Advances in Linear Algebra & Matrix Theory, 2015, 5, 76-89
Published Online September 2015 in SciRes. http://www.scirp.org/journal/alamt
http://dx.doi.org/10.4236/alamt.2015.53008

Scientific Research Publishing

A Subspace Iteration for Calculating a Cluster of Exterior Eigenvalues

Achiya Dax

Hydrological Service, Jerusalem, Israel
Email: dax20@water.gov.il

Received 25 June 2015; accepted 29 August 2015; published 1 September 2015

Abstract

In this paper we present a new subspace iteration for calculating eigenvalues of symmetric matrices. The method is designed to compute a cluster of *k* exterior eigenvalues. For example, *k* eigenvalues with the largest absolute values, the *k* algebraically largest eigenvalues, or the *k* algebraically smallest eigenvalues. The new iteration applies a Restarted Krylov method to collect information on the desired cluster. It is shown that the estimated eigenvalues proceed monotonically toward their limits. Another innovation regards the choice of starting points for the Krylov subspaces, which leads to fast rate of convergence. Numerical experiments illustrate the viability of the proposed ideas.

Keywords

Exterior Eigenvalues, Symmetric Matrices, Subspace Iterations, Interlacing, Restarted Krylov Methods

1. Introduction

In this paper we present a new subspace iteration for calculating a cluster of k exterior eigenvalues of a given symmetric matrix, $G \in \mathbb{R}^{n\times n}$. Other names for such eigenvalues are "peripheral eigenvalues" and "extreme eigenvalues". As with other subspace iterations, the method is best suited for handling large sparse matrices in which a matrix-vector product needs only $0(n)$ flops. Another underlying assumption is that k^2 is considerably smaller than n. Basically there are two types of subspace iterations for solving such problems. The first category regards "block" versions of the Power method that use frequent orthogonalizations. The eigenvalues are extracted with the Rayleigh-Ritz procedure. This kind of method is also called "orthogonal iterations" and "simultaneous iterations". The second category uses the Rayleigh-Ritz process to achieve approximation from a Krylov subspace. The Restarted Lanczos method turns this approach into a powerful tool. For detailed discus-

sions of these topics see, for example, [1]-[19].

The new iteration applies a Krylov subspace method to collect information on the desired cluster. Yet it has an additional flavor: it uses an interlacing theorem to improve the current estimates of the eigenvalues. This enables the method to gain speed and accuracy.

If G happens to be a singular matrix, then it has zero eigenvalues, and any orthonormal basis of Null (G) gives the corresponding eigenvectors. However, in many practical problems we are interested only in non-zero eigenvalues. For this reason the coming definitions of the term "**a cluster of *k* exterior eigenvalues**" do not include zero eigenvalues. Let r denote the rank of G and assume that $k < r$. Then G has r non-zero eigenvalues that can be ordered to satisfy

$$\left|\hat{\lambda}_1\right| \geq \left|\hat{\lambda}_2\right| \geq \cdots \geq \left|\hat{\lambda}_r\right| > 0 \tag{1.1}$$

or

$$\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_r. \tag{1.2}$$

The new algorithm is built to compute one of the following four types of **target clusters** that contain k extreme eigenvalues.

A **dominant cluster**

$$\left\{\hat{\lambda}_1, \cdots, \hat{\lambda}_k\right\}. \tag{1.3}$$

A **right-side cluster**

$$\left\{\tilde{\lambda}_1, \cdots, \tilde{\lambda}_k\right\}. \tag{1.4}$$

A **left-side cluster**

$$\left\{\tilde{\lambda}_{r+1-k}, \cdots, \tilde{\lambda}_{r-1}, \tilde{\lambda}_r\right\}. \tag{1.5}$$

A **two-side cluster** is a union of a right-side cluster and a left-side cluster. For example, $\left\{\tilde{\lambda}_1, \tilde{\lambda}_r\right\}, \left\{\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_r\right\}$, and so forth.

Note that although the above definitions refer to clusters of eigenvalues, the algorithm is carried out by computing the corresponding k eigenvectors of G. The subspace that is spanned by these eigenvectors is called the **target space**. The restriction of the target cluster to include only non-zero eigenvalues means that the target space is contained in Range(G). For this reason the search for the target space is restricted to Range(G).

Let us turn now to describe the basic iteration of the new method. The qth iteration, $q = 1, 2, \cdots$, is composed of the following five steps. The first step starts with a matrix $V_q \in \mathbb{R}^{n \times k}$ that contains "old" information on the target space, a matrix $Y_q \in \mathbb{R}^{n \times \ell}$ that contains "new" information, and a matrix $X_q = \left[V_q, Y_q\right] \in \mathbb{R}^{n \times (k+\ell)}$ that includes all the known information. The matrix X_q has $p = k + \ell$ orthonormal columns. That is

$$X_q^{\mathrm{T}} X_q = I \in \mathbb{R}^{p \times p}.$$

(Typical values for ℓ are $\ell = 2k$ or $\ell = 3k$.)

Step 1: Eigenvalues extraction. First compute the Rayleigh quotient matrix

$$S_q = X_q^{\mathrm{T}} G X_q.$$

Then compute k eigenpairs of S_q which correspond to the target cluster. (For example, if it is desired to compute a right-side cluster of G, then compute a right-side cluster of S_q.) The corresponding k eigenvectors of S_q are assembled into a matrix

$$U_q \in \mathbb{R}^{p \times k}, \quad U_q^{\mathrm{T}} U_q = I \in \mathbb{R}^{k \times k},$$

which is used to compute the related matrix of Ritz vectors,

$$V_{q+1} = X_q U_q.$$

Step 2: Collecting new information. Compute a matrix $B_q \in \mathbb{R}^{n \times \ell}$ that contains new information on the target space. The columns of B_q are forced to stay in Range(G).

Step 3: Discard redundant information. Orthogonalize the columns of B_q against the columns of V_{q+1}. There are several ways to achieve this task. In exact arithmetic the resulting matrix, Z_q, satisfies the Gram-Schmidt formula

$$Z_q = B_q - V_{q+1}\left(V_{q+1}^{\mathrm{T}} B_q\right).$$

Step 4: Build an orthonormal basis. Compute a matrix,

$$Y_{q+1} \in \mathbb{R}^{n\times \ell}, \quad Y_{q+1}^{\mathrm{T}} Y_{q+1} = I \in \mathbb{R}^{\ell\times\ell},$$

whose columns form an orthonormal basis of $\mathrm{Range}\left(Z_q\right)$. This can be done by a QR factorization of Z_q (if $\mathrm{rank}\left(Z_q\right)$ is smaller than ℓ, then ℓ is redefined as $\mathrm{rank}\left(Z_q\right)$).
Step 5: Define X_{q+1} by the rule

$$X_{q+1} = \left[V_{q+1}, Y_{q+1}\right],$$

which ensures that

$$X_{q+1}^{\mathrm{T}} X_{q+1} = I \in \mathbb{R}^{p\times p}.$$

The above description is aimed to clarify the purpose of each step. Yet there might be better ways to carry out the basic iteration. The restriction of the search to Range(G) is important when handling low-rank matrices. However, if G is known to be a non-singular matrix, then there is no need to impose this restriction.

The plan of the paper is as follows. The interlacing theorems that support the new method are given in the next section. Let $\lambda_j^{(q)}, j = 1,\cdots,k$, and denote the Ritz values which are computed at Step 1 of the qth iteration. Then it is shown that each iteration gives a better approximation of the target cluster. Moreover, the sequence $\lambda_j^{(q)}, q = 1,2,\cdots$ proceeds monotonously toward the desired eigenvalue of G. The rate of convergence depends on the information matrix B_q. Roughly speaking, the better information we get, the faster the convergence is. Indeed, the heart of the algorithm is the computation of B_q. It is well-known that a Krylov subspace which is generated by G gives valuable information on peripheral eigenvalues of G, e.g., [4] [5] [7] [13] [15]. The basic scheme of the new method uses this observation to define B_q, see Section 3. Difficulties that arise in the computation of non-peripheral clusters are discussed in Section 4. The fifth section considers the use of acceleration techniques; most of them are borrowed from orthogonal iterations. Another related iteration is the Restarted Lanczos method. The links with these methods are discussed in Sections 6 and 7. The paper ends with numerical experiments that illustrate the behavior of the proposed method.

2. Interlacing Theorems

In this section we establish a useful property of the proposed method. We start with two well-known interlacing theorems, e.g., [6] [7] [19].

Theorem 1 (Cauchy interlace theorem) *Let* $G \in \mathbb{R}^{n\times n}$ *be a symmetric matrix with eigenvalues*

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n. \tag{2.1}$$

Let the symmetric matrix $H \in \mathbb{R}^{k\times k}$ be obtained from G by deleting $n-k$ rows and the corresponding $n-k$ columns. Let

$$\eta_1 \geq \eta_2 \geq \cdots \geq \eta_k \tag{2.2}$$

denote the eigenvalues of H. Then

$$\lambda_j \geq \eta_j \quad \text{for } j = 1,\cdots,k, \tag{2.3}$$

and

$$\eta_{k+1-i} \geq \lambda_{n+1-i} \quad \text{for } i = 1,\cdots,k. \tag{2.4}$$

In particular, for $k = n-1$ we have the interlacing relations

$$\lambda_1 \geq \eta_1 \geq \lambda_2 \geq \eta_2 \geq \lambda_3 \geq \cdots \geq \lambda_{n-1} \geq \eta_{n-1} \geq \lambda_n. \tag{2.5}$$

Corollary 2 (Poincaré separation theorem) *Let the matrix* $V \in \mathbb{R}^{n\times k}$ *have k orthonormal columns. That is*

$V^{\mathrm{T}}V = I \in \mathbb{R}^{k\times k}$. *Let the matrix* $H = V^{\mathrm{T}}GV$ *have the eigenvalues* (2.2). *Then the eigenvalues of H and G satisfy* (2.3) *and* (2.4).

The next theorem seems to be new. It sharpens the above results by removing zero eigenvalues.

Theorem 3 *Assume that the non-zero eigenvalues of G satisfy* (1.2) *where* $r = \operatorname{rank}(G)$. *Let the matrix* $V \in \mathbb{R}^{n\times k}$ *satisfy*

$$k < r, \quad V^{\mathrm{T}}V = I \in \mathbb{R}^{k\times k}, \quad \text{and} \quad \operatorname{Range}(V) \subseteq \operatorname{Range}(G). \tag{2.6}$$

Let the matrix $H = V^{\mathrm{T}}GV$ has the eigenvalues (2.2). Then the eigenvalues of G and H satisfy the inequalities

$$\tilde{\lambda}_j \geq \eta_j \qquad \text{for } j = 1,\cdots,k, \tag{2.7}$$

and

$$\eta_{k+1-i} \geq \tilde{\lambda}_{r+1-i} \qquad \text{for } i = 1,\cdots,k. \tag{2.8}$$

Proof. Let the matrix $Y \in \mathbb{R}^{n\times r}$ be obtained by completing the columns of V to be an orthonormal basis of $\operatorname{Range}(G)$. Then $Y^{\mathrm{T}}GY$ is a full rank symmetric matrix whose eigenvalues are the non-zero eigenvalues of G which are given in (1.2). Since the first k columns of Y are the columns of V, the matrix $V^{\mathrm{T}}GV$ is obtained by deleting from $Y^{\mathrm{T}}GY$ the last $r-k$ rows and the last $r-k$ columns. Hence the inequalities (2.7) and (2.8) are direct corollary of Cauchy interlace theorem. □

Let us return now to consider the qth iteration of the new method, $q = 1,2,3,\cdots$. Assume first that the algorithm is aimed at computing a cluster of k right-side eigenvalues of G,

$$\left\{\tilde{\lambda}_1, \tilde{\lambda}_2, \cdots, \tilde{\lambda}_k\right\},$$

and let the eigenvalues of the matrix

$$S_q = X_q^{\mathrm{T}}GX_q = \left[V_q, Y_q\right]^{\mathrm{T}} G\left[V_q, Y_q\right]$$

be denoted as

$$\lambda_1^{(q)} \geq \lambda_2^{(q)} \geq \cdots \geq \lambda_k^{(q)} \geq \cdots \geq \lambda_p^{(q)}.$$

Then the Ritz values which are computed at Step 1 are

$$\lambda_1^{(q)} \geq \lambda_2^{(q)} \geq \cdots \geq \lambda_k^{(q)},$$

and these values are the eigenvalues of the matrix

$$V_{q+1}^{\mathrm{T}}GV_{q+1}.$$

Similarly,

$$\lambda_1^{(q-1)} \geq \lambda_2^{(q-1)} \geq \cdots \geq \lambda_k^{(q-1)},$$

are the eigenvalues of the matrix

$$V_q^{\mathrm{T}}GV_q.$$

Therefore, since the columns of V_q are the first k columns of X_q,

$$\lambda_j^{(q)} \geq \lambda_j^{(q-1)} \qquad \text{for } j = 1,\cdots,k.$$

On the other hand from Theorem 3 we obtain that

$$\tilde{\lambda}_j \geq \lambda_j^{(q)} \qquad \text{for } j = 1,\cdots,k.$$

Hence by combining these relations we see that

$$\tilde{\lambda}_j \geq \lambda_j^{(q)} \geq \lambda_j^{(q-1)} \tag{2.9}$$

for $j = 1,\cdots,k$ and $q = 2,3,\cdots$.

Assume now that the algorithm is aimed at computing a cluster of k left-side eigenvalues of G,

$$\left\{\tilde{\lambda}_{r+1-k},\cdots,\tilde{\lambda}_{r-1},\tilde{\lambda}_r\right\}.$$

Then similar arguments show that

$$\lambda_{p+1-i}^{(q-1)} \geq \lambda_{p+1-i}^{(q)} \geq \tilde{\lambda}_{r+1-i} \tag{2.10}$$

for $i=1,\cdots,k$, and $q=2,3,\cdots$.

Recall that a two-sides cluster is the union of a right-side cluster and a left-side one. In this case the eigenvalues of S_q that correspond to the right-side satisfy (2.9) while eigenvalues of S_q that correspond to the left-side satisfy (2.10). A similar situation occurs in the computation of a dominant cluster, since a dominant cluster is either a right-side cluster, a left-side cluster, or a two-sides cluster.

3. The Krylov Information Matrix

It is left to explain how the information matrices are computed. The first question to answer is how to define the starting matrix X_1. For this purpose we consider a Krylov subspace that is generated by the vectors

$$G\boldsymbol{r}, G^2\boldsymbol{r}, G^3\boldsymbol{r}, \cdots, G^p\boldsymbol{r}, \tag{3.1}$$

where $\boldsymbol{r}$ is a "random" vector. That is, a vector whose entries are uniformly distributed between −1 and 1. Then X_1 is defined to be a matrix whose columns provide an orthonormal basis for that space. This definition ensures that $\text{range}(X_1)$ is contained in range(G). The actual computation of X_1 can be done in a number of ways.

The main question is how to define the information matrix

$$B_q = \left[\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_\ell\right] \in \mathbb{R}^{n\times\ell} \tag{3.2}$$

which is needed in Step 2. Following the Krylov subspace approach, the columns of B_q are defined by the rule

$$\boldsymbol{b}_j = G\boldsymbol{b}_{j-1} \Big/ \left\|G\boldsymbol{b}_{j-1}\right\|, \; j=1,\cdots,\ell, \tag{3.3}$$

where $\|\cdot\|$ is some vector norm. In this way B_q is determined by the starting vector $\boldsymbol{b}_0$.

The ability of a Krylov subspace to approximate a dominant subspace is characterized by the Kaniel-Paige-Saad (K-P-S) bounds. See, for example, ([5], pp. 552-554), ([7], pp. 242-247), ([13], pp. 272-274), and the references therein. One consequence of these bounds regards the angle between $\boldsymbol{b}_1$ and the dominant subspace: The smaller the angle, the better approximation we get. This suggests that $\boldsymbol{b}_0$ should be defined as the sum of the current Ritz vectors. That is,

$$\boldsymbol{b}_0 = V_{q+1}\boldsymbol{e} \tag{3.4}$$

where vector of ones.

Another consequence of the K-P-S bounds is that a larger Krylov subspace gives better approximations. This suggests that using (3.2)-(3.3) with a larger ℓ is expected to result in faster convergence, since

$$\text{Range}\left(B_q\right) \subseteq \text{Range}\left(X_{q+1}\right). \tag{3.5}$$

A different argument that supports the last observation comes from the interlacing theorems: Consider the use of (3.2)-(3.3) with two values of ℓ say $\tilde{\ell} < \hat{\ell}$, and let $\tilde{X}_{q+1}$ and $\hat{X}_{q+1}$ denote the corresponding orthonormal matrices. Then the first $\tilde{\ell}$ columns of $\hat{X}_{q+1}$ can be obtained from the columns of $\tilde{X}_{q+1}$. Therefore, a larger value of ℓ is likely to give better approximations. However, the use of the above arguments to assess the expected rate of convergence is not straightforward.

4. Treating a Non-Peripheral Cluster

A cluster of eigenvalues is called **peripheral** if there exists a real number, α, for which the corresponding eigenvalues of the shifted matrix, $G-\alpha I$, turn out to be a dominant cluster. The theory developed in Section 2 suggests that the algorithm can be used to compute certain types of non-peripheral clusters (see the coming example). However, in this case it faces some difficulties. One difficulty regards the rate of convergence, as the K-P-S bounds tell us that approximations of peripheral eigenvalues are better than those of internal eigenvalues.

Hence when treating a non-peripheral cluster we expect a slower rate of convergence.

A second difficulty comes from the following phenomenon. To simplify the discussion we concentrate on a left-side cluster of a positive semi-definite matrix that has several zero eigenvalues. In this case the target cluster is composed from the k smallest non-zero eigenvalues of G. Then, once the columns of V_{q+1} become close to eigenvectors of G, the matrices B_q and Z_q turn out to be ill-conditioned. This makes Y_{q+1} vulnerable to rounding errors in the following manner. The orthogonality of Y_{q+1} is kept, but now $\text{Range}\left(Y_{q+1}\right)$ may fail to stay in Range(G). The same remark applies to X_{q+1}, since Y_{q+1} is part of X_{q+1}. In this case, when $\text{Range}\left(X_{q+1}\right)$ contains some vectors from Null(G), the smallest eigenvalues of the matrix $X_{q+1}^{\mathrm{T}} G X_{q+1}$ may be smaller than the smallest non-zero eigenvalue of G. Consequently the algorithm may converge toward zero eigenvalues of G. Hence it is necessary to take a precaution to prevent this possibility. (Similar ill-conditioning occurs when calculating a peripheral cluster, but in this case it does no harm.)

One way to overcome the last difficulty is to force $\text{Range}\left(Z_q\right)$ to stay inside Range(G). This can be done by the following modification of Step 3.

Step 3*: As before, the step starts by orthogonalizing the columns of B_q against the current Ritz vectors (the columns of V_{q+1}). This gives us an $n \times \ell$ matrix $\tilde{Z}_q$. Then the matrix $G\tilde{Z}_q$ is orthogonalized against the Ritz vectors, giving Z_q.

A second possible remedy is to force $\text{Range}\left(X_{q+1}\right)$ to stay inside Range(G). This can be done by correcting Step 5 in the following way.

Step 5*: Compute the matrices $\tilde{X}_{q+1} = \left[V_{q+1}, Y_{q+1}\right]$ and $\hat{X}_{q+1} = G\tilde{X}_{q+1}$. Then X_{q+1} is defined to be a matrix whose columns form an orthonormal basis of $\text{Range}\left(\hat{X}_{q+1}\right)$. This can be done by a QR factorization of $\hat{X}_{q+1}$.

In the experiments of Section 8, we have used Step 3* to compute a left-side cluster of Problem B, and Step 5* was used to compute a left-side cluster of Problem C. However the above modifications are not always helpful, and there might be better ways to correct the algorithm.

5. Acceleration Techniques

In this section we outline some possible ways to accelerate the rate of convergence. The acceleration is carried out in Step 2 of the basic iteration, by providing a "better" information matrix, B_q. All the other steps remain unchanged.

5.1. Power Acceleration

In this approach the columns of B_q are generated by replacing (3.3) with the rule

$$\boldsymbol{b}_j = G^m \boldsymbol{b}_{j-1} \Big/ \left\| G^m \boldsymbol{b}_{j-1} \right\|, \quad j = 1, \cdots, \ell, \tag{5.1}$$

where $m \geq 2$ is a small integer. Of course in practice the matrix G^m is never computed. Instead $\boldsymbol{b}_j$ is computed by a sequence of m matrix-vector multiplications and normalizations. The above acceleration is suitable for calculating a dominant cluster. In other exterior clusters it can be used with a shift. The main benefit of (5.1) is in reducing the portion of time that is spent on orthogonalizations and the Rayleigh-Ritz procedure (see **Table 4**).

5.2. Using a Shift

The shift operation is carried out by replacing (5.1) with

$$\boldsymbol{b}_j = \left(G - \alpha I\right)^m \boldsymbol{b}_{j-1} \Big/ \left\| \left(G - \alpha I\right)^m \boldsymbol{b}_{j-1} \right\|, \quad j = 1, \cdots, \ell, \tag{5.2}$$

where α is a real number. It is well known that Krylov subspaces are invariant under the shift operation. That is, for $m = 1$ the Krylov space that is generated by (5.2) equals that of (3.3), e.g., ([7], p. 238). Hence the shift operation does not accelerate the rate of convergence. Yet, it helps to reduce the deteriorating effects of rounding errors.

Assume first that G is a positive definite matrix and that we want to compute a left-side cluster (a cluster of the smallest eigenvalues). Then (5.1) is replaced by (5.2), where α is an estimate for the largest eigenvalue of G. Observe that the required estimate can be derived from the eigenvalues of the matrices S_q.

A similar tactic is used for calculating a two-sides cluster. In this case the shift is computed by the rule

$$\alpha = \left(\lambda_1^{(q)} + \lambda_\ell^{(q)}\right)/2.$$

In other words, the shift estimates the average value of the largest and the smallest (algebraically) eigenvalues of G. A more sophisticated way to implement the above ideas is outlined below.

5.3. Polynomial Acceleration

Let the eigenvalues of G satisfy (2.1) and let the real numbers

$$\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$$

define the monic polynomial

$$\Phi(\theta) = (\theta - \alpha_1)(\theta - \alpha_2)\cdots(\theta - \alpha_m).$$

Then the eigenvalues of the matrix polynomial

$$\Phi(G) = (G - \alpha_1 I)(G - \alpha_2 I)\cdots(G - \alpha_m I)$$

are determined by the relations

$$\varphi_j = \Phi(\lambda_j), \quad j = 1, \cdots, n.$$

Moreover, the matrices G and $\Phi(G)$ share the same eigenvectors: An eigenvector of G that corresponds to λ_j is an eigenvector of $\Phi(G)$ that corresponds to φ_j, and vice versa. In polynomial acceleration the basic scheme (3.3) is replaced with

$$\boldsymbol{b}_j = \Phi(G)\boldsymbol{b}_{j-1} \Big/ \left\|\Phi(G)\boldsymbol{b}_{j-1}\right\|, \quad j = 1, \cdots, \ell. \tag{5.3}$$

The idea here is to choose the points $\alpha_1, \cdots, \alpha_m$ in a way that enlarges the size of φ_j when λ_j belongs to the target cluster, and/or diminishes $|\varphi_j|$ when λ_j is outside the target cluster.

As with orthogonal iterations, the use of Chebyshev polynomials enables effective implementation of this idea. In this method there is no need in the numbers $\alpha_1, \cdots, \alpha_m$. Instead we have to supply the end points of the diminishing interval. Then the matrix-vector product $\Phi(G)\boldsymbol{b}_{j-1}$ is carried out by the Chebyshev recursion formula. See ([7], p. 294) for details. In our iteration the end points of the diminishing interval can be derived from the current Ritz values (the eigenvalues of S_q).

5.4. Inverse Iterations

This approach is possible only in certain cases, when G is invertible and the matrix-vector product $G^{-1}\boldsymbol{b}_j$ can be computed in $0(n)$ flops. In this case (3.3) is replaced with

$$\boldsymbol{b}_j = G^{-1}\boldsymbol{b}_{j-1} \Big/ \left\|G^{-1}\boldsymbol{b}_{j-1}\right\|, \quad j = 1, \cdots, \ell. \tag{5.4}$$

In practice G^{-1} is almost never computed. Instead the linear system $G\boldsymbol{x} = \boldsymbol{b}_{j-1}$ is solved for $\boldsymbol{x}$. For this purpose we need an appropriate factorization of G.

The use of (5.4) is helpful for calculating small eigenvalues of a positive definite matrix. If other clusters are needed then G^{-1} should be replaced with $(G - \alpha I)^{-1}$, where α is a suitable shift. This is possible when the shifted system $(G - \alpha I)\boldsymbol{x} = \boldsymbol{b}_{j-1}$ is easily solved, as with band matrices.

6. Orthogonal Iterations

In this section we briefly examine the similarity and the difference between the new method and the Orthogonal Iterations method. The last method is also called Subspace Iterations and Simultaneous Iterations, e.g., [1] [4] [5] [7]-[10] [13]-[15]. It is aimed at computing a cluster of k dominant eigenvalues, as defined in (1.3). The simplest way to achieve this goal is, perhaps, to apply a "block" version of the Power method on an $n \times k$ matrix. In this way the Power method is simultaneously applied on each column of the matrix. The Orthogonal Iterations method modifies this idea in a number of ways. First, as its name says, it orthogonalizes the iteration matrix, which

keeps it a full rank well-conditioned matrix. Second, although we are interested in k eigenpairs, the iteration is applied on a larger matrix that has p columns, where $p > k$. This leads to faster convergence (see below). As before,

$$p = k + \ell$$

where ℓ is a small multiple of k. The qth iteration of the resulting method, $q = 0,1,2,\cdots$, is composed of the following three steps. It starts with a matrix $X_q \in \mathbb{R}^{n\times p}$ that has orthonormal columns.

Step 1: Compute the product matrix

$$Y_q = GX_q.$$

Step 2: Compute the Rayleigh quotient matrix

$$S_q = X_q^{\mathrm{T}} Y_q = X_q^{\mathrm{T}} G X_q \in \mathbb{R}^{p\times p},$$

and its k dominant eigenvalues

$$\left|\lambda_1^{(q)}\right| \geq \left|\lambda_2^{(q)}\right| \geq \cdots \geq \left|\lambda_k^{(q)}\right|.$$

Step 3: Compute a matrix X_{q+1} whose columns provide an orthonormal basis of $\mathrm{Range}\left(Y_q\right)$. This can be done by a QR factorization of Y_q.

Let the eigenvalues of G satisfy (1.1). Then for $j = 1,\cdots,k$ the sequence $\left|\lambda_j^{(q)} - \hat{\lambda}_j\right|$, $q = 1,2,\cdots$, converges to zero at the same asymptotic rate as the sequence $\left|\hat{\lambda}_{p+1}/\hat{\lambda}_j\right|^q$, $q = 1,2,\cdots$. See, for example, ([4], p. 157) or ([5], p. 368). Therefore, the larger p is, the faster is the convergence. Moreover, let the spectral decomposition of S_q have the form

$$S_q = V_q D_q V_q^{\mathrm{T}}$$

where

$$V_q^{\mathrm{T}} V_q = I \in \mathbb{R}^{p\times p},\quad D_q = \mathrm{diag}\left\{\lambda_1^{(q)},\cdots,\lambda_p^{(q)}\right\},\ \text{and}\ \left|\lambda_1^{(q)}\right| \geq \left|\lambda_2^{(q)}\right| \geq \cdots \geq \left|\lambda_p^{(q)}\right|.$$

If at the end of Step 2 the matrix Y_q is replaced with $Y_q V_q$ then the columns of X_q converge toward the corresponding eigenvectors of G. However, in exact arithmetic both versions generate the same sequence of Ritz values. Hence the retrieval of eigenvectors can wait to the final stage. The practical implementation of orthogonal iterations includes several further modifications, such as skipping Steps 2 - 3 and "locking". For detailed discussions of these options, see [1] [7]-[10].

A comparison of the above orthogonal iteration with the new iteration shows that both methods need about the same amount of computer storage, but the new method doubles the computational effort per iteration. The adaptation of orthogonal iterations to handle other peripheral clusters requires the shift operation. Another difference regards the rate of convergence. In orthogonal iterations the rate is determined by the ratio $\left|\lambda_{p+1}\right|/\left|\lambda_k\right|$. The theory behind the new method is not that decisive, but our experiments suggest that it is quite faster.

7. Restarted Lanczos Methods

The current presentation of the new method is carried out by applying the Krylov information matrix (3.2)-(3.4). This version can be viewed as a "Restarted Krylov method". The Restarted Lanczos method is a sophisticated implementation of this approach that harnesses the Lanczos algorithm to reduce the computational effort per iteration. As before, the method is aimed at computing a cluster of k exterior eigenpairs, new information is gained from an ℓ-dimensional Krylov subspace, and

$$p = k + \ell.$$

The qth iteration, $q = 1,2,\cdots$, of the **Implicitly Restarted Lanczos method (IRLM)** is composed of the following four steps. It starts with a tridiagonal matrix, $T_q \in \mathbb{R}^{p\times p}$, and the related matrix of Lanczos vectors,

$$L_q \in \mathbb{R}^{n\times p},\quad L_q^{\mathrm{T}} L_q = I \in \mathbb{R}^{p\times p},$$

which have been obtained by applying p steps of Lanczos method on G.

Step 1: Compute the eigenvalues of T_q. Let

$$\lambda_1^{(q)},\cdots,\lambda_k^{(q)},\lambda_{k+1}^{(q)},\cdots,\lambda_p^{(q)},$$

denote the computed eigenvalues, where the first k eigenvalues correspond to the target cluster.
Step 2: Compute a $p\times k$ matrix with orthonormal columns,

$$V_q\in\mathbb{R}^{p\times k},\quad V_q^{\mathrm{T}}V_q=I\in\mathbb{R}^{k\times k},$$

such that $\mathrm{Range}\left(V_q\right)$ equals an invariant subspace of T_q which corresponds to $\lambda_1^{(q)},\cdots,\lambda_k^{(q)}$. The computation of V_q is carried out by conducting a QR factorization of the product matrix

$$\left(T_q-\lambda_{k+1}^{(q)}I\right)\cdots\left(T_q-\lambda_p^{(q)}I\right).$$

Step 3: The above QR factorization is used to build a new $k\times k$ tridiagonal matrix, $\tilde{T}_q\in\mathbb{R}^{k\times k}$, and the matrix

$$\tilde{L}_q=L_qV_q\in\mathbb{R}^{n\times k}.$$

This pair of matrices has the property that it can be obtained by applying k steps of Lanczos algorithm on G, starting from some (unknown) vector.
Step 4: Continue ℓ additional steps of Lanczos algorithm to obtain T_{q+1} and L_{q+1}.

The IRLM iterations are due to Sorensen [11]. The name "Implicitly restarted" refers to the fact that the starting vector (which initiates the restarted Lanczos process) is not computed. For detailed description of this iteration, see ([1], pp. 67-73), and [11]. See also [13] [15].

A different implementation of the Restarted Lanczos idea, the **Thick-Restarted Lanczos (TRLan) method**, was proposed by Wu and Simon [17]. The qth iteration, $q=1,2,\cdots$, of this method is composed of the following five steps, starting with T_q and L_q as above.
Step 1: Compute k eigenpairs of T_q which correspond to the target cluster. The corresponding k eigenvectors of T_q are assembled into a matrix

$$U_q\in\mathbb{R}^{p\times k},\quad U_q^{\mathrm{T}}U_q=I\in\mathbb{R}^{k\times k},$$

which is used to compute the related matrix of Ritz vectors,

$$X_q=L_qU_q\in\mathbb{R}^{n\times k}.$$

Step 2: Let $\lambda_1^{(q)},\cdots,\lambda_k^{(q)}$, and $\boldsymbol{x}_1^{(q)},\cdots,\boldsymbol{x}_k^{(p)}$, denote the computed Ritz pairs. The algorithm uses these pairs to compute a vector $\boldsymbol{r}_q\in\mathbb{R}^n$ and scalars, $\sigma_1,\cdots,\sigma_k$, that satisfy the equalities

$$G\boldsymbol{x}_j^{(q)}-\lambda_j^{(q)}\boldsymbol{x}_j=\sigma_j\boldsymbol{r}_q,\quad\text{for } j=1,\cdots,k,$$

and

$$X_q^{\mathrm{T}}\boldsymbol{r}_q=\boldsymbol{o}.$$

Step 3: The vector $\boldsymbol{y}_1^{(q)}=\boldsymbol{r}_q/\left\|\boldsymbol{r}_q\right\|_2$ is used to initiate a new sequence of Lanczos vectors. The second vector, $\boldsymbol{y}_2^{(q)}$, is obtained by orthogonalizing the vector $G\boldsymbol{y}_1$ against $\boldsymbol{y}_1$ and the columns of X_q.
Step 4: Continue $\ell-2$ additional steps of the Lanczos process that generate $\boldsymbol{y}_3,\cdots,\boldsymbol{y}_\ell$. At the end of this process we obtain an $n\times p$ matrix

$$\tilde{L}_q=\left[\boldsymbol{x}_1^{(q)},\cdots,\boldsymbol{x}_k^{(q)},\boldsymbol{y}_1^{(q)},\boldsymbol{y}_2^{(q)},\cdots,\boldsymbol{y}_\ell^{(q)}\right],$$

that has mutually orthonormal columns and satisfy

$$\tilde{L}_q^{\mathrm{T}}G\tilde{L}_q=\tilde{T}_q,$$

where $\tilde{T}_q$ is nearly tridiagonal: The $(k+1)\times(k+1)$ principal submatrix of $\tilde{T}_q$ has an "arrow-head" shape, with $\lambda_1^{(q)},\cdots,\lambda_k^{(q)}$, on the diagonal, and $\sigma_1,\cdots,\sigma_k$, on the sides.
Step 5: Use a sequence of Givens rotations to complete the reduction of $\tilde{T}_q$ into a $p\times p$ tridiagonal matrix T_{q+1}. Similarly $\tilde{L}_q$ is updated to give L_{q+1}.

For detailed description of the above iteration see [17] [18]. Both IRLM and TRLan are carried out with reor-

thogonalization of the Lanczos vectors. The TRLan method computes the Ritz vectors while IRLM avoids this computation. Yet the two methods are known to be mathematically equivalent.

One difference between the new method and the Restarted Lanczos approach lies in the computation of the Rayleigh quotient matrix. In our method this computation requires additional p matrix-vector products, which doubles the computational effort per iteration.

A second difference lies in the starting vector of the ℓ dimensional Krylov subspace that is newly generated at each iteration. In our method this vector is defined by (3.4). In TRLan this vector is $\boldsymbol{r}_q$. Yet it is difficult to reckon how this difference effects the rate of convergence.

A third difference arises when using acceleration techniques. Let us consider for example the use of power acceleration with G^m replacing G. In this case the Restarted Lanczos methods compute a tridiagonal reduction of G^m, and Ritz values of G^m, while our method computes Ritz eigenpairs of G. (Power acceleration allows us to use smaller ℓ and reduces the portion of time that is spent on the Rayleigh-Ritz procedure. See the next section.) A similar remark applies to the use of Polynomial acceleration.

The new method can be viewed as generalization of the Restarted Lanczos approach. The generalization is carried out by replacing the Lanczos process with standard orthogonalization. This simplifies the algorithm and clarifies the main reasons that lead to fast rate of convergence. One reason is that each iteration builds a new Krylov subspace, using an improved starting vector. A second reason comes from the orthogonality requirement: The new Krylov subspace is orthogonalized against the current Ritz vectors. It is this orthogonalization that ensures successive improvement. (The Restarted Lanczos algorithms achieve these tasks in implicit ways.)

8. Numerical Experiments

In this section we describe some experiments that illustrate the behavior of the proposed method. The test matrices have the form

$$G = VDV^{\mathrm{T}} \in \mathbb{R}^{n\times n}, \tag{8.1}$$

where $V \in \mathbb{R}^{n\times n}$ is a random orthonormal matrix, $V^{\mathrm{T}}V = I$, and D is a diagonal matrix,

$$D = \operatorname{diag}\{d_1,\cdots,d_n\} \in \mathbb{R}^{n\times n}. \tag{8.2}$$

The term "random orthonormal" means that V is obtained by orthonormalizing the columns of a random matrix, $R \in \mathbb{R}^{n\times n}$, whose entries are random numbers from the interval $[-1,1]$. The random numbers generator is of uniform distribution. **All the experiments were carried out with $n = 200$ and $k = 6$**. The values of ℓ are specified in the coming tables. (The diagonal entries of D are the eigenvalues of G. Hence the structure of D is the major factor that effects convergence. Other factors, like the size of the matrix or the sparsity pattern, have minor effects.) The computations were carried out with MATLAB. We have used the following four types of test matrices:

Type A matrices, where

$$d_j = 201 - j \quad \text{for } j = 1,\cdots,200. \tag{8.3}$$

Type B matrices, where

$$d_j = 101 - j \quad \text{for } j = 1,\cdots,100, \quad \text{and} \quad d_j = 0 \quad \text{for } j = 101,\cdots,200. \tag{8.4}$$

Type C matrices, where

$$d_1 = 101 - j \quad \text{for } j = 1,\cdots,50, \text{ and } d_j = 0 \quad \text{for } j = 51,\cdots,200. \tag{8.5}$$

Type D matrices, where

$$d_{2j-1} = 51 - j,\ d_{2j} = -(51 - j), \quad \text{for } j = 1,\cdots,50, \tag{8.6a}$$

and

$$d_j = 0 \quad \text{for } j = 101,\cdots,200. \tag{8.6b}$$

The difference between the computed Ritz values and the desired eigenvalues of G is computed as follows.

Let $\lambda_1,\cdots,\lambda_k$ denote the k eigenvalues of G which constitute the desired cluster. Let $\lambda_1^{(q)},\cdots,\lambda_k^{(q)}$, denote the corresponding Ritz values which are computed at the qth iteration, $q=0,1,2,\cdots$. Then the average difference between the corresponding eigenvalues is

$$\eta_q=\left(\sum_{j=1}^{k}\left|\lambda_j-\lambda_j^{(q)}\right|\right)\Big/k,\quad q=0,1,2,\cdots. \tag{8.7}$$

The figures in Tables 1-4 provide the values of η_q. They illustrate the way η_q converges to zero as q increases.

The new method is implemented as described in Section 3. It starts by orthonormalizing a random Krylov matrix of the form (3.1). The information matrix, B_q, is defined by (3.2)-(3.4). The Orthogonal iterations method is implemented as in Section 6. It starts from a random orthonormal matrix, which is a common default option, e.g., ([1], p. 55) and ([13], p. 60).

Table 1. Computing a dominant cluster with the new iteration.

Iter. No.	Type A		Type B		Type C		Type D	
	$\ell=12$	$\ell=18$	$\ell=12$	$\ell=18$	$\ell=12$	$\ell=18$	$\ell=12$	$\ell=18$
0	1.39E1	9.91E0	6.18E0	3.91E0	4.58E0	3.99E0	1.53E0	2.41E−1
1	4.28E0	1.96E0	1.12E0	4.35E−1	9.85E−1	6.11E−1	4.09E−2	3.02E−4
2	1.15E0	2.93E−1	6.66E−2	6.95E−4	1.14E−1	1.22E−2	6.82E−4	3.45E−7
3	4.10E−1	1.18E−1	8.36E−4	2.10E−6	3.28E−3	8.04E−6	1.78E−5	1.20E−9
4	1.69E−1	4.72E−3	1.76E−5	3.23E−8	1.36E−5	1.46E−7	4.71E−7	4.29E−12
5	5.07E−2	1.17E−4	1.47E−6	3.86E−10	2.74E−7	5.19E−9	2.19E−8	1.52E−13
6	1.61E−3	4.93E−6	3.50E−8	4.52E−11	2.60E−9	7.70E−11	3.92E−10	
7	3.42E−4	5.13E−7	3.68E−9	7.53E−12	5.75E−11	2.12E−12	3.94E−11	
8	4.82E−5	1.10E−8	9.13E−11	1.25E−12	1.67E−11	2.82E−13	4.72E−12	
10	2.25E−6	3.11E−10	6.21E−13		3.98E−13		9.00E−14	
12	1.49E−7	1.31E−11						
14	8.21E−9	1.13E−12						

Table 2. Computing a dominant cluster with orthogonal iterations.

Iter. No.	Type A		Type B		Type C		Type D	
	$\ell=18$	$\ell=30$	$\ell=18$	$\ell=30$	$\ell=18$	$\ell=30$	$\ell=18$	$\ell=30$
0	7.93E1	6.52E1	5.79E1	5.13E1	6.63E1	5.80E1	3.99E1	3.52E1
1	3.55E1	2.95E1	1.43E1	9.30E0	8.20E0	3.88E0	2.84E1	1.16E1
2	2.18E1	1.74E1	8.01E0	4.18E0	5.96E0	2.66E0	1.39E1	1.96E1
4	1.16E1	8.07E0	3.26E0	1.00E0	3.23E0	1.07E0	6.49E0	1.77E0
8	4.83E0	2.41E0	7.44E−1	8.19E−2	8.84E−1	1.39E−1	1.59E0	2.16E−1
12	2.54E0	8.20E−1	1.69E−1	1.05E−2	3.23E−1	1.44E−2	4.80E−1	1.91E−2
16	1.55E0	2.56E−1	3.57E−2	1.38E−3	9.38E−2	1.34E−3	1.30E−1	1.48E−3
20	9.95E−1	7.75E−2	7.61E−3	1.50E−4	2.70E−2	1.28E−4	3.73E−2	1.12E−4
24	6.43E−1	2.33E−2	1.69E−3	1.41E−5	7.92E−3	1.29E−5	1.11E−2	8.75E−6

Table 3. Computing a left-side cluster with the new iteration.

Iter. No.	Type A		Type B		Type C		Type D	
	$\ell = 12$	$\ell = 18$	$\ell = 12$	$\ell = 18$	$\ell = 12$	$\ell = 18$	$\ell = 12$	$\ell = 18$
0	3.19E1	2.94E1	1.11E1	9.97E0	4.99E0	4.70E0	6.29E0	2.55E0
1	1.30E1	7.68E0	4.93E0	2.83E0	7.45E−1	3.61E−1	1.17E0	5.72E−2
2	7.17E0	3.00E0	2.64E0	6.89E−1	4.49E−3	4.15E−5	7.38E−2	2.81E−5
3	3.13E0	8.79E−1	1.45E0	1.80E−1	4.52E−5	2.31E−7	9.17E−4	2.32E−8
4	1.37E0	1.28E−1	1.14E0	5.63E−2	4.03E−6	8.06E−9	1.69E−5	2.13E−11
5	6.58E−1	1.67E−2	4.80E−1	1.87E−3	1.80E−8	3.60E−10	3.68E−7	1.67E−13
6	1.05E−1	2.36E−4	6.56E−2	2.87E−4	9.35E−10	5.66E−11	8.31E−9	5.33E−14
7	1.30E−2	5.73E−6	2.69E−3	9.09E−5	3.65E−11	2.74E−12	1.91E−10	
8	7.44E−4	5.95E−7	9.73E−4	2.06E−5	3.14E−12	7.61E−13	4.54E−12	
10	3.67E−5	4.74E−10	2.14E−4	2.98E−6	2.75E−13		8.41E−14	
12	6.94E−7	9.70E−12	5.66E−5	5.84E−7				
14	2.09E−8	2.47E−13	1.82E−5	2.72E−7				

Table 4. The use of Power acceleration to compute a dominant cluster of Type A matrix.

Iter. No.	$\ell = 6$		$\ell = 12$			$\ell = 18$	
	m = 2	m = 4	m = 2	m = 3	m = 4	m = 2	m = 3
0	1.68E1	7.82E0	6.41E0	3.80E0	2.43E0	4.34E0	2.39E0
1	8.52E0	2.46E0	1.09E0	5.05E−1	1.84E−1	3.37E−1	1.62E−1
2	3.84E0	8.78E−1	1.94E−1	1.16E−1	1.38E−3	1.64E−2	1.30E−4
3	1.99E0	2.21E−1	3.14E−2	3.48E−3	6.72E−7	7.73E−5	2.11E−8
4	9.03E−1	1.55E−1	2.78E−4	5.04E−5	1.86E−8	7.46E−7	2.46E−10
5	3.51E−1	3.48E−2	8.74E−6	4.82E−7	1.49E−10	1.66E−8	2.96E−11
6	1.79E−1	1.83E−3	1.24E−6	3.41E−8	2.66E−11	1.71E−9	3.36E−13
7	1.59E−1	1.46E−4	2.33E−8	1.78E−10	3.21E−12	5.29E−11	
8	1.09E−1	1.34E−5	3.25E−9	3.24E−11	3.13E−13	1.61E−11	
10	1.33E−2	8.13E−7	9.34E−11	4.64E−13		3.93E−13	
12	2.28E−3	7.93E−8	2.42E−12				
14	2.76E−4	1.47E−9	2.61E−13				
18	8.14E−6	2.32E−11					

Table 1 describes the computation of dominant clusters with the new method. The reading of this table is simple: We see, for example, that after performing 6 iterations on a Type B matrix, $\eta_6 = 3.50E-8$ for $\ell = 12$, and $\eta_6 = 4.52E-11$ for $\ell = 18$. (The corresponding values of p are $p = 18$ and $p = 24$, respectively.) **Table 2** describes the computation of the same dominant clusters with orthogonal iterations. A comparison of the two tables suggests that the new method is considerably faster than Orthogonal iterations.

Another observation stems from the first rows of these tables: We see that a random Krylov matrix gives a better start than a random starting matrix.

The ability of the new method to compute a left-side cluster is illustrated in **Table 3**. Note that the Type B matrix and the Type C matrix are positive semidefinite matrices, in which the left-side cluster is composed from the smaller non-zero eigenvalues of G. Both matrices have several zero eigenvalues. In the Type B matrix the eigenvalues in the left-side cluster are much smaller than the dominant eigenvalues, but the new method is able to distinguish these eigenvalues from zero eigenvalues.

The merits of Power acceleration are demonstrated in **Table 4**. On one hand it enables us to use a smaller ℓ, which saves storage. On the other hand it reduces the portion of time that is spent on orthogonalizations and the Rayleigh-Ritz process.

9. Concluding Remarks

The new method is based on a modified interlacing theorem which forces the Rayleigh-Ritz approximations to move monotonically toward their limits. The current presentation concentrates on the Krylov information matrix (3.2)-(3.4), but the method can use other information matrices. The experiments that we have done are quite encouraging, especially when calculating peripheral clusters. The theory suggests that the method can be extended to calculate certain non-peripheral clusters, but in this case we face some difficulties due to rounding errors. Further modifications of the new method are considered in [3].

References

[1] Bai, Z., Demmel, J., Dongarra, J., Ruhe, A. and van der Vorst, H. (1999) Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, Philadelphia.

[2] Bauer, F.L. (1957) Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischers Eigenwertprobleme. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, **8**, 214-235. http://dx.doi.org/10.1007/BF01600502

[3] Dax, A. Restarted Krylov Methods for Calculating Exterior Eigenvalues of Large Matrices. Tech. Rep., Hydrological Service of Israel, in Preparation.

[4] Demmel, J.W. (1997) Applied Numerical Linear Algebra. SIAM, Philadelphia. http://dx.doi.org/10.1137/1.9781611971446

[5] Golub, G.H. and Van Loan, C.F. (1983) Matrix Computations. Johns Hopkins University Press, Baltimore.

[6] Horn, R.A. and Johnson, C.R. (1985) Matrix Analysis. Cambridge University Press, Cambridge. http://dx.doi.org/10.1017/CBO9780511810817

[7] Parlett, B.N. (1980) The Symmetric Eigenvalue Problem. Prentice-Hall, Englewood Cliffs.

[8] Reinsch, C.H. (1971) Simultaneous Iteration Method for Symmetric Matrices. In: Wilkinson, J.H. and Reinsch, C.H., Eds., *Handbook for Automatic Computation* (*Linear Algebra*), Springer-Verlag, New York, 284-302.

[9] Rutishauer, H. (1969) Computational Aspects of F. L. Bauer's Simultaneous Iteration Method. *Numerische Mathematik*, **13**, 4-13. http://dx.doi.org/10.1007/BF02165269

[10] Rutishauser, H. (1970) Simultaneous Iteration Method for Symmetric Matrices. *Numerische Mathematik*, **16**, 205-223. http://dx.doi.org/10.1007/BF02219773

[11] Sorensen, D.C. (1992) Implicit Application of Polynomial Filters in a *k*-Step Arnoldi Method. *SIAM Journal on Matrix Analysis and Applications*, **13**, 357-385. http://dx.doi.org/10.1137/0613025

[12] Stewart, G.W. (1969) Accelerating the Orthogonal Iteration for the Eigenvalues of a Hermitian Matrix. *Numerische Mathematik*, **13**, 362-376. http://dx.doi.org/10.1007/BF02165413

[13] Stewart, G.W. (2001) Matrix Algorithms, Volume II: Eigensystems. SIAM, Philadelphia.

[14] Trefethen, L.N. and Bau III, D. (1997) Numerical Linear Algebra. SIAM, Philadelphia. http://dx.doi.org/10.1137/1.9780898719574

[15] Watkins, D.S. (2007) The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods. SIAM, Philadelphia. http://dx.doi.org/10.1137/1.9780898717808

[16] Wilkinson, J.H. (1965) The Algebraic Eigenvalue Problem. Clarendon Press, Oxford.

[17] Wu, K. and Simon, H. (2000) Thick-Restarted Lanczos Method for Large Symmetric Eigenvalue Problems. *SIAM Journal on Matrix Analysis and Applications*, **22**, 602-616. http://dx.doi.org/10.1137/S0895479898334605

[18] Yamazaki, I., Bai, Z., Simon, H., Wang, L. and Wu, K. (2010) Adaptive Projection Subspace Dimension for the

Thick-Restart Lanczos Method. *ACM Transactions on Mathematical Software*, **37**, 1-18. http://dx.doi.org/10.1145/1824801.1824805

[19] Zhang, F. (1999) Matrix Theory: Basic Results and Techniques. Springer-Verlag, New York. http://dx.doi.org/10.1007/978-1-4757-5797-2