JOURNAL OF COMPUTATIONAL PHYSICS **146,** 203–226 (1998)
ARTICLE NO. CP986064

Computation of a Few Small Eigenvalues of a Large Matrix with Application to Liquid Crystal Modeling

J. Baglama,*[,1] D. Calvetti,†[,2] L. Reichel, ‡[,3] and A. Ruttan‡[,4]

**Department of Mathematics, Texas Tech University, Lubbock, Texas 79409; †Department of Mathematics, Case Western Reserve University, Cleveland, Ohio 44106; ‡Department of Mathematics and Computer Science, Kent State University, Kent, Ohio 44242*

Received January 28, 1997; revised March 13, 1998

Equilibrium configurations of liquid crystals in a finite containment are minimizers of the thermodynamic free energy of the system. It is important to be able to track an equilibrium configuration as the temperature of the liquid crystals is decreased. The path of the minimal energy configuration at a bifurcation point can be computed from the null space of a sparse symmetric matrix, which typically is very large, e.g., of order 3×10^5. We describe an implicitly restarted block Lanczos method designed for the computation of a few extreme multiple or close eigenvalues and associated eigenvectors of a large sparse symmetric matrix and apply this method to determine the desired null space. Our method generalizes the implicitly restarted Lanczos method introduced by Sorensen. The method requires that certain acceleration parameters, referred to as shifts, be chosen. The storage requirement depends on the choice of shifts. We propose a new strategy for choosing shifts. Numerical examples illustrate that the implicitly restarted block Lanczos method with shifts chosen in this manner gives rapid convergence, reliably detects extreme multiple or close eigenvalues, and requires little computer storage in addition to the storage used for the desired eigenvectors. These features make the method well suited for the application of tracking an equilibrium configuration of liquid crystals. © 1998 Academic Press

Key Words: bifurcation; block Lanczos method; continuation method; equilibrium configuration; polynomial acceleration.

[1] E-mail: baglama@math.ttu.edu. Research supported in part by NSF Grant F377 DMR-8920147 ALCOM.

[2] E-mail: dxc57@po.cwru.edu. Research supported in part by NSF Grant DMS-9404692.

[3] E-mail: reichel@mcs.kent.edu. Research supported in part by NSF Grants DMS-9404706 and ASC-9720221.

[4] E-mail: ruttan@mcs.kent.edu. Research supported in part by NSF Grants F377 DMR-8920147 ALCOM and ASC-9720221.

1. INTRODUCTION

The computation of an equilibrium configuration of liquid crystals and the tracking of such a configuration as the temperature of the liquid crystals decreases are computationally challenging problems. Tracking of the equilibrium configuration requires the determination of a few, say k, of the smallest eigenvalues and associated eigenvectors of a large sparse symmetric matrix $A \in \mathbf{R}^{n\times n}$, where n can be as large as 3×10^5 and k typically is 4. Some of the desired eigenvalues can be of multiplicity larger than one, or distinct and very close. It is the purpose of the present paper to describe a new method for computing the desired eigenvalue–eigenvector pairs.

We begin with a description of the liquid crystals problem. The problem under consideration is to determine the minimum energy equilibrium configuration of liquid crystals in a slab

$$\Omega = \{(x_1, x_2, x_3) : 0 \le x_1 \le a, 0 \le x_2 \le b, 0 \le x_3 \le c\} \tag{1.1}$$

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 Priestly *et al.* [27]. The free energy is given by

$$F(Q,\mathcal{T}) = F_{\text{vol}}(Q,\mathcal{T}) + F_{\text{surf}}(Q) = \int_\Omega f_{\text{vol}}(Q,\mathcal{T})\,dV + \int_{\partial\Omega} f_{\text{surf}}(Q)\,dS, \tag{1.2}$$

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

$$\begin{aligned} Q(p) = {} & q_1(p)\begin{pmatrix}1&0&0\\0&0&0\\0&0&-1\end{pmatrix} + q_2(p)\begin{pmatrix}0&1&0\\1&0&0\\0&0&0\end{pmatrix} + q_3(p)\begin{pmatrix}0&0&1\\0&0&0\\1&0&0\end{pmatrix}\\ & + q_4(p)\begin{pmatrix}0&0&0\\0&1&0\\0&0&-1\end{pmatrix} + q_5(p)\begin{pmatrix}0&0&0\\0&0&1\\0&1&0\end{pmatrix} \end{aligned} \tag{1.3}$$

and the q_i are real-valued functions on Ω. The q_i are to be determined so that the free energy (1.2) is minimal. The representation

$$\begin{aligned} f_{\text{vol}}(Q,\mathcal{T}) = {} & \frac{1}{2}\mathcal{L}_1 Q_{\alpha\beta,\gamma} Q_{\alpha\beta,\gamma} + \frac{1}{2}\mathcal{L}_2 Q_{\alpha\beta,\beta} Q_{\alpha\gamma,\gamma} + \frac{1}{2}\mathcal{L}_3 Q_{\alpha\beta,\gamma} Q_{\alpha\gamma,\beta}\\ & + \frac{1}{2}\mathcal{A}\,\text{trace}(Q^2) - \frac{1}{3}\mathcal{B}\,\text{trace}(Q^3) + \frac{1}{4}\mathcal{C}\,\text{trace}(Q^2)^2 \end{aligned} \tag{1.4}$$

uses the conventions that summation over repeated indices is implied and indices separated by commas represent partial derivatives. For example,

$$Q_{\alpha\beta,\gamma} Q_{\alpha\gamma,\beta} = \sum_{\alpha=1}^{3}\sum_{\beta=1}^{3}\sum_{\gamma=1}^{3} \frac{\partial Q[\alpha,\beta]}{\partial x_\gamma} \cdot \frac{\partial Q[\alpha,\gamma]}{\partial x_\beta}.$$

The bulk parameter $\mathcal{A}$ is assumed to be of the form $\mathcal{A} = \mathcal{A}_0(\mathcal{T} - \mathcal{T}_0)$, where $\mathcal{A}_0$ and $\mathcal{T}_0$ are constants and $\mathcal{T}$ is the normalized temperature of the liquid crystals. In this paper we take $\mathcal{T}_0 = 0$ and $\mathcal{A}_0 = 2$, which gives $\mathcal{T} = \frac{1}{2}\mathcal{A}$. The quantities $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ are elastic constants, and $\mathcal{B}$ and $\mathcal{C}$ are bulk constants. Moreover,

$$f_{\text{surf}}(Q) = \mathcal{W}\,\text{trace}\big((Q - Q_0)^2\big), \tag{1.5}$$

where $\mathcal{W}$ is a constant and the tensor Q_0 is determined by the boundary conditions for the functions q_i. We consider two kinds of boundary conditions which model strong and weak anchoring of the liquid crystals on the surface $\partial\Omega$, respectively. Strong anchoring is obtained by imposing the condition

$$Q(p) = Q_0(p), \quad p \in \partial\Omega. \tag{1.6}$$

This has the effect that

$$F_{\text{surf}}(Q) = \int_{\partial\Omega} f_{\text{surf}}(Q)\,dS = 0. \tag{1.7}$$

Weak anchoring of the liquid crystal molecules on the boundary is obtained by setting the constant $\mathcal{W}$ in (1.5) to a finite value. Details can be found in [9, 12, 27, 35].

The minimum energy equilibrium configuration of the liquid crystals is determined by solving the Euler–Lagrange equations associated with (1.2). 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 large order which we represent as

$$G(Q, \mathcal{T}) = 0. \tag{1.8}$$

We solve this system by Newton's method. Each iteration by Newton's method requires the solution of a linear system of equations with the matrix of partial derivatives $G_Q(Q, \mathcal{T})$ obtained by the discretized Euler–Lagrange equations. We are interested in tracking the minimal energy equilibrium configuration as the temperature $\mathcal{T}$ of the liquid crystals is varied. This gives rise to a path-following problem, which we solve by using the Euler–Newton continuation method; see, e.g., [1, 17] for discussions on continuation methods. Points on the solution path at which $G_Q(Q, \mathcal{T})$ is singular are referred to as singular points. We are interested in determining these points because they may be bifurcation points for the minimal energy equilibrium configuration. When the continuation method finds a point on a solution path close to a singular point, we use the secant method to accurately determine the location of the singular point. Thus, we use the secant method to determine a temperature $\mathcal{T}_0$ such that the minimum eigenvalue $\lambda_1(\mathcal{T}_0)$ of $G_Q(Q, \mathcal{T}_0)$ vanishes. This is described in the following algorithm.

ALGORITHM 1.1. Computation of a singular point.

```
Input: $\lambda_1^{(0)}$, $\mathcal{T}^{(0)}$, $\mathcal{T}^{(1)}$, $tol$; Output: $\mathcal{T}_0$, $\lambda_1(\mathcal{T}_0)$;
1. $j := 1$;
2. Minimize $F(Q, \mathcal{T}^{(j)})$;
3. Compute the smallest eigenvalue $\lambda_1^{(j)}$ of the matrix $G_Q(Q, \mathcal{T}^{(j)})$;
4. if $|\lambda_1^{(j)}| \le tol$ then $\mathcal{T}_0 := \mathcal{T}^{(j)}$; $\lambda_1(\mathcal{T}_0) := \lambda_1^{(j)}$; exit endif;
5. $\mathcal{T}^{(j+1)} := \mathcal{T}^{(j)} - \lambda_1^{(j)} \frac{\mathcal{T}^{(j)} - \mathcal{T}^{(j-1)}}{\lambda_1^{(j)} - \lambda_1^{(j-1)}}$; $j := j + 1$;
6. go to 2.
```

We use Algorithm 4.1 of Section 4 to compute the smallest eigenvalue of the matrix of partial derivatives in step 3 of Algorithm 1.1. At the singular point the solution path might bifurcate. One technique for switching paths at bifurcation points is to determine the tangent vectors to the different branches of the solution path by solving a nonlinear system of polynomial equations of small order. Specifically, one solves the *algebraic bifurcation equations* when the partial derivative $G_{\mathcal{T}}(Q, \mathcal{T})$ at $\mathcal{T} = \mathcal{T}_0$ is in the range of $G_Q(Q, \mathcal{T}_0)$, and the *limit point bifurcation equations* otherwise; see, e.g., [16, 17] and references therein for further discussions. We can follow the desired solution path from a bifurcation point by taking a Newton step in the direction of an appropriate tangent vector.

The eigenvectors associated with the zero eigenvalues of the matrix $G_Q(Q, \mathcal{T}_0)$ are required in the algebraic bifurcation equations for computing the tangent vectors of the solution path at a bifurcation point. Therefore it is necessary to determine the location of singular points and to compute the null space of $G_Q(Q, \mathcal{T}_0)$ at these points. In the application to liquid crystal modeling described in this paper, we also need to compute the dimension of the null space of $G_Q(Q, \mathcal{T}_0)$ at a singular point. Due to symmetry this dimension is frequently a multiple of three. In view of the large order of $G_Q(Q, \mathcal{T}_0)$ in our application, it is highly desirable that the numerical method used for computing the wanted eigenvalues and eigenvectors requires little computer storage in addition to the storage needed for the computed eigenvectors. In fact, in order to reduce the storage requirement, we do not store all nonzero entries of $G_Q(Q, \mathcal{T}_0)$ simultaneously, but instead calculate them as needed when evaluating matrix–vector products with $G_Q(Q, \mathcal{T}_0)$. The FORTRAN code for evaluating these matrix–vector products was generated by the symbolic formula manipulation language Maple V.

This paper describes the implicitly restarted block Lanczos (IRBL) method for the computation of a few extreme eigenvalues of a large sparse symmetric matrix and applies the method to the computation of a few of the smallest eigenvalues and associated eigenvectors of the matrix $G_Q(Q, \mathcal{T})$ introduced above. We denote the block size by r and sometimes refer to our block scheme as the IRBL(r) method. This method generalizes the implicitly restarted Lanczos (IRL) method introduced by Sorensen [33] and more recently studied by Lehoucq and Sorensen [18, 20]. The IRBL method is based on the recursions formula for the block Lanczos method, described, e.g., by Chatelin [7], Golub and Underwood [14], Grimes *et al.* [15], and Ruhe [29]. Similarly to the block Lanczos method, the IRBL method is well suited for the computation of multiple or very close eigenvalues. The main advantage of the IRBL method, when compared with the block Lanczos method, is its smaller storage requirement when both eigenvalues and associated eigenvectors are required.

The IRBL method can be regarded as a curtailed block QR algorithm for the symmetric eigenvalue problem; see, e.g., Bai and Demmel [4], and Dubrulle and Golub [10] for discussions of the latter. Similarly as in the block QR algorithm, the choice of shifts is important for the IRBL method. However, obvious generalizations of the Rayleigh or Wilkinson shifts that can be used in a block QR algorithm cannot be applied in the IRBL method, because the data required to compute these shifts is not available. We therefore propose a new shift selection strategy.

We remark that shift selection in the IRL method, and the closely related implicitly restarted Arnoldi (IRA) method, has received considerable attention and is studied in [3, 6, 20, 23, 33]. ARPACK by Lehoucq *et al.* [21] implements the IRL and IRA methods as described by Sorensen [33]. Computed examples in Section 5 illustrate that this implementation of the IRL method does not reliably detect multiple eigenvalues; neither does the IRL method described in [3]. A large number of numerical experiments, some of which are reported in Section 5, indicate that the IRBL method of the present paper reliably determines extreme eigenvalues with correct multiplicity and the associated eigenvectors. When the block size is chosen to be one, the IRBL method reduces to the IRBL(1) method, which is an IRL method. In our experience the IRBL(1) method also reliably determines extreme eigenvalues with correct multiplicity. The IRBL(1) method differs from the previously described IRL methods in the selection of Krylov subspace after an eigenvalue–eigenvector pair has been found and in the choice of shifts. The main advantage of the IRBL(1) method over the IRBL(r) methods for $r > 1$ is that it requires less computer storage. However, the IRBL(1) method may require more arithmetic operations to determine the desired eigenvalue–eigenvector pairs. This is illustrated in Section 5.

For any block size $r \geq 1$, the IRBL(r) method is a polynomial acceleration method with a special choice of accelerating polynomial. Polynomial acceleration for eigenvalue computation was first used by Flanders and Shortley [13], who applied Chebyshev polynomials. More recent applications of Chebyshev polynomials as accelerating polynomials are described by Chatelin [7] and Saad [30].

We note that when a suitable preconditioner for A is known, the Davidson method and extensions thereof can be competitive for the computation of a few eigenvalues; see Davidson [8], Morgan and Scott [24], Murray *et al.* [25], and Sleijpen and van der Vorst [32]. The determination of a suitable preconditioner for the liquid crystal problem that we focus on in this paper requires further study, and we therefore only consider methods that just require the matrix A.

This paper is organized as follows. In Section 2 we review the block Lanczos method and develop the recursion formulas for the IRBL method. Section 3 describes our strategies for subspace and shift selections, and Section 4 presents our new IRBL algorithm. Numerical examples are displayed in Section 5, and concluding remarks can be found in Section 6.

2. THE IRBL METHOD

Let $\{v_j\}_{j=1}^r$ be a given set of orthonomal n-vectors, and introduce the matrix $V_r = [v_1, v_2, \ldots, v_r]$. An application of m steps of the block Lanczos method with initial matrix V_r reduces the $n \times n$ symmetric matrix A to a symmetric block tridiagonal matrix T_{mr} with $r \times r$ blocks and upper triangular subdiagonal blocks, such that

$$AV_{mr} = V_{mr}T_{mr} + F_r E_r^{\mathrm{T}}, \tag{2.1}$$

where $V_{mr} \in \mathbf{R}^{n\times mr}$, $V_{mr} I_{mr\times r} = V_r$, $V_{mr}^{\mathrm{T}} V_{mr} = I_{mr}$, and $F_r \in \mathbf{R}^{n\times r}$ satisfies $V_{mr}^{\mathrm{T}} F_r = 0$. As usual I_{mr} denotes the $mr \times mr$ identity matrix, and $I_{mr\times r} \in \mathbf{R}^{mr\times r}$ consists of the first r columns of I_{mr}. The matrix $E_r \in \mathbf{R}^{mr\times r}$ consists of the last r columns of I_{mr}. We refer to (2.1) as a block Lanczos decomposition, and the space $\mathbf{K}_{mr} :=$ range V_{mr} as a Krylov subspace.

Let θ be an eigenvalue of the matrix T_{mr}, 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 $x = V_{mr}y$ 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

$$\|Ax - x\theta\| = \|(AV_{mr} - V_{mr}T_{mr})y\| = \left\|F_r E_r^{\mathrm{T}} y\right\|. \tag{2.2}$$

Throughout this paper $\|\cdot\|$ denotes the Euclidean vector norm as well as the associated induced matrix norm. Thus, the norm of the residual error can be determined without explicitly computing the Ritz vector x by evaluating the right-hand side of (2.2). When the norm (2.2) 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 block Lanczos method, one fixes the block size r and increases m until the right-hand side of (2.2) is sufficiently small. Then one computes the Ritz pair $\{\theta, x\}$. When the order of the matrix A and the number of Lanczos steps m are large, secondary computer storage may have to be used to store V_{mr}. This can slow down the computations significantly. The use of secondary computer storage can be avoided by restarting the block Lanczos process periodically, and the IRBL method provides recursion formulas for this purpose.

The IRBL method generalizes the IRL method by Sorensen [33]. Assume for definiteness that we are interested in computing the k smallest eigenvalues and associated eigenvectors of the matrix A, where k is a fixed and fairly small number. Let m steps of the block Lanczos method produce the block Lanczos decomposition (2.1).

Let $z \in \mathbf{R}$ and determine the QR factorization $T_{mr} - zI_{mr} = QR$, where $Q, R \in \mathbf{R}^{mr \times mr}$, $Q^{\mathrm{T}} Q = I_{mr}$, and R is upper triangular. We obtain

$$(A - zI)V_{mr} - V_{mr}(T_{mr} - zI_{mr}) = F_r E_r^{\mathrm{T}}, \tag{2.3.1}$$

$$(A - zI)V_{mr} - V_{mr}QR = F_r E_r^{\mathrm{T}}, \tag{2.3.2}$$

$$(A - zI)(V_{mr}Q) - (V_{mr}Q)(RQ) = F_r E_r^{\mathrm{T}} Q, \tag{2.3.3}$$

$$A(V_{mr}Q) - (V_{mr}Q)(RQ + zI_{mr}) = F_r E_r^{\mathrm{T}} Q. \tag{2.3.4}$$

Let $T_{mr}^+ = RQ + zI_{mr}$. Then T_{mr}^+ is a symmetric block tridiagonal matrix with the same band width as T_{mr}. The matrix Q in the QR factorization of $T_{mr} - zI_{mr}$ is a generalized upper Hessenberg matrix, whose lower triangular part has band width r.

The formulas (2.3) are similar to recursion formulas for the explicitly shifted block QR method, and in analogy with the terminology for the latter method, we refer to z as a shift. After applying the $m-1$ shifts $z_1, z_2, \ldots, z_{m-1}$, we obtain

$$AV_{mr}^+ = V_{mr}^+ T_{mr}^+ + F_r E_r^{\mathrm{T}} Q^+, \tag{2.4}$$

where

$$V_{mr}^+ = [v_1^+, v_2^+, \ldots, v_{mr}^+] = V_{mr}Q^+, \quad Q^+ = Q_1 Q_2 \cdots Q_{m-1}, \quad T_{mr}^+ = (Q^+)^T T_{mr} Q^+$$

and Q_j denotes the orthogonal matrix associated with the shift z_j. Introduce the partitioning

$$T_{mr}^+ = \left(\begin{array}{c|cccc} T_r^+ & B_r^{\mathrm{T}} & 0 & \cdots & 0 \\ \hline B_r & & & & \\ 0 & & & & \\ \vdots & & T_{mr-r}^+ & & \\ 0 & & & & \end{array}\right), \tag{2.5}$$

where $T_r^+ \in \mathbf{R}^{r\times r}$, $B_r \in \mathbf{R}^{r\times r}$ is upper triangular, and $T_{mr-r}^+ \in \mathbf{R}^{(mr-r)\times(mr-r)}$. Equate the first r columns on the right-hand side and left-hand side of (2.4). We then obtain

$$AV_r^+ = V_r^+ T_r^+ + F_r^+, \tag{2.6}$$

where $V_r^+ = [v_1^+, v_2^+, \ldots, v_r^+]$ and $F_r^+ = [v_{r+1}^+, \ldots, v_{2r}^+]B_r + F_r E_r^T Q^+ I_{n\times r}$. The mth shift z_m is applied according to

$$V_r^{++} = F_r^+ + V_r^+(T_r^+ - z_m I_r). \tag{2.7}$$

Introduce $\hat{R} = (R_m^r R_{m-1}^r \ldots R_1^r)^{-1}$, where R_j^r is the first r columns and r rows of the upper triangular matrix R_j in the QR factorization of $T_{mr} - z_j I$. Then

$$V_r^{++} = \psi_m(A) V_r \hat{R}, \tag{2.8}$$

where ψ_m is a polynomial of degree m with zeros $z_1, z_2, \ldots, z_m$. Formula (2.8) shows that we can multiply the initial matrix V_r for the Lanczos method by an accelerating polynomial in A of degree m without evaluating any matrix-vector products with the matrix A, in addition to those matrix–vector products that were computed during m steps of the block Lanczos method. The choice of accelerating polynomial ψ_m, i.e., the choice of the shifts $z_1, z_2, \ldots, z_m$, is discussed in Section 3. Here we only note that we wish to choose the z_j so that range V_r^{++} is in, or close to, an invariant subspace of A associated with all or a subset of the k desired eigenvalues of A.

Having computed V_r^{++} in the manner outlined, we orthonormalize the columns of V_r^{++} and denote the orthonormal matrix so obtained by V_r. The block Lanczos process is now restarted with the initial matrix V_r. If the number of desired eigenvalues k is not larger than the block size r, then the computations proceed by periodically applying m steps of the block Lanczos process and m shifts until the desired k eigenvalue–eigenvector pairs have been found. After having applied m shifts q times, the relation (2.8) is replaced by

$$\tilde{V}_r^{++} = \psi_{mq}(A) V_r \tilde{R}, \tag{2.9}$$

where ψ_{mq} is a polynomial of degree mq with zeros $z_1, z_2, \ldots, z_{mq}$, and $\tilde{R}$ is an upper triangular matrix.

In view of that the eigenvalues of a block tridiagonal matrix $T_{mr} \in \mathbf{R}^{mr\times mr}$ with block size r and of rank larger than or equal to $mr - r$ are of multiplicity less than or equal to r, see, e.g., [7, Lemma 6.41, p. 268], the block Lanczos method with block size r can determine eigenvalues of multiplicity at most r. If $k > r$, we therefore proceed as follows in order to be able to compute all the k smallest eigenvalues and associated eigenvectors. We carry out the computations as described above until r desired eigenvalue–eigenvector pairs have been found. We then generate r random vectors, orthogonalize them against the orthogonal eigenvectors already computed, as well as against each other, by the modified Gram–Schmidt method. This gives the matrix $V_r = [v_1, v_2, \ldots, v_r]$, which we use to restart the block Lanczos method. This selection of matrix V_r after r eigenvalues have converged makes it possible to determine eigenvalues of multiplicity larger than r and associated eigenvectors. We also use this choice of starting vector for the Lanczos process when $r = 1$.

We remark that when $r = 1$ the selection of initial vector for the Lanczos method differs from the choice of initial vector proposed in [3]. The choice advocated in [3] typically yields faster convergence to k small eigenvalues and associated eigenvectors; however, when there are multiple or very close eigenvalues the eigenvalues found are not always the k smallest ones. The initial vector should be selected as proposed in the present paper when it is important that the computed eigenvalues are the very smallest ones.

The computations in our present implementation of the block Lanczos method are organized as proposed by Ruhe [29]. In this implementation the vectors in the Krylov subspace bases generated are orthogonalized sequentially. Orthogonality of the basis vectors is secured by reorthogonalization when necessary. Results in [5, 28] show that at most one reorthogonalization is required.

We note that on some computers and for certain matrices A, it may be possible to simultaneously evaluate r matrix–vector products with the matrix A faster than to evaluate r matrix–vector products with A sequentially. This depends on whether simultaneous evaluation may require less data movement. For such computers and matrices, it can be attractive to consider variants of the block Lanczos method that allow simultaneous evaluation of r matrix–vector products. It is straightforward to modify our code for the IRBL method accordingly.

The recurrence formulas (2.3) are related to the explicitly shifted block QR algorithm. However, for reasons of numerical stability, we use an implementation based on recurrence formulas associated with an implicitly shifted block QR algorithm. These recursions can be modified to allow double shifts. The latter is attractive when A is a general real matrix. A code for a restarted block Arnoldi method for the computation of a few eigenvalues and associated eigenvectors of a general real matrix A is presently being developed by Lehoucq and Maschhoff [19].

Assume that we already have determined $jr < k$ eigenvalue–eigenvector pairs, and are to apply the block Lanczos method to the matrix $V_r \in \mathbf{R}^{n\times r}$ with orthonormal columns. The eigenvectors already found require njr storage locations. In order not to increase the demand of computer storage, we only apply $m - j$ steps of the block Lanczos algorithm, which are followed by $m - j$ shifts.

3. SHIFT SELECTION

The rate of convergence of the IRBL method depends on the choice of accelerating polynomial ψ_{mq} in (2.9). We determine ψ_{mq} by prescribing its zeros. The description of our selection of zeros requires some notation. Let $\mathbf{K}$ be a closed and bounded interval 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

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

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

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

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 **K**, or sometimes briefly Leja points for **K**. When $w(z) = 1$, the weighted Leja points agree with the "classical" Leja points studied by Leja [22].

We choose the zeros of ψ_{mq} to be Leja points for certain intervals **K** that do not contain any of the desired k smallest eigenvalues of A. Because we use these zeros as shifts in our algorithm, we also refer to them as Leja shifts. The purpose of the Leja shifts is to dampen eigenvector components associated with undesired eigenvalues in the columns of the matrix V_r in (2.9).

We now describe how intervals **K** that do not contain any of the k smallest eigenvalues of A can be determined from the eigenvalues of the matrices T_{mr} generated by the IRBL method. We may assume that the subdiagonal $r \times r$ blocks of the block tridiagonal matrix T_{mr} defined by (2.1) are nonsingular, which implies that $\operatorname{rank}(T_{mr}) \geq rm - r$, because otherwise we have found an invariant subspace.

PROPOSITION 3.1. *Let*

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n \tag{3.3}$$

denote the eigenvalues of A, and let

$$\theta_1 \leq \theta_2 \leq \cdots \leq \theta_{mr} \tag{3.4}$$

be the eigenvalues of the symmetric block tridiagonal matrix T_{mr}, with nonsingular subdiagonal $r \times r$ blocks, in a block Lanczos decomposition (2.1). *Then*

$$\lambda_j \leq \theta_j, \quad 1 \leq j \leq mr, \tag{3.5}$$

$$\lambda_n \geq \theta_{mr}. \tag{3.6}$$

Moreover,

$$\lambda_j < \theta_{j+r}, \quad 1 \leq j \leq (m-1)r. \tag{3.7}$$

Proof. We obtain from (2.1) that $T_{mr} = V_{mr}^{\mathrm{T}} A V_{mr}$ and, therefore,

$$\lambda_k = \max_{S,\dim(S)=k} \min_{\substack{x\in S\\ x\neq 0}} \frac{x^{\mathrm{T}}Ax}{x^{\mathrm{T}}x} \leq \max_{S,\dim(S)=k} \min_{\substack{x\in S\cap\operatorname{span}\{V_{mr}\}\\ x\neq 0}} \frac{x^{\mathrm{T}}Ax}{x^{\mathrm{T}}x} = \max_{S,\dim(S)=k} \min_{\substack{y\in S\\ y\neq 0}} \frac{y^{\mathrm{T}}T_{mr}y}{y^{\mathrm{T}}y} = \theta_k.$$

This shows (3.5). The inequality (3.6) can be shown similarly. Finally, (3.7) follows from the fact that the eigenvalues of T_{mr} have multiplicity at most r. ■

Assume that $(m-1)r \geq k$ and let the integer p satisfy

$$0 \leq p \leq (m-1)r - k. \tag{3.8}$$

Then, by (3.7), the interval $\mathbf{K} = [\theta_{k+r+p}, \theta_{mr}]$ does not contain any of the k smallest eigenvalues of the matrix A. This suggests the following choice of intervals **K** during the computation with the IRBL method. For now, we let p be an arbitrary integer that satisfies (3.8). We discuss different choices of p at the end of this section.

Let $V_r \in \mathbf{R}^{n\times r}$ be the initial matrix with orthonormal columns for the block Lanczos method, compute the decomposition (2.1), determine the eigenvalues θ_j of the matrix T_{mr} in (2.1), and order them according to (3.4). Define the endpoints of the interval $\mathbf{K} = [a, b]$ by

$$a = \theta_{k+r+p}, \quad b = \theta_{mr}. \tag{3.9}$$

We let the m first shifts $\{z_j\}_{j=1}^{m}$ be Leja points for $\mathbf{K}$. Application of the m shifts as described in Section 2 yields a new matrix $V_r^{++} \in \mathbf{R}^{n\times r}$ defined by (2.7). We orthonormalize its columns and this gives a new matrix, which we also refer to as V_r. We now apply m steps of the block Lanczos method with initial matrix V_r in order to obtain a new block Lanczos decomposition (2.1), with a new block tridiagonal matrix T_{mr}. Compute its eigenvalues θ_j and order them according to (3.4). The endpoints of $\mathbf{K} = [a, b]$ are updated by the formulas

$$a = \theta_{k+r+p}, \quad b = \max\{b, \theta_{mr}\}. \tag{3.10}$$

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

$$w(z) = |z - \theta_{k+r+p}|, \tag{3.11}$$

where p satisfies (3.8).

ALGORITHM 3.2. Compute m shifts as Leja points for $\mathbf{K}$, given $m(q-1)$ shifts.

```
Input: endpoints of $\mathbf{K}$, $q$, $m$, $\{z_j\}_{j=1}^{m(q-1)}$ ; Output: $\{z_j\}_{j=m(q-1)+1}^{mq}$;
1. $j := m(q-1)+1$;
2. if $j = 1$ then
       $z_0 :=$ point of largest magnitude of $\mathbf{K}$
   else
       Determine $z_j \in \mathbf{K}$, such that
       $w(z_j)\prod_{i=1}^{j-1}|z_j - z_i| = \max_{z\in\mathbf{K}} w(z)\prod_{i=1}^{j-1}|z - z_i|$,
       where $w(z)$ is defined by (3.11)
   endif;
3. $j := j+1$;
4. if $j < mq$ then go to 2 else stop.
```

The computation of the Leja points z_j, $j \geq 1$, by Algorithm 3.2 requires the maximization of a product over $\mathbf{K}$. In order to reduce the computational effort necessary to determine Leja points, we discretize each updated interval $\mathbf{K}$ using zeros of a Chebyshev polynomial of the first kind of degree ℓ for the interval $\mathbf{K}$, where ℓ is sufficiently large.

We turn to the selection of integer p in (3.9)–(3.11). In [3] we described an algorithm that corresponds to the case when $r = 1$ and considered eigenvalue problems in which the desired eigenvalues were fairly well separated from the undesired ones. We found $p = 0$ to be appropriate. This corresponds to $a = \theta_{k+1}$ in (3.9)–(3.10). However, when the largest desired eigenvalues of A are close to the smallest undesired eigenvalues of A and m is small, e.g. $m \leq 5$, faster convergence can be achieved with a larger value of p. Increasing p moves the left endpoint of the interval $\mathbf{K} = [a, b]$ away from the desired eigenvalues. For the examples of the present paper, we found

$$p = (m - 1)r - k - 1 \tag{3.12}$$

to give rapid convergence. This corresponds to $a = \theta_{mr-1}$ in (3.9)–(3.10). This is the largest value of p for which the interval $\mathbf{K}$, in general, will be a proper interval that does not contain the desired eigenvalues of A. Example 5.4 in Section 5 illustrates the effect of different choices of p.

4. THE IRBL ALGORITHM

We describe our algorithm for computing the k smallest eigenvalues $\{\lambda_j\}_{j=1}^k$ and associated orthonormal eigenvectors $\{u_j\}_{j=1}^k$ of a large symmetric matrix A. If the k smallest eigenvalues form clusters of very close or multiple eigenvalues, and it is known that the largest cluster contains ℓ eigenvalues, then block size $r = \ell$ is appropriate, because the matrix T_{mr} can then have as many multiple eigenvalues as there may be in the set $\{\lambda_j\}_{j=1}^k$. If all the k desired eigenvalues form a cluster, then, if possible, r should be chosen to be equal to k. However, in many applications the multiplicity of the desired eigenvalues is not known a priori, and choosing $r = k$ can be prohibitive due to the requirement of computer storage. The use of various block sizes is illustrated in the examples of Section 5.

Let $\{\theta_j, y_j\}_{j=1}^{mr}$ denote eigenvalue–eigenvector pairs of the symmetric block tridiagonal matrix T_{mr} defined by (2.1) and assume that the eigenvalues are ordered according to (3.4). Let $x_j = V_{mr} y_j$ be a Ritz vector of the matrix A, associated with the Ritz value θ_j. Then, analogously with (2.2), we obtain that

$$\|Ax_j - x_j\theta_j\| = \|(AV_{mr} - V_{mr}T_{mr})y_j\| = \left\|F_r E_r^{\mathrm{T}} y_j\right\|, \quad 1 \leq j \leq mr.$$

The stopping criterion

$$\max_{1\leq j\leq k} \left\|F_r E_r^{\mathrm{T}} y_j\right\| \leq \epsilon \|A\|, \tag{4.1}$$

where ϵ is a user supplied positive constant, would secure that each computed Ritz pair $\{\theta_j, x_j\}$ would be an eigenpair of a matrix $A + \Delta_j$, where $\Delta_j \in \mathbf{R}^{n\times n}$ satisfies $\|\Delta_j\| \leq \epsilon$.

The evaluation of $\|A\|$ is impractical for large matrices A, however, the IRBL method furnishes good approximations of $\|A\|$. Specifically, we approximate $\|A\|$ by the largest norm of all the matrices T_{mr} generated by the IRBL method. Thus, in our algorithm we use the stopping criterion

$$\max_{1\leq j\leq k} \left\|F_r E_r^{\mathrm{T}} y_j\right\| \leq \epsilon \max\|T_{mr}\|, \tag{4.2}$$

where the maximum in the right-hand side is taken over all the block tridiagonal matrices T_{mr} generated by the block Lanczos method.

In the following algorithm for computing the k smallest eigenvalues and associated eigenvectors of the matrix A, we typically restart the block Lanczos method with r random vectors when r eigenpairs have been found in order to be able to detect all eigenpairs. This corresponds to setting the parameter “random” to true in the algorithm. This choice is appropriate if we do not know how the k smallest eigenvalues of A are distributed.

In the numerical examples of Section 5, we also illustrate the behavior of the algorithm when the parameter random is set to false. In this case the most recently available Krylov subspace basis is orthogonalized against all determined eigenvectors and used as initial matrix when restarting the block Lanczos method. In our experience the latter approach reliably yields all the eigenvalues if the block size r is at least as large as the number of eigenvalues in the largest cluster among the desired eigenvalues. In this case often all desired eigenpairs are determined faster when random is set to false than when random is true; see Example 5.1.

ALOGRITHM 4.1. IRBL(r) method for computing k eigenpairs of A associated with the smallest eigenvalues.

```
Input: $A, k, m, r, \epsilon$, such that $(m-1)r \geq k$;
       random := true, if restart with random vectors,
       random := false, otherwise;
Output: eigenvalues $\{\lambda_j\}_{j=1}^{k}$, orthonormal eigenvectors $\{u_j\}_{j=1}^{k}$;
 1. $i_{\text{conv}} := 0$;
 2. Choose r random vectors $v_j$ and let $V_r = [v_1, v_2, \ldots, v_r]$;
 3. $i_{\text{shift}} := 0$;
 4. Orthogonalize the columns of $V_r$ against the $i_{\text{conv}}$ converged eigenvectors.
    Orthonormalize the columns of $V_r$;
 5. Apply m steps of the block Lanczos method to the matrix A with initial orthonor-
    mal matrix $V_r$ in order to determine the matrices $T_{mr}$, $V_{mr}$, and $F_r$ in (2.1);
    The vectors generated are reorthogonalized against already determined columns
    of $V_{mr}$ as well as against already converged eigenvectors;
 6. Compute the eigenvalues (3.4) of $T_{mr}$;
 7. Check whether any new eigenpairs have converged:
    Let $\|F_r E_r^{\mathrm{T}} y_j\| \leq \epsilon \max\|T_{mr}\|$[5] be satisfied for $\ell$ of the mr indices j;
    if $k - i_{\text{conv}} \leq \ell$ and not random then
        Store $\ell$ converged eigenpairs; exit
    endif;
    if $\ell \geq r$ then
        Store the eigenpairs associated with the r smallest of the $\ell$ newly
        converged eigenvalues;
        $i_{\text{conv}} := i_{\text{conv}} + r$; $m := m - 1$;
        if random then go to 2
    endif;
 8. if $i_{\text{shift}} = 0$ then define the interval $\mathbf{K} = [a, b]$ by (3.9) else by (3.10);
 9. Compute m Leja points $\{z_j\}_{j=i_{\text{shift}}+1}^{i_{\text{shift}}+m}$ for $\mathbf{K}$ in the presence of the points $\{z_j\}_{j=1}^{i_{\text{shift}}}$
    by Algorithm 3.2;
10. Apply shifts $\{z_j\}_{j=i_{\text{shift}}+1}^{i_{\text{shift}}+m}$ according to (2.3)-(2.7) and let $V_r := V_r^{++}$, where
    $V_r^{++}$ is defined by (2.7);
11. $i_{\text{shift}} := i_{\text{shift}} + m$; go to 4.
```

The design of Algorithm 4.1 is motivated by its performance in numerous numerical experiments. Theoretical results on the algorithm are still incomplete. Difficulties in the analysis stem from the fact that the interval **K** keeps changing during iterations. Our selection

[5] $\max\|T_{mr}\|$ denotes the maximum over all $1 + i_{\text{shift}}/m$ matrices T_{mr} generated so far.

of the interval **K** is based on extensive numerical experiments, some of which are reported in Section 5. The choice of weight function (3.11) also is motivated by numerical experiments; this weight function gave faster convergence than $w(z) = 1$.

Algorithm 4.1 can be enhanced. For instance, it is quite straightforward to implement a change of block size during the iterations. It may be attractive to reduce the block size when i_{conv} eigenpairs have been computed and $k - i_{\text{conv}} < r$.

5. NUMERICAL EXAMPLES

This section presents some computed examples that illustrate the performance of Algorithm 4.1. The algorithm was implemented in both MATLAB and FORTRAN. In Examples 5.5–5.8 we tracked the equilibrium configuration of the liquid crystals. Due to the large size of the matrices used in tracking the equilibrium configuration the FORTRAN code was used; the MATLAB code was used for all other examples. All numerical experiments were carried out on HP work stations using double precision arithmetic, i.e., with approximately 16 significant digits. Unless stated otherwise, the parameter p in (3.9) and (3.10) is defined by (3.12). We compare Algorithm 4.1 with subroutines in ARPACK by Lehoucq *et al.* [21] and with the subroutine DNLASO of the package LASO2 by Scott [31]. ARPACK implements the IRL method with “exact shifts” as described by Sorensen [33]. Thus, in order to compute the k smallest eigenvalues of A, ARPACK applies, say, mr steps of the Lanczos method to build up an orthogonal basis of a Krylov subspace of dimension mr, and to determine a Lanczos decomposition with a symmetric tridiagonal matrix $T \in \mathbf{R}^{mr \times mr}$. Then ARPACK applies the $mr - k$ largest eigenvalues of $T \in \mathbf{R}^{mr \times mr}$ as shifts z_j. Now a new Lanczos decomposition is computed, and the largest eigenvalues of a new symmetric tridiagonal matrix T are used as shifts, and so on. The use of exact shifts often requires mr to be chosen substantially larger than k; see the computed examples in [3, 6] for the case $r = 1$.

The subroutine DNLASO implements the Lanczos method with selective reorthogonalization (see [26]) and allows the user to specify the 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 associated eigenvectors. The subroutine allows the user to select block-size for the Lanczos method. 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 and MAXJ is required to be larger than or equal to $6 \cdot$ NBLOCK. The storage requirement for the block Lanczos vectors generated by DNLASO is $n \cdot$ MAXJ storage locations. The total storage requirement for DNLASO is larger than $n \cdot (\text{MAXJ} + \text{NBLOCK})$, in addition to the storage needed to represent the matrix A. DNLASO and ARPACK are more sophisticated than our experimental code for Algorithm 4.1 and have multiple stopping criteria. This makes a comparison between DNLASO, ARPACK and Algorithm 4.1 difficult. The subroutine DNLASO allows the specification of a parameter NFIG, the number of desired correct decimal digits in the computed eigenvalue approximations. In all examples, we chose NFIG so as to give the same accuracy as Algorithm 4.1. ARPACK is designed to terminate the computations when $|\lambda_j - \lambda_j^{(c)}| < \text{TOL}|\lambda_j^{(c)}|$ for $1 \leq j \leq k$, where $\lambda_j^{(c)}$ denotes a computed approximation of λ_j. The parameter TOL is chosen by the user; a large value of TOL led ARPACK to fail to detect some multiple eigenvalues. In all examples we chose the largest value of TOL for which ARPACK gave the same accuracy as Algorithm 4.1 without missing any desired eigenvalues. The iterations with our code for Algorithm 4.1 were terminated when condition (4.2) was satisfied.

In all computed examples with block size one we determined the first initial Lanczos vector by generating an n-vector with normally distributed random numbers N(0, 1) as entries, and then normalized the vector to have unit length. In experiments with block sizes larger than one, the first vector in the initial block is the initial vector used in experiments with block-size one. The entries of the other vectors in the initial block are generated analogously. The advantage of using normally distributed random numbers instead of uniformly distributed ones has been pointed out by Ericsson and Ruhe [11]. The initial vector used in DNLASO, ARPACK and our code is the same for each particular example, but may differ for different examples.

Several tables have a column labeled "maximum error." Let $\lambda_j^{(c)}$ denote the computed approximation of λ_j. This column displays $\max_{1\le j\le k}|\lambda_j^{(c)} - \lambda_j|$. The column labeled "No. matrix–vector products" reports the number of matrix–vector product evaluations with the matrix A required to satisfy the stopping criterion. The evaluation of the product of A with an $n \times r$ matrix counts as r matrix–vector products.

EXAMPLE 5.1. Let $A = \mathrm{diag}(a_{11}, a_{22}, \ldots, a_{100,100})$ have entries

$$a_{ii} = \begin{cases} 1 \times 10^{-10}, & \text{if } 1 \le i \le 4, \\ \frac{i^2}{100}, & \text{if } 5 \le i \le 100. \end{cases}$$

We wish to compute the five smallest eigenvalues and associated eigenvectors of A, and used Algorithm 4.1 with $\epsilon = 1 \times 10^{-9}$, block sizes $1 \le r \le 5$, and several values of m. Unless explicitly stated otherwise, the parameter random in Algorithm 4.1 is set to true. The algorithm determines the eigenvalues in increasing order. Results are displayed in Table I, which shows the number of matrix–vector products required for the computation of every set of r eigenpairs for different block sizes r, as well as the number of matrix-vector products required for the computation of the remaining five mod r eigenpairs associated with the largest of the k desired eigenvalues. For instance, with block size $r = 2$, the computation of the first two eigenpairs to desired accuracy required the evaluation of 298 matrix–vector products, and the computation of the next two eigenpairs required the evaluation of 318 additional matrix–vector products. The determination of the fifth eigenpair required the computation of 300 further matrix–vector products.

TABLE I
Example 5.1: IRBL Method

Block size r	Consecutive shifts m	No. Lanczos vectors mr	No. matrix–vector products λ_1	λ_2	λ_3	λ_4	λ_5	Total	Maximum error
1	10	10	169	161	183	174	186	873	8.33×10^{-16}
2	5	10	-	298	-	318	300	916	1.78×10^{-15}
3	5	15	-	-	312	-	324	636	4.44×10^{-16}
4	5	20	-	-	-	396	320	716	3.89×10^{-16}
5	4	20	-	-	-	-	400	400	5.55×10^{-16}
4	5	20	Random = false in Alg. 4.1					380	2.83×10^{-16}

TABLE II
Example 5.1: ARPACK

No. Lanczos vectors	No. matrix–vector products	Maximum error
10	2868	9.99×10^{-15}
15	946[a]	2.00×10^{-14}
20	786[a]	9.99×10^{-15}
25	978	1.40×10^{-14}
30	1017	1.66×10^{-14}

[a] Not all of the multiple eigenvalues found.

The last row of Table I shows the number of matrix–vector product evaluations required by Algorithm 4.1 when $r = 4$ and random is set to false. These choices of r and random are appropriate if we a priori know that there is a cluster of four eigenvalues that is well separated from the fifth eigenvalue. Table I shows that $r = 4$ and random = false yields the fastest convergence. This depends on that the matrix V_r used in the restart after four eigenpairs have been found, contains more useful information about the fifth desired eigenpair than a random matrix. We note that Algorithm 4.1 was able to compute all the desired eigenpairs for all choices of r and random.

In ARPACK we set $\text{TOL} = 1 \times 10^{-16}$ in order not to miss any of the multiple eigenvalues. Table II shows the number of matrix–vector products required by ARPACK and the largest error in any one of the the computed eigenvalues. ARPACK failed to detect one of the multiple eigenvalues when the number of Lanczos vectors used was 15 and 20. We remark that a locking-and-purging strategy for ARPACK, suggested by Lehoucq and Sorensen [20], may enable this method to detect more multiple eigenvalues. However, this strategy is not implemented in the available code for ARPACK.

Table III shows the number of matrix–vector products required and the largest error in any one of the eigenvalues that were computed by the subroutine DNLASO for block sizes $1 \leq \text{NBLOCK} \leq 5$, and $\text{NFIG} = 10$. For $1 \leq \text{NBLOCK} \leq 3$, we let $\text{MAXJ} = 20$, and for NBLOCK equal to 4 or 5, we use $\text{MAXJ} = 30$. These values of MAXJ satisfy $\text{MAXJ} \geq 6 \cdot \text{NBLOCK}$, as required by DNLASO. The subroutine DNLASO failed to detect all the multiple eigenvalues for block sizes 1 and 3. In this example Algorithm 4.1 required the least computer storage and matrix-vector evaluations, and accurately determined all desired eigenpairs.

TABLE III
Example 5.1: DNLASO

Block size NBLOCK	No. Lanczos vectors MAXJ	No. matrix–vector products	Maximum error
1	20	1439[a]	1.47×10^{-9}
2	20	1509	5.27×10^{-10}
3	20	1910[a]	7.49×10^{-11}
4	30	2069	1.61×10^{-11}
5	30	2630	2.55×10^{-12}

[a] Not all of the multiple eigenvalues found.

We emphasize that the purpose of the computed examples of this section is to show the performance of the algorithms when only few vectors can be stored in fast computer memory. An increase in m would reduce the matrix-vector products required to determine the desired eigenpairs. Related computed examples can be found in [3].

EXAMPLE 5.2. This example is identical with Example 5.1, except for the selection of shifts. In particular, we used the same matrix A, initial matrix V_r, and value of ϵ. Instead of Leja shifts, we used the zeros of the mth degree Chebyshev polynomials of the first kind for the intervals $\mathbf{K} = [a, b]$ generated by Algorithm 4.1 as shifts. Thus, for each interval $\mathbf{K} = [a, b]$, we applied the shifts

$$z_j = \frac{b-a}{2} \cos\left(\frac{2j-1}{2m}\pi\right) + \frac{b+a}{2}, \quad 1 \le j \le m. \tag{5.1}$$

This choice of shifts is quite natural, because among all monic polynomials of degree m, the monic mth degree Chebyshev polynomial is of smallest magnitude on $\mathbf{K}$. Thus, the polynomial ψ_{mq} in (2.9) is a product of Chebyshev polynomials of degree m for different intervals $\mathbf{K}$. Table IV displays the results of the computations, and is analogous to Table I. The results are typical for many numerical examples; Leja shifts yield faster convergence than Chebyshev shifts (5.1). This depends on that the Chebyshev shifts for a given interval only depend on the endpoints of that interval, while Leja shifts also depend on the location of previously applied shifts.

EXAMPLE 5.3. Let A be the 900×900 matrix obtained by discretizing the two-dimensional negative Laplace operator on the unit square by the standard 5-point stencil with Dirichlet boundary conditions. We wish to determine the eigenpairs associated with the six smallest eigenvalues of A. It is well known that the largest multiplicity of the desired eigenvalues is 2. Specifically,

$$\lambda_1 < \lambda_2 = \lambda_3 < \lambda_4 < \lambda_5 = \lambda_6 < \cdots;$$

see, e.g., [34, Section 8.4]. We choose block size $r = 2$ and set random to false. Letting $m = 5$ yields a subspace of dimension $mr = 10$. For $\epsilon = 1 \times 10^{-5}$. Algorithm 4.1 required the evaluation of 232 matrix–vector products with the matrix A and gave a maximum error over all computed eigenvalues of 5.82×10^{-9}.

TABLE IV

Example 5.2: IRBL Method with Zeros of Chebyshev Polynomials as Shifts

Block size r	Consecutive shifts m	No. Lanczos vectors mr	No. matrix–vector products λ_1	λ_2	λ_3	λ_4	λ_5	Total	Maximum error
1	10	10	199	188	199	195	258	1039	6.49×10^{-15}
2	5	10	-	358	-	550	792	1700	1.19×10^{-14}
3	5	15	-	-	432	-	444	876	3.44×10^{-16}
4	5	20	-	-	-	436	384	820	1.55×10^{-15}
5	4	20	-	-	-	-	580	580	9.43×10^{-16}
4	5	20	random = false in Alg. 4.1					484	2.22×10^{-16}

When using ARPACK to compute these eigenpairs, we had to set TOL to 1×10^{-10}; a larger value of TOL resulted in ARPACK missing multiple eigenvalues. When allowing 10 Lanczos vectors, i.e., $mr = 10$, ARPACK required the evaluation of 1329 matrix–vector products with the matrix A, and the largest error in a computed eigenvalues was 9.37×10^{-12}. Increasing the number of Lanczos vectors to 20 reduced the number of matrix–vector products required by ARPACK to 334 and gave a maximum error in the computed eigenvalues of 1.88×10^{-12}.

An application of DNLASO with block size 2, MAXJ = 20, and NFIG = 10 required the evaluation of 1094 matrix–vector products and gave a maximum error in the computed eigenvalues of 1.11×10^{-10}. Thus, Algorithm 4.1 required the fewest matrix–vector product evaluations with A, and the least computer storage.

EXAMPLE 5.4. Let $A = \text{diag}(a_{11}, a_{22}, \ldots, a_{100,100})$ with entries

$$a_{ii} = \begin{cases} 1 \times 10^{i-11}, & \text{if } 1 \le i \le 4, \\ \frac{i^2}{100}, & \text{if } 5 \le i \le 100. \end{cases}$$

We seek to compute the k smallest eigenvalues and associated eigenvectors for $1 \le k \le 4$ and use Algorithm 4.1 with $m = 5$, $r = 4$, $\epsilon = 1 \times 10^{-8}$, and random = false. We wish to illustrate the performance of the algorithm for different choices of the parameter p in (3.9) and (3.10).

Table V displays the results achieved with Algorithm 4.1. The largest error in any one of the computed eigenvalues was 4.71×10^{-14}. We allowed the evaluation of at most 5000 matrix–vector products with the matrix A, and the computations were terminated when this number was exceeded before the desired k eigenpairs had been found. Table V illustrates the effect of the choice of p, i.e., the effect of the choice of left endpoint of the interval $\mathbf{K} = [a, b]$. If this endpoint is close to the desired eigenvalues, then we get poor or no convergence. Faster convergence is achieved when the left endpoint is moved away from the desired eigenvalues.

TABLE V
Example 5.4: IRBL Method; $m = 5$, $r = 4$

	No. matrix–vector products			
p	$k = 1$	$k = 2$	$k = 3$	$k = 4$
0	900	760	620	560
1	680	640	580	600
2	600	540	540	500
4	540	500	480	440
6	460	460	480	420
8	440	420	400	340
10	400	340	340	360
11	340	340	360	340
12	340	360	340	
13	360	340		
14	340			

ARPACK required the evaluation of 570 matrix–vector products in order to find the smallest eigenvalue of A with a Krylov subspace of size 20 and $\mathrm{TOL} = 1 \times 10^{-7}$. The error in the computed eigenvalue was 3.45×10^{-14}. The computation of the four smallest eigenvalues by ARPACK required the evaluation of 889 matrix–vector products, with a maximum error over the computed eigenvalues of 2.31×10^{-14}.

When we tried to compute only the two smallest eigenvalues of A and associated eigenvectors with ARPACK, the scheme failed to converge. More precisely, setting $\mathrm{TOL} = 1 \times 10^{-7}$ and allowing a Krylov subspace with at most 20 Lanczos vectors and 180,020 matrix–vector products with the matrix A gave no convergence to the desired eigenpairs. Increasing the Krylov subspace to dimension 50 and allowing the evaluation of 480,050 matrix–vector products still did not produce the desired eigenpairs. Our experience was similar when we instead tried to determine the eigenpairs associated with the three smallest eigenvalues.

DNLASO with $\mathrm{NFIG} = 10$, $\mathrm{MAXJ} = 30$, and $\mathrm{NBLOCK} = 4$ determined the three smallest eigenvalues using 3487 evaluations of matrix–vector products with A, and the computation of the four smallest eigenvalues required the evaluation of 1304 matrix–vector products. However, DNLASO was not able to compute only the smallest or only the two smallest eigenvalues and associated eigenvectors before the computations were terminated because 16,000 matrix–vector products had been evaluated. Convergence could be achieved by reducing NFIG, but then the computed eigenvalues were inaccurate.

The remaining examples of this section are concerned with liquid crystal modeling. We used the values $\mathcal{L}_1 = \frac{1}{200}$, $\mathcal{L}_2 = \frac{3}{400}$, and $\mathcal{L}_3 = \frac{3}{400}$ for the elastic constants, and $\mathcal{B} = 2$ and $\mathcal{C} = \frac{2}{9}$ for the bulk constants in (1.4); see [36] for a discussion on the determination and relationship between these constants. We do not report details about the computational work required; such details have been provided in the previous examples already. Instead, we focus on how Algorithm 4.1 can be applied in the context of liquid crystal modeling.

EXAMPLE 5.5. We track the equilibrium configuration of the liquid crystals in the two-dimensional slab

$$\Omega = \{(x_1, x_2) : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}, \tag{5.2}$$

which is discretized by a 41×41 grid. Discretization of the Euler–Lagrange equations by finite differences on this grid, with the boundary condition determined by strong surface anchoring (1.6) with $Q_0 = 3vv^{\mathrm{T}} - I$, yields a matrix $G_Q(Q, \mathcal{T})$ of order $n = 5 \times 39^2 = 7605$. The initial tensors $Q = Q(p)$, $p \in \Omega$, are defined by $Q(p) = 3v(p)v(p)^{\mathrm{T}} - I$, where $v(p)$ is the unit vector directed toward the center of Ω.

We let the initial normalized temperature $\mathcal{T}$ of the liquid crystals be $\mathcal{T}_a = 0.375$ and track the equilibrium configuration for decreasing temperature until $\mathcal{T}_b = 0.280$. Figure 5.1 displays the smallest eigenvalue of the matrix $G_Q(Q, \mathcal{T})$ and the minimum energy of the liquid crystals versus $\mathcal{T}$. For temperatures larger than 1/3 the model presented here is not physically realistic as can be seen from the fact that the minimum energy of the system is a decreasing function of $\mathcal{T}$ for $\mathcal{T} \geq 1/3$; see Fig. 5.1b.

Four solution paths, path (1)–path (4), are determined. We followed path (1) to path (2). This was done with Algorithm 4.1 with input parameters $r = 1$, $\epsilon = 1 \times 10^{-9}$, $m = 10$, and $k = 1$. These parameter values are used during path following in Examples 5.7 and 5.8 as well. The initial temperature on path (1) is $\mathcal{T}_a = 0.375$ and the final temperature on path (2) is $\mathcal{T}_b = 0.280$. We found a singular point $(Q_0, \mathcal{T}_0)$ for $\mathcal{T}_0 = 0.3012$ with

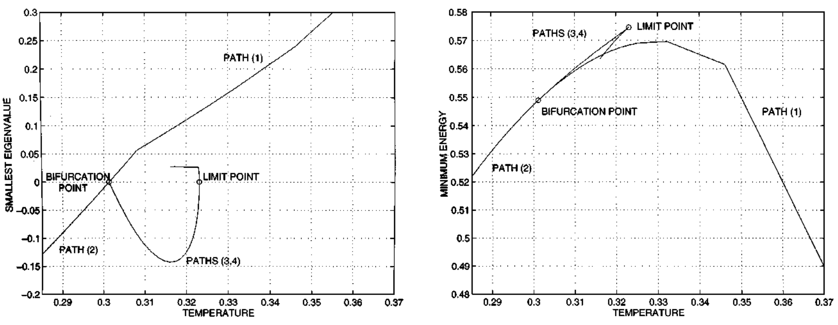

FIG. 5.1. Example 5.5: two-dimensional slab with strong surface anchoring of the liquid crystals. (a) Smallest eigenvalue of the matrix $G_Q(Q, \mathcal{T})$ vs temperature $\mathcal{T}$. Paths (3) and (4) are represented by the same curve. (b) Minimum energy vs temperature $\mathcal{T}$. The model is unrealistic for $\mathcal{T} \geq 1/3$.

Algorithm 1.1. A vector in the null space of $G_Q(Q_0, \mathcal{T}_0)$ is determined by Algorithm 4.1 simultaneously. We then used Algorithm 4.1 with input parameters $r = 3$, $\epsilon = 1 \times 10^{-9}$, $m = 9$, and $k = 3$ to compute the dimension and an orthonormal basis of the null space of $G_Q(Q_0, \mathcal{T}_0)$. For these computations, we chose an initial matrix $V_3 \in \mathbf{R}^{n\times 3}$ with random orthonormal columns that, moreover, are orthogonal to the vector in the null space already known. Null space computations with Algorithm 4.1 are carried out similarly in Examples 5.6–5.8. In the present example, we found the dimension of the null space to be one. The null space is orthogonal to $G_{\mathcal{T}}(Q_0, \mathcal{T}_0)$ and, therefore, the singular point is a bifurcation point.

We followed the intersecting paths through the point $(Q_o, \mathcal{T}_o)$, paths (3) and (4) in Fig. 5.1, by taking a step in the tangent direction along each path. Note that paths (3) and (4) are represented by the same curve. This is a consequence of the symmetry of the problem.

When traversing the intersecting curves, we passed another singular point $(Q_1, \mathcal{T}_1)$ for $\mathcal{T}_1 = 0.3232$; see Fig. 5.1. We used Algorithm 4.1 to compute the null space of the matrix $G_Q(Q_1, \mathcal{T}_1)$. The null space is of dimension one, and it is not orthogonal to $G_{\mathcal{T}}(Q_1, \mathcal{T}_1)$. Therefore the singular point is a limit point.

EXAMPLE 5.6. We track the equilibrium configuration of the liquid crystals in the two-dimensional slab (5.2) and model weak surface anchoring of the liquid crystal molecules with $\mathcal{W} = 10$ in (1.5). We use a symmetric finite difference discretization of the Euler–Lagrange equations described in [2] on a 41×41 grid and obtain a symmetric matrix $G_Q(Q, \mathcal{T})$ of order $n = 5 \times 41^2 = 8405$.

Let the initial temperature and tensors be Q_0 and $\mathcal{T}_0$ of Example 5.5 and minimize the free energy by using the Euler–Lagrange equations. The matrix $G_Q(Q_0, \mathcal{T}_0)$ so obtained has three negative eigenvalues. We follow the largest negative eigenvalue of $G_Q(Q_0, \mathcal{T}_0)$ with Algorithm 1.1 as we increase $\mathcal{T}$. This is the eigenvalue of smallest magnitude of $G_Q(Q, \mathcal{T})$, and this eigenvalue vanishes at the singular points $(Q, \mathcal{T})$ of this matrix. We obtain the singular point $(Q_2, \mathcal{T}_2)$ for $\mathcal{T}_2 = 0.3013$ and a vector in the null space of the matrix $G_Q(Q_2, \mathcal{T}_2)$. Algorithm 4.1 with $\epsilon = 1 \times 10^{-9}$, $m = 5$, $k = r = 4$, and random = false is applied to determine the entire null space of $G_Q(Q_2, \mathcal{T}_2)$. It is found to be of dimension one and is orthogonal to $G_{\mathcal{T}}(Q_2, \mathcal{T}_2)$. Therefore, the singular point is a bifurcation point.

Similarly as in Example 5.5, we tracked four different solution paths between the temperatures $\mathcal{T}_a = 0.37$ and $\mathcal{T}_b = 0.28$. Figure 5.2 shows the largest negative eigenvalue of the

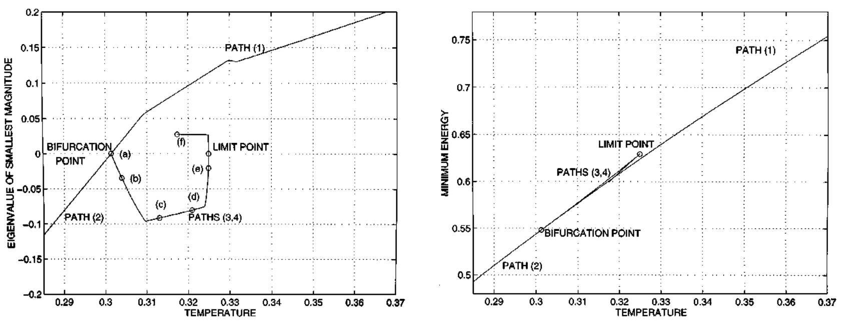

FIG. 5.2. Example 5.6: two-dimensional slab with weak surface anchoring of the liquid crystals. The points in the graphs labeled (a)–(f) correspond to the images (a)–(f) in Fig. 5.3 (a) Eigenvalue of smallest magnitude of the matrix $G_Q(Q, \mathcal{T})$ vs temperature $\mathcal{T}$. (b) Minimum energy vs temperature $\mathcal{T}$.

matrix $G_Q(Q, \mathcal{T})$ and the minimum energy of the liquid crystals versus the temperature. The model with weak surface anchoring of the liquid crystals displays physically reasonable behavior also for $\mathcal{T} \geq 1/3$.

Figures 5.3a–f display the orientation of the liquid crystal molecules in the slab (5.2). Let $\lambda(p)$ be an eigenvalue of largest magnitude of the tensor $Q(p)$, and let $w(p)$ be an associated normalized eigenvector. The figures show the directors $\lambda(p)w(p)$ for $p \in \Omega$. The lengths of the lines in the figures are proportional to the length of the directors. No line indicates random orientation of the molecules. Areas with liquid crystal molecules in random directions are often referred to as the defect of the equilibrium configuration.

Figure 5.3a shows the directors at the bifurcation point for $\mathcal{T}_2 = 0.3013$. Figures 5.3b–f are obtained when following path (3) as indicated in Fig. 5.2a. The defect of the equilibrium configuration splits and the two areas move towards the corners of Ω. The images obtained when following path (4) instead are the images for path (3) rotated 90°.

EXAMPLE 5.7. The equilibrium configuration of the liquid crystals is tracked for the three-dimensional slab (1.1) with $a = b = c = 1$. We discretize the slab by finite differences on a $21 \times 21 \times 21$ grid. Strong surface anchoring (1.6) of the liquid crystals is imposed, and we obtain a matrix $G_Q(Q, \mathcal{T})$ of order $n = 5 \times 19^3 = 34295$. The initial tensors are given by $Q(p) = 3v(p)v(p)^{\mathrm{T}} - I$, $p \in \Omega$, where $v(p)$ is the unit vector directed toward the center of Ω, analogously with Example 5.5.

The initial and final temperatures used were $\mathcal{T}_a = 0.1$ and $\mathcal{T}_b = 0.0$, respectively. Figure 5.4 shows the smallest eigenvalue of the matrix $G_Q(Q, \mathcal{T})$ and the minimum energy of the liquid crystals versus the temperature. We followed 16 solution paths, but Fig. 5.4a only shows five distinct paths. This is due to the symmetry of the problem.

We first followed path (1) starting with $\mathcal{T}_a = 0.1$ to path (2) ending with $\mathcal{T}_b = 0.0$. A singular point $(Q_3, \mathcal{T}_3)$ was found on the path for at $\mathcal{T}_3 = 0.0344$; see Fig. 5.4a. Using Algorithm 4.1, we found that the dimension of the null space of $G_Q(Q_3, \mathcal{T}_3)$ is three, as well as an orthonormal basis of the null space. Since the null space is orthogonal to $G_{\mathcal{T}}(Q_3, \mathcal{T}_3)$, the singular point is a bifurcation point.

We followed the intersecting curves, paths (3–16) in Fig. 5.4a, by taking a step in the tangent directions along these curves. No other singular points were found along these curves for temperatures between $\mathcal{T}_a = 0.1$ and $\mathcal{T}_b = 0.0$.

Figure 5.4b displays the minimum energy for the different paths. The graph shows that, starting from the bifurcation point, the minimum energy increases as we follow the paths

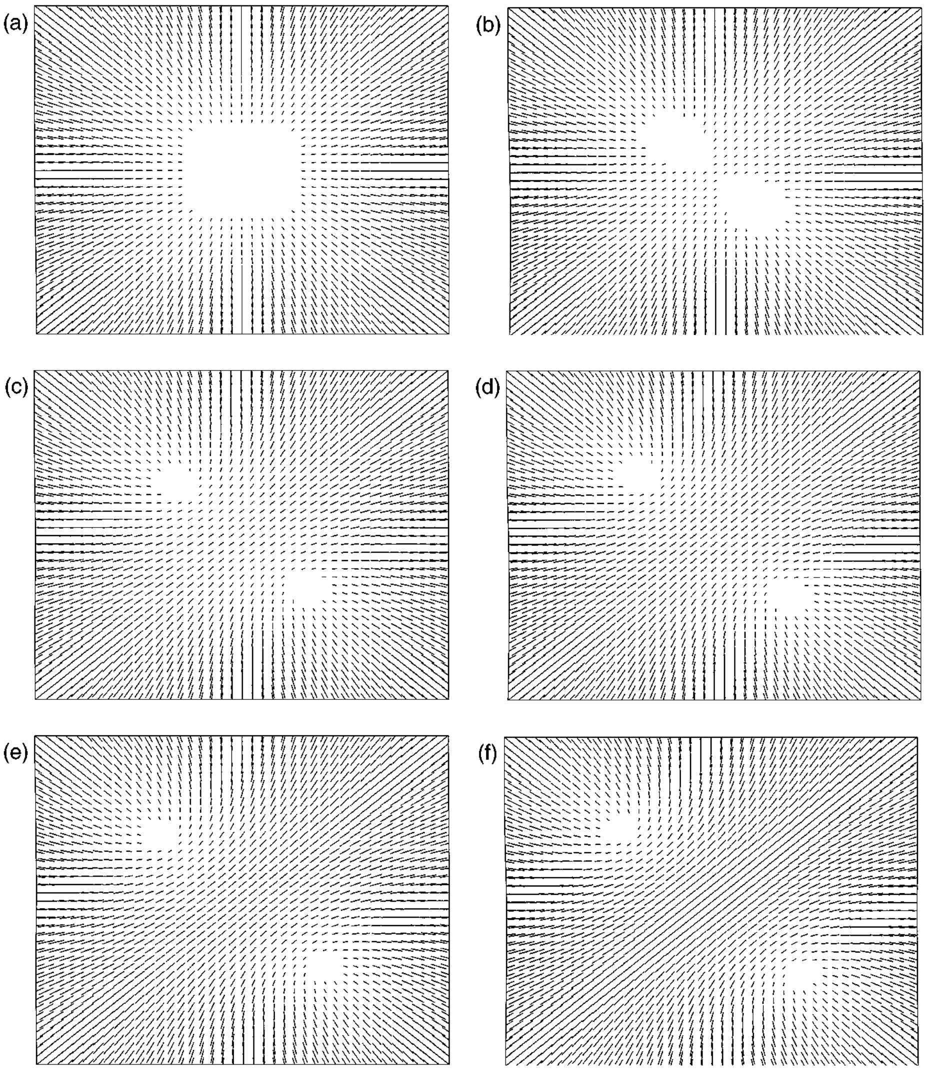

FIG. 5.3. Example 5.6: images of the equilibrium configurations of the liquid crystals for several temperatures: (a) $\mathcal{T} = 0.3013$; (b) $\mathcal{T} = 0.3039$; (c) $\mathcal{T} = 0.3131$; (d) $\mathcal{T} = 0.3210$; (e) $\mathcal{T} = 0.3249$; (f) $\mathcal{T} = 0.3174$.

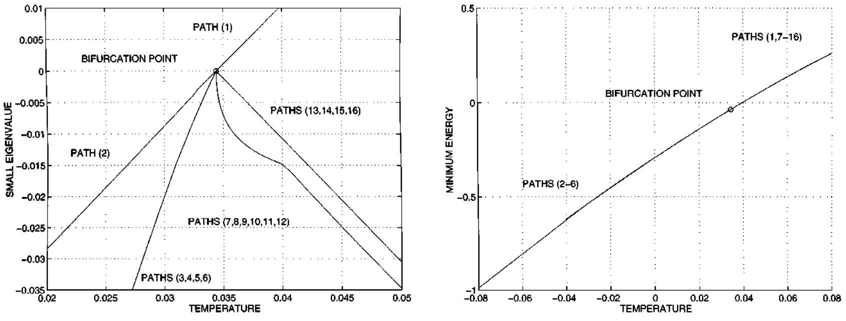

FIG. 5.4. Example 5.7: three-dimensional example with strong surface anchoring of the liquid crystals. (a) Smallest eigenvalue of the matrix $G_Q(Q, \mathcal{T})$ vs temperature $\mathcal{T}$. (b) Minimum energy vs temperature $\mathcal{T}$.

TABLE VI
Example 5.7: Symmetry Properties and Equivalence Classes for the Different Paths in Fig. 5.4a

	Symmetry of the field of the Q tensors			
Path	x_1x_2	x_1x_3	x_2x_3	Equivalence classes
1	√	√	√	path 1
2	√	√	√	path 2
3	√	√	√	path 3
4	√			paths 4, 5, 6
5			√	paths 4, 5, 6
6		√		paths 4, 5, 6
7	√			paths 7 – 12
8	√			paths 7 – 12
9			√	paths 7 – 12
10			√	paths 7 – 12
11		√		paths 7 – 12
12		√		paths 7 – 12
13	√	√	√	path 13
14	√			paths 14, 15, 16
15			√	paths 14, 15, 16
16		√		paths 14, 15, 16

(1,7–16) and the temperature increases. When we follow the paths (2–6) the minimum energy decreases as temperature decreases.

This three-dimensional problem illustrates symmetric properties of the field of Q tensors on different solution paths; see Table VI. The solution paths belong to equivalence classes. For example, the paths (4,5,6) are in the same equivalence class because these paths yield the same smallest eigenvalue graph; see Fig. 5.4a. These paths yield the same field of Q tensors up to a relabeling of the coordinate axes. The different equivalence classes are shown in the last column of Table VI.

EXAMPLE 5.8. The slab and boundary conditions for this example are the same as for Example 5.7, but we discretize the Euler–Lagrange equations on a finer mesh; the slab is discretized by a $41 \times 41 \times 41$ grid and this yields a matrix $G_Q(Q, \mathcal{T})$ of order $n = 5 \times 39^3 = 296{,}595$. We started with the same initial temperature and tensor field as in Example 5.7 and found a singular point $(Q_4, \mathcal{T}_4)$ for $\mathcal{T} = 0.0356$ and a vector in the null space of $G_Q(Q_4, \mathcal{T}_4)$. We applied Algorithm 4.1 to determine an orthonormal basis of the null space of $G_Q(Q_4, \mathcal{T}_4)$ and its dimension. The latter is three. Since the null space is orthogonal to $G_{\mathcal{T}}(Q_4, \mathcal{T}_4)$, the singular point is a bifurcation point.

6. CONCLUSION AND EXTENSION

The paper describes a new algorithm for computing multiple or very close eigenvalues of a large sparse symmetric matrix. A comparison shows Algorithm 4.1 to be competitive with the subroutine DNLASO in the LASO2 package and with ARPACK. In particular, Algorithm 4.1 has been used successfully to determine the smallest eigenvalues and associated eigenvectors of large and sparse Jacobian matrices that arise in liquid crystal modeling. The low storage requirement of Algorithm 4.1 is essential for this application due to the large size of the Jacobian matrices.

Algorithm 4.1 is designed for the computation of eigenpairs associated with the smallest eigenvalues of a matrix A. The algorithm can be used to compute the largest eigenvalues of A by applying it to the matrix $-A$. A restarted Lanczos method for computing a few nonextreme eigenvalues of a large sparse matrix has been described in [3], and it would appear possible to modify the IRBL method of the present paper so that it can be used to compute a few nonextreme eigenvalues as well.

ACKNOWLEDGMENT

We thank Richard Lehoucq and Axel Ruhe for comments and Reference [16].

REFERENCES

1. E. L. Allgower, C.-S. Chien, K. Georg, and C.-F. Wang, Conjugate gradient methods for continuation problems, *J. Comput. Appl. Math.* **38**, 1 (1991).
2. J. Baglama, *Krylov Subspace Methods with Application to Liquid Crystal Modeling*, Ph.D. thesis, Department of Mathematics and Computer Science, Kent State University, 1997.
3. J. Baglama, D. Calvetti and L. Reichel, Iterative methods for the computation of a few eigenvalues of a large symmetric matrix, *BIT* **36**, 400 (1996).
4. Z. Bai and J. Demmel, On a block implementation of Hessenberg QR iteration, *Int'l J. High Speed Comput.* **1**, 97 (1989).
5. Å. Björck, Numerics of Gram-Schmidt orthogonalization, *Linear Algebra Appl.* **197–198**, 297 (1994).
6. D. Calvetti, L. Reichel and D. C. Sorensen, An implicitly restarted Lanczos method for large symmetric eigenvalue problems, *Elec. Trans. Numer. Anal.* **2**, 1 (1994).
7. F. Chatelin, *Eigenvalues of Matrices* (Wiley, Chichester, 1993).
8. E. R. Davidson, The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices, *J. Comput. Phys.* **17**, 87 (1975).
9. T. A. Davis and E. G. Gartland, jr., Finite element analysis of the Landau-de Gennes minimization problem for liquid crystals, *SIAM J. Numer. Anal.*, to appear.
10. A. A. Dubrulle and G. H. Golub, A multishift QR iteration without computation of the shifts, *Numer. Algorithms* **7**, 173 (1994).
11. T. Ericsson and A. Ruhe, The spectral transformation Lanczos method for the solution of large sparse symmetric generalized eigenvalue problems, *Math. Comp.* **35**, 1251 (1980).
12. 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, in *Comput. Appl. Math., I*, edited by C. Brezinski and U. Kulish (Elsevier, Amsterdam, 1992), p. 137.
13. D. A. Flanders and G. Shortley, Numerical determination of fundamental modes, *J. Appl. Phys.* **21**, 1326 (1950).
14. G. H. Golub and R. Underwood, The block Lanczos method for computing eigenvalues, in *Mathematical Software III*, edited by J. R. Rice (Academic Press, New York, 1977), p. 361.
15. 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**, 228 (1994).
16. J. Huitfeldt, *Nonlinear Eigenvalue Problems — Prediction of Bifurcation Points and Branch Switching*, Report, Department of Computer Science, Chalmers University of Technology, Göteborg, 1991.
17. H. B. Keller, *Lectures on Numerical Methods in Bifurcation Problems* (Springer, Berlin, 1987).

18. R. B. Lehoucq, *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration*, Ph.D thesis, Rice University, Houston, 1995.
19. R. B. Lehoucq and K. J. Maschhoff, *Implicitly Re-started Block Arnoldi Methods*, in preparation.
20. R. B. Lehoucq and D. C. Sorensen, Deflation techniques for an implicitly restarted Arnoldi iteration, *SIAM J. Matrix Anal. Appl.* **17**, 789 (1996).
21. R. B. Lehoucq, D. C. Sorensen, P. A. Vu, and C. Wang, *ARPACK: An Implementation of an Implicitly Restarted Arnoldi Method for Computing Some of the Eigenvalues and Eigenvectors of a Large Sparse Matrix, 1996*. Code available from Netlib in directory scalapack.
22. F. Leja, Sur certaines suits liées aux ensemble plan et leur application à la representation conforme, *Ann. Polon. Math.* **4**, 8 (1957).
23. R. B. Morgan, On restarting the Arnoldi method for large nonsymmetric eigenvalue problems, *Math. Comp.* **65**, 1213 (1996).
24. 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**, 817 (1986).
25. C. R. Murray, S. C. Racine, and E. R. Davidson, Improved algorithms for the lowest few eigenvalues and associated eigenvectors of large matrices, *J. Comput. Phys.* **103**, 382 (1992).
26. B. N. Parlett and D. S. Scott, The Lanczos algorithm with selective orthogonalization, *Math. Comp.* **33**, 311 (1979).
27. E. B. Priestly, P. J. Wojyowicz, and P. Sheng (Eds.), *Introduction to Liquid Crystals* (Plenum New York, 1975).
28. L. Reichel and W. B. Gragg, Algorithm 686: FORTRAN subroutines for updating the QR decomposition of a matrix, *ACM Trans. Math. Software* **16**, 369 (1990).
29. A. Ruhe, Implementation aspect of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices, *Math. Comp.* **33**, 680 (1979).
30. Y. Saad, *Numerical Methods for Large Eigenvalue Problems* (Halstead Press, New York, 1992).
31. D. S. Scott, *LASO2-FORTRAN Implementation of the Lanczos Process with Selective Orthogonalization*. Code and documentation available from Netlib.
32. G. L. G. Sleijpen and H. A. van der Vorst, A Jacobi-Davidson iteration method for linear eigenvalue problems, *SIAM J. Matrix Anal. Appl.* **17**, 401 (1996).
33. D. C. Sorensen, Implicit application of polynomial filters in a k-step Arnoldi method, *SIAM J. Matrix Anal. Appl.* **13**, 357 (1992).
34. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed. (Springer, New York, 1993).
35. E. G. Virga, *Variational Theories for Liquid Crystals* (Chapman and Hall, London, 1994).
36. R. Zeller, *Parallel Numerical Solutions of the Landau-de Gennes Minimization Problem for Liquid Crystals in a Slab Geometry*, Ph.D. thesis, Kent State University, 1993.