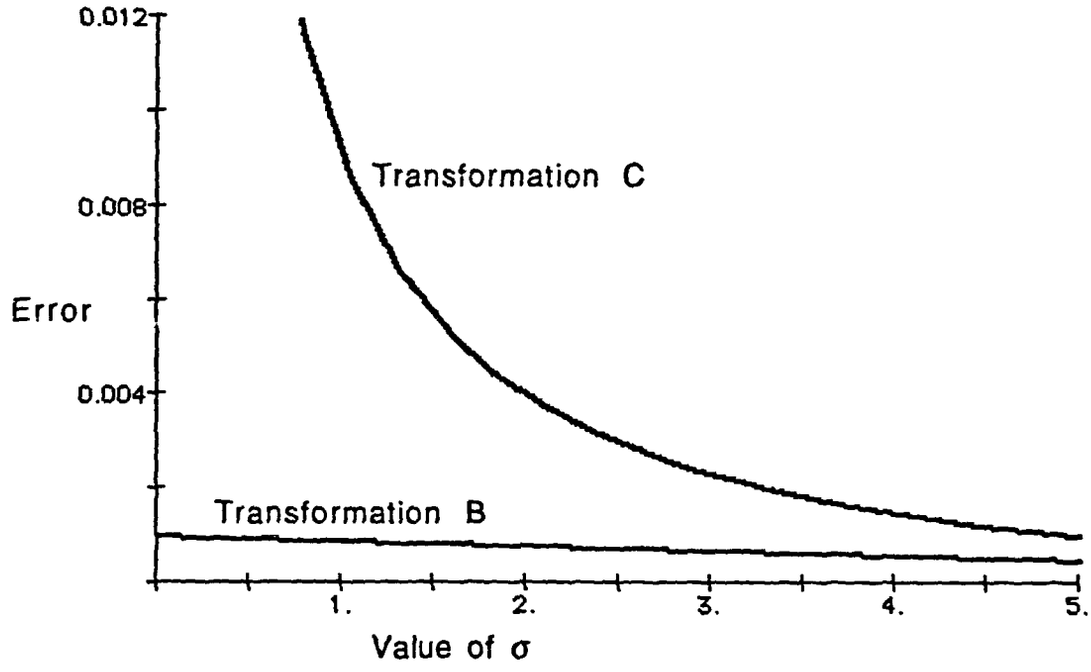

Figure 5: Errors in Transformations B and C

selecting shifts and deciding how many Lanczos steps to take at each shift [ER80]. The efficiency of the algorithm depends on how well these shifts are chosen, and how many Lanczos steps are taken at each shift. Normally, the most expensive step in the Lanczos process is the factorization of $(K - \sigma M)$, which must be done once for each shift. But if j becomes large, the calculation of an eigenvector, Equation 21, can be very expensive. In addition, many steps could be required to converge to an eigenvalue if the shift is far away from this eigenvalue, or if the eigenvalue is poorly separated from its neighbors. The method used by Ericcson and Ruhe first estimates that r eigenvalues will converge in j steps, where r is calculated based on the fact that the eigenvalues of Equation 1 are linearly distributed. A cost analysis of the algorithm is performed, and from this analysis, a determination of how many steps to take at a shift is made. Their choice of shift depends on the inertia calculation in the factorization step [ER80].

In the problems from the NASA structures testbed [Ste89a], the distribution of the eigenvalues is often anything but linear. This distribution makes the above estimates invalid and requires a different method for deciding how many steps to take at each shift. Instead of calculating the number of steps to take for a shift prior to execution, **LANZ** uses a dynamic criterion to decide when to stop working on a shift. Later, in Section 6, a cost analysis of the Lanczos algorithm shown in Figure 2 is given. This cost analysis is part of the basis for the dynamic shifting algorithm. The implementation of **LANZ**, however, uses execution timings, rather than a precalculated cost analysis, because the cost analysis is different for each architecture on which the implementation is run. These timings let **LANZ** know how long each Lanczos step takes and the cost of factorization at a new shift. In addition to the timing information, the estimated number of steps required for unconverged eigenvalues to converge is calculated. The step estimate is computed by tracking the eigenvalues of T_j (and the corresponding error bounds) throughout the execution of the Lanczos algorithm. The method for this tracking and computation of eigenvalues is described in Section 7. With estimates for the number of steps required for eigenvalues to converge and the time needed for a step (or new factorization) to execute, the decision to keep working on a shift or choose a new shift can be made efficiently.

The selection of a new shift depends on the information generated during the execution of **LANZ** on previous shifts and on inertia calculations at previous shifts. The inertia calculations are used to identify any eigenvalues

that were skipped over in previous steps, including eigenvalues of multiplicity greater than one. The estimated eigenvalues and their error bounds generated during the execution of Lanczos enable the selection of a new shift based on these estimates (if no eigenvalues were skipped in the previous run). Because convergence to an eigenvalue is faster if the shift, σ, is chosen to be close to that eigenvalue, **LANZ** seeks to choose a shift that is near to the desired unconverged eigenvalue. However, σ must not be chosen so close to an eigenvalue that the system $(K - \sigma M)x = y$ becomes very ill-conditioned. In the authors' experience, Lanczos's method generates approximations to all nearby eigenvalues, so that even if an eigenvalue is not converged to, an estimate along with an error bound for a nearby eigenvalue is generated. If the initial Lanczos vector is not deficient in the eigenvector corresponding to an eigenvalue, and if that eigenvalue is close to σ, then the Kaniel-Page-Saad theory shows that Lanczos's method will generate an estimate to that eigenvalue in a few steps [Kan66]. In practice, even if the initial Lanczos vector is deficient in the eigenvector, round-off error will quickly make that eigenvector a component of the Lanczos vectors [ER80].

When a new shift is chosen, the initial Lanczos vector is chosen to be a weighted linear combination of the Ritz vectors corresponding to the unconverged Ritz values of the previous shift, where the weights are chosen as the inverse of the error bounds of those Ritz values [PS79]. The number of Ritz vectors chosen is based on the number of eigenvectors still to be found and the number of Ritz vectors with "reasonable" error bounds. To examine the effect of using a linear combination of Ritz vectors rather than a random vector as the initial vector, several structural engineering problems (both buckling and vibration) were solved using both methods for selecting an initial vector (for an explanation of the problems used, see Section 9). The number of steps taken by each method to get the same number of eigenvalues is given in Figure 6 and from this it appears that using the linear combination of Ritz vectors is always as good or better than choosing a random vector.

To give the reader a clearer picture of the overall execution flow of **LANZ**, a flow chart is shown in Figure 7.

5.4 Selective Orthogonalization

The method used to maintain "semi-orthogonality" among the Lanczos vectors is a modification of selective orthogonalization as proposed by Parlett

Problem	Size	Random	Linear Combination
Mast (buckling)	$n = 1980$ $\beta = 58$	19 steps	19 steps
Mast (vibration) (diagonal M)	$n = 1980$ $\beta = 58$	10 steps	10 steps
Mast (vibration) (non-diagonal M)	$n = 1980$ $\beta = 58$	11 steps	8 steps
Cylinder (buckling)	$n = 7644$ $\beta = 385$	3 steps	2 steps

Figure 6: Methods for choosing initial vector

and Scott [PS79]. As described in Section 4, the Lanczos vectors do not lose orthogonality until a Ritz pair converges to an eigenpair of the system. At this point, significant components of the Ritz vector that have converged begin to creep into subsequent Lanczos vectors. Selective orthogonalization monitors the level of orthogonality between converged Ritz vectors and Lanczos vectors. If at step j, the orthogonality level between a converged Ritz vector and the Lanczos vector q_j is above a given threshold (Parlett and Scott suggest that $\epsilon^{1/2}$ be used), the Ritz vector is purged from both q_j and q_{j-1}.

Parlett and Scott use the following derivation to monitor the level of orthogonality [PS79]. In finite precision arithmetic the Lanczos recurrence relation is

$$\beta_{j+1}q_{j+1} = Bq_j - \alpha_j q_j - \beta_j q_{j-1} + f_i \tag{49}$$

where B is one of the operators described in Subsection 5.1 and f_i represents the roundoff error. The bound on $\| f_i \|$ is $c\epsilon \| B \|$ where c is a constant independent of j derived from B. If each side of Equation 49 is multiplied by y^T, where y is a Ritz vector computed from Equation 21, then

$$y^T\beta_{j+1}q_{j+1} = y^TBq_j - y^T\alpha_j q_j - y^T\beta_j q_{j-1} + y^T f_i. \tag{50}$$

Multiply each side of Equation 23 by s to yield

$$By = \theta y + r \tag{51}$$

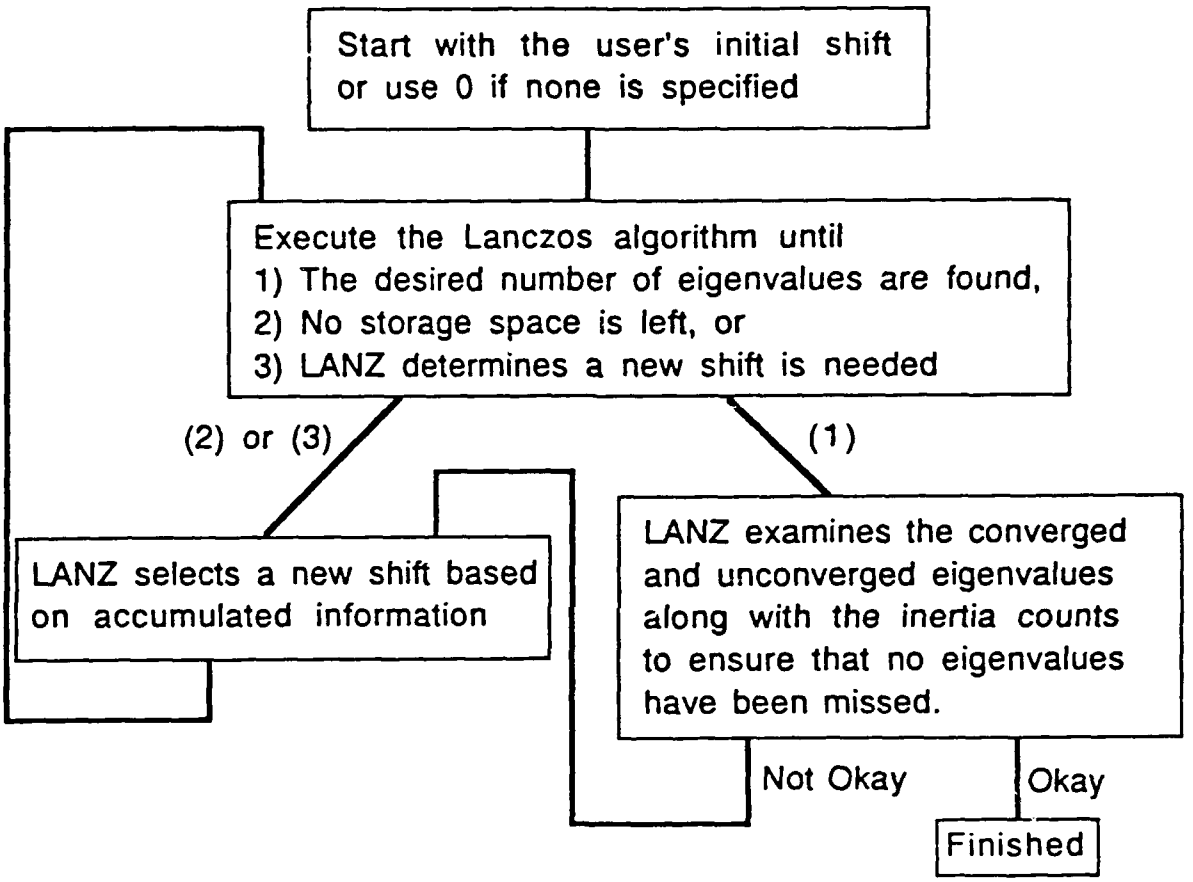

Figure 7: Execution flow of **LANZ**

where r is not r_j and will be discussed below. If the bound for $\| f_i \|$ and Equation 51 are substituted into Equation 50, Equation 52 results.

$$\beta_{j+1} y^T q_{j+1} = (\theta y^T + r^T - \alpha_j y^T) q_j - \beta_j y^T q_{j-1} + c\epsilon \| B \| \qquad (52)$$

Parlett and Scott assume that $r = q_{k+1}\beta_{ki}$ for some $k < j$, and therefore, that

$$| r^T q_j | \leq \beta_{ki} | q_{k+1}^T q_j | . \qquad (53)$$

They then state: 1) $| q_{k+1}^T q_j | \leq \epsilon^{1/2}$, because semi-orthogonality among the Lanczos vectors is maintained, and 2) if y is a converged Ritz vector, then β_{ji} is less than or equal to $\epsilon^{1/2} \| B \|$. Fact 2 is caused by the definition of a "good" Lanczos vector given in Section 4. If facts 1 and 2, along with $\tau_j = | y^T q_j |$, are substituted into Equation 52, then Equation 54 is derived.

$$\tau_{j+1} \leq (| \theta - \alpha_j | \tau_j + \beta_j \tau_{j-1} + \epsilon \| B \| + c\epsilon \| B \|)/\beta_{j+1} \qquad (54)$$

Because c and $\| B \|$ are small and not readily available, Parlett and Scott ignore these terms and derive the following recurrence relation for τ_{j+1}:

$$\tau_{j+1} \leq (| \theta - \alpha_j | \tau_j + \beta_j \tau_{j-1})/\beta_{j+1}. \qquad (55)$$

The values τ_0 and τ_1 are initialized to ϵ and whenever y is purged from q_j, τ_j is reset to ϵ. From this recurrence relation, the conclusion can be drawn that the Lanczos vectors should be orthogonalized in pairs.

This recurrence relation predicts the actual level of orthogonality very well in practice with two exceptions. The first problem occurs when calculating τ_{j+1} after y has been computed at step $j-1$ and purged from q_{j-1} and q_j. In this situation, a small increase in τ_{j+1} over τ_j and τ_{j-1} is expected. However, a large increase occurs. This increase is caused by the assumption on Parlett and Scott's part that $| q_{k+1}^T q_j | \leq \epsilon^{1/2}$, when in fact that equation only holds when $k < j-1$. The quantity, $| q_{k+1}^T q_j |$, is 1 when $k = j-1$. Thus, Equation 55 holds when $k < j-1$, but when $k = j-1$ Equation 55 becomes:

$$\tau_{j+1} \leq (| \theta - \alpha_j | \tau_j + \beta_j \tau_{j-1} + \beta_{j-1,i})/\beta_{j+1}. \qquad (56)$$

The second problem arises when using Equation 34 to compute y. In this case $r = \theta y + \beta_{j-1,i} q_j + \beta_{j-1,i} B q_j/\theta$ assuming y is computed at step $j-1$; therefore,

$$| r^T q_j | \leq \theta \tau_j + \beta_{j-1,i} + \beta_{j-1,i} \alpha_j/\theta. \qquad (57)$$

The recurrence relation used for τ_{j+1} then becomes:

$$\tau_{j+1} \leq (\mid \theta - \alpha_j \mid \tau_j + \beta_j \tau_{j-1} + \beta_{j-1,i} + \beta_{j-1,i}\alpha_j/\theta)/\beta_{j+1}. \quad (58)$$

If y has been computed at step $j-2$, then

$$\mid r^T q_j \mid \leq \theta\tau_j + \beta_{j-2,i}\epsilon^{1/2} + \beta_{j-2,i}\beta_j/\theta, \quad (59)$$

and, because the second term is very small, the recurrence relation for τ_{j+1} becomes

$$\tau_{j+1} \leq (\mid \theta - \alpha_j \mid \tau_j + \beta_j \tau_{j-1} + \beta_{j-2,i}\beta_j/\theta)/\beta_{j+1}. \quad (60)$$

5.5 Orthogonalization Methods

Selective orthogonalization and partial reorthogonalization are the two best known orthogonalization methods other than full reorthogonalization. Partial reorthogonalization monitors the orthogonality of q_j and q_{j+1} versus the other Lanczos vectors. Partial orthogonalization measures the orthogonality between q_j and q_k, ω_{jk}, via the recurrence relation defined in the following set of equations [Sim84]:

$$\omega_{kk} = 1 \quad for\ k = 1, ..., j, \quad (61)$$

$$\omega_{kk-1} = q_k^T q_{k-1} \quad for\ k = 2, ..., j, \quad (62)$$

$$\beta_{j+1}\omega_{j+1k} = \beta_{j+1}\omega_{jk+1} + (\alpha_k - \alpha_j)\omega_{jk} + \beta_k\omega_{jk-1} - \beta_j\omega_{jk-1} + q_j^T f_k - q_k^T f_j \quad (63)$$

$$and\ \omega_{jk+1} = \omega_{k+1j} \quad for\ 1 \leq k \leq j. \quad (64)$$

Simon has stated that the theoretical relationship between partial reorthogonalization and selective orthogonalization is not known [Sim84]. The following discussion explains the relationship between the two methods. If y_i^T on the left side of Equation 50 is expanded to $s_i^T Q_j^T$, and the resulting equation is divided by β_{j+1}, the right side becomes $\tau_{j+1,i}$, yielding

$$s_i^T Q_j^T q_{j+1} = \tau_{j+1,i} \quad (65)$$

From the recurrence relation for partial reorthogonalization, the product $Q_j^T q_{j+1}$ is the vector $\omega_{j+1,k}$, where k runs from 1 to j. Thus the relationship between the ω's and the τ's is governed by

$$s^T\omega_{j+1,k} = \tau_{j+1,k}, \quad where\ k = 1, ..., j. \quad (66)$$

This relationship in Equation 66 has also been observed in numerical experiments run by the authors. When a Ritz vector, y_i, is purged from q_{j+1} and therefore $\tau_{j+1,i}$ becomes small, the $\omega_{j+1,k}$'s for which the values of $s_{i,k}$ are the largest decrease significantly.

6 Execution-Time Cost Analysis

An analysis of the execution-time cost of the Lanczos algorithm when using transformation B is given in Appendix A. Because the costs for the other transformations are almost identical their cost will not be analyzed in this section. The analysis has two purposes: 1) to allow the computation of the tradeoff point between re-starting the Lanczos algorithm at a different shift and continuing with the current shift, and, 2) to allow analysis of performance on parallel and vector/parallel machines. Throughout the analysis, only the cost of floating point operations is included. The assumption is made for this analysis that the matrices have a band structure and that the Bunch-Kaufman method (or an LDL^T decomposition) is used for the solution of the linear systems. In order to simplify the analysis, the assumption is made that the bandwidth of K is greater than or equal to the bandwidth of M. This assumption has no effect on the analysis other than to avoid making some of the operation costs conditional on which matrix has the larger bandwidth. Much of this analysis does not take into account "end conditions," such as those that arise near the end of a factorization when less work needs to be done than in the middle of a factorization. Thus, some of the expressions are necessarily approximations.

Several observations regarding the shift tradeoff can be made from the cost analysis: 1) the single most expensive step in the algorithm is the factorization phase (2B) which is $O(n\mu_k^2)$, 2) the cost of the reorthogonalization phase increases as j increases because of the increasing number of "good" Ritz vectors to orthogonalize against, 3) the cost of computing a converged Ritz vector is based on j^2 and therefore increases rapidly as j increases, and 4) the cost of the rest of the operations in the program loop is not affected by growth in j (with exception of step 15 but this step is not costly enough to consider). To illustrate how the costs of the four operation groups *per Lanczos step* change, the number of floating point operations per step is plotted against j, the number of Lanczos steps, in Figure 8. The costs in the Figure 8 are from an actual **LANZ** run during which a new shift was selected beginning at step 22. These costs, of course, will differ for each problem. From the cost analysis and this graph it can be seen that a tradeoff exists between the benefits of a taking a new shift (smaller reorthogonalization and eigenvector computation cost as well as accelerated convergence to desired eigenvalues) and the benefit of continuing work on the current shift (avoiding

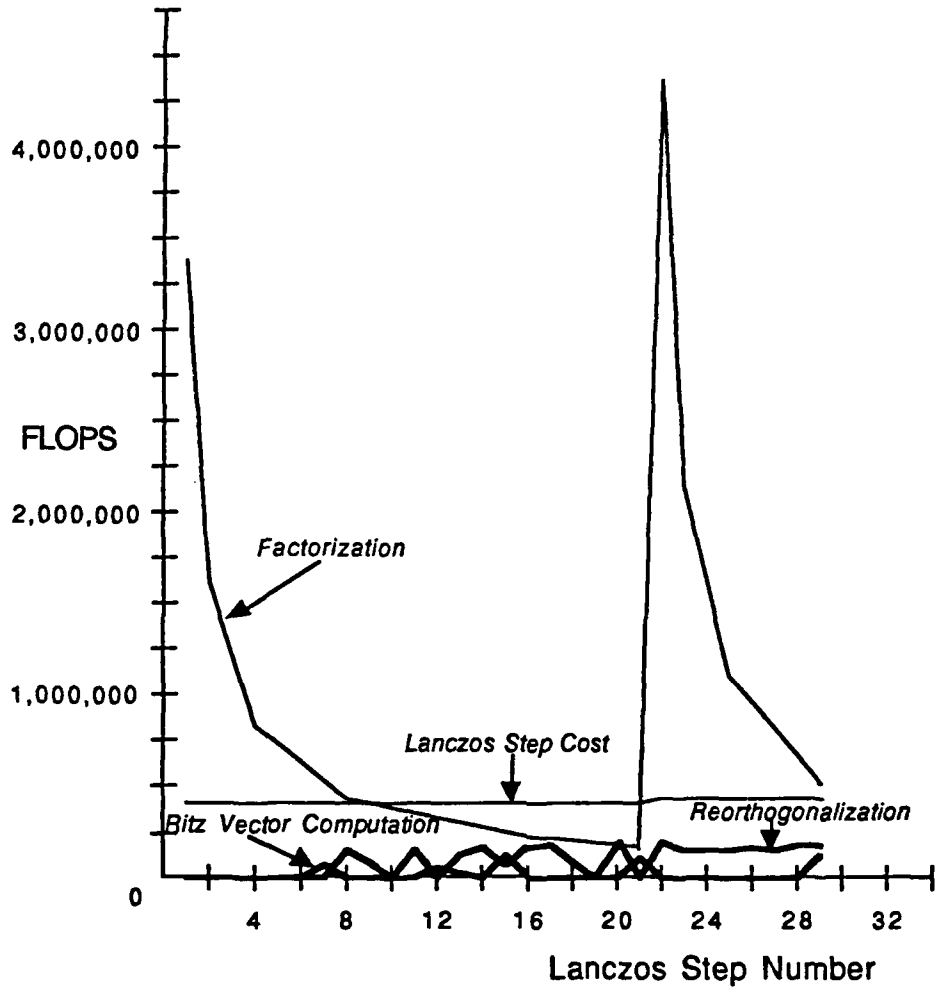

Figure 8: Operation costs plotted against Lanczos step number

the cost of refactorization).

The **LANZ** implementation uses actual timings of the various steps during the current run to analyze the tradeoff, rather than substituting values for the cost of various operations for the machine being used. The use of timings is simpler to implement and makes the code more portable.

7 Solution of the system $Ts = \theta s$

The size, j, of the tridiagonal system, $Ts = \theta s$, generated by the Lanczos algorithm is 1 at step 1 and increases by 1 at each Lanczos step. The size of T is usually very small compared to the size of the original problem, n. Therefore, the time used to solve the tridiagonal system does not greatly affect the sequential execution time of the **LANZ** algorithm. However, if the rest of the algorithm is parallelized, the solution of the tridiagonal system could well become a large factor in the parallel execution time. Parlett and Nour-Omid have proposed a method of tracking a small group of the eigenvalues of the T matrices as they are produced by the Lanczos algorithm. An inexpensive by-product of their method is the error bounds for the θ_i's [PNO85]. Their algorithm monitors the outermost θ's whose error bounds, β_{ji}, indicate that they will converge in the next 2 or 3 steps; it actually monitors 8 eigenvalues at a time. There are two phases: 1) the previous θ's and their error bounds are updated and any new θ's are detected, and 2) converged θ's are detected and removed from the data structure. This algorithm is not suitable for use by **LANZ** for 3 reasons: 1) it is not easily parallelizable, 2) it does not track an eigenvalue for many steps to get a convergence rate estimate, and 3) in tests run by the authors, it often failed.

The authors have developed a new solution method that is: 1) inherently parallel, 2) tracks all the eigenvalues of T from step to step, and 3) has been used successfully with **LANZ** to solve real structures problems. The method uses information from step $j-1$ to solve for all the eigenvalues and their error bounds at step j. It uses Cauchy's interlace theorem, shown in Equation 67, to find ranges for the all the eigenvalues (except the outermost eigenvalues) of T_j from the eigenvalues of T_{j-1}.

$$\theta_1^{j+1} < \theta_1^j < \theta_2^{j+1} < \theta_2^j \ldots \theta_j^{j+1} < \theta_j^j < \theta_{j+1}^{j+1} \tag{67}$$

Cauchy's interlace theorem states that the eigenvalues of T_j interlace those of T_{j+1} [Par80]. In addition to the interlace property, the error bounds, β_{ji}, from the previous step can be used to provide even smaller ranges for some eigenvalues. If good error bounds, β_{ji}, are not available for the outer eigenvalues (the interlace property only gives a starting point for these eigenvalues), they can be found by extending an interval from the previous extreme eigenvalue. However, a property of the Lanczos algorithm is that the extreme

eigenvalues are usually the first to stabilize. The algorithm for the method just described is given in Figure 9. For simplicity, the algorithm does not show the code for handling extreme eigenvalues. The algorithm requires two subroutines, the root finder described below and a function, numless, that uses spectrum slicing to determine the number of eigenvalues of T_j less than a value. Details of how to efficiently implement these subroutines are given by Parlett and Nour-Omid [PNO85].

A root finding method, such as bisection or Newton's method, can be used to find the eigenvalues in the ranges given by the algorithm in Figure 9 [PNO85]. Newton's method is preferred for its fast convergence and because it generates the jth element in s_i as a by-product, which allows for the inexpensive computation of the error bound for θ_i [PNO85]. For safety's sake, the Newton root finder is protected by a bisection root finder to ensure that Newton's method converges to the desired root. If the Ritz vector corresponding to a particular eigenvalue, θ_i, needs to be computed, inverse iteration can be used to compute s_i. Because the calculation of every eigenvalue is independent, this algorithm is inherently parallel. In order to save time, it may be beneficial to keep track of which eigenvalues of T have stabilized, as these do not need to be recomputed. The major difficulty in parallelizing this algorithm appears to be load balancing; it will take different numbers of Newton iterations to find each eigenvalue, and only occasionally will inverse iteration be necessary.

The algorithm developed by the authors for solving the tridiagonal system also tracks the eigenvalues of T_j from step to step. This tracking is necessary for two reasons. First, selective orthogonalization requires the computation of the Ritz vectors corresponding to the eigenvalues of T_j that become "good" (as defined in Section 4) at step j. Those Ritz vectors can be used from step $j+1$ until the end of the Lanczos run if the eigenvalues in T_{j+1} (and subsequent T_i's) that correspond to the eigenvalues in T_j can be identified. Second, the rate of convergence of a particular eigenvalue is predicted by tracking its convergence over several steps (the use of the convergence rate was described in a previous section).

```
bounded[i] = 0 , for i = 1, j
do i = 1, j − 1
   if ((2*β_ji < θ_i − θ_{i−1}) and (2*β_ji < θ_{i+1} − θ_i)) then
     probe = θ_i + β_ji
     less = numless(probe)
     if (less = i) then
       bounded[i] = i
     else /* i and i + 1 are the only values numless will return, if
       it returns something else, a grave error has occurred */
       bounded[i + 1] = i
     endif
   endif
enddo
do i = 1, j
   if (bounded[i] = 0) then
     leftbound = θ_{i−1}
     rightbound = θ_i
     newtonroot(leftbound,rightbound,newθ_i,newβ_ji)
   else if (bound[i] = i) then
     leftbound = θ_i − β_ji
     rightbound = θ_i
     newtonroot(leftbound,rightbound,newθ_i,newβ_ji)
   else if (bound[i] = i − 1) then
     leftbound = θ_{i−1}
     rightbound = θ_{i−1} + β_{ji−1}
     newtonroot(leftbound,rightbound,newθ_i,newβ_ji)
   endif
enddo
```

Figure 9: Tridiagonal Eigenvalue Solver

8 The Solution of the system $(K - \sigma M)x = y$

The solution to the possibly indefinite system,

$$(K - \sigma M)x = y, \tag{68}$$

is normally the most time-consuming step in the Lanczos algorithm for transformations A, B, and C (unless there are very few non-zero elements in $(K - \sigma M)$). Therefore, it makes sense to try to optimize this step of the algorithm as much as possible. Two approaches can be taken to solving this system: 1) the use of direct solution methods, or 2) the use of iterative solution methods. Because the problems under consideration can be very ill-conditioned, the use of iterative methods has been avoided.

Because this paper is focused on the problem in which K and M are banded, the discussion in this section is limited to the banded case. In the vibration problem, because K is positive definite and M is semi-positive definite, if $\sigma \leq 0$, then the system in Equation 68 is positive definite. In the buckling problem, because K is positive definite, if $\sigma = 0$, then only K must be factored because transformation D is used. Because K and M are always symmetric, Choleski's method can be used to solve these systems. Choleski's method is the direct method of choice for this class of banded linear systems because it is stable and results in no destruction of the bandwidth [BKP76]. Choleski's method is used by **LANZ** for the vibration problem whenever $\sigma \leq 0$ and in the buckling problem whenever $\sigma = 0$.

The system in Equation 68 can be indefinite whenever $\sigma > 0$ in the vibration problem and may be indefinite in the buckling problem when σ is non-zero. When the system is indefinite, Choleski factorization will fail because a square root of a negative number will be taken, and the LDL^T decomposition is not stable because the growth of elements in L cannot be bounded *a priori* [Wil65]. The methods of choice for factoring a *full* symmetric indefinite matrix are the Bunch-Kaufman method and Aasen's method [BG76]. It was believed that both methods, however, would destroy the structure of a banded system and not be competitive with Gaussian elimination with partial pivoting, which does not destroy the band structure but ignores symmetry [BK77]. To address this, the authors have developed a new method of implementing the Bunch-Kaufman algorithm which is the method of choice for factoring symmetric indefinite banded systems when

the systems have only a few negative eigenvalues [JP89]. This is exactly the case which arises when moving the shift in search of the lowest eigenvalues of Equation 1 in the vibration problem and is often the case in the buckling problem as well. The modified algorithm takes full advantage of the symmetry of the system, unlike Gaussian elimination, and is therefore faster to execute and takes less storage space. **LANZ** uses this algorithm whenever the system can be indefinite. As an additional benefit, the inertia of the system can be obtained virtually for free [BK77].

Regardless of which factorization method is used, the system is only factored once for each σ. After the factorization has taken place, each time the solution to Equation 68 is required, only back and forward triangular solutions (and a diagonal solution in the Bunch-Kaufman case) must be executed.

9 Performance Analysis

9.1 Vectorization

From the analysis in Appendix A it appears that vectorizing **LANZ** would result in significant speedup of the solution procedure. The **LANZ** code was compiled using the Convex Vectorizing Fortran compiler [Cor87]. The code was executed using double precision on a Convex 220 in both vector and scalar modes. Seven free vibration problems and five buckling problems of varying sizes from the NASA Langley testbed were run. The problems consisted of varying sizes of three different structures. The first structure is a thin circular, cylindrical shell simply supported along its edges. The buckling eigenvalues for this structure are closely spaced and present a challenge for eigensolvers. The actual finite element model only needs to model a small rectangle of the cylinder to correctly simulate the behavior of the structure. A plot of the entire cylinder that shows the 15 degree rectangle of the cylinder that is modeled is given Figure 20 of Appendix A. The two lowest buckling modes for an axially-compressed cylinder are are also plotted in Figure 21 of Appendix A. The second structure is a composite (graphite-epoxy) blade-stiffened panel with a discontinuous center stiffener. The finite element model for this structure is shown in Figure 22 of Appendix A. The third structure is a model of a deployable space mast constructed at NASA Langley Research Center. A picture of the deployable mast along with a plot of the finite element model is shown in Figure 23 of Appendix A. Descriptions of the first two structures can be found in [Ste89a]. A description of the deployable mast can be found in [HWHB86]. In every problem, at least ten eigenpairs were found. All times in this section are given in seconds. In each problem, n will refer to the number of equations, and β will refer to the semi-bandwidth of the K matrix. The execution times for the vibration problem with a diagonal mass matrix are given in Figure 10 where a speedup due to vectorization of up to 7.83 is shown. Speedups of up to 7.30 for the vibration problem with a consistent mass matrix are given in Figure 11. For the buckling problem, speedups of up to 7.79 can be observed in Figure 12. These speedups are similar to the speedups obtained by other linear algebra applications on the Convex 220. From these comparisons the conclusion can be drawn that significant speedup of the solution procedures due to vectorization can be achieved.

Problem	Size	Vector (seconds)	Scalar (seconds)	Speedup Factor
Mast	$n = 1980$ $\beta = 58$	2.80	9.28	3.31
Cylinder	$n = 216$ $\beta = 65$	0.40	0.92	2.30
Cylinder	$n = 1824$ $\beta = 185$	3.64	20.42	5.61
Cylinder	$n = 7644$ $\beta = 385$	34.77	256.47	7.38
Cylinder	$n = 12054$ $\beta = 485$	75.79	593.48	7.83
Panel	$n = 477$ $\beta = 142$	1.04	3.69	3.55
Panel	$n = 2193$ $\beta = 237$	6.04	35.49	5.88

Figure 10: Vectorization Results for the Vibration Problem with a Diagonal Mass Matrix

Problem	Size	Vector (seconds)	Scalar (seconds)	Speedup Factor
Mast	$n = 1980$ $\beta = 58$	4.05	10.52	2.60
Cylinder	$n = 216$ $\beta = 65$	0.44	0.96	2.18
Cylinder	$n = 1824$ $\beta = 185$	4.69	22.12	4.72
Cylinder	$n = 7644$ $\beta = 385$	39.26	263.73	6.72
Cylinder	$n = 12054$ $\beta = 485$	82.88	605.30	7.30
Panel	$n = 477$ $\beta = 142$	1.74	5.18	2.98
Panel	$n = 2193$ $\beta = 237$	10.61	43.82	4.13

Figure 11: Vectorization Results for the Vibration Problem with a Consistent Mass Matrix

Problem	Size	Vector (seconds)	Scalar (seconds)	Speedup Factor
Mast	$n = 1980$ $\beta = 58$	5.16	14.72	2.85
Cylinder	$n = 216$ $\beta = 65$	0.43	1.01	2.35
Cylinder	$n = 1824$ $\beta = 185$	5.18	27.54	5.32
Cylinder	$n = 7644$ $\beta = 385$	70.32	510.75	7.26
Cylinder	$n = 12054$ $\beta = 485$	150.57	1172.39	7.79

Figure 12: Vectorization Results for the Buckling Problem

9.2 Comparison With Subspace Iteration

The claim was made in Section 3 that Lanczos's method is significantly faster than subspace iteration. The results presented in this section support this claim. The **LANZ** code was compared with the EIG2 processor from the NASA Langley testbed code. The EIG2 processor uses the subspace iteration method [Ste89b]. Both codes were compiled and executed as in Subsection 9.1. The same problems that were solved in 9.1 were used for this comparison in which the the lowest ten eigenvalues were sought. Both programs were able to find the lowest ten eigenvalues in every case, although EIG2 took an unusually large number of iterations (over three times the recommended maximum) to find them in the Mast case for both the buckling and free vibration problems. The Mast problem has a difficult distribution of eigenvalues, and the **LANZ** code makes use of shifting to quickly find the eigenvalues. Both codes were directed to find the eigenvalues to a relative accuracy of 10^{-4}. However, the subspace iteration code used an accuracy measure which was more lax than that used in the **LANZ** code. The measure used in the subspace code,

$$(\lambda_i^{k+1} - \lambda_i^k)/\lambda_i^{k+1}, \tag{69}$$

where k is the iteration number, is only a check to determine whether an eigenvalue has stabilized relative to itself. In the **LANZ** code

$$(\parallel Ky_i - \theta_i My_i \parallel)/\theta_i \tag{70}$$

is used to check the relative accuracy of the combination of the eigenvalue and the eigenvector. Therefore, the **LANZ** code is at a disadvantage to the subspace code in this comparison because the eigenpairs are computed to greater accuracy than in the subspace iteration code.

When the results comparing the two codes are given, two times are reported for **LANZ**: the processing time required by the code and the total of the system and processing time required by the code. The two times are given because the EIG2 processor can only report its execution time as the total of system and processing time. For the free vibration problem with a diagonal mass matrix, **LANZ** is shown in Figure 13 to be about 7 to 14 times faster than subspace iteration. In Figure 14, **LANZ** is shown to be about 7 to 26 times faster than subspace iteration for the vibration problem

Problem	Size	**LANZ** Program (seconds)	**LANZ** Total (seconds)	Subspace Iteration Total (seconds)	Ratio
Mast	$n = 1980$ $\beta = 58$	2.80	2.85	29.40	10.32
Cylinder	$n = 216$ $\beta = 65$	0.40	0.41	5.70	13.90
Cylinder	$n = 1824$ $\beta = 185$	3.64	3.73	46.40	12.44
Cylinder	$n = 7644$ $\beta = 385$	34.77	35.18	313.80	8.92
Cylinder	$n = 12054$ $\beta = 485$	75.79	76.65	541.50	7.06
Panel	$n = 477$ $\beta = 142$	1.04	1.07	12.30	11.50
Panel	$n = 2193$ $\beta = 237$	6.04	6.17	82.30	13.34

Figure 13: **LANZ** vs. Subspace Iteration: Vibration Problem with Diagonal Mass Matrix

Problem	Size	**LANZ** Program (seconds)	**LANZ** Total (seconds)	Subspace Iteration Total (seconds)	Ratio
Mast	$n = 1980$ $\beta = 58$	4.05	4.08	107.60	26.37
Cylinder	$n = 216$ $\beta = 65$	0.44	0.44	5.90	13.41
Cylinder	$n = 1824$ $\beta = 185$	4.69	4.77	51.60	10.82
Cylinder	$n = 7644$ $\beta = 385$	39.26	39.74	357.10	8.99
Cylinder	$n = 12054$ $\beta = 485$	82.88	83.80	585.10	6.98
Panel	$n = 477$ $\beta = 142$	1.74	1.77	20.50	11.58
Panel	$n = 2193$ $\beta = 237$	10.61	10.72	109.80	10.24

Figure 14: **LANZ** vs. Subspace Iteration: Vibration Problem with Consistent Mass Matrix

Problem	Size	**LANZ** Program (seconds)	**LANZ** Total (seconds)	Subspace Iteration Total (seconds)	Ratio
Mast	$n = 1980$ $\beta = 58$	5.16	5.22	108.9C	20.86
Cylinder	$n = 216$ $\beta = 65$	0.43	0.44	5.60	12.73
Cylinder	$n = 1824$ $\beta = 185$	5.18	5.30	92.80	17.51
Cylinder	$n = 7644$ $\beta = 385$	70.32	70.84	523.90	7.40
Cylinder	$n = 12054$ $\beta = 485$	150.57	151.44	992.30	6.55

Figure 15: **LANZ** vs. Subspace Iteration: Buckling Problem

with a consistent mass matrix. **LANZ** is shown to be about 6 to 21 times faster than subspace iteration for the buckling problem in Figure 15.

LANZ's advantage over subspace iteration appears to be diminishing as the problem sizes increase because the factorization of the matrices takes a larger proportion of the time as the matrix size increases. Because each code could use the same factorization technique, the time spent in factorization distorts the advantage that **LANZ** holds over subspace iteration. To more clearly illustrate the advantage of **LANZ** over subspace iteration, the time for factorizing $(K - \sigma M)$ was removed from the results in Figures 13, 14, and 15. Only the totals of system and processing time were accessible when computing the modified times. Although the time for triangular linear system solutions (the backward, forward, and diagonal linear solutions required at each step) is still included, the modified times will give the reader a better comparison of the time spent in the eigensolving routines. In Figure 16, **LANZ** now shows an advantage of up to 47.18 for the vibration problem with a diagonal mass matrix. For the vibration problem with a consistent mass matrix, a speedup of up to 31.31 can be observed in Figure 17. A speedup for the buckling problem of up to 23.64 is shown in Figure 18. In Figures 16 and 17 the **LANZ** code used only one factorization per problem except for

Problem	Size	**LANZ** (seconds)	Subspace Iteration (seconds)	Ratio
Mast	$n = 1980$ $\beta = 58$	2.13	121.80	47.18
Cylinder	$n = 216$ $\beta = 65$	0.36	5.30	14.72
Cylinder	$n = 1824$ $\beta = 185$	2.07	39.20	18.94
Cylinder	$n = 7644$ $\beta = 385$	12.65	235.50	18.62
Cylinder	$n = 12054$ $\beta = 485$	23.45	362.00	15.44
Panel	$n = 477$ $\beta = 142$	0.87	11.50	13.22
Panel	$n = 2193$ $\beta = 237$	3.63	71.70	19.75

Figure 16: Comparison without Factorization: Vibration Problem with a Diagonal Mass Matrix

the mast problem where two factorizations were required for ten eigenvalues to converge. In Figure 18 the **LANZ** code used only one factorization per problem to converge to ten eigenvalues except in the two large cylinder problems and the mast problem, where two factorization were required.

9.3 Performance Benefits of Tracking Eigenvalues

The value of tracking the eigenvalues will now be shown. In Section 7 an algorithm for tracking and computing the eigenvalues of T_j is given. The code was run on a Convex C-1 computer for five free vibration problems from the NASA Langley testbed. Ten eigenvalues were sought for each problem. To assess the benefits of the tracking algorithm, the code was run with the tracking algorithm first turned on and then turned off. The M matrices in this experiment are diagonal; however, the benefits would be even greater for non-diagonal M matrices. Reductions in execution time of up to 23

Problem	Size	**LANZ** (seconds)	Subspace Iteration (seconds)	Ratio
Mast	$n = 1980$ $\beta = 58$	3.36	105.20	31.31
Cylinder	$n = 216$ $\beta = 65$	0.39	5.50	14.10
Cylinder	$n = 1824$ $\beta = 185$	3.11	43.90	14.16
Cylinder	$n = 7644$ $\beta = 385$	17.21	284.60	16.54
Cylinder	$n = 12054$ $\beta = 485$	30.60	407.80	13.33
Panel	$n = 477$ $\beta = 142$	1.57	19.80	12.61
Panel	$n = 2193$ $\beta = 237$	8.18	99.20	12.73

Figure 17: *Comparison without Factorization:* Vibration Problem with a Consistent Mass Matrix

Problem	Size	**LANZ** (seconds)	Subspace Iteration (seconds)	Ratio
Mast	$n = 1980$ $\beta = 58$	4.50	106.40	23.64
Cylinder	$n = 216$ $\beta = 65$	0.39	5.20	13.33
Cylinder	$n = 1824$ $\beta = 185$	3.64	86.00	23.63
Cylinder	$n = 7644$ $\beta = 385$	25.78	445.90	17.30
Cylinder	$n = 12054$ $\beta = 485$	45.04	808.70	17.96

Figure 18: Comparison without Factorization: Buckling Problem

Problem	Tracking	No Tracking
$n = 486$ $\beta = 16$	1.440	1.870
$n = 476$ $\beta = 117$	8.540	8.880
$n = 1980$ $\beta = 58$	25.070	26.210
$n = 1824$ $\beta = 239$	36.990	40.150
$n = 3548$ $\beta = 259$	82.920	88.610

Figure 19: Execution time with and without tracking

percent are shown in Figure 19. The data in Figure 19 are from a version of the program that existed prior to changes made in early 1989. The current version of the program will not work with the tracking algorithm turned off. The gain in execution time would actually be more marked if the the tracking algorithm could be turned off in the current version because other parts of the **LANZ** algorithm that aren't affected by the tracking algorithm have been optimized.

9.4 Multiple Eigenvalues

Although the test problems from NASA Langley had no low eigenvalues of multiplicity greater than one, some of the eigenvalues in the Mast case were very closely clustered. However, the performance of the algorithm with exact multiple eigenvalues is of interest. Therefore, diagonal test matrices with multiple eigenvalues were constructed to test whether **LANZ** would reveal their presence. In these test cases, the correct number of copies of each eigenvalue were found by **LANZ**.

10 Concluding Remarks

10.1 Conclusions

For the large, generalized eigenvalue problem arising from two structural engineering applications, the vibration and buckling problems, the **LANZ** algorithm was shown to be superior to the subspace iteration method. Results from several structural engineering problems were given to support this claim. **LANZ** is based on the Lanczos algorithm and makes use of spectral transformations, dynamic movement of a shift, and a modified version of selective reorthogonalization to quickly converge to desired eigenpairs. The dynamic shift-moving algorithm used by **LANZ** was described. The shifting-moving algorithm is based on a cost analysis of the Lanczos algorithm with spectral transformations and selective reorthogonalizations. A parallel algorithm for efficiently solving the tridiagonal matrices that arise when using Lanczos's method was also given.

10.2 Future Work

LANZ has been shown to perform well on vector machines, an important class of scientific computing machines. These classes show the most promise for solving very large problems. The next step is to shown that **LANZ** will perform well on parallel and vector/parallel computers. An examination of the **LANZ** algorithm based on the analysis in Section 6 is the logical first step in determining a strategy for parallelizing **LANZ**. A possible next step is to use the Force programming language to parallelize the code [Jor87]. This language allows parallel loops to be easily expressed and can be used on several different shared-memory computers. The Force has been shown to be a good language for parallel linear algebra applications [JPV89]. The outlined approach would most likely provide a good barometer with which to assess the performance of **LANZ** on parallel machines.

Acknowledgements The authors would like to thank Dr. Axel Ruhe for several comments that led to the improvement of Section 5. The authors would also like to thank Dr. Robert M. Jones whose comments were helpful throughout the paper and especially improved Section 2.

References

[Bat82] K. Bathe. *Finite Element Procedures in Engineering Analysis.* Prentice-Hall, Englewood Clifs, NJ, 1982.

[BG76] Victor Barwell and Alan George. "A Comparison of Algorithms for Solving Symmetric Indefinite Systems of Linear Equations". *ACM Transactions on Mathematical Software*, 2(3):242–251, September 1976.

[BH87] Charles P. Blankenship and Robert J. Hayduk. "Potential Supercomputer Needs for Structural Analysis". Presentation at the Second International Conference on Supercomputing, May 3-8 1987. Santa Clara, CA.

[BK77] James R. Bunch and Linda Kaufman. "Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems". *Mathematics of Computation*, 31(137):163–179, January 1977.

[BKP76] J. Bunch, L. Kaufman, and B. Parlett. "Decomposition of a Symmetric Matrix". *Numerische Mathematik*, 27:95–109, 1976.

[Cor87] Convex Corporation. CONVEX FORTRAN User's Guide, 1987. Dallas, TX.

[CW85] Jane K. Cullum and Ralph A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Vol I Theory.* Birkhauser, Boston, MA, 1985.

[ER80] Thomas Ericsson and Axel Ruhe. "The Spectral Transformation Lanczos Method for the Numerical Solution of Large Sparse Generalized Symmetric Eigenvalue Problems". *Mathematics of Computation*, 35(152):1251–1268, October 1980.

[GL83] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 1983.

[GU77] G. H. Golub and R. Underwood. "The Block Lanczos Method for Computing Eigenvalues". In J. Rice, editor, *Mathematical Software III*, pages 361–377. Academic Press, New York, 1977.

[HWHB86] L. G. Horta, J. L. Walsh, G. C. Horner, and J. P. Bailey. "Analysis and Simulation of the MAST (COFS-1 Flight Hardware)". *NASA CP 2447, Part 1*, pages 515–532, Nov. 18-21, 1986.

[Jon89] Robert M. Jones. Personal communciation, 1989. Virginia Polytechnic Institute, Blacksburg, VA.

[Jor87] Harry Jordan. "The Force". Computer systems design group, University of Colorado, 1987.

[JPar] Mark T. Jones and Merrell L. Patrick. "Bunch-Kaufman Factorization for Real Symmetric Indefinite Banded Matrices". Technical report, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, 1989 to appear.

[JPV89] Mark T. Jones, Merrell L. Patrick, and Robert G. Voigt. "Language Comparison for Scientific Computing on MIMD Architectures". Report no. 89-6, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, 1989.

[Kan66] Shmuel Kaniel. "Estimates for Some Computational Techniques in Linear Algebra". *Mathematics of Computation*, 20:369–378, 1966.

[Kni89] Norman Knight. Personal communciation, 1989. Computational Structural Mechanics, NASA Langley Reseach Center, Hampton, VA.

[Lan50] Cornelius Lanczos. "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators". *Journal of Research of the National Bureau of Standards*, 45(4):255–282, October 1950.

[Ma88] Shing Chong Ma. "A Parallel Algorithm Based on the Sturm Sequence Solution Method for the Generalized Eigenproblem, $Ax = \lambda Bx$". Master's thesis, Department of Computer Science, Duke University, 1988.

[Mah87] Umesh Mahajan. "Parallel Subspace Iteration for Solving the Generalized Eigenvalue Problem". Master's thesis, Department of Computer Science, Duke University, 1987.

[NOC85] Bahram Nour-Omid and Ray W. Clough. "Block Lanczos Method for Dynamic Analysis of Structures". *Earthquake Engineering and Structural Dynamics*, 13:271–275, 1985.

[NOPEJ87] Bahram Nour-Omid, Beresford N. Parlett, Thomas Ericsson, and Paul S. Jensen. "How to Implement the Spectral Transformation". *Mathematics of Computation*, 48(178):663–673, April 1987.

[NOPT83] Bahram Nour-Omid, Beresford N. Parlett, and Robert L. Taylor. "Lanczos Versus Subspace Iteration For Solution of Eigenvalue Problems". *International Journal for Numerical Methods in Engineering*, 19:859–871, 1983.

[Pai71] C. C. Paige. *The Computations of Eigenvalues and Eigenvectors of Very Large Sparse Matrices.* PhD thesis, Institute of Computer Science, University of London, 1971.

[Par80] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[PNO85] Beresford N. Parlett and Bahram Nour-Omid. "The Use of a Refined Error Bound When Updating Eigenvalues of Tridiagonals". *Linear Algebra and its Applications*, 68:179–219, 1985.

[PS79] B. N. Parlett and D. S. Scott. "Lanczos Algorithm With Selective Orthogonalization". *Mathematics of Computation*, 33:217–238, 1979.

[Ruh74] Axel Ruhe. "SOR-Methods for the Eigenvalue Problem with Large Sparse Matrices". *Mathematics of Computation*, 28(127):695–710, July 1974.

[Ruh89] Axel Ruhe. Personal communciation, 1989. Department of Numerical Analysis, University of Umea, Umea, Sweden.

[Saa80] Y. Saad. "On The Rates of Convergence of the Lanczos and the Block-Lanczos Methods." *SIAM Journal of Numerical Analysis*, 17(5):687-706, October 1980.

[Sch74] H. R. Schwarz. "The Method of Coordinate Overrelaxation for $(A - \lambda B)x = 0$". *Numerische Mathematik*, 23:135-151, 1974.

[Sch89] H. R. Schwarz. "Rayleigh Quotient Minimization with Preconditioning". *Journal of Computational Physics, 81:53-69, 1989.*

[Sim84] Horst D. Simon. "The Lanczos Algorithm With Partial Reorthogonalization". *Mathematics of Computation*, 42:(165):115-142, January 1984.

[Ste89a] Caroline B. Stewart. "The Computational Structural Mechanics Testbed Procedure Manual", 1989. Computational Structural Mechanics, NASA Langley Research Center, NASA TM-100646, Hampton, VA.

[Ste89b] Caroline B. Stewart. "The Computational Structural Mechanics Testbed User's Manual", 1989. Computational Structural Mechanics, NASA Langley Research Center, NASA TM-100644, Hampton, VA.

[SW82] Ahmed H. Sameh and John A. Wisniewski. "A Trace Minimization Algorithm for the Generalized Eigenvalue Problem". *SIAM Journal of Numerical Analysis, 19(6):1243-1259, December 1982.*

[Szy83] Daniel B. Szyld. *A Two-level Iterative Method for Large Sparse Generalized Eigenvalue Calculations.* Ph.D thesis, Department of Mathematics, New York University, 1983.

[Wil65] J. H. Wilkinson. *The Algebraic Eigenvalue Problem* Oxford University, Oxford, 1965.

A Sequential and Vector Cost Analysis

A step-by-step cost analysis for the Lanczos algorithm when using transformation B (shown in Figure 2) is given below for sequential and vector machines.

Definitions:

μ_K: The semi-bandwidth of the K matrix.

μ_M: The semi-bandwidth of the M matrix.

n: The number of equations in the system.

daxpy: A double precision vector operation that computes $ax+y$, where a is a scalar and x and y are vectors.

Initialization

1.) Choose an initial guess, *guess*

Small cost ($O(cn)$), but might be larger depending on the method used for choosing the guess.

2.) $r_0 = (K - \sigma M)^{-1} M\,guess$ (Purifying r_0)

A.) Formation of the matrix $(K - \sigma M)$

The matrix is formed from K and M and made available to the factorization routine.

Sequential: $\mu_M n$ subtractions and multiplications

Vector: 1 $\mu_M n$-length daxpy operation

B.) Factorization of $(K - \sigma M)$

Using Bunch-Kaufman (or LDL^T decomposition) as described in Section 8

Sequential: n divisions

$O(n\mu_K)$ multiplications

$O(n\mu_K^2/2)$ multiplications and additions

Vector: n scalar divisions

n μ_K-length vector by scalar multiplications

$n\mu_K$ daxpy operations of average length μ_K

C.) Forward Solve using factored matrix from B

Sequential: $O(n\mu_K)$ multiplications and divisions

Vector: n μ_K-length daxpy operations

D.) Diagonal Solve using factored matrix from B

Sequential: $3n$ multiplications

$2n$ additions

Vector: 3 n-length daxpy operations

E.) Back Solve using factored matrix from B

Sequential: $O(n\mu_K)$ multiplications and divisions

Vector: n μ_K-length inner products

3.) $p_1 = Mr_0$

An $n \times n$ banded matrix-vector multiplication

Sequential: $O((2\mu_K + 1)n)$ multiplications

$O(2\mu_K n)$ additions

Vector: n $\mu_K + 1$-length inner products

n μ_K-length daxpy operations

4.) $\beta_1 = (r_0^T p_1)^{1/2}$

An n-length inner product and a square root

Sequential: n multiplications

$n - 1$ additions

1 square root

Vector: 1 n-length inner product

1 square root

Program Loop

5.) For $j = 1$, maximum number of iterations

6.) Reorthogonalization phase

Orthogonalize r_{j-1} and q_{j-1} against "good" Ritz vectors if necessary (see section on orthogonalization for details). Steps A and B are done only once and only if reorthogonalization is needed. Steps C and E are done for each Ritz vector that is orthogonalized against r_{j-1}. Steps D and F are done for each Ritz vector that is orthogonalized against q_{j-1}.

A.) $t_1 = Mr_{j-1}$

Same cost as 3

B.) $t_2 = Mq_{j-1}$

Same cost as 3

C.) $\gamma_i = y_i^T t_1$

Multiplication of an n-length vector by a scalar

Sequential: n multiplications

Vector: 1 n-length vector by scalar multiplication

D.) $\psi_i = y_i^T t_2$
Same cost as C

E.) $r_{j-1} = r_{j-1} - \gamma_i y_i$
Orthogonalize r_j against y_i
Sequential: n multiplications
n additions
Vector: 1 n-length daxpy operation

F.) $q_{j-1} = q_{j-1} - \psi_i y_i$
Same cost as 6E
Orthogonalize r_j against y_i

7.) $\underline{q_j = r_{j-1}/\beta_j}$
Division of an n-length vector by a scalar
Sequential: n multiplications
1 division
Vector: 1 n-length vector by scalar multiplication
1 division

8.) $\underline{p_j = p_j/\beta_j}$
Same cost as 7

9.) $\underline{(K - \sigma M)r_j = p_j}$
Same cost as parts C, D, and E of 3

10.) $\underline{r_j = r_j - q_{j-1}\beta_j}$
Orthogonalize r_j against q_{j-1}
Same cost as 6E

11.) $\underline{\alpha_j = r_j^T p_j}$
Same cost as 4 except no square root is needed

12.) $\underline{r_j = r_j - q_j \alpha_j}$
Same cost as 10

13.) $\underline{p_{j+1} = M r_j}$
Same cost as 3

14.) $\underline{\beta_{j+1} = (r_j^T p_{j+1})^{1/2}}$

15.) $\underline{\text{Compute the eigenvalue of } T_j \text{ and the corresponding error bounds}}$

A.) Calculate j eigenvalues via Newton's method
Sequential and *Vector*: Variable, but very small ($O(j^2)$)

B.) Calculate j error bounds, β_{ji}
Sequential: j multiplications

16.) Compute any converged Ritz Vectors

Cost per Ritz vector of computing y_i

A.) Compute correction factor, $w_i = 1/\theta_i(T_j s_i)$

Sequential: $3j$ multiplications
$2j$ additions
j multiplications
1 division

Vector: 3 j-length daxpy operations
1 j-length vector by scalar multiplication
1 division

B.) Compute $y_i = Q_{j+1} w_i$

Sequential: nj multiplications
$n(j-1)$ additions

Vector: n j-length inner products

17.) Halt if enough eigenvalues have been found

18.) End of Loop

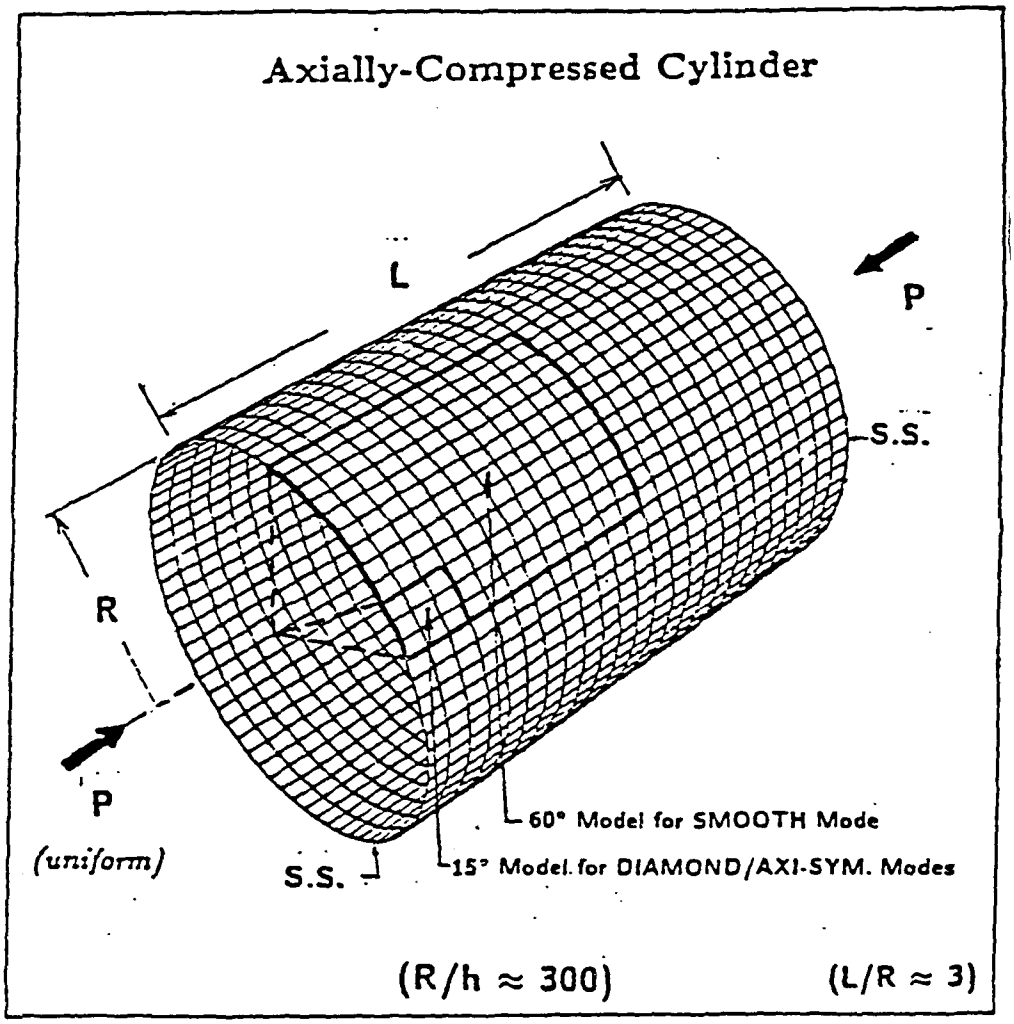

Figure 20: Axially-compressed circular cylindrical shell

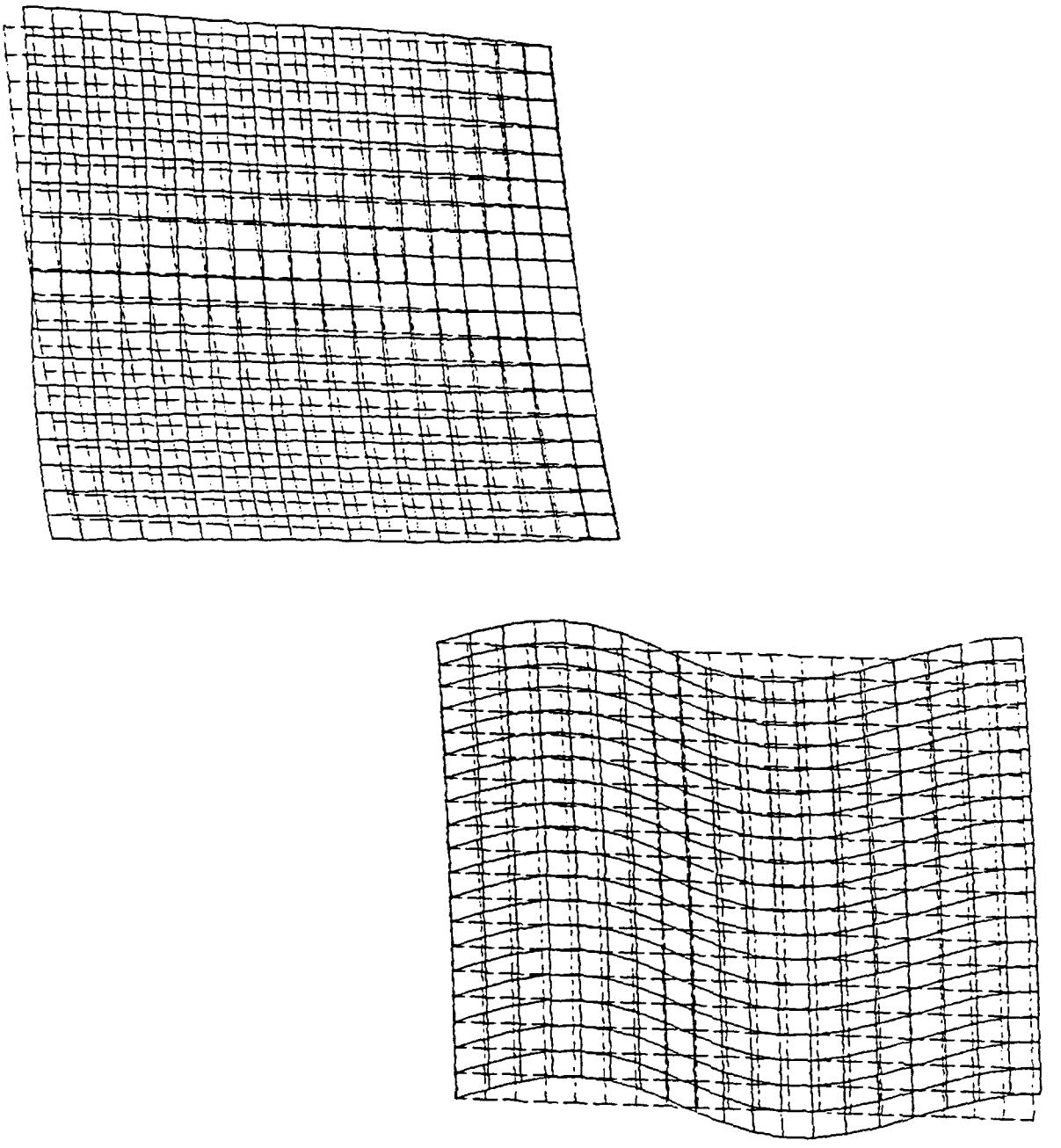

Figure 21: Lowest two buckling modes of an axially-compressed cylinder

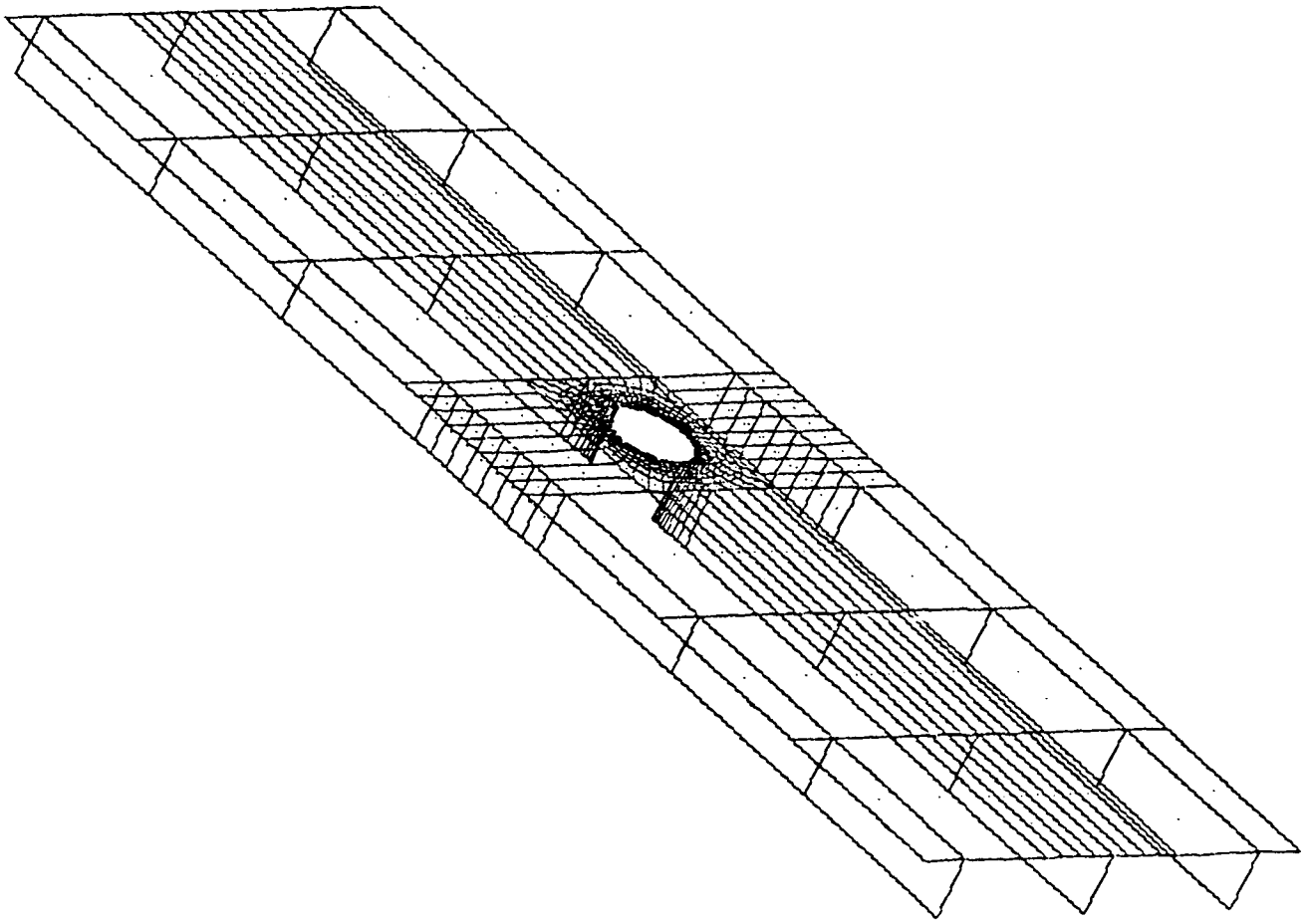

Figure 22: Composite blade-stiffened panel with a discontinous stiffener

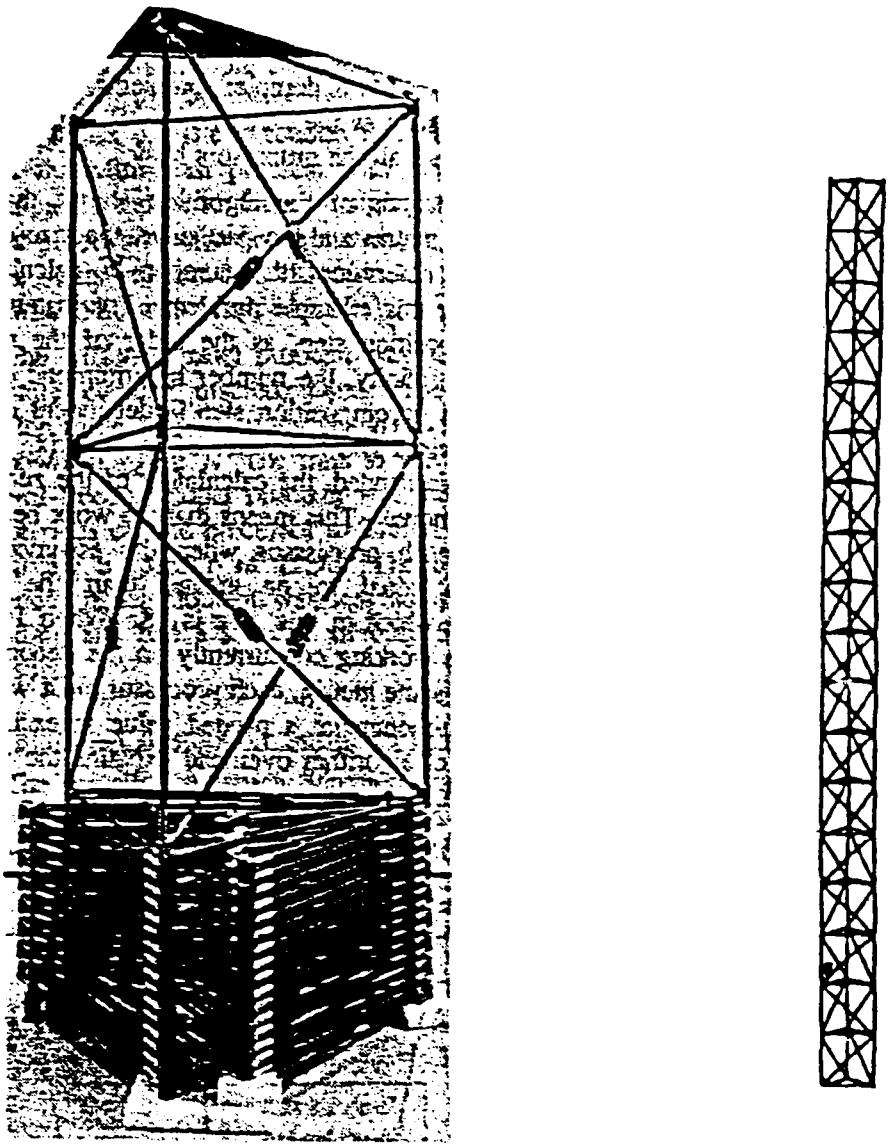

Figure 23: Deployable space mast

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-181914 ICASE Report No. 89-69		

4. Title and Subtitle	5. Report Date
THE USE OF LANCZOS'S METHOD TO SOLVE THE LARGE GENERALIZED SYMMETRIC DEFINITE EIGENVALUE PROBLEM	September 1989
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Mark T. Jones Merrell L. Patrick	89-69
	10. Work Unit No.
	505-90-21-01

9. Performing Organization Name and Address	11. Contract or Grant No.
Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225	NAS1-18605
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	Contractor Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Richard W. Barnwell

Final Report

16. Abstract

The generalized eigenvalue problem, $Kx = \lambda Mx$, is of significant practical importance, especially in structural engineering where it arises as the vibration and buckling problems. A new algorithm, LANZ, based on Lanczos's method is developed. LANZ uses a technique called dynamic shifting to improve the efficiency and reliability of the Lanczos algorithm. A new algorithm for solving the tridiagonal matrices that arise when using Lanczos's method is described. A modification of Parlett and Scott's selective orthogonalization algorithm is proposed. Results from an implementation of LANZ on a Convex C-220 show it to be superior to a subspace iteration code.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
generalized eigenvalue problem, Lanczo's method, buckling and vibration problems	61 - Computer Programming and Software 64 - Numerical Analysis 39 - Structural Mechanics Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	59	A04

NASA FORM 1626 OCT 86