Templates for Band Lanczos Methods and for the Complex Symmetric Lanczos Method*

Roland W. Freund
Bell Laboratories
Lucent Technologies
Room 2C–420
700 Mountain Avenue
Murray Hill, NJ 07974–0636

Abstract

The classical Lanczos process is a powerful tool for various computations involving large square matrices. The band Lanczos method is an extension of the classical Lanczos process for a pair of single starting vectors, to blocks of starting vectors. The method generates bi-orthogonal basis vectors for the right and left block-Krylov subspaces induced by the right and left starting blocks and the given square matrix, as well as an oblique projection of the given matrix onto the right block-Krylov subspace and orthogonal to the left block-Krylov subspace. In this paper, we present algorithmic descriptions of the Hermitian band Lanczos method, of the non-Hermitian band Lanczos method, and of a variant of the Lanczos process tailored to complex symmetric matrices.

The three sections of this paper are invited sections for the book project *Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide.*

1 The Hermitian Band Lanczos Method

The standard Hermitian Lanczos algorithm uses the Krylov subspaces induced by the matrix A and a single starting vector b, to produce approximate solutions of the Hermitian eigenproblem $Ax = \lambda x$. However, there are situations where the use of a block of p starting vectors, instead of a single starting vector, is preferable. One such case is matrices with multiple or closely clustered eigenvalues. To obtain basis vectors for the eigenspace corresponding to such a cluster of p eigenvalues, block-Krylov subspaces induced by A and a block of p

*Numerical Analysis Manuscript No. 99–3–01, Bell Laboratories, Murray Hill, New Jersey, January 1999. Available on WWW at `http://cm.bell-labs.com/cs/doc/99`.

starting vectors need to be used. Employing block-Krylov subspaces is also beneficial whenever computing matrix-matrix products AV, where V is a matrix with p columns, is cheaper than sequentially computing matrix-vector products Av for p vectors. Lanczos methods based on block-Krylov subspaces allow to compute all necessary multiplications with A as matrix-matrix products AV with blocks V of size p, whereas in the standard Hermitian Lanczos algorithm multiplications with A have to be computed as matrix-vector products Av.

In this section, we describe a band Lanczos method for the Hermitian eigenvalue problem,

$$Ax = \lambda x, \quad \text{where} \quad A = A^H, \tag{1}$$

that is based on block-Krylov subspaces induced by the matrix A and a block of p starting vectors,

$$b_1, b_2, \ldots, b_p. \tag{2}$$

The matrix A and the starting vectors (2) induce the block-Krylov sequence

$$b_1, b_2, \ldots, b_p, Ab_1, Ab_2, \ldots, Ab_p, \ldots, A^{k-1}b_1, A^{k-1}b_2, \ldots, A^{k-1}b_p, \ldots\ . \tag{3}$$

The goal of the band Lanczos algorithm is to construct orthonormal vectors,

$$v_1, v_2, \ldots, v_j, \tag{4}$$

that build a basis for the subspace spanned by the first j linearly independent vectors of the block-Krylov sequence (3).

1.1 Deflation

The use of $p > 1$ multiple starting vectors causes an additional difficulty that does not arise in the single-vector case $p = 1$. The standard Lanczos algorithm for a single starting vector b terminates after j iterations if the next Krylov vector, $A^j b$, is linearly dependent on the previous Krylov vectors, i.e., $A^j b \in K^j(A, b)$. In this case, the Lanczos vectors build a basis of the A-invariant subspace $K^j(A, b)$ and all eigenvalues of the Lanczos tridiagonal matrix are also eigenvalues of A. The subspace $K^j(A, b)$ being A-invariant means that the Krylov sequence has been fully exhausted, and so adding further Krylov vectors would not expand the Krylov subspace. This termination after j iterations is thus natural.

In the case $p > 1$, however, the occurrence of a first linearly dependent vector in (3) does not mean that the block-Krylov sequence is exhausted. It simply means that the linearly dependent vector and all its following A-multiples do not contain any new information, and therefore, these vectors should be removed from (3). This process of detecting and deleting linearly dependent vectors is called *exact deflation*. For example, suppose the starting vectors (2) are such that the vector Ab_1 is a linear combination of $b_1, b_2, \ldots, b_p$. Then Ab_1 is certainly linearly dependent on the Krylov vectors to the left of Ab_1 in (3); in fact, each vector $A^i b_1$ with $i \geq 1$ is linearly dependent on vectors to the left of $A^i b_1$ in (3). In this case, all vectors $A^i b_1$, $i \geq 1$, need to be deleted from (3), resulting in the deflated block-Krylov sequence

$$b_1, b_2, \ldots, b_p, Ab_2, Ab_3, \ldots, Ab_p, \ldots, A^{k-1}b_2, A^{k-1}b_3, \ldots, A^{k-1}b_p, \ldots\ . \tag{5}$$

For $p > 1$, in contrast to the case $p = 1$, the first occurrence of an exact deflation does not imply that any of the eigenvalues of the Lanczos matrix produced by the band Lanczos method is also an eigenvalue of A. However, after p exact deflations have occurred, then, just as in the case $p = 1$, the Lanczos vectors span an A-invariant subspace and all the eigenvalues of the Lanczos matrix are also eigenvalues of A.

Of course, in finite-precision arithmetic, it is impossible to distinguish between exactly linearly dependent and almost linearly dependent vectors. Therefore, in practice, almost linearly dependent vectors also have to be detected and deleted. In the sequel, we will refer to the process of detecting and deleting linearly dependent and almost linearly dependent vectors as *deflation*.

1.2 Basic Properties

After the first j iterations, the band Lanczos algorithm has generated the first j Lanczos vectors (4). These vectors are constructed to be orthonormal:

$$V_j^H V_j = I_j, \quad \text{where} \quad V_j = [\; v_1 \quad v_2 \quad \cdots \quad v_j \;]. \tag{6}$$

Here, I_j denotes the $j \times j$ identity matrix. In addition to (4), the algorithm has constructed the vectors

$$\widehat{v}_{j+1}, \widehat{v}_{j+2}, \ldots, \widehat{v}_{j+p_c}, \tag{7}$$

which are the candidates for the next p_c Lanczos vectors, $v_{j+1}, v_{j+2}, \ldots, v_{j+p_c}$. Here, p_c is the number of starting vectors, p, minus the number of deflations that have occurred during the first j iterations. The vectors (7) are constructed so that they satisfy the orthogonality relations

$$V_j^H \widehat{v}_k = 0 \quad \text{for all} \quad k = j+1, j+2, \ldots, j+p_c. \tag{8}$$

The band Lanczos algorithm has a very simple built-in deflation procedure based on the vectors (7). In fact, an exact deflation at iteration $j+1$ is equivalent to $\widehat{v}_{j+1} = 0$. Therefore, in the algorithm, one checks if $\|\widehat{v}_{j+1}\|_2$ is smaller than some suitable deflation tolerance. If yes, the vector $\widehat{v}_{j+1}$ is deflated and p_c is reduced by one. Otherwise, $\widehat{v}_{j+1}$ is normalized to become the next Lanczos vector v_{j+1}.

The recurrences that are used in the algorithm to generate the vectors (4) and (7) can be summarized compactly as follows:

$$AV_j = V_j T_j + [\; 0 \quad \cdots \quad 0 \quad \widehat{v}_{j+1} \quad \widehat{v}_{j+2} \quad \cdots \quad \widehat{v}_{j+p_c} \;] + \widehat{V}_j^{(\mathrm{dl})}. \tag{9}$$

Here, T_j is an $j \times j$ matrix whose entries are chosen so that the orthogonality conditions (6) and (8) are satisfied. The matrix $\widehat{V}_j^{(\mathrm{dl})}$ in (9) contains mostly zero columns, together with the vectors that have been deflated during the first j iterations. Recall that $p - p_c$ is the number of deflated vectors. It turns out that orthogonality only has to be explicitly enforced among $2p_c + 1$ consecutive Lanczos vectors and, once deflation has occurred, against $p - p_c$ fixed earlier vectors. As a result, the matrix T_j is "essentially" banded with bandwidth $2p_c + 1$, where the bandwidth is reduced by 2 every time a deflation occurs.

In addition, each deflation causes T_j to have nonzero elements in a fixed row outside and to the right of the banded part. More precisely, the row index of each such row caused by deflation is given by $k-p_c(k)$, where k is the number of the iteration at which the deflation has occurred and $p_c(k)$ is the corresponding value of p_c at that iteration. The matrix T_j can thus be written as

$$T_j = T_j^{(\mathrm{b})} + T_j^{(\mathrm{d})}, \tag{10}$$

where $T_j^{(\mathrm{b})}$ is a banded matrix and $T_j^{(\mathrm{d})}$ contains the horizontal "spikes" outside the band of T_j due to deflation. In particular, if no deflation has occurred, then $T_j^{(\mathrm{d})}$ is the zero matrix. Finally, we note that the banded part of T_j is Hermitian:

$$T_j^{(\mathrm{b})} = \left(T_j^{(\mathrm{b})}\right)^H. \tag{11}$$

For example, consider the case of $p = 5$ starting vectors and assume that during the first $j = 15$ iterations, deflations have occurred at steps $k = 8$, $k = 11$, and $k = 13$. These three deflations correspond to deleting the vectors Ab_3, A^2b_2, and A^3b_1, as well as the following A-multiples of these three vectors, from the block-Krylov sequence (3). In this case, the matrix $T_{15} = T_{15}^{(\mathrm{b})} + T_{15}^{(\mathrm{d})}$ has the following sparsity structure:

$$T_{15} = \begin{bmatrix}
* & * & * & * & * & * & & & & & & & & & \\
* & * & * & * & * & * & * & & & & & & & & \\
* & * & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
* & * & * & * & * & * & * & * & & & & & & & \\
* & * & * & * & * & * & * & * & * & & & & & & \\
* & * & * & * & * & * & * & * & * & * & & & & & \\
 & * & * & * & * & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
 & & & * & * & * & * & * & * & * & * & & & & \\
 & & & & * & * & * & * & * & * & * & * & & & \\
 & & & & & * & * & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
 & & & & & & & * & * & * & * & * & * & & \\
 & & & & & & & & * & * & * & * & * & * & \\
 & & & & & & & & & & * & * & * & * & * \\
 & & & & & & & & & & & * & * & * & * \\
 & & & & & & & & & & & & * & * & *
\end{bmatrix}. \tag{12}$$

Here, the $*$'s denote potentially nonzero entries within the banded part, $T_{15}^{(\mathrm{b})}$, and the $\mathtt{d}$'s denote potentially nonzero entries of $T_{15}^{(\mathrm{d})}$ due to the deflations at iterations $k = 8$, $k = 11$, and $k = 13$. Note that the three deflations have reduced the initial bandwidth $2p + 1 = 11$ to $2p_c + 1 = 5$ at iteration $j = 15$.

After j iterations of the band Lanczos algorithm, approximate eigensolutions for the Hermitian eigenvalue problem (1) are obtained by computing eigensolutions of $T_j^{(\mathrm{pr})}$,

$$T_j^{(\mathrm{pr})} z_i^{(j)} = \theta_i^{(j)} z_i^{(j)}, \quad i = 1, 2, \ldots, j. \tag{13}$$

Here, $T_j^{(\mathrm{pr})}$ is the projection of A onto the space spanned by the Lanczos basis matrix V_j, i.e.,

$$T_j^{(\mathrm{pr})} = V_j^H A V_j. \tag{14}$$

Each value $\theta_i^{(j)}$ and its Ritz vector, $x_i^{(j)} = V_j z_i^{(j)}$, yield an approximate eigenpair of A. Of course, the matrix $T_j^{(\mathrm{pr})}$ is not computed via its definition (14). Instead, we use the formula

$$T_j^{(\mathrm{pr})} = T_j + \left(T_j^{(\mathrm{d})}\right)^H. \tag{15}$$

By (15), we simply have to conjugate and transpose the part of T_j outside its banded part and add it to T_j in order to obtain $T_j^{(\mathrm{pr})}$. To show that (15) indeed holds true, note that by multiplying (9) from the left by V_j^H and by using the orthogonality relations (6) and (8), we get

$$T_j^{(\mathrm{pr})} = V_j^H A V_j = T_j + S_j, \quad \text{where} \quad S_j = V_j^H \widehat{V}_j^{(\mathrm{dl})}. \tag{16}$$

Since the matrices $T_j^{(\mathrm{pr})}$ and $T_j^{(\mathrm{b})}$ are both Hermitian, it follows from (10) that $S_j = \left(T_j^{(\mathrm{d})}\right)^H$ in (16). Thus (16) reduces to (15).

For example, for T_{15} in (12) the associated matrix $T_{15}^{(\mathrm{pr})} = T_{15} + \left(T_{15}^{(\mathrm{d})}\right)^H$ is of the form

$$T_{15}^{(\mathrm{pr})} = \left[\begin{array}{ccccccccccccccc}
* & * & * & * & * & * & & & & & & & & & \\
* & * & * & * & * & * & * & & & & & & & & \\
* & * & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
* & * & * & * & * & * & * & * & & & & & & & \\
* & * & * & * & * & * & * & * & * & & & & & & \\
* & * & * & * & * & * & * & * & * & * & & & & & \\
 & * & * & * & * & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
 & & \bar{\mathtt{d}} & * & * & * & * & * & * & * & * & & & & \\
 & & \bar{\mathtt{d}} & & * & * & * & * & * & * & * & * & & & \\
 & & \bar{\mathtt{d}} & & & * & * & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
 & & \bar{\mathtt{d}} & & & & \bar{\mathtt{d}} & * & * & * & * & * & * & & \\
 & & \bar{\mathtt{d}} & & & & \bar{\mathtt{d}} & & * & * & * & * & * & * & \\
 & & \bar{\mathtt{d}} & & & & \bar{\mathtt{d}} & & & \bar{\mathtt{d}} & * & * & * & * & * \\
 & & \bar{\mathtt{d}} & & & & \bar{\mathtt{d}} & & & \bar{\mathtt{d}} & & * & * & * & * \\
 & & \bar{\mathtt{d}} & & & & \bar{\mathtt{d}} & & & \bar{\mathtt{d}} & & & * & * & *
\end{array}\right]. \tag{17}$$

Here, the $\bar{\mathtt{d}}$'s below the banded part were obtained by conjugating and transposing the corresponding entries above the banded part in (12).

Finally, we note that the band Lanczos algorithm terminates as soon as $p_c = 0$ is reached. This means that p deflations have occurred, and thus the block-Krylov sequence is exhausted. At termination due to $p_c = 0$, the relation (9) for the Lanczos vectors reduces to

$$A V_j = V_j T_j + \widehat{V}_j^{(\mathrm{dl})}. \tag{18}$$

Using (16), we can rewrite (18) as follows:

$$AV_j - V_j T_j^{(\mathrm{pr})} = \left(I - V_j V_j^H\right) \widehat{V}_j^{(\mathrm{dl})}. \tag{19}$$

Now let $\theta_i^{(j)}$ and $z_i^{(j)}$ be any of the eigenpairs of $T_j^{(\mathrm{pr})}$, and assume that $z_i^{(j)}$ is normalized so that $\left\| z_i^{(j)} \right\|_2 = 1$. Recall that the pair $\theta_i^{(j)}$ and $x_i^{(j)} = V_j z_i^{(j)}$ is used as an approximate eigensolution of A. By multiplying (19) from the right by $z_i^{(j)}$ and taking norms, it follows that the residual of the approximate eigensolution $\theta_i^{(j)}$ and $x_i^{(j)}$ can be bounded as follows:

$$\left\| Ax_i^{(j)} - \theta_i^{(j)} x_i^{(j)} \right\|_2 = \left\| \left(I - V_j V_j^H\right) \widehat{V}_j^{(\mathrm{dl})} z_i^{(j)} \right\|_2 \leq \left\| \widehat{V}_j^{(\mathrm{dl})} \right\|_2. \tag{20}$$

In particular, if only exact deflation is performed, then $\widehat{V}_j^{(\mathrm{dl})} = 0$ and (20) shows that each eigenvalue $\theta_i^{(j)}$ of $T_j^{(\mathrm{pr})}$ is indeed an eigenvalue of A. For general deflation, $\widehat{V}_j^{(\mathrm{dl})} \neq 0$, however $\left\| \widehat{V}_j^{(\mathrm{dl})} \right\|_2$ is of the order of the deflation tolerance. More precisely, if the deflation check (22) below is used, then

$$\left\| \widehat{V}_j^{(\mathrm{dl})} \right\|_2 \leq \sqrt{p}\ \mathtt{dtol}, \tag{21}$$

where `dtol` denotes the deflation tolerance.

1.3 The Algorithm

A complete statement of the band Lanczos algorithm for Hermitian matrices A is as follows.

ALGORITHM 1.1: **Hermitian band Lanczos**

Input: *Starting vectors* $b_1, b_2, \ldots, b_p$.

Output: *The* $j \times j$ *Lanczos matrix* $T_j^{(\mathrm{pr})}$.

(1) *For* $k = 1, 2, \ldots, p$, *set* $\widehat{v}_k = b_k$. *Set* $p_c = p$ *and* $\mathcal{I} = \emptyset$.
(2) *For* $j = 1, 2, \ldots,$ *until convergence or* $p_c = 0$ *do*:
(3) *Compute* $\|\widehat{v}_j\|_2$.
(4) *Decide if* $\widehat{v}_j$ *should be deflated. If yes, do the following*:
a) *If* $j - p_c > 0$, *set* $\mathcal{I} = \mathcal{I} \cup \{j - p_c\}$.
b) *Set* $p_c = p_c - 1$. *If* $p_c = 0$, *set* $j = j - 1$ *and stop.*
c) *For* $k = j, j+1, \ldots, j + p_c - 1$, *set* $\widehat{v}_k = \widehat{v}_{k+1}$.
d) *Return to Step* (3).
(5) *Set* $t_{j,j-p_c} = \|\widehat{v}_j\|_2$ *and normalize* $v_j = \widehat{v}_j / t_{j,j-p_c}$.
(6) *For* $k = j+1, j+2, \ldots, j + p_c - 1$, *set*
$t_{j,k-p_c} = v_j^H \widehat{v}_k$ *and* $\widehat{v}_k = \widehat{v}_k - v_j t_{j,k-p_c}$.
(7) *Compute* $\widehat{v}_{j+p_c} = A v_j$.
(8) *Set* $k_0 = \max\{1, j - p_c\}$. *For* $k = k_0, k_0 + 1, \ldots, j - 1$, *set*
$t_{kj} = \overline{t_{jk}}$ *and* $\widehat{v}_{j+p_c} = \widehat{v}_{j+p_c} - v_k t_{kj}$.
(9) *For* $k \in \mathcal{I} \cup \{j\}$ *(in ascending order), set*
$t_{kj} = v_k^H \widehat{v}_{j+p_c}$ *and* $\widehat{v}_{j+p_c} = \widehat{v}_{j+p_c} - v_k t_{kj}$.
(10) *For* $k \in \mathcal{I}$, *set* $s_{jk} = \overline{t_{kj}}$.
(11) *Set* $T_j^{(\mathrm{pr})} = T_j + S_j = [\, t_{ik} \,]_{i,k=1,2,\ldots,j} + [\, s_{ik} \,]_{i,k=1,2,\ldots,j}$.
(12) *Test for convergence.*

Next, we discuss some of the steps of Algorithm 1.1 in more detail.

Step (4) Generally, the decision if $\widehat{v}_j$ needs to be deflated should be based on checking if

$$\|\widehat{v}_j\|_2 \le \texttt{dtol}, \tag{22}$$

where `dtol` is a suitably chosen small deflation tolerance. The vector $\widehat{v}_j$ is then deflated if (22) is satisfied. In this case, the value $j - p_c$, if positive, is added to the index set $\mathcal{I}$, which contains the indices of the nonzero rows in the part $T_j^{(\mathrm{d})}$ of T_j, and the current block size is updated to $p_c = p_c - 1$. If $p_c = 0$, then the block-Krylov sequence (3) is exhausted, and the algorithm terminates naturally. If $p_c > 0$, the vector $\widehat{v}_j$ is deleted, the index $k+1$ of each of the remaining candidate vectors $\widehat{v}_{k+1}$ is reset to k, and finally, the algorithm returns to Step (3). If (22) is not satisfied, then no deflation is necessary and the algorithm proceeds with Step (5).

We note that the algorithm performs only exact deflation if one sets $\texttt{dtol} = 0$ in (22).

Step (5) The vector $\widehat{v}_j$ has passed the deflation check and is now normalized to become the next Lanczos vector v_j. The normalization is such that

$$\|v_j\|_2 = 1 \quad \text{for all} \quad j. \tag{23}$$

Step (6) The remaining candidate vectors, $\widehat{v}_{j+1}, \widehat{v}_{j+1}, \ldots, \widehat{v}_{j+p_c-1}$, are explicitly orthogonalized against the latest Lanczos vector v_j.

Step (7) The block-Krylov sequence is advanced by computing a new candidate vector, $\widehat{v}_{j+p_c}$, as the A-multiple of the latest Lanczos vector v_j.

Step (8) The vector $\widehat{v}_{j+p_c}$ is orthogonalized against the previous Lanczos vectors $v_{k_0}, v_{k_0+1}, \ldots, v_{j-1}$, where $k_0 = \max\{1, j - p_c\}$. Here, we exploit the fact that the matrix $T_j^{(\mathrm{pr})}$ is Hermitian and instead of explicitly computing t_{kj}, we set $t_{kj} = \overline{t_{jk}}$. Note that the t_{jk}'s were computed explicitly in Step (6).

Step (9) The vector $\widehat{v}_{j+p_c}$ is explicitly orthogonalized against the Lanczos vectors v_k, $k \in \mathcal{I}$, and against v_j. Note that modified Gram-Schmidt is used to do this orthogonalization.

Step (11) All the nonzero elements in the j-th row and the j-th column have now been computed and they are added to the matrix $T_{j-1}^{(\mathrm{pr})}$ of the previous iteration $j-1$, to yield the current matrix $T_j^{(\mathrm{pr})}$. Here, we use the convention that entries t_{ik} and s_{ik} that are not explicitly defined in Algorithm 1.1 are set to be zero.

Step (12) To test for convergence, the eigenvalues $\theta_i^{(j)}$, $i = 1, 2, \ldots, j$, of the Hermitian $j \times j$ matrix $T_j^{(\mathrm{pr})}$ are computed, and the algorithm is stopped if some of the $\theta_i^{(j)}$'s are good enough approximations to the desired eigenvalues of A.

1.4 Variants

A band Lanczos method for Hermitian matrices was first proposed in [21]. The Hermitian band Lanczos Algorithm 1.1 described here, however, is somewhat different in that it uses both the first j Lanczos vectors (4) and the candidate vectors (7) for the following p_c Lanczos vectors. This allows to only build the $j \times j$ matrix $T_j^{(\mathrm{pr})}$. The algorithm in [21] constructs the first $j + p_c$ Lanczos vectors and as a result, it needs to build a somewhat larger $(j + p_c) \times j$ matrix, instead of $T_j^{(\mathrm{pr})}$. The version of the band Lanczos method described here was first derived in [14] as a special case of the general nonsymmetric band Lanczos method proposed in [20].

The version of the band Lanczos method stated as Algorithm 1.1 performs all multiplications with A as matrix-vector products with single vectors. However, at any stage of the algorithm, it is also possible to pre-compute the next p_c matrix-vector products, which will be needed in the next p_c iterations, as a single matrix-matrix product AV with a block V of p_c vectors. The procedure is as follows. Instead of performing Step (7) in Algorithm 1.1, we compute the matrix-matrix product

$$A\left[\begin{array}{ccccc} v_j & \widehat{v}_{j+1} & \widehat{v}_{j+2} & \cdots & \widehat{v}_{j+p_c-1} \end{array}\right]. \tag{24}$$

This gives us the vector $\widehat{v}_{j+p_c} = Av_j$, which is used in the remaining steps of iteration j, as well as the vectors

$$A\widehat{v}_{j+1}, A\widehat{v}_{j+2}, \ldots, A\widehat{v}_{j+p_c-1}. \tag{25}$$

In the following $p_c - 1$ iterations, we will need the vectors

$$Av_{j+1}, Av_{j+2}, \ldots, Av_{j+p'_c-1}. \tag{26}$$

Here, p'_c is defined as p_c minus the number of deflations that will have occurred during these following $p_c - 1$ iterations. In particular, $p'_c = p_c$ if no deflations will occur. The matrix-vector products (26) we need are not quite the ones (25) we computed. However, the vectors (25) and (26) are connected by the relation

$$\begin{aligned} &\left[\begin{array}{cccc} A\widehat{v}_{j+1} & A\widehat{v}_{j+2} & \cdots & A\widehat{v}_{j+p_c-1} \end{array}\right] \\ &\qquad = \left[\begin{array}{cccc} Av_{j+1} & Av_{j+2} & \cdots & Av_{j+p'_c-1} \end{array}\right] T_{j+1:j+p'_c-1,\,j-p_c+1:j-1}. \end{aligned} \tag{27}$$

Here, $T_{j+1:j+p'_c-1,\,j-p_c+1:j-1}$ is the submatrix of $T_{j+p'_c-1}$ that consists of the entries in rows $j+1, \ldots, j+p'_c-1$ and columns $j-p_c+1, \ldots, j-1$ of $T_{j+p'_c-1}$. If $p'_c < p_c$, then some of the $\widehat{v}_k$'s in (27) will lead to deflated vectors. Let $\widehat{v}_{j_1}, \widehat{v}_{j_2}, \ldots, \widehat{v}_{j_{p'_c-1}}$, be the vectors that are not deflated. Furthermore, let $\widetilde{T}$ be the matrix obtained from $T_{j+1:j+p'_c-1,\,j-p_c+1:j-1}$ by deleting the columns that correspond to the deflated $\widehat{v}_k$ vectors in (27). With these notations, by (27), we have

$$\left[\begin{array}{cccc} A\widehat{v}_{j_1} & A\widehat{v}_{j_2} & \cdots & A\widehat{v}_{j_{p'_c-1}} \end{array}\right] = \left[\begin{array}{cccc} Av_{j+1} & Av_{j+2} & \cdots & Av_{j+p'_c-1} \end{array}\right] \widetilde{T}. \tag{28}$$

The matrix $\widetilde{T}$ is nonsingular and upper triangular. Therefore, using (28), we can obtain each of the vectors Av_{j+k}, $k = 1, 2, \ldots, p'_c - 1$, in (26) as a suitable linear combination of k of the pre-computed vectors (25).

For Hermitian positive definite matrices A, yet another variant of Algorithm 1.1, based on coupled recurrences, was proposed in [4]. The motivation for this variant was the observation that for ill-conditioned Hermitian positive definite matrices A, the smallest eigenvalue obtained from the Lanczos matrix $T_j^{(\mathrm{pr})}$ may become negative; see [3, 4] for examples. In exact arithmetic, this would be impossible, since the positive definiteness of A implies that $T_j^{(\mathrm{pr})} = V_j^H A V_j$ is also positive definite. However, in finite-precision arithmetic, round-off may cause the matrix $T_j^{(\mathrm{pr})}$ generated by Algorithm 1.1 to be slightly indefinite. This can be avoided easily by using coupled recurrences to generate a Cholesky factor of $T_j^{(\mathrm{pr})}$, instead of $T_j^{(\mathrm{pr})}$ itself. More precisely, the recurrences summarized in (9) are replaced by coupled recurrences of the form

$$\begin{aligned} AP_j &= V_j L_j D_j + \left[\begin{array}{cccccccc} 0 & \cdots & 0 & \widehat{v}_{j+1} & \widehat{v}_{j+2} & \cdots & \widehat{v}_{j+p_c} \end{array}\right] + \widehat{V}_j^{(\mathrm{dl})}, \\ V_j &= P_j U_j. \end{aligned} \tag{29}$$

Here, P_j is a second basis matrix whose columns span the same subspace as V_j, and it is constructed to be A-orthogonal:

$$P_j^H A P_j = D_j, \tag{30}$$

where D_j is a positive definite diagonal matrix. Furthermore, in (29), the matrix L_j is unit lower triangular, and the matrix U_j is unit upper triangular. After j iterations, the approximate eigenpairs of A are then obtained by computing the eigensolutions of the Hermitian positive definite matrix

$$V_j^H A V_j = U_j^H D_j U_j, \tag{31}$$

which is given in terms of D_j and U_j.

Finally, we note that the block Lanczos method is the more traditional way of extending the standard Lanczos algorithm for a single starting vector to multiple starting vectors. When the block Lanczos method is combined with deflation (as described, e.g., in [6]), then the resulting algorithm is mathematically equivalent to the band Lanczos method described here. However, the band Lanczos method has two distinguished advantages over the block Lanczos method. First, deflation is extremely simple, since it requires only to check the norm of a single vector. In contrast, in the block Lanczos method, one needs to check if a whole block of p_c vectors has full rank, and if not, find the linearly dependent columns. Typically, this requires the computation of a QR factorization of each block. Second, the band Lanczos method actually performs slightly fewer explicit orthogonalizations, resulting in a Lanczos matrix $T_j^{(\mathrm{pr})}$ that is slightly sparser than the one produced by the block Lanczos method.

2 The Non-Hermitian Band Lanczos Method

The standard non-Hermitian Lanczos algorithm [19] uses the Krylov subspaces induced by the matrix A and a pair of single right and left starting vectors b and c, to produce approximate solutions of the non-Hermitian eigenproblem,

$$Ax = \lambda x. \tag{32}$$

Here, A is a square, in general non-Hermitian matrix.

There are situations where the use of blocks of right and left starting vectors, instead of a pair of single starting vectors, is preferable. One such case is eigenvalue computations for matrices with multiple or closely clustered eigenvalues. Another important application is reduced-order modeling of linear dynamical systems. Here, the right and left starting blocks are given as part of the problem, as is described in more detail in Section 2.4 below. Finally, the use of blocks of starting vectors is also beneficial whenever computing matrix-matrix products AV and $A^T W$, where V and W are blocks of vectors, is cheaper than sequentially computing matrix-vector products Av and $A^T w$ for all the columns of V and W.

In this section, we describe the non-Hermitian band Lanczos method, which extends the standard non-Hermitian Lanczos algorithm for single starting vectors, to blocks of m right and p left starting vectors,

$$b_1, b_2, \ldots, b_m \quad \text{and} \quad c_1, c_2, \ldots, c_p. \tag{33}$$

The matrix A and the vectors (33) are allowed to be real or complex. However, even when they are complex, we will state the algorithm in terms of A and A^T, rather than A and A^H, since this way, we can avoid unnecessary complex conjugations in the recurrence relations used in the algorithm. We stress that both formulations are equivalent.

The matrix A and the starting vectors (33) induce the right block-Krylov sequence

$$b_1, b_2, \ldots, b_m, Ab_1, Ab_2, \ldots, Ab_m, \ldots, A^{k-1}b_1, \ldots, A^{k-1}b_m, \ldots, \tag{34}$$

and the left block-Krylov sequence

$$c_1, c_2, \ldots, c_p, A^T c_1, A^T c_2, \ldots, A^T c_p, \ldots, (A^T)^{k-1} c_1, \ldots, (A^T)^{k-1} c_p, \ldots\ . \tag{35}$$

The goal of the band Lanczos algorithm is to construct suitable right and left Lanczos vectors,

$$v_1, v_2, \ldots, v_j \quad \text{and} \quad w_1, w_2, \ldots, w_j, \tag{36}$$

that build bases for the subspaces spanned by the first j linearly independent vectors of the block-Krylov sequence (34) and (35), respectively.

The non-Hermitian band Lanczos method discussed in this section can also be viewed as an extension of the Hermitian band Lanczos method described in Section 1, to general square non-Hermitian matrices. For the special case of right and left starting blocks of the same size, i.e., $m = p$, the band Lanczos method is also related to the block Lanczos method. However, the band Lanczos method is more general in that it can handle the case of arbitrary block sizes $m, p \geq 1$. Even for the special case $m = p$, there are advantages of the band method over the block Lanczos method; see Section 2.5 below.

2.1 Deflation

As in the case of the Hermitian band Lanczos method, the use of multiple starting vectors necessitates a suitable *deflation* procedure to delete linearly and almost linearly dependent vectors in the block-Krylov sequences. We refer to Section 1.1 for a discussion of deflation in the Hermitian case.

In the non-Hermitian case, deflation in general occurs independently in the right and left block-Krylov sequences. An *exact deflation* in the right block-Krylov sequence (34) means that a vector, say $A^i b_j$, in the sequence (34) is linearly dependent on vectors to the left of $A^i b_j$ in (34) and that this vector and all its A-multiples are removed from (34). Similarly, an *exact deflation* in the left block-Krylov sequence (35) means that a vector in the sequence (35) is linearly dependent on previous Krylov vectors in (35) and that this vector and all its A^T-multiples are removed from (35). After m exact deflations in the right block-Krylov sequence (34) have occurred, the remaining non-deflated vectors of (34) span an A-invariant subspace. Furthermore, the right Lanczos vectors in (36) build a suitable basis for this A-invariant subspace and all the eigenvalues of the Lanczos matrix $T_j^{(\mathrm{pr})}$ defined in Section 2.2 below are also eigenvalues of A. Similarly, after p exact deflations in the left block-Krylov sequence (35) have occurred, the remaining non-deflated vectors of (35) span an A^T-invariant subspace. The left Lanczos vectors in (36) build a suitable basis for this A^T-invariant subspace and all the eigenvalues of the Lanczos matrix $T_j^{(\mathrm{pr})}$ are also eigenvalues of A.

Of course, in finite-precision arithmetic, it is impossible to distinguish between exactly linearly dependent and almost linearly dependent vectors. Therefore, in practice, almost linearly dependent vectors also have to be detected and deleted. In the sequel, we will refer to the process of detecting and deleting linearly dependent and almost linearly dependent vectors as *deflation*.

2.2 Basic Properties

After j iterations, the algorithm has generated the vectors (36). It will be convenient to introduce the notation

$$V_j = \begin{bmatrix} v_1 & v_2 & \cdots & v_j \end{bmatrix} \quad \text{and} \quad W_j = \begin{bmatrix} w_1 & w_2 & \cdots & w_j \end{bmatrix} \tag{37}$$

for the matrices whose columns are just the right and left Lanczos vectors (36), respectively. The vectors (36) are constructed to be bi-orthogonal. Using the notation (37), the bi-orthogonality can be stated compactly as follows:

$$W_j^T V_j = D_j = \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_j). \tag{38}$$

In order to enforce bi-orthogonality of the next Lanczos vectors, the algorithm involves division by the diagonal entries, δ_k, in (38). As a result, the algorithm has to be stopped as soon as

$$\delta_k = w_k^T v_k = 0, \quad \text{but} \quad v_k,\ w_k \neq 0, \tag{39}$$

occurs. The situation (39) is called a *breakdown* of the algorithm. While breakdowns can be remedied by incorporating so-called *look-ahead* into the algorithm, here, for simplicity, we discuss only the band Lanczos algorithm without look-ahead.

After j iterations, in addition to (36), the algorithm has constructed the vectors

$$\widehat{v}_{j+1}, \widehat{v}_{j+2}, \ldots, \widehat{v}_{j+m_c} \quad \text{and} \quad \widehat{w}_{j+1}, \widehat{w}_{j+2}, \ldots, \widehat{w}_{j+p_c}. \tag{40}$$

The vectors $\widehat{v}_{j+1}, \widehat{v}_{j+2}, \ldots, \widehat{v}_{j+m_c}$ are the candidates for the next m_c right Lanczos vectors, $v_{j+1}, v_{j+2}, \ldots, v_{j+m_c}$, and the vectors $\widehat{w}_{j+1}, \widehat{w}_{j+2}, \ldots, \widehat{w}_{j+p_c}$ are the candidates for the next p_c left Lanczos vectors, $w_{j+1}, w_{j+2}, \ldots, w_{j+p_c}$. Here, m_c is the number of right starting vectors, m, minus the number of deflations in the right block-Krylov sequence that have occurred during the first j iterations, and p_c is the number of left starting vectors, p, minus the number of deflations in the left block-Krylov sequence that have occurred during the first j iterations. The vectors (40) are constructed so that they satisfy the bi-orthogonality relations

$$\begin{aligned} W_j^T \widehat{v}_k &= 0 \quad \text{for all} \quad k = j+1, j+2, \ldots, j+m_c, \\ V_j^T \widehat{w}_k &= 0 \quad \text{for all} \quad k = j+1, j+2, \ldots, j+p_c. \end{aligned} \tag{41}$$

The algorithm has a very simple built-in deflation procedure based on the vectors (40). In fact, an exact deflation in the right block-Krylov sequence at iteration $j+1$ is equivalent to $\widehat{v}_{j+1} = 0$. Therefore, in the algorithm, one checks if $\|\widehat{v}_{j+1}\|_2$ is smaller than some suitable deflation tolerance. If yes, the vector $\widehat{v}_{j+1}$ is deflated and m_c is reduced by one. Otherwise, $\widehat{v}_{j+1}$ is normalized to become the next right Lanczos vector v_{j+1}. Similarly, an exact deflation in the left block-Krylov sequence at iteration $j+1$ is equivalent to $\widehat{w}_{j+1} = 0$. In the algorithm, one checks if $\|\widehat{w}_{j+1}\|_2$ is smaller than the deflation tolerance. If yes, the vector $\widehat{w}_{j+1}$ is deflated and p_c is reduced by one. Otherwise, $\widehat{w}_{j+1}$ is normalized to become the next left Lanczos vector w_{j+1}.

The recurrences that are used in the algorithm to generate the vectors (36) and (40) can be summarized compactly as follows:

$$\begin{aligned} AV_j &= V_j T_j + \begin{bmatrix} 0 & \cdots & 0 & \widehat{v}_{j+1} & \widehat{v}_{j+2} & \cdots & \widehat{v}_{j+m_c} \end{bmatrix} + \widehat{V}_j^{(\mathrm{dl})}, \\ A^T W_j &= W_j \widetilde{T}_j + \begin{bmatrix} 0 & \cdots & 0 & \widehat{w}_{j+1} & \widehat{w}_{j+2} & \cdots & \widehat{w}_{j+p_c} \end{bmatrix} + \widehat{W}_j^{(\mathrm{dl})}. \end{aligned} \tag{42}$$

Here, T_j and $\widetilde{T}_j$ are $j \times j$ matrices whose entries are chosen so that the bi-orthogonality conditions (38) and (41) are satisfied. The matrix $\widehat{V}_j^{(\mathrm{dl})}$ in (42) contains mostly zero columns, together with the $\widehat{v}_k$ vectors that have been deflated during the first j iterations. The matrix $\widehat{W}_j^{(\mathrm{dl})}$ in (42) contains mostly zero columns, together with the $\widehat{w}_k$ vectors that have been deflated during the first j iterations. We remark that $m - m_c$ is the number of deflated $\widehat{v}_k$ vectors and that $p - p_c$ is the number of deflated $\widehat{w}_k$ vectors. It turns out that bi-orthogonality only has to be explicitly enforced among $m_c + p_c + 1$ consecutive Lanczos vectors and, once deflation has occurred, against $p - p_c$ fixed earlier left Lanczos vectors, respectively $m - m_c$ fixed earlier right Lanczos vectors. As a result, the matrices T_j and $\widetilde{T}_j$ are "essentially" banded. More precisely, T_j has lower bandwidth $m_c + 1$ and upper bandwidth $p_c + 1$, where the lower bandwidth is reduced by 1 every time a $\widehat{v}_k$ vector is deflated, and the upper bandwidth is reduced by 1 every time a $\widehat{w}_k$ vector is deflated. In addition, each deflation of a $\widehat{w}_k$ vector causes T_j to have nonzero elements in a fixed row outside and to the right of the banded part. More precisely, the row index of each such row caused by deflation of a $\widehat{w}_k$ vector is given by $k - p_c(k)$, where k is the number of the iteration at which the deflation has occurred and $p_c(k)$ is the corresponding value of p_c at that iteration. The matrix T_j can thus be written as

$$T_j = T_j^{(\mathrm{b})} + T_j^{(\mathrm{d})}, \tag{43}$$

where $T_j^{(\mathrm{b})}$ is a banded matrix and $T_j^{(\mathrm{d})}$ contains horizontal "spikes" above the band of T_j due to deflation of $\widehat{w}_k$ vectors. Similarly,

$$\widetilde{T}_j = \widetilde{T}_j^{(\mathrm{b})} + \widetilde{T}_j^{(\mathrm{d})}, \tag{44}$$

where the banded part $\widetilde{T}_j^{(\mathrm{b})}$ has lower bandwidth $p_c + 1$ and upper bandwidth $m_c + 1$ and $\widetilde{T}_j^{(\mathrm{d})}$ contains horizontal "spikes" above the band of $\widetilde{T}_j$ due to deflation of $\widehat{v}_k$ vectors.

The entries of the matrices T_j and $\widetilde{T}_j$ are not independent of each other. More precisely, setting

$$T_j^{(\mathrm{pr})} = T_j + D_j^{-1}\left(\widetilde{T}_j^{(\mathrm{d})}\right)^T D_j \quad \text{and} \quad \widetilde{T}_j^{(\mathrm{pr})} = \widetilde{T}_j + D_j^{-1}\left(T_j^{(\mathrm{d})}\right)^T D_j, \tag{45}$$

we have

$$T_j^{(\mathrm{pr})} = D_j^{-1}\left(\widetilde{T}_j^{(\mathrm{pr})}\right)^T D_j, \tag{46}$$

where D_j is the diagonal matrix given by (38). Inserting (43) into the definition of $T_j^{(\mathrm{pr})}$ in (45), we obtain the relation

$$T_j^{(\mathrm{pr})} = T_j^{(\mathrm{b})} + T_j^{(\mathrm{d})} + D_j^{-1}\left(\widetilde{T}_j^{(\mathrm{d})}\right)^T D_j, \tag{47}$$

which shows that $T_j^{(\mathrm{pr})}$ consists of the banded part $T_j^{(\mathrm{b})}$, horizontal spikes due to deflation of $\widehat{w}_k$ vectors above the banded part, and vertical spikes due to deflation of $\widehat{v}_k$ vectors below the banded part.

For example, consider the case of $m = 3$ right and $p = 5$ left starting vectors. Assume that during the first $j = 15$ iterations, deflations of $\widehat{w}_k$ vectors have occurred at iterations $k = 8$, $k = 11$, and $k = 13$, and deflations of $\widehat{v}_k$ vectors have occurred at iterations $k = 7$ and $k = 12$. In this case, the matrix $T_{15}^{(\mathrm{pr})}$ has the following sparsity structure:

$$T_{15}^{(\mathrm{pr})} = \begin{bmatrix}
* & * & * & * & * & * & & & & & & & & & \\
* & * & * & * & * & * & * & & & & & & & & \\
* & * & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
* & * & * & * & * & * & * & * & & & & & & & \\
 & * & * & * & * & * & * & * & * & & & & & & \\
 & & * & * & * & * & * & * & * & * & & & & & \\
 & & & \tilde{\mathtt{d}} & * & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
 & & & \tilde{\mathtt{d}} & & * & * & * & * & * & * & & & & \\
 & & & \tilde{\mathtt{d}} & & & * & * & * & * & * & * & & & \\
 & & & \tilde{\mathtt{d}} & & & & * & * & * & * & * & \mathtt{d} & \mathtt{d} & \mathtt{d} \\
 & & & \tilde{\mathtt{d}} & & & & & * & * & * & * & * & & \\
 & & & \tilde{\mathtt{d}} & & & & & & \tilde{\mathtt{d}} & * & * & * & * & \\
 & & & \tilde{\mathtt{d}} & & & & & & \tilde{\mathtt{d}} & & * & * & * & * \\
 & & & \tilde{\mathtt{d}} & & & & & & \tilde{\mathtt{d}} & & & * & * & * \\
 & & & \tilde{\mathtt{d}} & & & & & & \tilde{\mathtt{d}} & & & & * & *
\end{bmatrix}. \tag{48}$$

Here, the $*$'s denote potentially nonzero entries within the banded part, $T_{15}^{(\mathrm{b})}$, the $\mathtt{d}$'s denote potentially nonzero entries due to the deflations of $\widehat{w}_k$ vectors at iterations $k = 8$, $k = 11$, and $k = 13$, and the $\tilde{\mathtt{d}}$'s denote potentially nonzero entries due to the deflations of $\widehat{v}_k$ vectors at iterations $k = 7$ and $k = 12$. Note that the deflations have reduced the initial lower bandwidth $m + 1 = 4$ to $m_c + 1 = 2$ at iteration $j = 15$ and the initial upper bandwidth $p + 1 = 6$ to $p_c + 1 = 3$ at iteration $j = 15$.

After j iterations of the band Lanczos algorithm, approximate eigensolutions for the non-Hermitian eigenvalue problem (32) are obtained via an oblique projection of the matrix A onto the subspace spanned by the columns of V_j and orthogonal to the subspace spanned by the columns of W_j. More precisely, this means that we are seeking approximate eigenvectors of (32) of the form $x_i^{(j)} = V_j z_i^{(j)}$ and that, after inserting this ansatz for $x_i^{(j)}$ into (32), we multiply the resulting relation from the left by W_j^T. This yields the generalized eigenvalue problem

$$W_j^T A V_j z_i^{(j)} = \theta_i^{(j)} W_j^T V_j z_i^{(j)}, \quad i = 1, 2, \ldots, j. \tag{49}$$

Using the bi-orthogonality relations (38) and (41), it is easy to verify that the matrix $T_j^{(\mathrm{pr})}$ defined in (45) satisfies

$$T_j^{(\mathrm{pr})} = \left(W_j^T V_j\right)^{-1} W_j^T A V_j. \tag{50}$$

By (50), the generalized eigenvalue problem (49) is equivalent to the standard eigenvalue problem

$$T_j^{(\mathrm{pr})} z_i^{(j)} = \theta_i^{(j)} z_i^{(j)}, \quad i = 1, 2, \ldots, j. \tag{51}$$

We stress that, in the algorithm, we use the formula in (45) to obtain the entries of $T_j^{(\mathrm{pr})}$, rather than (50).

The band Lanczos algorithm terminates as soon as $m_c = 0$ or $p_c = 0$ is reached. In the case $m_c = 0$, m deflations of $\widehat{v}_k$ vectors have occurred, and thus the right block-Krylov sequence (34) is exhausted. In the case $p_c = 0$, p deflations of $\widehat{w}_k$ vectors have occurred, and thus the left block-Krylov sequence (35) is exhausted.

First consider termination due to $m_c = 0$. Then, the relation for the right Lanczos vectors in (42) can be rewritten as follows:

$$AV_j - V_j T_j^{(\mathrm{pr})} = P_j \widehat{V}_j^{(\mathrm{dl})}. \tag{52}$$

Here, the matrix

$$P_j = I - V_j D_j^{-1} W_j^T, \quad \text{where} \quad D_j = W_j^T V_j, \tag{53}$$

represents the oblique projection characterized by $P_j V_j = 0$ and $P_j x = x$ for all x in the null space of W_j^T. Now let $\theta_i^{(j)}$ and $z_i^{(j)}$ be any of the eigenpairs of $T_j^{(\mathrm{pr})}$, and assume that $z_i^{(j)}$ is normalized so that $\left\| z_i^{(j)} \right\|_2 = 1$. Recall that the pair $\theta_i^{(j)}$ and $x_i^{(j)} = V_j z_i^{(j)}$ is used as an approximate eigensolution of A. From (52), it follows that the residual of this approximate eigensolution can be bounded as follows:

$$\left\| Ax_i^{(j)} - \theta_i^{(j)} x_i^{(j)} \right\|_2 = \left\| P_j \widehat{V}_j^{(\mathrm{dl})} z_i^{(j)} \right\|_2 \leq \| P_j \|_2 \cdot \left\| \widehat{V}_j^{(\mathrm{dl})} \right\|_2. \tag{54}$$

In particular, if only exact deflation is performed, then $\widehat{V}_j^{(\mathrm{dl})} = 0$ and (54) shows that each eigenvalue $\theta_i^{(j)}$ of $T_j^{(\mathrm{pr})}$ is indeed an eigenvalue of A.

Similarly, in the case of termination due to $p_c = 0$, the relation for the left Lanczos vectors in (42) can be rewritten as follows:

$$A^T W_j - W_j \widetilde{T}_j^{(\mathrm{pr})} = P_j^T \widehat{W}_j^{(\mathrm{dl})}. \tag{55}$$

Here, P_j is again the matrix defined in (53). Now let $\theta_i^{(j)}$ and $\widetilde{z}_i^{(j)}$ be any of the eigenpairs of $\left(T_j^{(\mathrm{pr})} \right)^T$, and assume that $\widetilde{z}_i^{(j)}$ is normalized such that $\left\| D_j^{-1} \widetilde{z}_i^{(j)} \right\|_2 = 1$. Note that the complex conjugate of $\widetilde{z}_i^{(j)}$ is a left eigenvector of $T_j^{(\mathrm{pr})}$. The pair $\theta_i^{(j)}$ and $y_i^{(j)} = W_j D_j^{-1} \widetilde{z}_i^{(j)}$ then represents an approximate eigensolution of A^T. From (55), it follows that the residual of this approximate eigensolution can be bounded as follows:

$$\left\| A^T y_i^{(j)} - \theta_i^{(j)} y_i^{(j)} \right\| = \left\| P_j^T \widehat{W}_j^{(\mathrm{dl})} D_j^{-1} \widetilde{z}_i^{(j)} \right\|_2 \leq \left\| P_j^T \right\|_2 \cdot \left\| \widehat{W}_j^{(\mathrm{dl})} \right\|_2. \tag{56}$$

In particular, if only exact deflation is performed, then $\widehat{W}_j^{(\mathrm{dl})} = 0$ and (56) shows that each eigenvalue $\theta_i^{(j)}$ of $T_j^{(\mathrm{pr})}$ is indeed an eigenvalue of A^T and thus of A.

2.3 The Algorithm

A complete statement of the band Lanczos algorithm is as follows.

ALGORITHM 2.2: **Non-Hermitian band Lanczos**

Input: *Right starting vectors* $b_1, b_2, \ldots, b_m$.
Left starting vectors $c_1, c_2, \ldots, c_p$.

Output: *The* $j \times j$ *Lanczos matrix* $T_j^{(\mathrm{pr})}$.

(1) *For* $k = 1, 2, \ldots, m$, *set* $\widehat{v}_k = b_k$.
For $k = 1, 2, \ldots, p$, *set* $\widehat{w}_k = c_k$.
Set $m_c = m$, $p_c = p$, *and* $\mathcal{I}_v = \mathcal{I}_w = \emptyset$.

(2) *For* $j = 1, 2, \ldots,$ *until convergence or* $m_c = 0$ *or* $p_c = 0$ *do*:

(3) *Compute* $\|\widehat{v}_j\|_2$.

(4) *Decide if* $\widehat{v}_j$ *should be deflated. If yes, do the following*:
a) *If* $j - m_c > 0$, *set* $\mathcal{I}_w = \mathcal{I}_w \cup \{j - m_c\}$.
b) *Set* $m_c = m_c - 1$. *If* $m_c = 0$, *set* $j = j - 1$ *and stop.*
c) *For* $k = j, j+1, \ldots, j + m_c - 1$, *set* $\widehat{v}_k = \widehat{v}_{k+1}$.
d) *Return to Step* (3).

(5) *Compute* $\|\widehat{w}_j\|_2$.

(6) *Decide if* $\widehat{w}_j$ *should be deflated. If yes, do the following*:
a) *If* $j - p_c > 0$, *set* $\mathcal{I}_v = \mathcal{I}_v \cup \{j - p_c\}$.
b) *Set* $p_c = p_c - 1$. *If* $p_c = 0$, *set* $j = j - 1$ *and stop.*
c) *For* $k = j, j+1, \ldots, j + p_c - 1$, *set* $\widehat{w}_k = \widehat{w}_{k+1}$.
d) *Return to Step* (5).

(7) *Set* $t_{j,j-m_c} = \|\widehat{v}_j\|_2$ *and* $\widetilde{t}_{j,j-p_c} = \|\widehat{w}_j\|_2$.
Normalize $v_j = \widehat{v}_j / t_{j,j-m_c}$ *and* $w_j = \widehat{w}_j / \widetilde{t}_{j,j-p_c}$.

(8) *Compute* $\delta_j = w_j^T v_j$. *If* $\delta_j = 0$, *stop.*

(9) *Compute* $\widehat{v}_{j+m_c} = A v_j$.

(10) *Set* $k_v = \max\{1, j - p_c\}$.
Set $\mathcal{I}_v^{(\mathrm{e})} = \mathcal{I}_v \cup \{\, k_v, k_v + 1, \ldots, j - 1 \,\}$.
For $k \in \mathcal{I}_v^{(\mathrm{e})}$ *(in ascending order), set*
$t_{kj} = w_k^T \widehat{v}_{j+m_c} / \delta_k$ *and* $\widehat{v}_{j+m_c} = \widehat{v}_{j+m_c} - v_k t_{kj}$.

(11) *Compute* $\widehat{w}_{j+p_c} = A^T w_j$.

(12) *Set* $k_w = \max\{1, j - m_c\}$.
Set $\mathcal{I}_w^{(\mathrm{e})} = \mathcal{I}_w \cup \{\, k_w, k_w + 1, \ldots, j - 1 \,\}$.
For $k \in \mathcal{I}_w^{(\mathrm{e})}$ *(in ascending order), set*
$\widetilde{t}_{kj} = \widehat{w}_{j+p_c}^T v_k / \delta_k$ *and* $\widehat{w}_{j+p_c} = \widehat{w}_{j+p_c} - w_k \widetilde{t}_{kj}$.

(13) *For* $k = j+1, j+2, \ldots, j + m_c$, *set*
$t_{j,k-m_c} = w_j^T \widehat{v}_k / \delta_j$ *and* $\widehat{v}_k = \widehat{v}_k - v_j t_{j,k-m_c}$.

(14) *For* $k = j+1, j+2, \ldots, j + p_c$, *set*
$\widetilde{t}_{j,k-p_c} = \widehat{w}_k^T v_j / \delta_j$ *and* $\widehat{w}_k = \widehat{w}_k - w_j \widetilde{t}_{j,k-p_c}$.

(15) *For* $k \in \mathcal{I}_w$, *set* $t_{jk} = \widetilde{t}_{kj}\, \delta_k / \delta_j$.

(16) *Set* $T_j^{(\mathrm{pr})} = [\, t_{ik} \,]_{i,k=1,2,\ldots,j}$.

(17) *Test for convergence.*

At the time of this writing, there is no software for the non-Hermitian band Lanczos method in public domain. However, Matlab implementations of the method, even with look-ahead, will be made available soon, and can then be obtained via `http://cm.bell-labs.com/who/freund`.

We stress that the non-Hermitian band Lanczos method can be implemented by directly following the statement in Algorithm 2.2, together with the further details of some of the steps given below. To keep the length of the statement to one page, in Algorithm 2.2, all potentially nonzero entries of the banded parts of T_j and $\widetilde{T}_j$ are computed as inner products. However, only roughly half of these entries need to be explicitly computed as inner products. The remaining entries can be obtained via the relation

$$t_{ik} = \widetilde{t}_{ki}\, \delta_k / \delta_i, \tag{57}$$

which follows from the connection (46) of $T_j^{(\mathrm{pr})}$ and $\widetilde{T}_j^{(\mathrm{pr})}$. In particular, in the following discussion of Steps (10), (12), (13), and (14), we give formulas for the entries of T_j and $\widetilde{T}_j$ that use (57) whenever possible. Employing these formulas not only minimizes the number of inner products, but it typically also enhances numerical stability. Therefore, we recommend to use these formulas in an actual implementation of the non-Hermitian band Lanczos method.

Next, we discuss some of the steps of Algorithm 2.2 in more detail.

Step (4) Generally, the decision if $\widehat{v}_j$ needs to be deflated should be based on checking if

$$\|\widehat{v}_j\|_2 \le \texttt{dtol}, \tag{58}$$

where `dtol` is a suitably chosen small deflation tolerance. The vector $\widehat{v}_j$ is then deflated if (58) is satisfied. In this case, the value $j - m_c$, if positive, is added to the index set $\mathcal{I}_w$, which contains the indices of the nonzero rows in the part $\widetilde{T}_j^{(\mathrm{d})}$ of $\widetilde{T}_j$, and the current right block size is updated to $m_c = m_c - 1$. If $m_c = 0$, then the right block-Krylov sequence (34) is exhausted, and the algorithm terminates naturally. If $m_c > 0$, the vector $\widehat{v}_j$ is deleted, the index $k+1$ of each of the remaining right candidate vectors $\widehat{v}_{k+1}$ is reset to k, and finally, the algorithm returns to Step (3). If (58) is not satisfied, then no deflation is necessary and the algorithm proceeds with Step (5).

Step (6) In analogy to (58), the decision if $\widehat{w}_j$ needs to be deflated is based on checking if

$$\|\widehat{w}_j\|_2 \le \texttt{dtol}. \tag{59}$$

The vector $\widehat{w}_j$ is then deflated if (59) is satisfied. In this case, the value $j - p_c$, if positive, is added to the index set $\mathcal{I}_v$, which contains the indices of the nonzero rows in the part $T_j^{(\mathrm{d})}$ of T_j, and the current left block size is updated to $p_c = p_c - 1$. If $p_c = 0$, then the left block-Krylov sequence (35) is exhausted, and the algorithm terminates naturally. If $p_c > 0$, the vector $\widehat{w}_j$ is deleted, the index $k+1$ of each of the remaining left candidate vectors $\widehat{w}_{k+1}$ is reset to k, and finally, the algorithm returns to Step (5). If (59) is not satisfied, then no deflation is necessary and the algorithm proceeds with Step (7).

Step (7) Both vectors $\widehat{v}_j$ and $\widehat{w}_j$ have passed the deflation check and are now normalized to become the next right and left Lanczos vectors v_j and w_j, respectively. The normalization is such that

$$\|v_j\|_2 = \|w_j\|_2 = 1 \quad \text{for all} \quad j. \tag{60}$$

Step (8) Here, we compute δ_j and check for breakdown. If $\delta_j = 0$, then look-ahead would be needed in order to continue the algorithm.

Step (9) In this step, we advance the right block-Krylov sequence by computing a new candidate vector, $\widehat{v}_{j+m_c}$, as the A-multiple of the latest right Lanczos vector v_j.

Step (10) The vector $\widehat{v}_{j+m_c}$ is bi-orthogonalized against the left Lanczos vectors w_k, $k \in \mathcal{I}_v^{(\mathrm{e})}$. Note that a two-sided modified Gram-Schmidt procedure is used to do this bi-orthogonalization. One of the inner products required in the computation of t_{kj} can be saved by employing (57). More precisely, the following formula for t_{kj} should be used:

$$t_{kj} = \begin{cases} \widetilde{t}_{jk}\, \delta_j / \delta_k & \text{if} \quad k = j - p_c, \\ \\ w_k^T \widehat{v}_{j+m_c} / \delta_k & \text{otherwise.} \end{cases} \tag{61}$$

Note that the necessary entry $\widetilde{t}_{jk}$ in (61) is available from Step (7).

Step (11) In this step, we advance the left block-Krylov sequence by computing a new candidate vector, $\widehat{w}_{j+p_c}$, as the A^T-multiple of the latest left Lanczos vector w_j.

Step (12) The vector $\widehat{w}_{j+p_c}$ is bi-orthogonalized against the right Lanczos vectors v_k, $k \in \mathcal{I}_w^{(\mathrm{e})}$. Note that this bi-orthogonalization is done by means of a two-sided modified Gram-Schmidt procedure. Again, to save one inner product, the following formula for $\widetilde{t}_{kj}$ should be used:

$$\widetilde{t}_{kj} = \begin{cases} t_{jk}\, \delta_j / \delta_k & \text{if} \quad k = j - m_c, \\ \\ \widehat{w}_{j+p_c}^T v_k / \delta_k & \text{otherwise.} \end{cases} \tag{62}$$

Step (13) The right candidate vectors, $\widehat{v}_{j+1}, \widehat{v}_{j+2}, \ldots, \widehat{v}_{j+m_c}$, are bi-orthogonalized against the latest left Lanczos vector w_j. To save inner products and also enhance numerical stability, the following formula for $t_{j,k-m_c}$ should be used:

$$t_{j,k-m_c} = \begin{cases} w_j^T \widehat{v}_k / \delta_j & \text{if} \quad k - m_c \leq 0 \quad \text{or} \quad k - m_c = j, \\ \\ \widetilde{t}_{k-m_c,j}\, \delta_{k-m_c} / \delta_j & \text{otherwise.} \end{cases} \tag{63}$$

Step (14) The left candidate vectors, $\widehat{w}_{j+1}, \widehat{w}_{j+2}, \ldots, \widehat{w}_{j+p_c}$, are bi-orthogonalized against the latest right Lanczos vector v_j. To save inner products and also enhance numerical stability, the following formula for $\widetilde{t}_{j,k-p_c}$ should be used:

$$\widetilde{t}_{j,k-p_c} = \begin{cases} \widehat{w}_k^T v_j / \delta_j & \text{if } \; k - p_c \leq 0, \\[2ex] t_{k-p_c,j} \, \delta_{k-p_c} / \delta_j & \text{otherwise.} \end{cases} \tag{64}$$

Step (15) The entries t_{jk} computed in this step are the potentially nonzero entries in the j-th row of $T_j^{(\mathrm{pr})}$ due to the vertical spikes caused by the deflation of $\widehat{v}_k$ vectors. Note that again formula (57) is employed.

Step (16) All the potentially nonzero elements in the j-th row and the j-th column of $T_j^{(\mathrm{pr})}$ have now been computed and they are added to the matrix $T_{j-1}^{(\mathrm{pr})}$ of the previous iteration $j-1$, to yield the current matrix $T_j^{(\mathrm{pr})}$. Here, we use the convention that entries t_{ik} that are not explicitly defined in Algorithm 2.2 are set to be zero. We remark that in the initial iterations, i.e., as long as $j \leq m_c$, respectively $j \leq p_c$, Algorithm 2.2 also produces entries t_{jk}, respectively $\widetilde{t}_{jk}$, with negative indices $k \leq 0$. These entries arise from the bi-orthogonalization of the starting vectors, and they are not part of the matrix $T_j^{(\mathrm{pr})}$. In particular, these entries are not needed when Algorithm 2.2 is used for eigenvalue computations. However, they are crucial when Algorithm 2.2 is applied to reduced-order modeling, as we will discuss in Section 2.4 below.

Step (17) For eigenvalue computations, one tests for convergence by computing the eigenvalues $\theta_i^{(j)}$, $i = 1, 2, \ldots, j$, of the $j \times j$ matrix $T_j^{(\mathrm{pr})}$. The algorithm is stopped if some of the $\theta_i^{(j)}$'s are good enough approximations to the desired eigenvalues of A. For reduced-order modeling, the algorithm is stopped if the j-th order model generated by the algorithm is a good enough approximation of the original linear dynamical system.

2.4 Application to Reduced-Order Modeling

For eigenvalue computations, one is usually free to choose the right and left starting vectors for the band Lanczos method. However, there are other important applications where the starting vectors are given as part of the problem. One such application is reduced-order modeling of time-invariant linear dynamical systems with multiple inputs and multiple outputs; see, e.g., [13] for a recent survey. Such systems are characterized by matrix-valued transfer functions of the form

$$H(s) = C^T \left(I - sA\right)^{-1} B, \quad s \in \mathbb{C}, \tag{65}$$

Here, A is a square, in general non-Hermitian matrix, and

$$B = \begin{bmatrix} b_1 & b_2 & \cdots & b_m \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} c_1 & c_2 & \cdots & c_p \end{bmatrix} \tag{66}$$

are rectangular matrices with m and p columns, respectively. Moreover, m is the number of inputs, p is the number of outputs, and m and p are different in general. The band Lanczos method (applied to A and with the columns of the matrices (66), B and C, as right and left starting vectors) can be used to generate a reduced-order model of the linear dynamical system described by (65).

For reduced-order modeling, the entries t_{jk} and $\widetilde{t}_{jk}$ with negative indices $k \leq 0$ are also used. More precisely, in this case, Step (16) in Algorithm 2.2 is augmented by the following six lines:

If $j \leq m_c$,
set $\rho_{jk} = t_{j,k-m}$ *for all* k *with* $j - m_c + m \leq k \leq m$,
and set $\rho = [\,\rho_{ik}\,]_{i=1,2,\ldots,j,\,k=1,2,\ldots,m}$.
If $j \leq p_c$,
set $\eta_{jk} = \widetilde{t}_{j,k-p}$ *for all* k *with* $j - p_c + p \leq k \leq p$,
and set $\eta = [\,\eta_{ik}\,]_{i=1,2,\ldots,j,\,k=1,2,\ldots,p}$.

Here, again we use the convention that entries ρ_{ik} and η_{ik} that are not explicitly defined in Algorithm 2.2 are set to be zero.

The matrix ρ is upper triangular and of size $m_1 \times m$. Here, m_1 is defined as the value of $m_c = m_c(j)$ at the iteration j of Algorithm 2.2 for which $j = m_c$ is reached. It turns out that m_1 is just the value of m minus the number of right initial vectors $b_1, b_2, \ldots, b_m$ that have been deflated. In particular, $m_1 \leq m$, and $m_1 = m$ if, and only if, none of the right starting vectors has been deflated. The matrix η is upper triangular and of size $p_1 \times p$. Here, p_1 is defined as the value of $p_c = p_c(j)$ at the iteration j of Algorithm 2.2 for which $j = p_c$ is reached. The number p_1 is the value of p minus the number of left initial vectors $c_1, c_2, \ldots, c_p$ that have been deflated. In particular, $p_1 \leq p$, and $p_1 = p$ if, and only if, none of the left starting vectors has been deflated. The entries of ρ are the coefficients used to turn the right starting vectors into the first m_1 right Lanczos vectors, and the entries of η are the coefficients used to turn the left starting vectors into the first p_1 left Lanczos vectors.

Now let $j \geq \max\{m_1, p_1\}$, and let $T_j^{(\mathrm{pr})}$, ρ, and η be the matrices generated after j iterations of Algorithm 2.2. These three matrices then define a j-th order reduced model of the original transfer function (65) as follows:

$$H_j(s) = \eta_j^T D_j \left(I_j - s T_j^{(\mathrm{pr})} \right)^{-1} \rho_j, \quad s \in \mathbb{C}. \tag{67}$$

Here,

$$\rho_j = \begin{bmatrix} \rho \\ 0 \end{bmatrix} \in \mathbb{C}^{j \times m} \quad \text{and} \quad \eta_j = \begin{bmatrix} \eta \\ 0 \end{bmatrix} \in \mathbb{C}^{j \times p} \tag{68}$$

are the matrices ρ and η with $j - m_1$, respectively $j - p_1$, rows of zeros added. The reduced-order model (67) can be shown to be characterized as a certain matrix-Padé approximation of the original transfer function (65); see [13] and the references given there.

2.5 Variants

The traditional approach to extending Krylov-subspace techniques for single starting vectors to multiple starting vectors is to design a block version of the Krylov-subspace algorithm. A block version of the non-Hermitian Lanczos process was first proposed in [17, 18]. However, this block Lanczos algorithm assumes that the right and left starting blocks have the same size, i.e., $m = p$, and it does not have deflation, nor look-ahead. It is easy to see that *any* block version of the non-Hermitian Lanczos process requires all right and left blocks to have the same size. In particular, any block version is restricted to the special case that $m = p$ and that possible deflations occur simultaneously in the right and left block Krylov sequences, so that $m_c = p_c$ throughout the algorithm.

The non-Hermitian band Lanczos algorithm described in this section, however, suffers from none of these restrictions. In particular, it can be used for starting blocks of arbitrary sizes $m,\, p \geq 1$, and it allows deflations that occur independently in the right and left block Krylov sequences.

The version of the band Lanczos method stated in Algorithm 2.2 first appeared in this form in [12]. A somewhat different version, which also includes a look-ahead procedure to remedy possible breakdowns, is described in [1].

The version of the band Lanczos method stated as Algorithm 2.2 performs all multiplications with A and A^T as matrix-vector products with single vectors. However, at any stage of the algorithm, it is also possible to pre-compute the next m_c matrix-vector products with A, which will be needed in the next m_c iterations, as a single matrix-matrix product AV with a block V of m_c vectors. Indeed, instead of performing Step (9) in Algorithm 2.2, one computes the matrix-matrix product

$$A \left[\begin{array}{ccccc} v_j & \widehat{v}_{j+1} & \widehat{v}_{j+2} & \cdots & \widehat{v}_{j+m_c-1} \end{array} \right]. \tag{69}$$

This gives us the vector $\widehat{v}_{j+m_c} = Av_j$, which is used in the remaining steps of iteration j, as well as the vectors

$$A\widehat{v}_{j+1}, A\widehat{v}_{j+2}, \ldots, A\widehat{v}_{j+m_c-1}. \tag{70}$$

In the following $m_c - 1$ iterations, we will need the vectors

$$Av_{j+1}, Av_{j+2}, \ldots, Av_{j+m'_c-1}. \tag{71}$$

Here, m'_c is defined as m_c minus the number of deflations of $\widehat{v}_k$ vectors that will have occurred during these following $m_c - 1$ iterations. Using the procedure outlined in Section 1.4, one can easily obtain the vectors (71) as suitable linear combinations of the pre-computed vectors (70). Similarly, at any stage of the algorithm, it is possible to pre-compute the next p_c matrix-vector products with A^T, which will be needed in the next p_c iterations, as a single matrix-matrix product $A^T W$ with a block W of p_c vectors. Indeed, instead of performing Step (11) in Algorithm 2.2, one computes the matrix-matrix product

$$A^T \left[\begin{array}{ccccc} w_j & \widehat{w}_{j+1} & \widehat{w}_{j+2} & \cdots & \widehat{w}_{j+p_c-1} \end{array} \right] \tag{72}$$

and then proceeds analogously as for the vectors (70).

3 The Complex Symmetric Lanczos Method

In this section, we consider the complex symmetric eigenvalue problem,

$$Ax = \lambda x, \quad \text{where} \quad A = A^T \in \mathbb{C}^{n\times n}. \tag{73}$$

3.1 Properties of Complex Symmetric Matrices

Complex symmetry is a purely algebraic property, and it has no effect on the spectrum of the matrix. Indeed, for any given set of n numbers,

$$\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{C}, \tag{74}$$

there exists a complex symmetric $n \times n$ matrix A whose eigenvalues are just the prescribed numbers (74); see, e.g., [16, Theorem 4.4.9].

A complex symmetric matrix may not even be diagonalizable. For example, consider the complex symmetric matrix

$$A = \begin{bmatrix} 2\mathtt{i} & 1 \\ 1 & 0 \end{bmatrix}, \quad \text{where} \quad \mathtt{i} = \sqrt{-1}. \tag{75}$$

The only eigenvalue of this matrix is $\lambda = \mathtt{i}$, with algebraic multiplicity 2, but geometric multiplicity 1. In fact, the Jordan normal form of A is as follows:

$$Z^{-1}AZ = \begin{bmatrix} \mathtt{i} & 1 \\ 0 & \mathtt{i} \end{bmatrix}, \quad \text{where} \quad Z = \begin{bmatrix} \mathtt{i} & 1 \\ 1 & 0 \end{bmatrix}.$$

Thus, A is not diagonalizable.

If a complex symmetric matrix A is diagonalizable, then it has an eigendecomposition that reflects the complex symmetry; see, e.g., [16, Theorem 4.4.13]. More precisely, a complex symmetric matrix A is diagonalizable if, and only if, its eigenvector matrix, Z, can be chosen such that

$$Z^TAZ = \Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \quad \text{and} \quad Z^TZ = I_n. \tag{76}$$

An matrix Z with n columns that satisfies $Z^TZ = I_n$ is called *complex orthogonal.* The complex orthogonality of Z in (76) reflects the complex symmetry of A.

We remark that the eigendecomposition (76) is the suitable adaptation of the corresponding decomposition for Hermitian matrices. Recall that for any matrix $A = A^H$, the eigenvector matrix Z can always be chosen to be unitary:

$$Z^HAZ = \Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \quad \text{and} \quad Z^HZ = I_n. \tag{77}$$

The unitarity of Z in (77) reflects the fact that A is Hermitian.

The reason why an eigendecomposition (76) does not always exist is that there are vectors z with

$$z^Tz = 0, \quad \text{but} \quad z \neq 0. \tag{78}$$

Indeed, suppose A has an eigenvalue with a one-dimensional eigenspace and the vector z spanning that space satisfies (78). Then one of the columns of any eigenvector matrix Z of A would be of the form $z_k = \gamma z$, where $\gamma \neq 0$ is a scalar. Then, by (78), $z_k^T z_k = 0$, while the complex orthogonality condition, $Z^T Z = I_n$, in (76) would imply $z_k^T z_k = 1$. Note that for example (75), the vector $z = \begin{bmatrix} \mathtt{i} & 1 \end{bmatrix}^T$ spans the one-dimensional eigenspace associated with $\lambda = \mathtt{i}$ and it satisfies (78).

3.2 Properties of the Algorithm

While the complex symmetry of A has no effect on the eigenvalues of A, this particular structure can be exploited to halve the work and storage requirement of the general non-Hermitian Lanczos method [19]. Indeed, while the non-Hermitian Lanczos method involves one matrix-vector product with A and one with A^H at each iteration, the complex symmetric Lanczos method only requires one matrix-vector product with A at each iteration.

After j iterations, the complex symmetric Lanczos method has generated j Lanczos vectors,

$$v_1, v_2, \ldots, v_j, \tag{79}$$

that span the j-th Krylov subspace $K^j(A, b)$ induced by the complex symmetric matrix A and any nonzero starting vector b. The vectors (79) are constructed to be complex orthogonal:

$$V_j^T V_j = I_j, \quad \text{where} \quad V_j = \begin{bmatrix} v_1 & v_2 & \cdots & v_j \end{bmatrix}. \tag{80}$$

Note that, in view of the eigendecomposition (76) of diagonalizable complex symmetry matrices A, the complex orthogonality (80) of the Lanczos vectors is natural.

The complex symmetric Lanczos algorithm computes the vectors (79) by means of three-term recurrences that can be summarized as follows:

$$AV_j = V_j T_j + \begin{bmatrix} 0 & \cdots & 0 & \widehat{v}_{j+1} \end{bmatrix}. \tag{81}$$

Here,

$$T_j = T_j^T = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_j \\ & & \beta_j & \alpha_j \end{bmatrix} \tag{82}$$

is a complex symmetric tridiagonal matrix whose entries are the coefficients of the three-term recurrences. The vector $\widehat{v}_{j+1}$ is the candidate for the next Lanczos vector, v_{j+1}. It is constructed so that the orthogonality condition

$$V_j^T \widehat{v}_{j+1} = 0 \tag{83}$$

is satisfied, and it only remains to be normalized so that $v_{j+1}^T v_{j+1} = 1$. However, it cannot be excluded that

$$\widehat{v}_{j+1}^T \widehat{v}_{j+1} = 0, \quad \text{but} \quad \widehat{v}_{j+1} \neq 0. \tag{84}$$

If (84) occurs, then a next vector v_{j+1} cannot be obtained by simply normalizing $\widehat{v}_{j+1}$, as it would require division by zero. Therefore, (84) is called a *breakdown* of the complex symmetric Lanczos algorithm. Breakdowns can be remedied by incorporating look-ahead into the algorithm. Here, for simplicity, we restrict ourselves to the complex symmetric Lanczos algorithm without look-ahead, and we simply stop the algorithm in case a breakdown (84) is encountered.

After j iterations of the complex symmetric Lanczos algorithm, approximate eigensolutions for the complex symmetric eigenvalue problem (73) are obtained by computing eigensolutions of T_j,

$$T_j z_i^{(j)} = \theta_i^{(j)} z_i^{(j)}, \quad i = 1, 2, \ldots, j. \tag{85}$$

Each value $\theta_i^{(j)}$ and its Ritz vector, $x_i^{(j)} = V_j z_i^{(j)}$, yield an approximate eigenpair of A. Note that T_j is the complex orthogonal projection of A onto the space spanned by the Lanczos basis matrix V_j, i.e.,

$$T_j = V_j^T A V_j. \tag{86}$$

Indeed, the relation follows by multiplying (81) from the left by V_j^T and by using the orthogonality relations (80) and (83). Of course, in the complex symmetric Lanczos algorithm, the matrix T_j is not computed via the relation (86). Instead, the symmetric tridiagonal structure in the definition (82) is exploited and only the diagonal and subdiagonal entries of T_j are explicitly generated.

3.3 The Algorithm

A complete statement of the complex symmetric Lanczos algorithm (without look-ahead) is as follows.

ALGORITHM 3.3: **Complex symmetric Lanczos**

Input: *Starting vector b.*

Output: *The $j \times j$ Lanczos matrix T_j.*

(1) *Set $\widehat{v}_1 = b$ and $v_0 = 0$.*
(2) *For $j = 1, 2, \ldots,$ until convergence or $\beta_j = 0$ do:*
(3) *If $\widehat{v}_j = 0$, set $j = j - 1$ and stop:*
All eigenvalues of T_j are also eigenvalues of A.
(4) *Compute $\beta_j = \sqrt{\widehat{v}_j^T \widehat{v}_j}$.*
(5) *If $\beta_j = 0$, set $j = j - 1$ and stop: breakdown.*
(In this case, look-ahead would be needed to continue.)
(6) *Normalize $v_j = \widehat{v}_j / \beta_j$.*
(7) *Compute $\widehat{v}_{j+1} = A v_j$.*
(8) *Subtract $\widehat{v}_{j+1} = \widehat{v}_{j+1} - v_{j-1}\beta_j$.*
(9) *Compute $\alpha_j = v_j^T \widehat{v}_{j+1}$.*
(10) *Subtract $\widehat{v}_{j+1} = \widehat{v}_{j+1} - v_j \alpha_j$.*
(11) *Test for convergence.*

Next, we comment on a few of the steps of Algorithm 3.3.

Step (3) If $\widehat{v}_j = 0$ occurs, then the algorithm has fully exhausted the Krylov sequence generated by A and b and thus termination is natural. In fact, in this case, the Lanczos vectors generated so far span an A-invariant subspace, and all eigenvalues of the Lanczos tridiagonal matrix are also eigenvalues of A.

Step (5) In practice, one also needs to stop if a so-called *near-breakdown*, i.e., $\beta_j \approx 0$, occurs. A look-ahead version of the algorithm remedies both exact breakdowns, i.e., $\beta_j = 0$, and near-breakdowns; see, e.g., [10, 15].

Step (11) To test for convergence, the eigenvalues $\theta_i^{(j)}$, $i = 1, 2, \ldots, j$, of the complex symmetric tridiagonal matrix T_j are computed, and the algorithm is stopped if some of the $\theta_i^{(j)}$'s are good enough approximations to the desired eigenvalues of A.

3.4 Notes

A detailed discussion of basic properties of complex symmetric matrices can be found in [5] and [16, Section 4.4].

Algorithm 3.3 can also be used for the iterative solution of complex symmetric linear systems. In [11], a QMR iterative solver based on Algorithm 3.3 is described. Some further theoretical results on complex symmetric matrices can also be found in [11].

A Fortran implementation of Algorithm 3.3, even with the option to perform look-ahead if necessary, is included in the software package QMRPACK [15]. The corresponding routines in QMRPACK are CSLAL (for single precision complex) and ZSLAL (for double precision complex). The optional look-ahead in these routines is based on the look-ahead strategy described in [10]. QMRPACK also contains implementations of the complex symmetric QMR iterative solver based on Algorithm 3.3. QMRPACK is available from *netlib* or directly from `http://cm.bell-labs.com/who/freund`.

Recall that Algorithm 3.3 produces a complex symmetric tridiagonal matrix T_j and that, in order to obtain approximate eigenvalues of A, at least some of the eigenvalues of T_j need to be computed. Using the QR algorithm for the task of computing all eigenvalues of T_j does not exploit the tridiagonal structure of T_j. Indeed, after one step of the QR algorithm, in general, the tridiagonal structure of T_j will have been destroyed and one obtains a full upper Hessenberg matrix, just as in the case of QR for general non-Hermitian eigenvalue problems. Cullum and Willoughby [7, 8, 9] developed a QL procedure for complex symmetric tridiagonal matrices that exploits this special structure. The algorithm is based on factorizations of complex symmetric tridiagonal matrices into a complex orthogonal matrix Q and a lower triangular matrix L. However, since Q is not unitary in general, carefully chosen heuristics are needed in order to monitor and maintain numerical stability; we refer the reader to [9] for a detailed description of the QL procedure.

If the complex symmetric Lanczos method is run with look-ahead, then the complex symmetric tridiagonal structure of the Lanczos matrix T_j will be destroyed as soon as a look-ahead step occurs. The matrix V_j of Lanczos vectors is then no longer complex orthogonal, but instead $D_j = V_j^T V_j$ is a complex symmetric block diagonal matrix, where the sizes of the diagonal blocks correspond to the lengths of the look-ahead steps. In particular, a 1×1 " block" occurs whenever a regular Lanczos step (without look-ahead) is possible, and each "true" block of size bigger than 1 corresponds to a look-ahead step. Furthermore, instead of (86), one now has the relation

$$D_j T_j = V_j^T A V_j, \tag{87}$$

which shows that $D_j T_j$ is complex symmetric. In addition, the matrix $D_j T_j$ is block tridiagonal with diagonal blocks of the same size as D_j. Therefore, in the look-ahead case, instead of the standard eigenvalue problem (85), one could solve the equivalent complex symmetric generalized eigenvalue problem

$$D_j T_j z_i^{(j)} = \theta_i^{(j)} D_j z_i^{(j)}, \quad i = 1, 2, \ldots, j. \tag{88}$$

Here, both matrices $D_j T_j$ and D_j are complex symmetric, $D_j T_j$ is block tridiagonal, and D_j is block diagonal. However, it is not clear how to exploit this special structure when solving (88).

Finally, we note that it is also possible to adapt other eigenvalue algorithms to complex symmetric matrices. For example, a Jacobi-type method for complex symmetric matrices is described in [2] .

References

[1] J.I. Aliaga, D.L. Boley, R.W. Freund, and V. Hernández. A Lanczos-type algorithm for multiple starting vectors, revised version. Technical Report Numerical Analysis Manuscript No. 98–3–05, Bell Laboratories, Murray Hill, New Jersey, USA, 1998.

[2] P.J. Anderson and G. Loizou. A Jacobi type method for complex symmetric matrices. *Numer. Math.*, 25:347–363, 1976.

[3] Z. Bai, P. Feldmann, and R.W Freund. How to make theoretically passive reduced-order models passive in practice. In *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference*, pages 207–210. IEEE, 1998.

[4] Z. Bai and R.W Freund. A band symmetric Lanczos process based on coupled recurrences with applications. Technical Report Numerical Analysis Manuscript, Bell Laboratories, Murray Hill, New Jersey, USA, 1998.

[5] B.D. Craven. Complex symmetric matrices. *J. Austral. Math. Soc.*, 10:341–354, 1969.

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

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

[8] J.K. Cullum and R.A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 2, Programs.* Birkhäuser, Boston, 1985.

[9] J.K. Cullum and R.A. Willoughby. A QL procedure for computing the eigenvalues of complex symmetric tridiagonal matrices. *SIAM J. Matrix Anal. Appl.*, 17:83–109, 1996.

[10] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comput.*, 14:137–158, 1993.

[11] R.W. Freund. Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Statist. Comput.*, 13:425–448, 1992.

[12] R.W Freund. Computing minimal partial realizations via a Lanczos-type algorithm for multiple starting vectors. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pages 4394–4399. IEEE, 1997.

[13] R.W. Freund. Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation. Technical Report Numerical Analysis Manuscript No. 98–3–02, Bell Laboratories, Murray Hill, New Jersey, USA, 1998. To appear in *Applied and Computational Control, Signals, and Circuits.*

[14] R.W. Freund and P. Feldmann. Reduced-order modeling of large linear passive multi-terminal circuits using matrix-Padé approximation. In *Proceedings of the Design, Automation and Test in Europe Conference 1998*, pages 530–537. IEEE Computer Society Press, 1998.

[15] R.W. Freund and N.M. Nachtigal. QMRPACK: a package of QMR algorithms. *ACM Trans. Math. Software*, 22:46–77, 1996.

[16] R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, UK, 1985.

[17] H.M. Kim and R.R. Craig, Jr. Structural dynamics analysis using an unsymmetric block Lanczos algorithm. *Internat. J. Numer. Methods Engrg.*, 26:2305–2318, 1988.

[18] H.M. Kim and R.R. Craig, Jr. Computational enhancement of an unsymmetric block Lanczos algorithm. *Internat. J. Numer. Methods Engrg.*, 30:1083–1089, 1990.

[19] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linea r differential and integral operators. *J. Res. Nat. Bur. Stand.*, 45:255–282, 1950.

[20] Feldmann P. and R.W. Freund. Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm. In *Proceedings of the 32nd Design Automation Conference*, pages 474–479. Association for Computing Machinery, Inc., 1995.

[21] A. Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Math. Comp.*, 33:680–687, 1979.