ITERATIVE METHODS FOR THE COMPUTATION OF A FEW EIGENVALUES OF A LARGE SYMMETRIC MATRIX *†

J. BAGLAMA[1], D. CALVETTI[2], and L. REICHEL[3]

[1]*Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242. E-mail: jbaglama@mcs.kent.edu*

[2]*Department of Mathematical Sciences, Stevens Institute of Technology, Hoboken, NJ 07030. E-mail: na.calvetti@na-net.ornl.gov*

[3]*Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242. E-mail: reichel@mcs.kent.edu*

Dedicated to Åke Björck on the occasion of his 60th birthday

Abstract.

The task of computing a few eigenvalues and associated eigenvectors of a large sparse symmetric matrix arises in many applications. We present new iterative methods designed for the determination of a few extreme or non-extreme eigenvalues and associated eigenvectors. Our methods are based on the recursion formulas of the Implicitly Restarted Lanczos method introduced by Sorensen [37], but differ from previous applications of these formulas in the selection of accelerating polynomial. The methods of the present paper require very little computer storage. Numerical examples illustrate that the methods can give rapid convergence.

AMS subject classification: 65F15

Key words: Restarted Lanczos method, Leja points, polynomial acceleration.

1 Introduction.

The determination of a few, say k, eigenvalues and associated eigenvectors of a large sparse symmetric matrix $A \in \mathbf{R}^{n\times n}$, $n \gg k$, is an important computational problem that arises in many applications. A large number of algorithms for the solution of this problem are based on the Lanczos process. However, when applying the basic Lanczos method, one may encounter the following difficulties: i) large storage requirement for the Krylov subspace basis generated, ii) low accuracy of the computed approximate eigenvalues and eigenvectors due to loss of orthogonality of the computed Krylov subspace basis, and iii) poor or

*Received March 1995. Revised February 1996.

†This work was supported by NSF grants F377 DMR-8920147 ALCOM, DMS-9409422 and DMS-9205531.

no convergence towards eigenvalues in the interior of the spectrum unless the Lanczos process is combined with inverse iteration. Inverse iteration requires factorization of the matrix $A - zI$ into triangular matrices, and possibly a diagonal matrix, for some $z \in \mathbf{R}$ close to the desired eigenvalues.

This paper presents new iterative methods for the computation of a few selected eigenvalues of a large sparse symmetric matrix A. These methods do not require the factorization of matrices of the form $A - zI$, $z \in \mathbf{R}$. Iterative methods of this kind are of interest because the computational effort required for the factorization may be prohibitive when the order n of A is large. Also, the storage requirement for the factors may make their computation and use unattractive. An example that arises in the computation of equilibrium configurations of liquid crystals is described in Example 5.2 of Section 5.

The difficulties associated with applying the basic Lanczos method to the computation of a few extreme eigenvalues and associated eigenvectors of A have spurred considerable research aimed at improving the method or at developing alternative methods; see, e.g., [6, 13, 14, 23, 25, 27, 28, 29, 35] and references therein. Recently, Sorensen [37] proposed the Implicitly Restarted Lanczos (IRL) method for the computation of a few eigenvalues of a large sparse symmetric matrix, and the closely related Implicitly Restarted Arnoldi (IRA) method for the computation of a few eigenvalues of a large sparse nonsymmetric matrix. These methods can be regarded as curtailed QR algorithms for the symmetric and nonsymmetric eigenvalue problems, respectively. Similarly, as in the QR algorithms, the choice of shifts is crucial for the performance of the IRL and IRA methods. However, the Rayleigh or Wilkinson shifts, popular choices of shifts for the QR algorithm, cannot be applied in the IRL and IRA methods, because the data required to compute these shifts is not available. Therefore other shift selection strategies have been studied by Sorensen [37] for the IRL and IRA methods, and by Calvetti et al. [4] for the IRL method.

The IRL and IRA methods require the user to select a strategy for choosing a sequence of Krylov subspaces used to determine the invariant subspace associated with the desired eigenvalues. Similarly as the QR algorithm, the IRL and IRA methods have to be supplemented with deflation techniques in order to make the computation of multiple eigenvalues possible. Lehoucq and Sorensen [18, 19] describe deflation techniques applicable to the subspace selection strategies described in [4, 37].

The present paper describes modifications of the methods in [4, 37] for the computation of a few eigenvalues and associated eigenvectors of a large sparse symmetric matrix. We use the recursion formulas of the IRL method, but propose a new strategy for choosing the sequence of Krylov subpaces in the course of the computations. When a few extreme eigenvalues are desired, then our shift selection strategy is closely related to the one proposed in [4]. We also describe a new shift selection strategy applicable when a few non-extreme eigenvalues are desired. Our choices of subspaces and shifts avoid the difficulties i)-iii) of the basic Lanczos method discussed above. The storage requirements of the algorithms of the present paper are smaller than for previously described algorithms

based on the IRL recursions.

The computations for the algorithms of this paper proceed as follows. We apply m steps of the Lanczos method to an initial basis vector v_1, where m typically is chosen to be slightly larger than the number of desired eigenvalues k, e.g., $m = k + 2$. Orthogonality of the m basis vectors of the Krylov subspace generated by the Lanczos method is secured by reorthogonalization whenever necessary. In general only one reorthogonalization is necessary; this has been shown by Björck [1], see also the results by Hoffmann [15]. Assume first that we are interested in the k smallest eigenvalues and associated eigenvectors of A. The recursion coefficients of the Lanczos process then yield an $m \times m$ symmetric tridiagonal matrix T_m, whose eigenvalues are used to determine an interval $\mathbf{K}$ on the real axis that does not contain the k desired eigenvalues. Assume that no eigenvalue of T_m approximates any of the desired eigenvalues of A with sufficient accuracy. We then use the recursion formulas of the IRL method to apply an accelerating polynomial $\psi_m(A)$ to v_1 by using the orthogonal Krylov subspace basis, without evaluating new matrix-vector products with the matrix A. The purpose of the accelerating polynomial is to produce a vector $\psi_m(A)v_1$ in the invariant subspace associated with the k desired eigenvalues. The zeros of ψ_m are the shifts mentioned above; their choice affects the rate of convergence of the algorithm. We choose the m shifts as Leja points for the set $\mathbf{K}$. Having applied these m shifts, we carry out m steps of the Lanczos process with initial vector $\psi_m(A)v_1$.

If instead k non-extreme eigenvalues and the associated invariant subspace of A are desired, then we use the recursion coefficients generated by the Lanczos process to determine two symmetric tridiagonal matrices of order m and $m+1$, respectively. Eigenvalues of these matrices are used to determine a set $\mathbf{K}$ that consists of two intervals, and does not contain any of the k desired eigenvalues. The shifts, i.e., the zeros of ψ_m, are chosen to be Leja points for $\mathbf{K}$. Our algorithm only requires the user to provide a subroutine for the evaluation of matrix-vector products with the matrix A. In particular, the computation of matrix-vector products of the form $(A - zI)^{-1}u$ is not necessary, and therefore factorization of matrices of the form $A - zI$ is not required. Algorithms described in the literature for the computation of a few non-extreme eigenvalues require the factorization of $A - zI$ for one or several values of $z \in \mathbf{R}$; see [8, 27, 33, 39].

Numerical examples illustrate that the iterative methods of the present paper can give rapid convergence. They require only $O(mn)$ storage locations in addition to the storage space required for the matrix A.

Polynomial acceleration for eigenvalue computation was first used by Flanders and Shortley [10], who applied Chebyshev polynomials. A more recent application is described by Saad [34]. The algorithms of the present paper differ from the schemes in [10, 34] both in the organization of the computations and in the selection of accelerating polynomials. A restarted Lanczos method is described by Karush [16]. Applications that require the computation of a few eigenvalues of a large symmetric matrix are discussed in [5, 24, 30, 38, 40]. The recursion formulas of the Implicitly Restarted Lanczos method have recently also been

applied by Björck et al. [2] for the solution of ill-posed problems.

This paper is organized as follows. In Section 2 we review the Lanczos method and the recursion formulas of the IRL method. Section 3 describes our strategies for subspace and shift selections, and Section 4 presents the algorithms that result. Illustrative numerical examples are displayed in Section 5, and concluding remarks are found in Section 6.

2 The implicitly restarted Lanczos method.

The Lanczos process is a computational method for reducing an $n \times n$ symmetric matrix A to tridiagonal form, given an initial basis vector v_1, which we may assume to be of unit length. If we truncate the Lanczos process after $m < n$ steps, then we obtain the truncated reduction of A to tridiagonal form

$$AV_m = V_m T_m + f_m e_m^T, \tag{2.1}$$

where $V_m \in \mathbf{R}^{n\times m}$, $V_m e_1 = v_1$, $V_m^T V_m = I$, $T_m \in \mathbf{R}^{m\times m}$ is a symmetric tridiagonal matrix, and $f_m \in \mathbf{R}^n$ satisfies $V_m^T f_m = 0$. Throughout this paper e_j denotes the jth axis vector of appropriate dimension, and I denotes an identity matrix of suitable order. For future reference, we define

$$\beta_m = \|f_m\|, \tag{2.2}$$

where $\|\cdot\|$ denotes the Euclidean norm. Let θ be an eigenvalue of the matrix T_m and let y be an associated eigenvector. Then θ is an approximate eigenvalue of A, and is commonly referred to as a Ritz value of A. The vector $x = V_m y$ is an approximate eigenvector of A and is referred to as a Ritz vector of A. It follows from (2.1) that the residual error $Ax - x\theta$ associated with the Ritz pair $\{\theta, x\}$ satisfies

$$\|Ax - x\theta\| = \|(AV_m - V_m T_m)y\| = |\beta_m e_m^T y|. \tag{2.3}$$

Thus, the norm of the residual error can be determined without explicitly computing the Ritz vector x, by evaluating the right-hand side of (2.3). When the norm (2.3) is small, the Ritz value θ is an accurate approximation of an eigenvalue of A. The determination of how well x approximates an eigenvector of A requires further spectral information of A. In the basic Lanczos method, one increases m until the right-hand side of (2.3) is sufficiently small, and then computes the Ritz pair $\{\theta, x\}$. The latter requires that the matrix V_m be stored. The storage of V_m may require the use of secondary computer storage when the matrix A is large.

This difficulty can be avoided by restarting the Lanczos process periodically. The algorithms of the present paper use the recursion formulas of the IRL method of Sorensen [37] for this purpose. Our algorithms define new polynomial acceleration methods, in which the accelerating polynomials are determined by using the spectral information of A gained by the Lanczos process. The purpose of the accelerating polynomials is to determine a vector in the invariant subspace spanned by the eigenvectors that are associated with the desired eigenvalues.

We describe the recursion formulas of the IRL method. Throughout this paper, k denotes the number of desired eigenvalues. We assume that k is fixed and small. The number of steps of the Lanczos process taken between restarts is denoted by m, where we assume that $m > k$. After m steps of the Lanczos process with initial vector v_1, we have determined the quantities in formula (2.1). We now apply the following updating formulas, which are analogous to the explicitly shifted QR algorithm.

Let z be a chosen shift and determine the QR factorization $T_m - zI = QR$, where $Q, R \in \mathbf{R}^{m\times m}$, $Q^TQ = I$ and R is upper triangular. Putting $V = V_m$ and $T = T_m$, we obtain

$$(2.4.1)\quad (A - zI)V - V(T - zI) = f_m e_m^T,$$
$$(2.4.2)\quad (A - zI)V - VQR = f_m e_m^T,$$
$$(2.4.3)\quad (A - zI)(VQ) - (VQ)(RQ) = f_m e_m^T Q,$$
$$(2.4.4)\quad A(VQ) - (VQ)(RQ + zI) = f_m e_m^T Q.$$

Let $V' = VQ$ and $T' = RQ + zI$. Then T' is a symmetric tridiagonal matrix, which is similar to T. Multiplication of equation (2.4.2) by e_1 yields

$$(2.5)\qquad (A - zI)v_1 = v_1'\rho_{11},$$

where $\rho_{11} = e_1^T R e_1$ and $v_1' = V'e_1$. Equation (2.5) displays the relationship between the initial Lanczos vector v_1 and v_1'. After applying the $m - 1$ shifts $z_1, z_2, \ldots, z_{m-1}$, we obtain

$$(2.6)\qquad AV_m^+ = V_m^+ T_m^+ + f_m e_m^T Q^+,$$

where $V_m^+ = (v_1^+, v_2^+, \ldots, v_m^+) = V_m Q^+$, $T_m^+ = (Q^+)^T T_m Q^+$ and $Q^+ = Q_1 Q_2 \cdots Q_{m-1}$. Here Q_j denotes the orthogonal matrix associated with the shift z_j. Introduce the partitioning

$$(2.7)\qquad T_m^+ = \begin{pmatrix} \alpha_1^+ & \beta_1^+ e_1^T \\ \beta_1^+ e_1 & T_{m-1}^+ \end{pmatrix},$$

and equate the first column on the right-hand side and left-hand side of (2.6). We then obtain

$$(2.8)\qquad Av_1^+ = v_1^+\alpha_1^+ + f_1^+ e_1^T,$$

where $f_1^+ = v_2^+\beta_1^+ + f_m e_m^T Q^+ e_1$. It follows from $(v_1^+)^T f_1^+ = 0$ and $f_1^+ \in \text{span}\{v_1^+, Av_1^+\}$ that f_1^+ can be determined by the Lanczos process applied to the matrix A with initial vector v_1^+. The mth shift z_m is applied according to

$$(2.9)\qquad v_1^{++} = f_1^+ + (\alpha_1^+ - z_m)v_1^+.$$

By construction, $v_1^{++} = \psi_m(A)v_1$, where ψ_m is a polynomial of degree m with zeros $z_1, z_2, \ldots, z_m$. Note that v_1^{++} has been computed from v_1 without evaluation of matrix-vector products with the matrix A. Having computed v_1^{++} in the manner indicated, our algorithms set

$$(2.10)\qquad v_1 = v_1^{++}/\|v_1^{++}\|$$

and restart the Lanczos process with the vector (2.10) as initial vector. The algorithms proceeds in this manner to alternatively apply m steps of the Lanczos process and m shifts until an initial vector v_1 has been determined that lies in the invariant subspace associated with the k desired eigenvalues. Since we only keep at most $m+1$ orthogonal basis vectors of a Krylov subspace in memory during the computations, we can afford to secure their orthogonality by reorthogonalization whenever necessary.

As soon as an eigenvector has converged, it is stored, and subsequently generated Krylov subspace bases are orthogonalized against it. Assume that we have determined $j < k$ eigenvectors. Then the Lanczos process is only applied $m - j$ steps at a time, in order not to increase the memory requirement of the algorithms. The orthogonalization of Krylov subspaces against converged eigenvectors makes it possible to determine multiple eigenvalues.

Our algorithms differ from previous applications of the recursions of the IRL method in that we apply as many shifts as possible, i.e., until only the vector v_1^{++} remains. This makes deflation of converged eigenpairs trivial, and reduces requirements of computer storage.

3 Selection of accelerating polynomial.

The rate of convergence of our iterative methods is determined by the accelerating polynomial. We determine this polynomial by prescribing its zeros. Sometimes we refer to the zeros as shifts, because as shown in Section 2, they are shifts applied by a curtailed QR algorithm.

The description of the selection of zeros requires some notation. Let the set $\mathbf{K}$ consist of one or two closed and bounded intervals on the real axis, and let $w(z)$ be a nonnegative continuous function on $\mathbf{K}$. We refer to $w(z)$ as a weight function. Define a sequence $\{z_j\}_{j=1}^{\infty}$ of points in $\mathbf{K}$ as follows. Let z_1 be a point such that

$$w(z_1)|z_1| = \max_{z\in\mathbf{K}} w(z)|z|, \qquad z_1 \in \mathbf{K}, \tag{3.1}$$

and let z_j, for $j = 2, 3, \ldots$, satisfy

$$w(z_j)\prod_{l=1}^{j-1} |z_j - z_l| = \max_{z\in\mathbf{K}} w(z)\prod_{l=1}^{j-1} |z - z_l|, \qquad z_j \in \mathbf{K}. \tag{3.2}$$

The points z_j determined by (3.1)-(3.2) might not be unique. We call any sequence of points $\{z_j\}_{j=1}^{\infty}$ that satisfies (3.1)-(3.2) a sequence of weighted Leja points for $\mathbf{K}$, or sometimes briefly Leja points for $\mathbf{K}$. Because we will use these points as shifts in our algorithms, we also refer to them as Leja shifts. When $w(z) = 1$, the weighted Leja points agree with the "classical" Leja points studied by Leja [21], and probably first introduced by Edrei [7].

The sets $\mathbf{K}$ used in our algorithms are chosen so that they contain none of the desired k eigenvalues and all or most of the $n - k$ undesired ones. The motivation for choosing the shifts to be Leja points for such sets is that we want to dampen eigenvector components associated with undesired eigenvalues

in the initial Lanczos vectors v_1 determined by (2.10). We now describe how the eigenvalues of the Lanczos matrices generated by the algorithms help us determine such sets.

Assume first that we wish to compute the k smallest of the eigenvalues

$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_k < \lambda_{k+1} \leq \ldots \leq \lambda_n \tag{3.3}$$

of A, and let

$$\theta_1 < \theta_2 < \ldots < \theta_k < \theta_{k+1} < \ldots < \theta_m. \tag{3.4}$$

be eigenvalues of the matrix

$$T_m = \begin{pmatrix} \alpha_1 & \beta_1 & & & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & & \\ & \beta_2 & \alpha_3 & \beta_3 & & \\ & & & \ddots & & \\ & & & \ddots & \ddots & \beta_{m-1} \\ 0 & & & & \beta_{m-1} & \alpha_m \end{pmatrix} \tag{3.5}$$

defined by (2.1). By the Cauchy interlacing theorem,

$$\lambda_k \leq \theta_k, \qquad \theta_m \leq \lambda_n,$$

and therefore none of the k smallest eigenvalues of A lies in the interval $[\theta_{k+1}, \theta_m]$. This suggests the following choice of the set $\mathbf{K}$. The first time we determine a tridiagonal matrix (3.5) by the Lanczos process, we compute its spectrum (3.4) and define the endpoints of the interval $\mathbf{K} = [a, d]$ by

$$a = \theta_{k+1}, \qquad d = \theta_m. \tag{3.6}$$

We let the m shifts $\{z_j\}_{j=1}^m$ be Leja points for $\mathbf{K}$, i.e., we define the shifts by (3.1)-(3.2). Application of the shifts as described in Section 2 yields a new vector v_1 defined by (2.10). We now apply m steps of the Lanczos process to the matrix A with this vector v_1 as initial vector. This gives rise to a new tridiagonal matrix (3.5), defined by (2.1). The eigenvalues (3.4) of this tridiagonal matrix are computed and the endpoints of $\mathbf{K} = [a, d]$ are updated according to

$$a = \theta_{k+1}, \qquad d = \max\{d, \theta_m\}. \tag{3.7}$$

We then select m shifts $z_{m+1}, z_{m+2}, \ldots, z_{2m}$ as Leja points for this new set $\mathbf{K} = [a, d]$ in the presence of the points $z_1, z_2, \ldots, z_m$. More precisely, assume that we already have determined the points $\{z_j\}_{j=1}^r$. The next set of m points $\{z_j\}_{j=r+1}^{r+m}$ then is defined by the following algorithm. The weight function in the algorithm is chosen to be

$$w(z) = |z - \theta_{k+1}|. \tag{3.8}$$

ALGORITHM 3.1. Compute m shifts as Leja points for $\mathbf{K}$, given shifts $\{z_j\}_{j=1}^{r}$:

Input: *Endpoints of* $\mathbf{K}$, r, m, $\{z_j\}_{j=1}^{r}$; Output: $\{z_j\}_{j=r+1}^{r+m}$;

1. $j{:=}r{+}1$;

2. if $j{=}1$ *then*

z_0 := *point of largest magnitude of* $\mathbf{K}$

else

Determine $z_j \in \mathbf{K}$, *such that*

$$w(z_j)\prod_{l=1}^{j-1}|z_j - z_l| = \max_{z\in\mathbf{K}} w(z)\prod_{l=1}^{j-1}|z - z_l|,$$

where $w(z)$ *is defined by (3.8) or (3.17)*

endif;

3. $j := j + 1$;

4. if $j < r + m$ *then go to 2 else stop;* □

The determination of each Leja points, except z_0, by Algorithm 3.1 requires the maximization of a product over $\mathbf{K}$. In order to reduce the computational effort we replace $\mathbf{K}$ by a discrete point set.

We now describe the determination of the set $\mathbf{K}$ when k non-extreme eigenvalues of A are desired. We assume that A is indefinite and describe how to compute k eigenvalues close to the origin. By shifting A by a suitable multiple of the identity we can use the approach outlined to compute any set of k adjacent eigenvalues of A. Let the index s satisfy

$$\lambda_1 \leq \ldots \leq \lambda_{s-1} < 0 \leq \lambda_s \leq \ldots \leq \lambda_n, \tag{3.9}$$

where, as usual, the λ_j denote eigenvalues of A. For definiteness, assume that we are interested in the k_1 largest negative eigenvalues and in the k_2 smallest positive eigenvalues of A, where $k = k_1 + k_2$. An application of m steps of the Lanczos process to the matrix A yields a tridiagonal matrix T_m and a constant β_m; see (2.1) and (2.2). We can form the $(m+1)\times(m+1)$ matrix

$$\tilde{T}_{m+1} = \begin{pmatrix} \alpha_1 & \beta_1 & & & & & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & & & \\ & \beta_2 & \alpha_3 & \beta_3 & & & \\ & & & \ddots & & & \\ & & & \ddots & \ddots & \beta_{m-1} & \\ & & & & \beta_{m-1} & \alpha_m & \beta_m \\ 0 & & & & & \beta_m & \tilde{\alpha}_{m+1} \end{pmatrix}, \tag{3.10}$$

where the α_j and β_j are the same as in (3.5) and

$$\tilde{\alpha}_{m+1} = \beta_m^2 e_m^T T_m^{-1} e_m.$$

We assume that m is such that the matrix T_m is non-singular. Denote the eigenvalues of (3.10) by

$$\tilde{\theta}_1 < \tilde{\theta}_2 < \ldots < \tilde{\theta}_{m+1}. \tag{3.11}$$

One can show, see below, that the matrix $\tilde{T}_{m+1}$ is singular. Thus, there is an index q such that

$$\tilde{\theta}_q = 0. \tag{3.12}$$

The spectra of the matrices T_m and $\tilde{T}_{m+1}$ can be used to determine a set $\mathbf{K} = [a, b] \cup [c, d]$ that contains most or all of the $n - k$ undesired eigenvalues of A and none of the k wanted ones. In order to dampen eigenvector components associated with the undesired eigenvalues in the Lanczos vectors, we let the shifts be Leja points for $\mathbf{K}$. The endpoints a and d of $\mathbf{K}$ are determined as above, i.e., by formulas (3.6) or (3.7). We now consider the determination of the endpoints b and c. We would like them to be such that

$$\lambda_{s-k_1+i} \in [b, c], \qquad 0 \leq i < k_1 + k_2, \tag{3.13}$$

where the index s is given by (3.9). The following theorem sheds light on how to select these endpoints.

THEOREM 3.1. *Assume that the eigenvalues $\tilde{\theta}_j$ of $\tilde{T}_{m+1}$ are ordered according to (3.11), and let the index q be determined by (3.12). Then the interval $[0, \tilde{\theta}_{q+i}]$ contains at least i eigenvalues of A for $1 \leq i \leq m + 1 - q$, and the interval $[\tilde{\theta}_{q-i}, 0]$ contains at least i eigenvalues of A for $1 \leq i \leq q - 1$. The matrix (3.10) can be associated with a Gauss-Radau quadrature rule with a node at $z = 0$.*

PROOF: The intervals $[0, \tilde{\theta}_{q+i}]$ and $[\tilde{\theta}_{q-i}, 0]$ are Lehmann intervals discussed in [17, 26, 27]. The theorem follows by combining results by Golub [12], Lehmann [17], Morgan [22] and Paige et al. [26]. Details are presented in [3]. □

We are in a position to define the endpoints b and c for the set $\mathbf{K}$. It follows from Theorem 3.2 that the choice

$$b = \tilde{\theta}_{q-k_1-p}, \qquad c = \tilde{\theta}_{q+k_2+p} \tag{3.14}$$

satisfies (3.13) for $p \geq 1$.

Thus, after the first m steps of the Lanczos process, we compute the eigenvalues $\{\theta_j\}_{j=1}^m$ and $\{\tilde{\theta}_j\}_{j=1}^{m+1}$ of the tridiagonal matrices (3.5) and (3.10) generated, and define the initial set $\mathbf{K} = [a, b] \cup [c, d]$ by

$$\begin{aligned} a &= \theta_1, \qquad b = \tilde{\theta}_{q-k_1-p}, \\ c &= \tilde{\theta}_{q+k_2+p}, \qquad d = \theta_m \end{aligned} \tag{3.15}$$

for some integer $1 \leq p \leq \frac{1}{2}(m - k_1 - k_2) - 1$. We have found that $p = 1$ or $p = 2$ are suitable choices of p; see the computed examples of Section 5. Note that

it may happen that the endpoints a and b determined by (3.15) satisfy $b < a$. Then we let $\mathbf{K} = [c, d]$. Analogously, we let $\mathbf{K} = [a, b]$ if $d < c$. Our selection of endpoints secures that the set $\mathbf{K}$ consists of at least one interval.

In the course of the iterations new symmetric tridiagonal matrices T_m and $\tilde{T}_{m+1}$ are generated and their eigenvalues are computed. For each new set of eigenvalues computed, we update the endpoints of $\mathbf{K}$ according to

$$\begin{aligned} a &= \min\{a, \theta_1\}, \qquad b = \tilde{\theta}_{q-k_1-p}, \\ c &= \tilde{\theta}_{q+k_2+p}, \qquad d = \max\{d, \theta_m\}. \end{aligned} \tag{3.16}$$

The shifts are determined by Algorithm 3.1 with the weight function

$$w(z) = \begin{cases} |z - \tilde{\theta}_{q-k_1-p}|, & \text{if } z \in [a, b], \\ |z - \tilde{\theta}_{q+k_2+p}|, & \text{if } z \in [c, d]. \end{cases} \tag{3.17}$$

We remark that it is not necessary to compute all eigenvalues of the matrices T_m and $\tilde{T}_{m+1}$; only the extreme eigenvalues are required, and this may reduce the computational effort necessary. However, the work spent determining all eigenvalues of T_m and $\tilde{T}_{m+1}$ is negligible when $n \gg m$ and therefore we will not discuss this issue further in the present paper.

4 Algorithms.

We describe two algorithms: one for computing the k smallest eigenvalues $\{\lambda_j\}_{j=1}^k$ and associated eigenvectors $\{u_j\}_{j=1}^k$ of a large symmetric matrix A, and another one for computing the $k = k_1 + k_2$ eigenvalues $\{\lambda_j\}_{j=s-k_1}^{s+k_2-1}$ close to the origin and associated eigenvectors $\{u_j\}_{j=s-k_1}^{s+k_2-1}$ of a large indefinite symmetric matrix A, where the index s is defined by (3.9).

Let $\{\theta_j, y_j\}_{j=1}^m$ denote eigenvalue-eigenvector pairs of the symmetric tridiagonal matrix $T_m \in \mathbf{R}^{m\times m}$ defined by (2.1), and assume that the eigenvalues are ordered according to (3.4). We may assume that the off-diagonal elements β_j of T_m are nonvanishing, because otherwise we have found an invariant subspace. It follows from $\beta_j \neq 0$, $1 \leq j < m$, that the eigenvalues θ_j of T_m are distinct.

Let $x_j = V_m y_j$ be a Ritz vector of the matrix A, associated with the Ritz value θ_j. Then, analogously with (2.3), we obtain that

$$\|Ax_j - x_j\theta_j\| = |\beta_m e_m^T y_j|, \qquad 1 \leq j \leq m,$$

where β_m is defined by (2.2). In our algorithms the computations are terminated as soon as

$$\max_{1\leq j\leq k} |\beta_m e_m^T y_j| \leq \epsilon, \tag{4.1}$$

where ϵ is a given positive constant. Formula (4.1) also is used to determine whether an eigenvector has converged. We are in a position to describe our first algorithm. The parameter i_{conv} counts the number of converged eigenvalues, and r counts the number of Leja shifts that have been applied.

ALGORITHM 4.1. Computation of k smallest eigenvalues:

Input: A, k, m, v_1, ϵ; Output: $\{\lambda_j\}_{j=1}^{k}$, $\{u_j\}_{j=1}^{k}$;

1. $i_{\text{conv}} := 0$; $r := 0$;
2. *Apply m steps of the Lanczos process to the matrix A with initial unit vector $v_1 := v_1/\|v_1\|$ in order to determine the matrices T_m and V_m and the vector f_m in (2.1);*
3. *Compute the eigenvalues (3.4) of T_m;*
4. **If** *(4.1) is satisfied* **then** *done;*
5. *Determine whether any new eigenpairs have converged. Assume that $|\beta_m e_m^T y_j| \leq \epsilon$ for ℓ indices j. Let $i_{\text{conv}} := i_{\text{conv}} + \ell$; $m := m - \ell$; $k := k - \ell$;*
6. **If** $r = 0$ **then** *define the interval* $\mathbf{K} = [a, d]$ *by (3.6)* **else** *by (3.7);*
7. *Compute m Leja points $\{z_j\}_{j=r+1}^{r+m}$ for $\mathbf{K}$ by Algorithm 3.1;*
8. *Apply shifts $\{z_j\}_{j=r+1}^{r+m}$ according to (2.4)-(2.9). Compute v_1 by (2.10);*
9. *Orthogonalize vector v_1 against the i_{conv} converged eigenvectors. Denote the vector so obtained also by v_1. Let $r := r + m$; Go to 2;* □

We now turn to our second algorithm of this section. This algorithm is used to determine the $k = k_1 + k_2$ eigenvalues $\{\lambda_j\}_{j=s-k_1}^{s+k_2-1}$ close to the origin and associated eigenvectors $\{u_j\}_{j=s-k_1}^{s+k_2-1}$ of a large indefinite symmetric matrix A, where s is defined by (3.9).

ALGORITHM 4.2. Computation of k non-extreme eigenvalues:

Input: A, k_1, k_2, m, p, v_1, ϵ; Output: $\{\lambda_j\}_{j=s-k_1}^{s+k_2-1}$, $\{u_j\}_{j=s-k_1}^{s+k_2-1}$;

1. $k := k_1 + k_2$; $i_{\text{conv}} := 0$; $r := 0$;
2. *Apply m steps of the Lanczos process to the matrix A with initial unit vector $v_1 := v_1/\|v_1\|$ in order to determine the matrices T_m and V_m and the vector f_m in (2.1) as well as the matrix $\tilde{T}_{m+1}$ given by (3.10);*
3. *Compute the eigenvalues (3.4) of $T_m \in \mathbf{R}^{m\times m}$ and the eigenvalues (3.11) of $\tilde{T}_{m+1}$;*
4. **If** *(4.1) is satisfied* **then** *done;*
5. *Determine whether any new eigenpairs have converged. Assume that $|\beta_m e_m^T y_j| \leq \epsilon$ for ℓ indices j. Let $i_{\text{conv}} := i_{\text{conv}} + \ell$; $m := m - \ell$; $k := k - \ell$;*
6. **If** $r = 0$ **then** *define the set* $\mathbf{K} := [a, b] \cup [c, d]$ *by (3.15)* **else** *by (3.16);*
7. *Compute m Leja points $\{z_j\}_{j=r+1}^{r+m}$ for $\mathbf{K}$ by Algorithm 3.1;*

8. Apply shifts $\{z_j\}_{j=r+1}^{r+m}$ *according to (2.4)-(2.9). Compute* v_1 *by (2.10);*

9. Orthogonalize vector v_1 *against the* i_{conv} *converged eigenvectors. Denote the vector so obtained also by* v_1*. Let* $r := r + m$*; Go to 2;* □

The design of Algorithms 4.1 and 4.2 is motivated by their performance in numerous numerical experiments, a few of which are reported in Section 5. Theoretical results on the algorithms are incomplete. Difficulties in the analysis stem from that the sets $\mathbf{K}$ keep changing during the iterations. Consider a modification of Algorithm 4.1, in which the updating formula (3.7) for the endpoint a is replaced by $a = \min\{a, \theta_1\}$. Then the length of the interval $\mathbf{K}$ typically increases monotonically until $\mathbf{K}$ contains all undesired eigenvalues, and none of the desired ones. When such a set $\mathbf{K}$ has been determined, the set remains unchanged until all the wanted eigenvalues have been computed with desired accuracy. The convergence results presented in [4] for fixed $\mathbf{K}$ are now applicable. However, computed examples indicate that this modification generally gives much slower convergence than Algorithm 4.1. We therefore have presented the faster algorithm despite the lack of theoretical justification. The choice of weight functions (3.8) and (3.17) also is motivated by numerical experiments; the weight functions used give faster convergence than the choice $w(z) = 1$.

5 Computed examples.

This section describes some computed examples which illustrate the behavior of Algorithms 4.1 and 4.2. The computations were carried out on an HP 9000/770 computer using double precision arithmetic, i.e., with approximately 15 significant digits.

Our first example compares Algorithm 4.1 with the subroutine DNLASO of the FORTRAN package LASO2 by Scott [36] and with a subroutine in ARPACK by Lehoucq, Sorensen and Vu [20]. The subroutine DNLASO implements the Lanczos process with selective orthogonalization, see [29], and allows the user to specify the maximal amount of computer storage available for the code to use. Typically, the more storage available, the fewer restarts necessary and the faster convergence to the desired eigenvalues and eigenvectors. The subroutine allows the user to select block-size for the Lanczos process; if the block-size, denoted by NBLOCK, is larger than one, then DNLASO implements a block Lanczos algorithm. The parameter MAXJ of DNLASO specifies the order of the largest symmetric block-tridiagonal Lanczos matrix generated by the algorithm before restart. The largest order of this matrix is MAXJ*NBLOCK. The storage requirement for the (block) Lanczos vectors generated by DNLASO is n*MAXJ*NBLOCK storage locations. The columns labeled "# Lanczos vectors" in the tables display MAXJ*NBLOCK. The total storage requirement for DNLASO is larger than n * (MAXJ+2)*NBLOCK in addition to the storage needed to represent the matrix A.

The subroutine DNLASO is more advanced than our experimental code for Algorithm 4.1 and has multiple stopping criteria. The iterations may terminate

before desired accuracy is achieved and this makes a comparison between the subroutine DNLASO and our code for Algorithm 4.1 difficult. The performance of DNLASO is therefore displayed in tables for different choices of the maximal number of Lanczos vectors. The subroutine DNLASO allows the specification of a parameter NFIG, the number of desired correct decimal digits in the computed eigenvalue approximations. We show the performance of the DNLASO for NFIG=4 and NFIG=10. The columns "# of matrix-vector products" displays the number of matrix-vector products with the matrix A. The number of matrix-vector products shown in the tables is for vectors consisting of one column only, also when the block-size is larger than 1. The tables also display the magnitude of the errors in the computed eigenvalues. The stopping criterion is seen to be more reliable for block-size 3 than for block-size 1.

The iterations with our code for Algorithm 4.1 were terminated when condition (4.1) was satisfied. This stopping criterion gave in general at least $-2\log_{10}(\epsilon)$ correct decimal digits in the computed approximate eigenvalues. In all computed examples, the entries of the initial Lanczos vector v_1 were uniformly distributed random numbers in an interval $(0, \rho]$, where $\rho > 0$ is chosen so that $\|v_1\| = 1$. The initial vector is the same for all runs with all codes in each example, but it may differ between different examples. In experiments with DNLASO with block-size larger than 1, the first vector in the initial block is the initial vector used in experiments with block-size 1. The other vectors in the initial block have randomly generated uniformly distributed entries. The column "# Lanczos vectors" shows the parameter m in Algorithms 4.1 and 4.2; the algorithms require storage of $m + 1$ vectors of length n.

From ARPACK we use a subroutine that implements the IRL method with "exact shifts" as described by Sorensen [37] and Calvetti et al. [4]. We refer to this method as ARPACK. Thus, when interested in computing the k smallest eigenvalues of A, we use the $m - k$ largest eigenvalues of the matrices T_m generated as shifts z_j. The application of exact shifts often requires m to be chosen substantially larger than k. This phenomenon is discussed in Example 5.1 as well as in [4].

EXAMPLE 5.1. We wish to compute the three smallest eigenvalues of the matrix

$$A = \mathrm{diag}(1, 2, 3, \ldots, n). \tag{5.1}$$

Tables 5.1-5.6 compare Algorithm 4.1 with DNLASO and ARPACK when the matrices (5.1) are of order $n = 2500$. Table 5.1 shows the performance of Algorithm 4.1. Figure 5.1 displays the number of matrix-vector products required by Algorithm 4.1 for increasing values of m, and illustrates that the convergence of Algorithm 4.1 is fairly insensitive to the choice of m. The figure also displays the number of matrix-vector products required by ARPACK for different choices of m. We can see that ARPACK requires a larger value of m, i.e., storage of more n-vectors, in order to perform well. Analogous numerical experiments with ARPACK are reported in [4].

Table 5.2 shows the number of matrix-vector products required by ARPACK. The stopping criterion implemented in ARPACK is designed to terminate the

computations when $|\lambda_e - \lambda_c| < \epsilon|\lambda_c|$, where λ_e and λ_c denotes an exact eigenvalue and a computed approximation, respectively. With the choice $\epsilon = 1 \cdot 10^{-4}$, ARPACK gives an approximation of the smallest eigenvalue λ_1 of about the same accuracy as Algorithm 4.1. However, the computed approximations of λ_2 and λ_3 obtained by ARPACK were not as accurate as those determined by Algorithm 4.1. ARPACK required the evaluation of more matrix-vector products than Algorithm 4.1. Decreasing ϵ, in order to obtain more accurate approximations of λ_2 and λ_3, increases the number of matrix-vector products required by ARPACK.

Table 5.3 displays the number of matrix-vector products required by the subroutine DNLASO when storage is limited to 10, 20 and 30 Lanczos vectors and NFIG=10. The block-size is 1. The accuracy is seen to increase with the storage size for the Krylov subspace basis. The storage requirement as well as the number of matrix-vector products are much larger than for Algorithm 4.1. The accuracy achieved in the computed approximations of λ_1 and λ_2 when the iterations were terminated is lower than for Algorithm 4.1.

The performance of the block Lanczos algorithm is illustrated by Table 5.4. The accuracy achieved is higher than in Table 5.3, but so is the number of matrix-vector products required.

Tables 5.5 and 5.6 differ from Tables 5.3 and 5.4 only in the choice of NFIG. The tables show that even when much lower accuracy is demanded, the subroutine DNLASO still requires the evaluation of a substantial number of matrix-vector products and fairly large storage space.

Figure 5.2 compares Algorithm 4.1 and ARPACK when the order n of the matrix (5.1) is increased from 1000 to 10000. We let $m = 10$, i.e., we keep the dimension of the largest Krylov subspace used fixed, and compute the 3 smallest eigenvalues for each one of the matrices. The figure shows that the number of matrix-vector products required for both Algorithm 4.1 and ARPACK increases with n, but the rate of increase is faster for ARPACK.

EXAMPLE 5.2. In this example we wish to determine the minimum energy equilibrium configuration of liquid crystals in a slab

$$\Omega = \{(x, y, z) : 0 \leq x \leq a, 0 \leq y \leq b, 0 \leq z \leq c\}$$

with surface $\partial\Omega$. Using the Landau-de Gennes formulation, the free energy can be expressed in terms of a tensor order parameter field Q; see Gartland [11] and Priestly et al. [31]. The free energy is given by

$$F(Q) = F_{\text{vol}}(Q) + F_{\text{surf}}(Q) = \int_{\Omega} f_{\text{vol}}(Q)dV + \int_{\partial\Omega} f_{\text{surf}}(Q)dS, \tag{5.2}$$

where $Q = Q(p)$, $p \in \Omega$, is a 3×3 symmetric traceless tensor, which is represented

by

$$
\begin{aligned}
Q(p) \;=\; & q_1(p)\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} + q_2(p)\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
& + q_3(p)\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + q_4(p)\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \\
& + q_5(p)\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}
\end{aligned}
$$

and the q_i are real valued functions on Ω. The q_i are to be determined so that the free energy (5.2) is minimal. The representation

$$
\begin{aligned}
f_{\text{vol}}(Q) \;=\; & \tfrac{1}{2}L_1 Q_{\alpha\beta,\gamma}Q_{\alpha\beta,\gamma} + \tfrac{1}{2}L_2 Q_{\alpha\beta,\beta}Q_{\alpha\gamma,\gamma} + \tfrac{1}{2}L_3 Q_{\alpha\beta,\gamma}Q_{\alpha\gamma,\beta} \\
& + \tfrac{1}{2}A\ \text{trace}(Q^2) - \tfrac{1}{3}B\ \text{trace}(Q^3) + \tfrac{1}{4}C\ \text{trace}(Q^2)^2 \\
& + \tfrac{1}{5}D\ \text{trace}(Q^2)\text{trace}(Q^3) + \tfrac{1}{6}M\ \text{trace}(Q^2)^3 + \tfrac{1}{6}M'\ \text{trace}(Q^3)^2 \\
& - \Delta_{\chi_{max}} H_\alpha Q_{\alpha\beta} H_\beta - \Delta_{\epsilon_{max}} E_\alpha Q_{\alpha\beta} E_\beta
\end{aligned}
$$

uses the conventions that summation over repeated indices is implied and indices separated by commas represent partial derivatives. Here L_1, L_2 and L_3 are elastic constants; A, B, C, D, M and M' are bulk constants; and H, $\Delta_{\chi_{max}}$, E and $\Delta_{\epsilon_{max}}$ are field terms and the constants associated with the magnetic and electrical fields, respectively. Moreover,

$$f_{\text{surf}}(Q) = W\,\text{trace}((Q - Q_0)^2),$$

where W is a constant and the tensor Q_0 is determined by the boundary conditions for the functions q_i. We impose the boundary condition

$$F_{\text{surf}}(Q) = \int_{\partial\Omega} f_{\text{surf}}(Q)dS = 0, \tag{5.3}$$

which implies that the values of the q_i are prescribed on $\partial\Omega$. This boundary condition models strong anchoring of the liquid crystals on the surface $\partial\Omega$. Details can be found in [9, 11, 31].

The minimum energy equilibrium configuration of the liquid crystals is determined by solving the Euler-Lagrange equations associated with (5.2) and (5.3). These equations yield a boundary value problem for a system of nonlinear partial differential equations for the q_i. Discretization by finite differences gives rise to a system of nonlinear equations of finite, but large, order. We solve this system by Newton's method. Each iteration by Newton's method requires the solution of a linear system of equations, the matrix of which is the Jacobian obtain from

the discretized Euler-Lagrange equations. The purpose of the computations is to track the minimal energy equilibrium configuration as the temperature of the liquid crystals is varied. For this reason it is essential to determine a few of the eigenvalues closest to the origin of the Jacobian matrix. In order to reduce the storage space required, we do not store all the nonzero entries of the Jacobian matrix simultaneously. We use Maple V to manipulate the formulas for the Jacobian, and to generate FORTRAN code for the evaluation of matrix-vector products with the Jacobian matrix. The FORTRAN code only requires that a few of the entries of the Jacobian matrix be stored in fast computer memory at any given time. This approach is motivated by the fact that the order of the Jacobian matrices of interest is very large; we discretize Ω by a $40 \times 40 \times 40$ grid and obtain a Jacobian matrix of order $296,595$. The smallest eigenvalues of the Jacobian used in the present example are, roughly, $\lambda_1 = -1.2 \cdot 10^{-2}$, $\lambda_2 = 2.2 \cdot 10^{-1}$ and $\lambda_3 = 2.3 \cdot 10^{-1}$. We used Algorithm 4.1 with $\epsilon = 1 \cdot 10^{-5}$ and $m = 5$ to compute accurate approximations of these eigenvalues. This required the evaluation of 649 matrix-vector products with the Jacobian matrix. For comparison, we note that 664 matrix-vector products are required when $m = 10$. Thus, increasing the value of m does not decrease the computational work required. Due to the large size of the matrix, only a basis of a Krylov subspace of small dimension can be stored in fast computer memory. In particular, inverse iteration is unfeasible. This examples illustrates that Algorithm 4.1 can determine eigenvalues of a very large matrix by using a Krylov subspace of low dimension only.

EXAMPLE 5.3. Let $A = \mathrm{diag}(a_1, a_2, \ldots, a_{100})$ with entries $a_i = \frac{i^2}{100}$, $1 \leq i \leq 100$. The computation of the three smallest eigenvalues and associated eigenvectors by Algorithm 4.1 with $\epsilon = 1 \cdot 10^{-3}$ requires 243 matrix-vector products when $m = 5$, and 264 matrix-vector products when $m = 9$. Similarly as in Example 5.1 and Example 5.2, the number of matrix-vector products is fairly insensitive to the choice of m.

The computation of the six smallest eigenvalues of A with $m = 8$ and $\epsilon = 1 \cdot 10^{-3}$ requires 456 matrix-vector products by Algorithm 4.1. The poor separation between the smallest eigenvalues is responsible for the large number of shifts that have to be applied in order to determine these eigenvalues. For comparison, Algorithm 4.1 requires only the evaluation of 114 matrix-vector products to determine the six smallest eigenvalues of the matrix $-A$ for $m = 8$ and $\epsilon = 1 \cdot 10^{-3}$, because the smallest eigenvalues of $-A$ are better separated than the smallest eigenvalues of A.

EXAMPLE 5.4. The nontrivial entries of the 500×500 diagonal matrix A used in this example are equidistant points in the interval $[1, 10]$. The computation of the four smallest eigenvalues by Algorithm 4.1 with $m = 6$ and $\epsilon = 1 \cdot 10^{-3}$ requires 211 matrix-vector products. Increasing the parameter m to 8 increases the number of matrix-vector products to 242. Thus, the solution effort does not always decrease as m increases.

EXAMPLE 5.5. Let A be the matrix obtained by discretizing the 2-dimensional negative Laplace operator on the unit square by the standard 5-point stencil

with Dirichlet boundary conditions. We wish to determine the two smallest eigenvalues of A by Algorithm 4.1 and choose $\epsilon = 1 \cdot 10^{-4}$. When A is of order 900 the number of matrix-vector products required is 104, both for $m = 4$ and $m = 8$. Thus, the performance of the algorithm is quite insensitive to the choice of m.

The next examples illustrate the behavior of Algorithm 4.2 for the computation of a few eigenvalues close to the origin of a symmetric indefinite matrix. A few eigenvalues close to any value μ between the largest and smallest eigenvalues of A can be determined by applying Algorithm 4.2 to the matrix $A - \mu I$.

EXAMPLE 5.6. Let $A = \mathrm{diag}(a_1, a_2, \ldots, a_{500})$ with $a_{2i} = \sqrt{i}$ and $a_{2i-1} = -\sqrt{i}$, $1 \leq i \leq 250$. We want to compute the smallest positive and largest negative eigenvalues of A and corresponding eigenvectors by Algorithm 4.2, with $m = 8$ and $\epsilon = 1 \cdot 10^{-3}$. We select the third largest negative and the third smallest positive Ritz values as endpoints closest to the origin of the set $\mathbf{K}$ used to determine Leja shifts, i.e., we let $k_1 = k_2 = 1$ and $p = 2$. The computation of the desired eigenpairs requires 291 matrix-vector products with the matrix A. We remark that a larger separation between the eigenvalues of interest and those corresponding to eigenvector components that we want to damp will yield faster convergence. For example, the computation of the eigenpairs corresponding to the two smallest positive and two largest negative eigenvalues of A by Algorithm 4.2, with $m = 10, k_1 = k_2 = 2$, $p = 2$ and $\epsilon = 1 \cdot 10^{-3}$, requires 282 matrix-vector products.

EXAMPLE 5.7. In this example we consider the 800×800 matrix

$$A = \begin{pmatrix} I & M \\ M^T & 0 \end{pmatrix}, \tag{5.4}$$

where $M = \mathrm{diag}(m_1, m_2, \ldots, m_{400})$ with entries

$$m_i = \begin{cases} i & \text{if } 1 \leq i \leq 4, \\ 5 + \frac{i}{20} & \text{if } 5 \leq i \leq 400. \end{cases}$$

Matrices of the form (5.4) arise, for example, when solving constrained least squares problems. The matrix A has 400 positive and 400 negative eigenvalues given by

$$\lambda_{\pm i} = \frac{1}{2} \pm \sqrt{\frac{1}{4} + m_i^2}, \qquad 1 \leq i \leq 400.$$

The computation of the two smallest positive and the two largest negative eigenvalues of A, and corresponding eigenvectors, using Algorithm 4.2 with $m = 12$, $k_1 = k_2 = 2$, $p = 1$ and $\epsilon = 1 \cdot 10^{-3}$ requires 231 matrix-vector products with A. If instead $m = 16$, then the number of matrix-vector products required is reduced slightly to 222.

The computation of the eigenpairs corresponding to the smallest positive and largest negative eigenvalues of A, with $m = 6, k_1 = k_2 = 1$, $p = 1$ and $\epsilon = 1 \cdot 10^{-3}$ requires 218 matrix-vector products. If we increase the value of m to 8, the two desired eigenpairs can be computed with 165 matrix-vector products. However,

if we further increase m, the number of matrix-vector products needed remains essentially unchanged. This example illustrates that the number of matrix-vector product evaluations is fairly insensitive to the choice of m, for m slightly larger than $k = k_1 + k_2$.

6 Conclusion.

The paper describes new algorithms for the computation of a few eigenvalues of a large symmetric matrix. A comparison of Algorithm 4.1 with the subroutine DNLASO in the LASO2 package and ARPACK shows our algorithm to be competitive. Both Algorithms 4.1 and 4.2 require only very little computer storage in addition to storage used for the desired eigenvectors and for a representation of the matrix A.

REFERENCES

1. Å. Björck, *Numerics of Gram-Schmidt orthogonalization*, Linear Algebra Appl., 197-198 (1994), pp. 297–316.
2. Å. Björck, E. Grimme and P. Van Dooren, *An implicit shift bidiagonalization algorithm for ill-posed problems*, BIT, 34 (1994), pp. 510–534.
3. D. Calvetti and L. Reichel, *An adaptive Richardson iteration method for indefinite linear systems*, Numerical Algo., to appear.
4. D. Calvetti, L. Reichel and D. C. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Elec. Trans. Numer. Anal., 2 (1994), pp. 1–21.
5. A. K. Cline, G. H. Golub and G. W. Platzman, *Calculation of normal modes of oceans using the Lanczos method*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 409–426.
6. J. Cullum and R. A. Willoughby, Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1, Birkhäuser, Boston, 1985.
7. A. Edrei, *Sur les déterminants récurrents et les singularités d'une fonction donnée par son développement de Taylor*, Composito Math., 7 (1939), pp. 20–88.
8. T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. Comp., 35 (1980), pp. 1251–1268.
9. P. A. Farrell, A. Ruttan and R. R. Zeller, *Finite difference minimization of the Landau-de Gennes free energy for liquid crystals in rectangular regions*, Comp. Appl. Math., I, C. Brezinski and U. Kulish, eds., Elsevier, Amsterdam, 1992, pp. 137–146.
10. D. A. Flanders and G. Shortley, *Numerical determination of fundamental modes*, J. Appl. Phys., 21 (1950), pp. 1326–1332.
11. E. C. Gartland, *On some mathematical and numerical aspects of the Landau-de Gennes minimization problem for liquid crystals*, Report, Institute for Computational Mathematics, Kent State University, Kent, 1993.
12. G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Review, 15 (1973), pp. 318–334.

13. G. H. Golub, R. Underwood and J. H. Wilkinson, *The Lanczos algorithm for the symmetric $Ax = \lambda Bx$ problem*, Report STAN-CS-72-270, Department of Computer Science, Stanford University, Stanford, 1972.
14. R. G. Grimes, J. L. Lewis and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal., 15 (1994), pp. 228–272.
15. W. Hoffmann, *Iterative algorithms for Gram-Schmidt orthogonalization*, Computing, 41 (1989), pp. 335–348.
16. W. Karush, *An iterative method for finding characteristic vectors of a symmetric matrix*, Pacific J. Math., 1 (1951), pp. 233–248.
17. N. J. Lehmann, *Optimale Eigenwerteinschliessungen*, Numer. Math., 5 (1963), pp. 246–272.
18. R. Lehoucq, *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration*, Ph.D. Thesis, Rice University, Houston, 1995.
19. R. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly re-started Arnoldi iteration*, Report TR94-13, Department of Computational and Applied Mathematics, Rice University, Houston, 1995.
20. R. Lehoucq, D. C. Sorensen and P.A. Vu, ARPACK: An implementation of the implicitly restarted Arnoldi and the implicitly restarted Lanczos methods. Code available from Netlib, in directory scalapack.
21. F. Leja, *Sur certaines suits liées aux ensemble plan et leur application à la representation conforme*, Ann. Polon. Math., 4 (1957), pp. 8–13.
22. R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl., 154-156 (1991), pp. 289–309.
23. R. B. Morgan and D. S. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 817–825.
24. R. Natarajan and D. Vanderbilt, *A new iterative scheme for obtaining eigenvectors of large, real-symmetric matrices*, J. Comp. Phys., 82 (1989), pp. 218–228.
25. C. C. Paige, *Computational variants of the Lanczos method for the eigenproblem*, J. Inst. Math. Appl., 10 (1972), pp. 373–381.
26. C. C. Paige, B. N. Parlett and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Lin. Alg. Appl., to appear.
27. B. N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, 1980.
28. B. N. Parlett and B. Nour-Omid, *Towards a black box Lanczos program*, Comput. Phys. Comm., 53 (1989), pp. 169–179.
29. B. N. Parlett and D. S. Scott, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 311–328.
30. A. Pothen, H. D. Simon, and K.-P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.
31. E. B. Priestly, P. J. Wojyowicz, and P. Sheng, eds., Introduction to Liquid Crystals, Plenum Press, New York, 1975.
32. L. Reichel, *The application of Leja points to Richardson iteration and polynomial preconditioning*, Linear Algebra Appl., 154-156 (1991), pp. 389–414.
33. A. Ruhe, *Rational Krylov sequence methods for eigenvalue computations*, Linear Algebra Appl., 58 (1984), pp. 391–405.

34. Y. Saad, *Iterative solution of indefinite symmetric linear systems by methods using orthogonal polynomials over two disjoint intervals*, SIAM J. Numer. Anal., 20 (1983), pp. 784–811.

35. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halstead Press, New York, 1992.

36. D. S. Scott, LASO2 - FORTRAN implementation of the Lanczos process with selective orthogonalization. Code and documentation available from Netlib.

37. D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

38. D. C. Sorensen, *Minimization of a large scale quadratic function subject to an ellipsoidal constraint*, Report TR94-27, Department of Computational and Applied Mathematics, Rice University, Houston, 1994.

39. D. B. Szyld, *Criteria for combining inverse and Rayleigh quotient iteration*, SIAM J. Numer. Anal., 25 (1988), pp. 1369–1375.

40. X. Yang, T. K. Sarkar and E. Arvas, *A survey of conjugate gradient algorithms for solution of extreme eigen-problems of a symmetric matrix*, IEEE Trans. Acoust. Speech Signal Proc., 37 (1989), pp. 1550–1555.

Table 5.1: Example 5.1. Algorithm 4.1: $k = 3$, $\epsilon = 1 \cdot 10^{-4}$

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
5	525	$6.4 \cdot 10^{-11}$	$1.6 \cdot 10^{-10}$	$3.8 \cdot 10^{-10}$
10	583	$3.2 \cdot 10^{-10}$	$9.4 \cdot 10^{-11}$	$2.5 \cdot 10^{-11}$
15	497	$2.8 \cdot 10^{-11}$	$1.2 \cdot 10^{-11}$	$9.5 \cdot 10^{-11}$

Table 5.2: Example 5.1. ARPACK: $k = 3$, $\epsilon = 1 \cdot 10^{-4}$

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
5	3724	$6.4 \cdot 10^{-10}$	$1.9 \cdot 10^{-4}$	$5.8 \cdot 10^{-5}$
10	904	$5.5 \cdot 10^{-10}$	$1.1 \cdot 10^{-4}$	$2.4 \cdot 10^{-5}$
15	614	$5.7 \cdot 10^{-10}$	$7.8 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$

Table 5.3: Example 5.1. DNLASO: NFIG=10, block-size = 1

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
10	1691	$5.7 \cdot 10^{-7}$	$5.6 \cdot 10^{-7}$	$1.7 \cdot 10^{-11}$
20	1922	$1.6 \cdot 10^{-8}$	$1.5 \cdot 10^{-8}$	$2.6 \cdot 10^{-11}$
30	1721	$1.3 \cdot 10^{-9}$	$1.2 \cdot 10^{-9}$	$6.1 \cdot 10^{-11}$

Table 5.4: Example 5.1. DNLASO: NFIG=10, block-size = 3

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
18	3813	$1.3 \cdot 10^{-11}$	$8.3 \cdot 10^{-12}$	$4.7 \cdot 10^{-11}$
30	2952	$5.6 \cdot 10^{-11}$	$1.2 \cdot 10^{-11}$	$2.4 \cdot 10^{-11}$
39	2274	$9.3 \cdot 10^{-12}$	$5.9 \cdot 10^{-12}$	$1.3 \cdot 10^{-11}$

Table 5.5: Example 5.1. DNLASO: NFIG=4, block-size = 1

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
10	353	$7.9 \cdot 10^{-4}$	$7.1 \cdot 10^{-1}$	$5.9 \cdot 10^{-1}$
20	709	$4.3 \cdot 10^{-3}$	$9.4 \cdot 10^{-3}$	$7.2 \cdot 10^{-3}$
30	570	$6.6 \cdot 10^{-4}$	$1.9 \cdot 10^{-4}$	$3.5 \cdot 10^{-4}$

Table 5.6: Example 5.1. DNLASO: NFIG=4, block-size = 3

# Lanczos vectors	# matrix-vector products	magnitude of error in computed approx. of		
		λ_1	λ_2	λ_3
18	2715	$3.3 \cdot 10^{-5}$	$2.7 \cdot 10^{-5}$	$4.1 \cdot 10^{-5}$
30	1887	$2.6 \cdot 10^{-5}$	$2.4 \cdot 10^{-5}$	$1.5 \cdot 10^{-5}$
39	1050	$6.4 \cdot 10^{-6}$	$7.8 \cdot 10^{-6}$	$1.5 \cdot 10^{-5}$

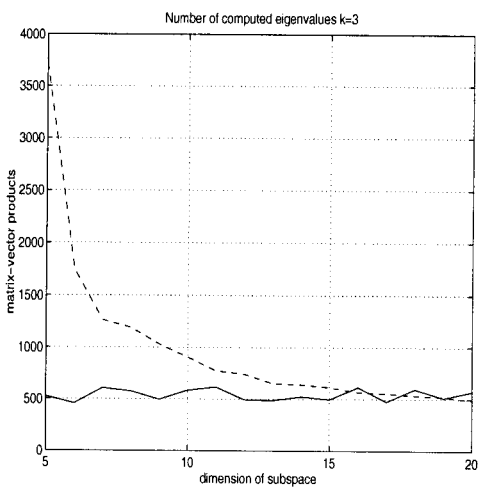

Figure 5.1: Number of matrix-vector products required by Algorithm 4.1 (—) and ARPACK (- -) for computing the 3 smallest eigenvalues for the matrix of Example 5.1 with $n = 2500$ as the dimension of the Krylov subspace is increased.

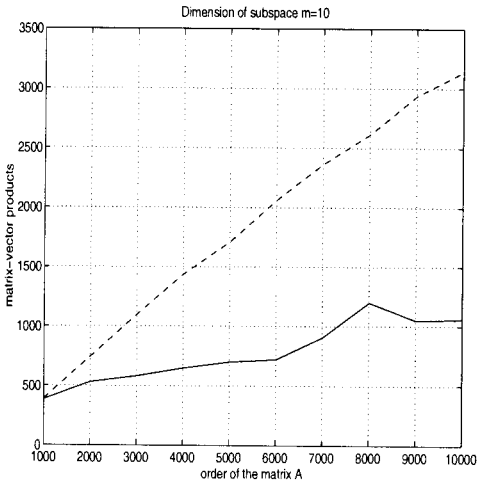

Figure 5.2: Number of matrix-vector products required by Algorithm 4.1 (—) and ARPACK (- -) for computing the 3 smallest eigenvalues for matrices (5.1) as their order is increased.