A LANCZOS-TYPE METHOD FOR MULTIPLE STARTING VECTORS[1]

J. I. ALIAGA, D. L. BOLEY, R. W. FREUND, AND V. HERNÁNDEZ

Abstract. Given a square matrix and single right and left starting vectors, the classical nonsymmetric Lanczos process generates two sequences of biorthogonal basis vectors for the right and left Krylov subspaces induced by the given matrix and vectors. In this paper, we propose a Lanczos-type algorithm that extends the classical Lanczos process for single starting vectors to multiple starting vectors. Given a square matrix and two blocks of right and left starting vectors, our algorithm generates two sequences of biorthogonal basis vectors for the right and left block Krylov subspaces induced by the given data. The algorithm can handle the most general case of right and left block Krylov subspaces with arbitrary sizes of the starting blocks, while all previously proposed extensions of the Lanczos process are restricted to right and left starting blocks of identical sizes. Other features of our algorithm include a built-in deflation procedure to detect and delete linearly dependent or almost linearly dependent vectors in the block Krylov sequences, and the option to employ look-ahead in order to avoid the potential breakdowns that are typical for nonsymmetric Lanczos-type methods.

1. Introduction

1.1. **The Lanczos process for single starting vectors.** Given a square matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$ and two nonzero starting vectors $\mathbf{r}, \mathbf{l} \in \mathbb{C}^N$, the classical nonsymmetric Lanczos process [28] is an algorithm that uses three-term recurrences to generate two sequences of biorthogonal basis vectors for the right, respectively left, Krylov subspaces induced by $\mathbf{A}$ and $\mathbf{r}$, respectively $\mathbf{A}^T$ and $\mathbf{l}$. Furthermore, the coefficients of the three-term recurrences define a sequence of $n \times n$ matrices $\mathbf{T}_n$, $n = 1, 2, \ldots$, that constitute approximations to the given matrix $\mathbf{A}$. More precisely, the n-th Lanczos matrix $\mathbf{T}_n$ represents the oblique projection of $\mathbf{A}$ onto the n-th right Krylov subspace and orthogonally to the n-the left Krylov subspace.

In the Lanczos process in its original form [28], breakdowns or near-breakdowns—triggered by division by zero or a number close to zero—cannot be excluded. Fortunately, the problem of potential breakdowns and near-breakdowns can be remedied by incorporating so-called look-ahead techniques into the Lanczos process. The possibility of such a remedy was first observed by Taylor [39] and Parlett, Taylor, and Liu [34], who also coined the term "look-ahead". Since then, there has been extensive research activities in this area, and as a result, the look-ahead Lanczos

[1] Numerical Analysis Manuscript No. 96–18, Bell Laboratories, Murray Hill, New Jersey, September 1996. Available on WWW at `http://cm.bell-labs.com/cs/doc/96`.

1991 *Mathematics Subject Classification.* Primary 65F10, 65F15; Secondary 65F25, 65F30.

Key words and phrases. Lanczos algorithm, nonsymmetric matrix, block Krylov subspaces, biorthogonalization, oblique projection, deflation, breakdown, look-ahead.

The first and the last author were supported in part by the European ESPRIT III Basic Research Project GEPPCOM #9072. The second author was supported in part by the U.S. NSF Grant #CCR-9405380.

process is now well understood; see, e.g., [5, 6, 14, 17, 24, 25, 33] and the references given therein. The basic principle of the look-ahead Lanczos process is to continue the algorithm in the event of a breakdown or near-breakdown by relaxing the vector-wise biorthogonality of the Lanczos basis vectors to a cluster-wise biorthogonality and by resorting, for the next few iteration steps, to recurrences that are slightly longer than the three-term recurrences in the classical algorithm.

When applied to large $N \times N$ matrices $\mathbf{A}$, the $n \times n$ Lanczos matrices $\mathbf{T}_n$ are often very good approximations to $\mathbf{A}$ already for $n \ll N$, and this makes the Lanczos process a powerful tool for various computational tasks for large matrices $\mathbf{A}$. We now briefly mention three such applications of the Lanczos process in large-scale matrix computations.

The first application is the computation of approximate eigenvalues of a matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$. Starting with arbitrary (for example, random) nonzero vectors $\mathbf{r}, \mathbf{l} \in \mathbb{C}^N$, one runs the Lanczos process for n steps to obtain $\mathbf{T}_n$. The eigenvalues of $\mathbf{T}_n$ are then used as approximate eigenvalues of the matrix $\mathbf{A}$; see, e.g., [11].

The second application is the solution of large systems of linear equations,

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}. \tag{1.1}$$

The biconjugate gradient (BCG) algorithm [29] and the quasi-minimal residual (QMR) algorithm [19, 20] are iterative methods that generate approximations $\mathbf{x}_n$ for the solution of (1.1), starting from an arbitrary initial guess $\mathbf{x}_0$ and an arbitrary nonzero left vector $\mathbf{l} \in \mathbb{C}^N$. Both algorithms are intimately connected to the Lanczos process applied to the matrix $\mathbf{A}$ and the starting vectors $\mathbf{r} = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$ and $\mathbf{l}$. Indeed, for BCG, the n-th iterate $\mathbf{x}_n$ is defined by a Galerkin-type condition that is mathematically equivalent to solving a small $n \times n$ linear system with $\mathbf{T}_n$ as coefficient matrix, instead of the large $N \times N$ system (1.1). For QMR, the n-th iterate $\mathbf{x}_n$ is defined by a quasi-minimization of the residual norm that is mathematically equivalent to solving a small $(n+1) \times n$ least-squares problem whose matrix is $\mathbf{T}_n$, extended by one more row. This additional row avoids possible singularity or near-singularity of $\mathbf{T}_n$, and results in a much smoother convergence behavior of QMR, compared to BCG.

A third application is Padé approximation of transfer functions describing large single-input single-output time-invariant linear dynamical systems. Such transfer functions are rational functions $H : \mathbb{C} \mapsto \mathbb{C} \cup \{\infty\}$ of the form

$$H(s) = \mathbf{l}^T(\mathbf{I}_N - s\mathbf{A})^{-1}\mathbf{r}, \tag{1.2}$$

where $\mathbf{A} \in \mathbb{C}^{N \times N}$, $\mathbf{r}, \mathbf{l} \in \mathbb{C}^N$ is given data, and $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. An n-th Padé approximant, H_n, to a given transfer function H of the type (1.2) is defined as a rational function with numerator and denominator degree at most $n-1$ and n, respectively, such that the Taylor expansions of H_n and H about $s = 0$ match in as many leading Taylor coefficients as possible. It is well known (see, e.g.,[22, 23, 24]) that H_n can be directly obtained from the Lanczos process applied to $\mathbf{A}$, $\mathbf{r}$, and $\mathbf{l}$. Indeed, assuming for simplicity that no look-ahead steps occur in the Lanczos algorithm, the n-th Padé approximant is simply given by

$$H_n(s) = \mathbf{l}^T\mathbf{r}\;\mathbf{e}_1^T(\mathbf{I}_n - s\mathbf{T}_n)^{-1}\mathbf{e}_1, \tag{1.3}$$

where $\mathbf{T}_n$ is the n-th Lanczos matrix and $\mathbf{e}_1$ denotes the first unit vector (in $\mathbb{R}^n$). A formula similar to (1.3) holds for the case that look-ahead steps do occur. We remark that the numerical computation of H_n via the Lanczos-Padé connection (1.3)

is significantly more stable than the standard approach of obtaining H_n via explicit calculations of the leading Taylor coefficients of H; see, e.g., [12].

1.2. **Handling multiple starting vectors.** All three applications described in Section 1.1 have extensions that involve multiple starting vectors.

For eigenvalue computations of a matrix $\mathbf{A}$ with multiple or clusters of eigenvalues, it is usually preferable to employ a Lanczos-type method that iterates on blocks of, say m, vectors, rather than on single vectors; see, e.g., [10, 21, 36]. Such a procedure then involves m right and m left starting vectors.

There are important applications where linear systems (1.1) need to be solved repeatedly with the same matrix $\mathbf{A}$, but different right-hand sides, say $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$. If all right-hand sides are available simultaneously, then these m systems can be summarized in block form as follows:

$$\mathbf{A}\,\mathbf{X} = \mathbf{B}, \quad \text{where} \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_m \end{bmatrix}. \tag{1.4}$$

Applying a suitable block version of an iterative method directly to (1.4) is often significantly more efficient than solving the m linear systems summarized in (1.4) individually; see, e.g., [18, 30, 31, 32]. Block versions of BCG and QMR now involve a block of m right starting vectors, namely $\mathbf{R} = \mathbf{B} - \mathbf{A}\,\mathbf{X}_0$, where $\mathbf{X}_0 \in \mathbb{C}^{N\times m}$ is an arbitrary initial guess for (1.4).

Finally, multi-input multi-output time-invariant linear dynamical systems are characterized by matrix-valued transfer functions $\mathbf{H} : \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{p\times m}$ of the form

$$\mathbf{H}(s) = \mathbf{L}^T(\mathbf{I}_N - s\mathbf{A})^{-1}\mathbf{R}. \tag{1.5}$$

Here, $\mathbf{A} \in \mathbb{C}^{N\times N}$, $\mathbf{R} \in \mathbb{C}^{N\times m}$, $\mathbf{L} \in \mathbb{C}^{N\times p}$, and m and p denote the number of inputs and outputs, respectively. Transfer functions (1.5) arise in different areas, such as control theory [4, 37, 38] and circuit simulation [13]. For further background on transfer functions (1.5) and the need to approximate them by reduced-order models, we refer the reader to [4, 13, 37, 38] and the references given therein. For matrix-valued transfer functions $\mathbf{H}$, one can again define n-th Padé approximants, $\mathbf{H}_n$, which, however, are now also matrix-valued functions, i.e., $\mathbf{H}_n : \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{p\times m}$. Extending the Lanczos-Padé connection (1.3) for the single-input single-output case, $m = p = 1$, to the general m-input p-output case, $m, p \geq 1$, requires a Lanczos-type process that can handle m right and p left starting vectors, namely the columns of $\mathbf{R}$ and $\mathbf{L}$, respectively. We note that, in general, m and p are different.

These three applications clearly show that there is a need for a Lanczos-type algorithm for multiple starting vectors. Furthermore, the algorithm should be an extension of the classical Lanczos process for single starting vectors, and thus generate two sequences of basis vectors for the right and left block Krylov subspaces induced by the given matrix and the block of right and left starting vectors. However, in order to obtain a robust Lanczos-type algorithm for multiple starting vectors, the following three key difficulties need to be resolved.

(i) The algorithm needs to include a *deflation* procedure in order to detect and delete linearly dependent or almost linearly dependent vectors in the right, respectively left, block Krylov subspaces. Moreover, in general, deflations in the right and left block Krylov subspaces occur independently of each other,

and consequently, the block sizes of both subspaces may become different in the course of the algorithm, even if they were identical at the beginning.

Note that for the Lanczos process for single starting vectors, deflation is not an issue. Encountering a linearly dependent right, respectively left, vector simply means that the right, respectively left, Krylov subspace is fully exhausted, and thus the algorithm terminates naturally in this situation.

(ii) The algorithm needs to be able to handle different block sizes in the right and left block Krylov subspaces. These different block sizes may be due to different numbers of right and left starting vectors, i.e., $m \neq p$, or due to deflation as mentioned in (i).

(iii) Just as in the classical Lanczos algorithm for single starting vectors, it cannot be excluded that breakdowns or near-breakdowns occur in a Lanczos-type process for multiple starting vectors. As a result, in the general case, look-ahead techniques need to be incorporated.

In this paper, we propose a Lanczos-type algorithm that extends the classical Lanczos process for single starting vectors to multiple starting vectors, and that can handle all three difficulties (i)–(iii) listed above. Given a matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$, a block of right starting vectors $\mathbf{R} \in \mathbb{C}^{N \times m}$, and a block of left starting vectors $\mathbf{L} \in \mathbb{C}^{N \times p}$, our algorithm generates two sequences of biorthogonal basis vectors for the right, respectively left, block Krylov subspaces induced by $\mathbf{A}$ and $\mathbf{R}$, respectively $\mathbf{A}^T$ and $\mathbf{L}$. The algorithm includes a simple built-in deflation procedure, and it can handle the most general case of right and left block Krylov subspaces with arbitrary sizes m and p of the starting blocks. We will actually describe two versions of the algorithm. First, we present the *generic* algorithm without look-ahead, and then we state the *general* algorithm with look-ahead.

The key property of our algorithm, which allows us to resolve the issues (i)–(iii), is the vector-wise construction of the basis vectors for the block Krylov subspaces. The idea of using a vector-wise approach appears to originate from Ruhe [36] who used it to derive a Lanczos algorithm for Hermitian $\mathbf{A}$ and multiple starting vectors. However, we stress that, for Hermitian matrices, the problem of handling multiple starting vectors is a lot easier for the following two reasons. First, in the Hermitian case, the right and left block Krylov subspaces are (up to complex conjugation if $\mathbf{A}$ is complex) identical, and thus the complication due to different right and left block sizes does not arise. Second, for Hermitian matrices, the possibility of breakdowns can be excluded, and hence no look-ahead is needed. We note that, to the best of our knowledge, we seem to be the first to extend Ruhe's vector-wise construction of basis vectors for the Hermitian case to the general case of non-Hermitian matrices.

We remark that early versions of the Lanczos-type algorithm described in this paper had been developed independently by Aliaga, Boley, and Hernández, and by Freund, and were presented by Boley [2] and Freund [16] at the same Oberwolfach meeting in 1994. It was then that we decided to write this joint paper. However, we would like to stress that the algorithm presented in this paper has evolved quite a bit from the earlier versions we had in 1994. Finally, we note that, in his doctoral thesis [1], Aliaga investigated variants of the algorithm that are tailored to parallel computers.

1.3. **Related work on block Lanczos methods.** The problem of extending the Lanczos process for single starting vectors to multiple starting vector is, of course, not new, and a number of algorithms have been developed over the years. However,

with the exception of Ruhe's algorithm [36] for the Hermitian case, all previously proposed Lanczos-type methods for multiple starting vectors use a block-wise construction of block-biorthogonal basis vectors for the underlying block Krylov subspaces. It is easy to see that any such block-wise approach requires all right and left blocks to have the same size. In particular, block Lanczos algorithms are restricted to the special case that $p = m$ and that possible deflation occurs simultaneously in the right and left block Krylov subspaces.

Block Lanczos algorithms for Hermitian matrices were first proposed by Cullum and Donath [9], and Golub and Underwood [21, 40]. Further and more recent work for the Hermitian case is described in [10, 31, 36] and the references given therein. We remark that only the algorithms in [9, 31] and Ruhe's algorithm [36] include a proper deflation procedure.

For non-Hermitian matrices, O'Leary—with her block BCG algorithm [32]—was the first to develop a block Lanczos-type method. A block version of the original three-term Lanczos algorithm [28] was first presented in [26, 27], and a more recent variant was proposed in [3]. As already pointed out above, all these algorithms are restricted to the case $p = m$. Furthermore, none of the existing block Lanczos-type methods for non-Hermitian matrices has a built-in deflation procedure, nor are there any look-ahead variants to remedy possible breakdowns.

1.4. **Outline.** The remainder of this article is organized as follows. In §2, we introduce our notion of block Krylov subspaces associated with multiple starting vectors. In §3, we state same basic properties of the Lanczos basis vectors generated by our Lanczos-type algorithm and describe the concept of history indices. In §4, we state the *generic* Lanczos-type algorithm without look-ahead for the case that no breakdowns occur. In §5, we present the *general* Lanczos-type algorithm with look-ahead included to avoid possible breakdowns. In §6, we establish the various properties of the Lanczos basis vectors. In §7, we discuss a few computational aspects of the proposed Lanczos-type algorithm. In §8, we make some concluding remarks. Finally, in an Appendix, we present a specific example to familiarize the reader with the notation used in the statement of the Lanczos-type algorithm.

1.5. **Notation.** Throughout this article, all vectors and matrices are allowed to have real or complex entries. We use boldface letters to denote vectors and matrices. As usual, $\overline{\mathbf{M}} = \big[\overline{m_{jk}}\big]$, $\mathbf{M}^T = \big[m_{kj}\big]$, and $\mathbf{M}^H = \overline{\mathbf{M}}^T = \big[\overline{m_{kj}}\big]$ denote the complex conjugate, transpose, and the conjugate transpose, respectively, of the matrix $\mathbf{M} = \big[m_{jk}\big]$. The vector norm $\|\mathbf{x}\| := \sqrt{\mathbf{x}^H \mathbf{x}}$ is always the Euclidean norm, and $\|\mathbf{M}\| := \max_{\|\mathbf{x}\|=1} \|\mathbf{M}\,\mathbf{x}\|$ is the corresponding induced matrix norm.

2. Block Krylov subspaces

From now on, it is always assumed that $\mathbf{A} \in \mathbb{C}^{N\times N}$ is a given $N \times N$ matrix,

$$\mathbf{R} = \begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \cdots & \mathbf{r}_m\end{bmatrix} \in \mathbb{C}^{N\times m} \tag{2.1}$$

is a given matrix of m *right starting vectors*, $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_m$, and

$$\mathbf{L} = \begin{bmatrix}\mathbf{l}_1 & \mathbf{l}_2 & \cdots & \mathbf{l}_p\end{bmatrix} \in \mathbb{C}^{N\times p} \tag{2.2}$$

is a given matrix of p *left starting vectors*, $\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_m$. We stress that $m \geq 1$ and $p \geq 1$ are *arbitrary* integers, and in particular, m and p need not be identical.

In this section, we introduce our notion of block Krylov subspaces induced by the data $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{L}$. We start by defining the *right block Krylov matrix*

$$\mathbf{K}(\mathbf{A},\mathbf{R}) := \begin{bmatrix}\mathbf{R} & \mathbf{A}\,\mathbf{R} & \mathbf{A}^2\,\mathbf{R} & \cdots & \mathbf{A}^{N-1}\,\mathbf{R}\end{bmatrix} \tag{2.3}$$

and the *left block Krylov matrix*

$$\mathbf{K}(\mathbf{A}^T,\mathbf{L}) := \begin{bmatrix}\mathbf{L} & \mathbf{A}^T\,\mathbf{L} & (\mathbf{A}^T)^2\,\mathbf{L} & \cdots & (\mathbf{A}^T)^{N-1}\,\mathbf{L}\end{bmatrix}. \tag{2.4}$$

Our goal is to construct two sequences of Lanczos basis vectors for the ascending n-dimensional subspaces, $n = 1, 2, \ldots$, spanned by the first n linearly independent columns of the matrices $\mathbf{K}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}(\mathbf{A}^T,\mathbf{L})$, respectively. To properly define these subspaces, we need to delete the linearly dependent (and possibly nearly linearly dependent) columns in (2.3) and (2.4). This is done by scanning the columns of each of the matrices $\mathbf{K}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}(\mathbf{A}^T,\mathbf{L})$, from left to right and deleting each column that is either linearly dependent or in some sense "almost" linearly dependent on earlier columns within the same matrix. This process of deleting linearly dependent and almost linearly dependent columns is referred to as *deflation* in the sequel. Moreover, we say that the deflation is *exact* if only the linearly dependent vectors are deleted, and we call it *inexact* if also nearly linearly dependent vectors are deleted. Applying deflation to (2.3) and (2.4), we obtain the *deflated* right and left block Krylov matrices $\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L})$, respectively.

By the structure (2.3) of $\mathbf{K}(\mathbf{A},\mathbf{R})$, a column $\mathbf{A}^{j-1}\mathbf{r}_i$ being linearly (or nearly linearly dependent) on earlier columns implies that all columns $\mathbf{A}^k\mathbf{r}_i$, $j \le k \le N-1$, are also linearly (or nearly linearly dependent) on earlier columns. An analogous statement holds for (2.4). Consequently, the deflated block Krylov matrices are of the form

$$\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R}) = \begin{bmatrix}\mathbf{R}_1 & \mathbf{A}\mathbf{R}_2 & \mathbf{A}^2\mathbf{R}_3 & \cdots & \mathbf{A}^{j_{\max}-1}\mathbf{R}_{j_{\max}}\end{bmatrix} \in \mathbb{C}^{N\times n^{(r)}} \tag{2.5}$$

and

$$\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L}) = \begin{bmatrix}\mathbf{L}_1 & \mathbf{A}^T\mathbf{L}_2 & (\mathbf{A}^T)^2\mathbf{L}_3 & \cdots & (\mathbf{A}^T)^{k_{\max}-1}\mathbf{L}_{k_{\max}}\end{bmatrix} \in \mathbb{C}^{N\times n^{(l)}}. \tag{2.6}$$

Here, for each $j = 1, 2, \ldots, j_{\max}$, $\mathbf{R}_j$ is a submatrix of $\mathbf{R}_{j-1}$, with $\mathbf{R}_j \neq \mathbf{R}_{j-1}$ if, and only if, deflation occurs within the j-th right Krylov block $\mathbf{A}^{j-1}\mathbf{R}$ in (2.3). (For $j = 1$, we set $\mathbf{R}_0 = \mathbf{R}$.) Similar, for each $k = 1, 2, \ldots, k_{\max}$, $\mathbf{L}_k$ is a submatrix of $\mathbf{L}_{k-1}$, with $\mathbf{L}_k \neq \mathbf{L}_{k-1}$ if, and only if, deflation occurs within the k-th left Krylov block $(\mathbf{A}^T)^{k-1}\mathbf{L}$ in (2.4). (For $k = 1$, we set $\mathbf{L}_0 = \mathbf{L}$.)

Note that, by construction, the columns of each deflated block Krylov matrix $\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L})$ are linearly independent. For $n = 1, 2, \ldots, n^{(r)}$, we denote by $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ the subspace of $\mathbb{C}^N$ spanned by the first n columns of the matrix $\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ in (2.5). We call $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ the n-th *right block Krylov subspace* (induced by $\mathbf{A}$ and $\mathbf{R}$). Similarly, for $n = 1, 2, \ldots, n^{(l)}$, the n-th *left block Krylov subspace* (induced by $\mathbf{A}^T$ and $\mathbf{L}$), denoted by $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L})$, is defined as the subspace of $\mathbb{C}^N$ spanned by the first n columns of the matrix $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L})$ in (2.6). By construction, both $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ and $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L})$ are subspaces of dimension n.

The goal of our Lanczos-type algorithm is to generate two sequences of basis vectors for the block Krylov subspaces $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ and $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L})$, $n \ge 1$. As in the classical nonsymmetric Lanczos process, the basis vectors are computed in pairs. At pass n of the algorithm, the n-th pair of vectors $\mathbf{v}_n$ and $\mathbf{w}_n$ is built, where $\mathbf{v}_n$ is the basis vector that advances $\mathcal{K}_{n-1}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ to $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$, and $\mathbf{w}_n$ advances $\mathcal{K}_{n-1}^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L})$ to $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A}^T,\mathbf{L})$. Clearly, the process of constructing basis vectors in

pairs has to be terminated as soon as the one of the two block Krylov subspaces is exhausted. This termination happens at pass $n = n_{\max}$, where

$$n_{\max} := \min\{\, n^{(r)},\, n^{(l)} \,\} = \min\{\, \operatorname{rank} \mathbf{K}^{\mathrm{dl}}(\mathbf{A}, \mathbf{R}),\ \operatorname{rank} \mathbf{K}^{\mathrm{dl}}(\mathbf{A}^T, \mathbf{L}) \,\}.$$

For the case that $n^{(r)} \neq n^{(l)}$, it would be possible to continue the construction of single basis vectors for the non-exhausted block Krylov subspace $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A}, \mathbf{R})$ if $n^{(r)} > n_{\max}$, respectively $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A}^T, \mathbf{L})$ if $n^{(l)} > n_{\max}$. However, this is not done in our algorithm, and we simply stop the process at $n = n_{\max}$.

3. Lanczos basis vectors

In this section, we state some basic properties of the vectors generated by our Lanczos-type algorithm.

3.1. **Biorthogonal bases.** The algorithm generates two sequences of *right* and *left Lanczos basis vectors*

$$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \quad \text{and} \quad \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n, \quad n = 1, 2, \ldots, n_{\max}, \tag{3.1}$$

for the n-th right and left block Krylov subspaces, i.e.,

$$\operatorname{span}\{\, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \,\} = \mathcal{K}_n^{\mathrm{dl}}(\mathbf{A}, \mathbf{R}) \tag{3.2}$$

and

$$\operatorname{span}\{\, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n \,\} = \mathcal{K}_n^{\mathrm{dl}}(\mathbf{A}^T, \mathbf{L}), \tag{3.3}$$

respectively. Furthermore, in the generic case, the vectors (3.1) are constructed to be *biorthogonal*, i.e.,

$$\mathbf{w}_i^T \mathbf{v}_n = \begin{cases} \delta_n & \text{if } i = n, \\ 0 & \text{if } i \neq n, \end{cases} \qquad \text{for all} \quad i, n = 1, 2, \ldots, n_{\max}. \tag{3.4}$$

For the statement of various properties of the vectors (3.1), it turns out to be convenient to introduce the notation

$$\mathbf{V}_n := \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} \quad \text{and} \quad \mathbf{W}_n := \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_n \end{bmatrix} \tag{3.5}$$

for the $N \times n$ matrices whose columns are just the first n right and left Lanczos vectors, respectively. For example, using (3.5) and setting

$$\boldsymbol{\Delta}_n := \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_n), \quad \text{where} \quad \delta_k = \mathbf{w}_k^T \mathbf{v}_k \quad \text{for all} \quad k,$$

the biorthogonality relations (3.4) can be stated in compact matrix form as follows:

$$\mathbf{W}_n^T \mathbf{V}_n = \boldsymbol{\Delta}_n \quad \text{for all} \quad n = 1, 2, \ldots, n_{\max}. \tag{3.6}$$

Enforcing the biorthogonality conditions (3.4) is only possible in the so-called *generic case* when

$$\delta_n \neq 0 \quad \text{for all} \quad n = 1, 2, \ldots, n_{\max} - 1.$$

Indeed, as the *generic* Lanczos-type Algorithm 4.1 below shows, constructing Lanczos vectors (3.1) that satisfy (3.2)–(3.4) involves division by δ_n. However, in general, it cannot be excluded that

$$\delta_n = \mathbf{w}_n^T \mathbf{v}_n = 0 \tag{3.7}$$

might occur, and thus any algorithm that tries to enforce (3.4) may break down due to division by zero. The event (3.7) will be referred to as an *exact breakdown*

of the Lanczos type-algorithm. In finite-precision arithmetic, one also needs to deal with so-called *near-breakdowns* due to division by nonzero numbers

$$\delta_n = \mathbf{w}_n^T \mathbf{v}_n \approx 0, \quad \delta_n \neq 0, \tag{3.8}$$

that are in some sense close to zero.

The key to devise an algorithm for the *general case*, where exact breakdowns and near-breakdowns are not excluded, is to relax the biorthogonality conditions (3.4) for the individual Lanczos vectors to a biorthogonality condition between suitably chosen *clusters* of Lanczos vectors. More precisely, these clusters are submatrices

$$\mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \ldots, \mathbf{V}^{(\gamma_{\max})} \quad \text{and} \quad \mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \ldots, \mathbf{W}^{(\gamma_{\max})} \tag{3.9}$$

that form a partition

$$\begin{aligned} \mathbf{V}_{n_{\max}} &= \begin{bmatrix} \mathbf{V}^{(1)} & \mathbf{V}^{(2)} & \cdots & \mathbf{V}^{(\gamma_{\max})} \end{bmatrix} \\ \text{and} \quad \mathbf{W}_{n_{\max}} &= \begin{bmatrix} \mathbf{W}^{(1)} & \mathbf{W}^{(2)} & \cdots & \mathbf{W}^{(\gamma_{\max})} \end{bmatrix} \end{aligned} \tag{3.10}$$

of the matrices $\mathbf{V}_{n_{\max}}$ and $\mathbf{W}_{n_{\max}}$, respectively, of all Lanczos basis vectors (3.1). By (3.10), each submatrix in (3.9) contains consecutive Lanczos vectors. Furthermore, the clusters (3.9) are chosen such that each pair $\mathbf{V}^{(\gamma)}$ and $\mathbf{W}^{(\gamma)}$ with identical index γ contains right and left Lanczos vectors with identical indices. Clusters $\mathbf{V}^{(\gamma)}$ and $\mathbf{W}^{(\gamma)}$ containing more than one vector each are built every time (3.7) or (3.8) occurs.

Instead of the vector-wise biorthogonality (3.4), in the general case only a cluster-wise biorthogonality is enforced:

$$\left(\mathbf{W}^{(k)}\right)^T \mathbf{V}^{(\gamma)} = \begin{cases} \mathbf{\Delta}^{(\gamma)} & \text{if } k = \gamma, \\ 0 & \text{if } k \neq \gamma, \end{cases} \qquad \text{for all} \quad k, \gamma = 1, 2, \ldots, \gamma_{\max}. \tag{3.11}$$

The *general* Lanczos-type Algorithm 5.2 below is a computational procedure for constructing Lanczos vectors (3.1) that are defined by (3.2), (3.3), and (3.11). Algorithm 5.2 now involves the solution of small linear systems with coefficient matrices $\mathbf{\Delta}^{(\gamma)}$, $\gamma < \gamma_{\max}$. Therefore, the clusters (3.9) need to be chosen such that

$$\mathbf{\Delta}^{(\gamma)} = \left(\mathbf{W}^{(\gamma)}\right)^T \mathbf{V}^{(\gamma)} \quad \text{is nonsingular for all} \quad \gamma = 1, 2, \ldots, \gamma_{\max} - 1. \tag{3.12}$$

Note that the cluster-biorthogonality conditions (3.11) can again be stated in the compact form (3.6), where $\mathbf{\Delta}_n$ is now defined as the $n \times n$ leading principal submatrix of the block-diagonal matrix

$$\mathbf{\Delta}_{n_{\max}} := \operatorname{diag}\left(\mathbf{\Delta}^{(1)}, \mathbf{\Delta}^{(2)}, \ldots, \mathbf{\Delta}^{(\gamma_{\max})}\right). \tag{3.13}$$

The purpose of forming clusters (3.9) is to avoid possible exact and near-breakdowns. More precisely, a pair of clusters $\mathbf{V}^{(\gamma)}$ and $\mathbf{W}^{(\gamma)}$ containing more than one vector is built every time an exact or near-breakdown would occur in the generic algorithm. In particular, in the absence of exact or near-breakdowns, each cluster consists of exactly one vector, the vector-wise and cluster-wise biorthogonality conditions (3.4) and (3.11) coincide, and the generic and general algorithms are identical.

3.2. **History indices.** Recall from (3.2) that the right Lanczos vectors build a basis for the subspaces $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$. Reflecting the block Krylov structure of $\mathcal{K}_n^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$, in our algorithm, each right Lanczos vector $\mathbf{v}_n$ is generated from a suitable $\mathbf{A}$-multiple of a previously constructed right Lanczos vector, except for the initial stages of the process when $\mathbf{v}_n$ is generated from one of the right starting vectors (2.1). Similarly, each left Lanczos vector $\mathbf{w}_n$ in (3.1) is generated either from a suitable $\mathbf{A}^T$-multiple of a previous vector or from one of the left starting vectors (2.2). Furthermore, the form of the actual recurrences for generating the Lanczos vectors (3.1) depends on the block structure of the deflated block Krylov matrices in (2.5) and (2.6). This structure could be recorded by keeping track of the sizes of the blocks $\mathbf{A}^{j-1}\mathbf{R}_j$ and $(\mathbf{A}^T)^{k-1}\mathbf{L}_k$ in (2.5) and (2.6), together with pointers for the positions of $\mathbf{v}_n$ and $\mathbf{w}_n$ relative to the current blocks in (2.5) and (2.6), respectively. However, for the exposition of our algorithm, we find it simpler to use a different encoding of the block structure of (2.5) and (2.6), based on *history indices* for the individual Lanczos vectors. Next, we describe these indices.

We use $n = 1, 2, \ldots$, as the counter for the main loop of the algorithm. During the n-th pass through the main loop, the n-th pair of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$ is being computed, together with their associated history indices, μ_n and ϕ_n.

For each n, the index μ_n records the index of the vector from which the n-th right Lanczos vector, $\mathbf{v}_n$, was generated, in the following sense. If $\mathbf{v}_n$ was generated from one of the right starting vectors, say $\mathbf{r}_j$, we set $\mu_n := j - m$; otherwise, $\mathbf{v}_n$ is generated from an $\mathbf{A}$-multiple, say $\mathbf{A} \cdot \mathbf{v}_\mu$, of a previously constructed vector $\mathbf{v}_\mu$, and we set $\mu_n = \mu$. Note that $\mu_n \leq 0$ if, and only if, $\mathbf{v}_n$ was generated from a starting vector. The sequence $\{\mu_n\}_{n\geq 1}$ is strictly increasing:

$$\mu_1 < \mu_2 < \cdots < \mu_n < \mu_{n+1} < \cdots < \mu_{n^{(r)}+1}. \tag{3.14}$$

Here, for each $1 \leq n \leq n^{(r)}$, a gap bigger than one, i.e., $\mu_{n+1} - \mu_n > 1$, occurs if, and only if, $\mu_{n+1} - \mu_n - 1$ consecutive deflation steps in the right Krylov blocks (2.3) were performed in between the construction of $\mathbf{v}_n$ and $\mathbf{v}_{n+1}$. Furthermore, we note that $\mu_n < n$ for all $n \leq n^{(r)}$, while $\mu_n = n$ for $n = n^{(r)} + 1$. The latter case means that the right Krylov blocks (2.3) are exhausted, and thus the algorithm terminates.

The following example illustrates the concept of the history indices $\{\mu_n\}_{n\geq 1}$.

Example 3.1. Suppose the indices (3.14) are given as

$$\{\, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6 \,\} = \{\, -1, 0, 1, 3, 4, 6 \,\}.$$

This means $\mathbf{v}_1$ and $\mathbf{v}_2$ came from the starting vectors (of which there were two), $\mathbf{v}_3$, $\mathbf{v}_4$, and $\mathbf{v}_5$ came from the $\mathbf{A}$-multiples $\mathbf{A}\mathbf{v}_1$, $\mathbf{A}\mathbf{v}_3$, and $\mathbf{A}\mathbf{v}_4$, respectively, and the remaining $\mathbf{A}$-multiples $\mathbf{A}\mathbf{v}_2$ and $\mathbf{A}\mathbf{v}_5$ were never used to generate any subsequent Lanczos vector because they were deflated out. Since $\mu_6 = 6$ in this example, the right Krylov sequence is exhausted at pass $n = 6$, thus terminating the algorithm.

The sequence of history indices,

$$\phi_1 < \phi_2 < \cdots < \phi_n < \phi_{n+1} < \cdots < \phi_{n^{(l)}+1}, \tag{3.15}$$

associated with the left Lanczos vectors $\mathbf{w}_n$, $n = 1, 2, \ldots$, is defined analogously. An index $\phi_n \leq 0$ means that $\mathbf{w}_n$ was generated from the left starting vector $\mathbf{l}_{\phi_n+p}$, while $\phi_n > 0$ means that $\mathbf{w}_n$ was generated from $\mathbf{A}^T\mathbf{w}_{\phi_n}$. In (3.15), a gap $\phi_{n+1} - \phi_n > 1$, occurs if, and only if, $\phi_{n+1} - \phi_n - 1$ consecutive deflation steps in the left Krylov

blocks (2.4) were performed in between the construction of $\mathbf{w}_n$ and $\mathbf{w}_{n+1}$. Again, we have $\phi_n < n$ for all $n \leq n^{(l)}$ and $\phi_n = n$ for $n = n^{(l)} + 1$. In the latter case, the left Krylov blocks (2.4) are exhausted, and the algorithm terminates.

We remark that, in the general algorithm with look-ahead, there is a third sequence of history indices, $\gamma(n)$, $n = 1, 2, \ldots, n_{\max}$, to record the sizes of the look-ahead clusters (3.9). Specifically, $\gamma(n)$ is defined to be the sequence number of the clusters $\mathbf{V}^{(\gamma(n))}$ and $\mathbf{W}^{(\gamma(n))}$ containing the individual vectors $\mathbf{v}_n$ and $\mathbf{w}_n$, respectively. However, in the generic case, we have $\gamma(n) = n$ for all n, and thus there is no need to record $\gamma(n)$ in the generic algorithm without look-ahead. The indices $\gamma(n)$ will be discussed further in §5 below.

We conclude this subsection with two remarks.

Remark 3.2. For the special case of the classical Lanczos algorithm with single starting vectors, the history indices (3.14) and (3.15) reduce to $\mu_n = \phi_n = n - 1$ for all n, except at termination when $\mu_n = n = n^{(r)} + 1$ or $\phi_n = n = n^{(l)} + 1$.

Remark 3.3. The history indices (3.14) and (3.15) can be used to determine the sizes of the blocks $\mathbf{R}_j$ and $\mathbf{L}_k$ in the deflated block Krylov matrices (2.5) and (2.6). We now briefly show how to do this for the right blocks $\mathbf{R}_j$; the case of the left blocks is analogous. For any $\mathbf{v}_n$, there is an index j_n depending on n such that $\mathbf{v}_n$ lies in the span of the columns of the matrix $\begin{bmatrix}\mathbf{R}_1 & \mathbf{A}\mathbf{R}_2 & \cdots & \mathbf{A}^{j_n-1}\mathbf{R}_{j_n}\end{bmatrix}$, but not in the span of the columns of $\begin{bmatrix}\mathbf{R}_1 & \mathbf{A}\mathbf{R}_2 & \cdots & \mathbf{A}^{j_n-2}\mathbf{R}_{j_n-1}\end{bmatrix}$. It is easy to see that the sequence $\{j_n\}_{n\geq 1}$ can be computed from the sequence (3.14) as follows:

$$j_n = \begin{cases} 1 & \text{if } \mu_n \leq 0, \\ j_{\mu_n} + 1 & \text{if } \mu_n > 0, \end{cases} \qquad \text{for all} \quad n = 1, 2, \ldots, n^{(r)}.$$

For a given j, the rank (which is equal to the number of columns) of the matrix $\begin{bmatrix}\mathbf{R}_1 & \mathbf{A}\mathbf{R}_2 & \cdots & \mathbf{A}^{j-1}\mathbf{R}_j\end{bmatrix}$ is equal to the largest value of n such that $j_n = j$, or equivalently, the value of n such that $j_n = j$ but $j_{n+1} > j$. (For $n = n^{(r)}$, we set $j_{n_r+1} = \infty$.) If we denote this value of n as n_j, making explicit its dependence on j, then the number of columns of $\mathbf{R}_j$ is just $n_j - n_{j-1}$.

3.3. **Structure of the algorithm.** After having introduced the history indices, we can now show the basic structure of our Lanczos-type algorithm. The structures of the generic Algorithm 4.1 and the general Algorithm 5.2 are identical, and in Figure 1, we show a flow chart that is valid for both versions of the algorithm. In Figure 1, the step numbers 0)–5) are the same as the ones used in Algorithms 4.1 and 5.2. Note that n is the counter for the main loop. Within each n-th pass through the main loop, it is possible that there are multiple passes through the sub-loop 1)–1f); in fact, this happens if, and only if, $\mathbf{v}$ vectors are deflated while building $\mathbf{v}_n$. The integer μ is used as a counter for these multiple passes through the sub-loop 1)–1f). Similarly, multiple passes through the sub-loop 2)–2f) occur if, and only if, $\mathbf{w}$ vectors are deflated while building $\mathbf{w}_n$, and the counter ϕ records these multiple passes.

3.4. **Recurrence relations.** Next, we state the recurrence relations that are employed in our Lanczos-type algorithm to generate the Lanczos vectors (3.1). Using the matrix notation $\mathbf{V}_n$ and $\mathbf{W}_n$ introduced in (3.5), we will write these recurrences in compact matrix form for *all* Lanczos vectors computed during the first n passes through the main loop of the algorithm.

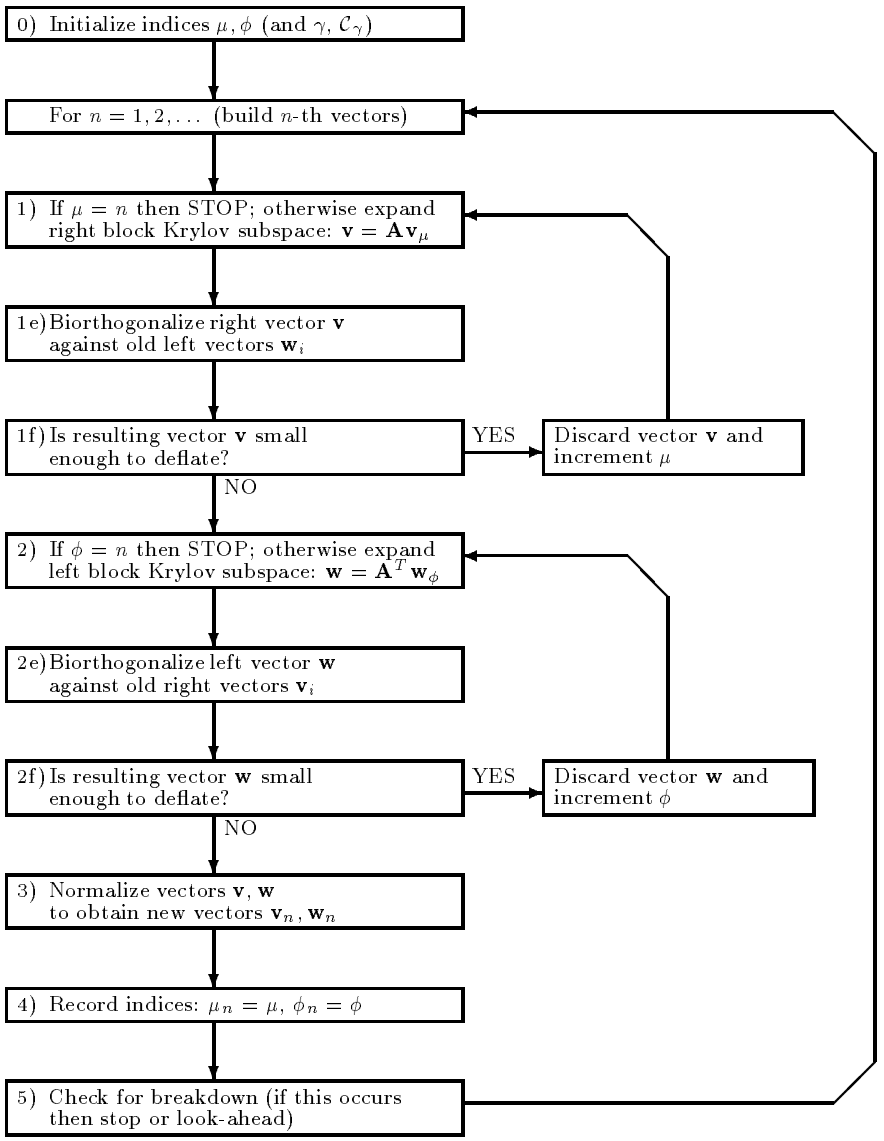


FIGURE 1. Flow chart of the Lanczos-type algorithm.

To motivate this compact matrix form of the recurrences, we first briefly review the case of the Lanczos process for single starting vectors $\mathbf{r}$ and $\mathbf{l}$. In this case, for all $n = 1, 2, \ldots, n_{\max}$, the recurrences for the first n right and left Lanczos vectors can be summarized as

$$\mathbf{V}_n \mathbf{T}_{n-1}^{(e)} = \begin{cases} \mathbf{r} & \text{if } n = 1, \\ \mathbf{A}\mathbf{V}_{n-1} & \text{if } n > 1, \end{cases} \tag{3.16}$$

and

$$\mathbf{W}_n \widetilde{\mathbf{T}}_{n-1}^{(e)} = \begin{cases} \mathbf{l} & \text{if } n = 1, \\ \mathbf{A}^T \mathbf{W}_{n-1} & \text{if } n > 1, \end{cases} \tag{3.17}$$

respectively. Here, for $n = 1$, $\mathbf{T}_0^{(e)}$ and $\widetilde{\mathbf{T}}_0^{(e)}$ are scalars that record the normalization of the starting vectors $\mathbf{r}$ and $\mathbf{l}$, respectively. For $n > 1$, $\mathbf{T}_{n-1}^{(e)}$ and $\widetilde{\mathbf{T}}_{n-1}^{(e)}$ are $n \times (n-1)$ matrices that contain the recurrence coefficients; both matrices are tridiagonal if no look-ahead steps occurs, and they are simultaneously upper Hessenberg and block tridiagonal if look-ahead steps are performed; see, e.g., [17].

We now present the corresponding extensions of (3.16) and (3.17) for our Lanczos-type algorithm for multiple starting vectors. Recall from Figure 1 that for each n, there can be multiple values of μ, respectively ϕ. Thus, we now need two indices n and μ, respectively n and ϕ, to state the recurrences for the right, respectively left, Lanczos vectors.

For all $n = 1, 2, \ldots, n_{\max}$ and $\mu = \mu_n, \mu_n + 1, \ldots, \mu_{n+1} - 1$, the right Lanczos vectors satisfy the recurrences

$$\mathbf{V}_n \mathbf{T}_\mu^{(e)} + \mathbf{V}_\mu^{\mathrm{dl}} = \begin{cases} [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \cdots \quad \mathbf{r}_{\mu+m}] & \text{if } \mu \le 0, \\ \mathbf{A}\mathbf{V}_\mu & \text{if } \mu > 0. \end{cases} \tag{3.18}$$

For all $n = 1, 2, \ldots, n_{\max}$ and $\phi = \phi_n, \phi_n + 1, \ldots, \phi_{n+1} - 1$, the left Lanczos vectors satisfy the recurrences

$$\mathbf{W}_n \widetilde{\mathbf{T}}_\phi^{(e)} + \mathbf{W}_\phi^{\mathrm{dl}} = \begin{cases} [\mathbf{l}_1 \quad \mathbf{l}_2 \quad \cdots \quad \mathbf{l}_{\phi+p}] & \text{if } \phi \le 0, \\ \mathbf{A}^T \mathbf{W}_\phi & \text{if } \phi > 0. \end{cases} \tag{3.19}$$

The matrices $\mathbf{T}_\mu^{(e)}$ and $\widetilde{\mathbf{T}}_\phi^{(e)}$ in (3.18) and (3.19) are given by

$$\mathbf{T}_\mu^{(e)} = \begin{cases} [t_{i,j}]_{1 \le i \le n,\, 1-m \le j \le \mu} & \text{if } \mu \le 0, \\ [t_{i,j}]_{1 \le i \le n,\, 1 \le j \le \mu} & \text{if } \mu > 0, \end{cases} \tag{3.20}$$

and

$$\widetilde{\mathbf{T}}_\phi^{(e)} = \begin{cases} \left[\tilde{t}_{i,j}\right]_{1 \le i \le n,\, 1-p \le j \le \phi} & \text{if } \phi \le 0, \\ \left[\tilde{t}_{i,j}\right]_{1 \le i \le n,\, 1 \le j \le \phi} & \text{if } \phi > 0. \end{cases} \tag{3.21}$$

The nonzero entries $t_{i,j}$ and $\tilde{t}_{i,j}$ in (3.20) and (3.21) are defined in equations (4.1), (4.3), and (4.4) below in the case of the generic Algorithm 4.1 without look-ahead, and in equations (5.5), (5.7), and (5.8) below in the case of the general Algorithm 5.2 with look-ahead. Furthermore, the elements $t_{i,j}$ and $\tilde{t}_{i,j}$ that are not explicitly defined in Algorithms 4.1 and 5.2 are set to be zero.

We remark that the nature of $\mathbf{T}_\mu^{(e)}$ is different for $\mu \le 0$ and $\mu > 0$. For $\mu \le 0$, the columns of $\mathbf{T}_\mu^{(e)}$ contain the recurrence coefficients used to process the right starting vectors; for example, if $\mathbf{R}$ has full column rank, then $\mathbf{V}_m \mathbf{T}_0^{(e)} = \mathbf{R}$. For $\mu > 0$, the columns of $\mathbf{T}_\mu^{(e)}$ contain the recurrences used to advance the right block Krylov subspaces by multiplications with $\mathbf{A}$, after the right starting vectors have been processed. Similarly, the columns of $\widetilde{\mathbf{T}}_\phi^{(e)}$ contain the recurrence coefficients used to process the left starting vectors $\mathbf{L}$ if $\phi \le 0$, and the ones used to advance the left block Krylov subspaces by multiplications with $\mathbf{A}^T$ if $\phi > 0$.

In (3.18), the term $\mathbf{V}_\mu^{\mathrm{dl}}$ is an $N \times \mu$ matrix that is built up as follows:

$$\mathbf{V}_\mu^{\mathrm{dl}} = \begin{cases} \begin{bmatrix} \mathbf{V}_{\mu-1}^{\mathrm{dl}} & \mathbf{v} \end{bmatrix} & \text{if } \mathbf{v} \neq 0 \text{ and } \mathbf{v} \text{ is deflated,} \\ \begin{bmatrix} \mathbf{V}_{\mu-1}^{\mathrm{dl}} & 0 \end{bmatrix} & \text{otherwise,} \end{cases}$$

where $\mathbf{V}_0$ is the empty matrix. In other words, $\mathbf{V}_\mu^{\mathrm{dl}}$ contains the vectors $\mathbf{v}$ that are discarded due to inexact deflation, together with lots of zero vectors. In particular, if no deflation occurs or only exact deflation is performed, then $\mathbf{V}_\mu^{\mathrm{dl}}$ is the $N \times \mu$ zero matrix. Similarly, in (3.19), the term $\mathbf{W}_\mu^{\mathrm{dl}}$ is an $N \times \phi$ matrix that contains the vectors $\mathbf{w}$ that are discarded due to inexact deflation, together with lots of zero vectors. It is defined by

$$\mathbf{W}_\mu^{\mathrm{dl}} = \begin{cases} \begin{bmatrix} \mathbf{W}_{\phi-1}^{\mathrm{dl}} & \mathbf{w} \end{bmatrix} & \text{if } \mathbf{w} \neq 0 \text{ and } \mathbf{w} \text{ is deflated,} \\ \begin{bmatrix} \mathbf{W}_{\phi-1}^{\mathrm{dl}} & 0 \end{bmatrix} & \text{otherwise,} \end{cases}$$

where $\mathbf{W}_0$ is the empty matrix. If no deflation occurs or only exact deflation is performed, then $\mathbf{W}_\phi^{\mathrm{dl}}$ is the $N \times \phi$ zero matrix.

In the case of inexact deflation, the matrices $\mathbf{V}_\mu^{\mathrm{dl}}$ and $\mathbf{W}_\mu^{\mathrm{dl}}$ are no longer zero, but they are still small in norm. First, we note that, at any stage of our algorithm, the number of deflations of $\mathbf{v}$ vectors during the first n passes is given by $m-n+\mu$; similarly, the number of deflations of $\mathbf{w}$ vectors during the first n passes is given by $p-n+\phi$. Now, suppose we deflate whenever $\|\mathbf{v}\| \leq \texttt{dtol}$, respectively $\|\mathbf{w}\| \leq \texttt{dtol}$, where `dtol` is some small deflation tolerance. Thus $\mathbf{V}_\mu^{\mathrm{dl}}$ has at most $m-n+\mu$ nonzero columns and each of these columns has Euclidean norm at most `dtol`. Similarly, $\mathbf{W}_\phi^{\mathrm{dl}}$ has at most $p-n+\phi$ nonzero columns and each of these columns has Euclidean norm at most `dtol`. It follows that

$$\left\|\mathbf{V}_\mu^{\mathrm{dl}}\right\| \leq \texttt{dtol}\sqrt{m-n+\mu} \quad \text{and} \quad \left\|\mathbf{W}_\phi^{\mathrm{dl}}\right\| \leq \texttt{dtol}\sqrt{p-n+\phi}.$$

We conclude this section with some remarks on the zero structure of the matrices $\mathbf{T}_\mu^{(e)}$ and $\widetilde{\mathbf{T}}_\phi^{(e)}$ in (3.20) and (3.21), respectively. In the simplest case of no deflation and no look-ahead, the matrix $\mathbf{T}_\mu^{(e)}$ is banded with a lower bandwidth $m+1$ and an upper bandwidth $p+1$, and $\widetilde{\mathbf{T}}_\phi^{(e)}$ is banded with a lower bandwidth $p+1$ and an upper bandwidth $m+1$. Look-ahead steps result in additional "bulges" in $\mathbf{T}_\mu^{(e)}$ and $\widetilde{\mathbf{T}}_\phi^{(e)}$ above their upper bands. Finally, each deflation of a $\mathbf{v}$ vector reduces both the lower bandwidth of $\mathbf{T}_\mu^{(e)}$ and the upper bandwidth of $\widetilde{\mathbf{T}}_\phi^{(e)}$ by one. Similarly, each deflation of a $\mathbf{w}$ vector reduces both the lower bandwidth of $\widetilde{\mathbf{T}}_\phi^{(e)}$ and the upper bandwidth of $\mathbf{T}_\mu^{(e)}$ by one. Furthermore, each inexact deflation of a $\mathbf{v}$ vector requires that all successive left Lanczos vectors need to be explicitly biorthogonalized against a certain vector $\mathbf{v}_i$, respectively all vectors of the cluster containing $\mathbf{v}_i$ in the look-ahead case. Similarly, inexact deflation of a $\mathbf{w}$ vector requires explicit biorthogonalization of all successive right Lanczos vectors against a certain vector $\mathbf{w}_i$, respectively all vectors of the cluster containing $\mathbf{w}_i$ in the look-ahead case. The indices of the vectors, respectively of the look-ahead clusters, against which we need to explicit biorthogonalize due to inexact deflation of $\mathbf{v}$ and $\mathbf{w}$ vectors are stored in the index sets $\mathcal{I}_\mathbf{w}$ and $\mathcal{I}_\mathbf{v}$ in Algorithm 4.1, respectively Algorithm 5.2, below. These additional biorthogonalizations are reflected in nonzeros

in rows of $\mathbf{T}_\mu^{(e)}$ of $\widetilde{\mathbf{T}}_\phi^{(e)}$ whose row indices correspond to $\mathcal{I}_\mathbf{v}$ and $\mathcal{I}_\mathbf{w}$, respectively. However, these nonzeros only appear to the right of the bands. In particular, even in the most general case of deflation and look-ahead, $\mathbf{T}_\mu^{(e)}$ and $\widetilde{\mathbf{T}}_\phi^{(e)}$ always have lower bandwidth $m+1$ and $p+1$, respectively. Finally, we refer the reader to the Appendix where the zero structures of $\mathbf{T}_\mu^{(e)}$ and $\widetilde{\mathbf{T}}_\phi^{(e)}$ are illustrated for a specific example.

4. The generic Lanczos-type algorithm

In this section, we present the generic Lanczos-type algorithm in a form that will lead naturally to the look-ahead algorithm in §5 below.

Algorithm 4.1. (Lanczos-type method with deflation, but without look-ahead.)
INPUT: Matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$;
m right starting vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_m \in \mathbb{C}^N$;
p left starting vectors $\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_p \in \mathbb{C}^N$.

0) *Set* $\mu = -m$ *and* $\phi = -p$.
(μ is the index of the currently expanded vector in the $\mathbf{v}$ sequence; ϕ is the index of the currently expanded vector in the $\mathbf{w}$ sequence. A non-positive μ or ϕ means we are still fetching starting vectors.)
Set $\mathcal{I}_\mathbf{v} = \emptyset$ *and* $\mathcal{I}_\mathbf{w} = \emptyset$.
($\mathcal{I}_\mathbf{v}$ and $\mathcal{I}_\mathbf{w}$ record indices of vectors that must be preserved due to inexact deflation. If $i \in \mathcal{I}_\mathbf{v}$, respectively $i \in \mathcal{I}_\mathbf{w}$, then the currently constructed Lanczos vector $\mathbf{v}$, respectively $\mathbf{w}$, needs to be biorthogonalized against $\mathbf{w}_i$, respectively $\mathbf{v}_i$.)

For $n = 1, 2, \ldots,$ *do* (Build n-th pair of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$.) :

1) (Build the unnormalized right Lanczos vector $\mathbf{v}$.)
1a) *Set* $\mu = \mu + 1$.
1b) (Check if the right Krylov blocks are exhausted.)
If $\mu = n$, *then stop.* (There are no more right Krylov vectors.)
1c) (Advance the right block Krylov subspace.)
Set

$$\mathbf{v} = \begin{cases} \mathbf{r}_{\mu+m} & \text{if } \mu \le 0, \\ \mathbf{A}\,\mathbf{v}_\mu & \text{if } \mu > 0. \end{cases}$$

1d) (Determine against which vectors $\mathbf{v}$ needs to be biorthogonalized.)
Set

$$i_\mathbf{v} = \begin{cases} 1 & \text{if } \mu \le 0, \\ \max\{\, 1, \phi_\mu \,\} & \text{if } \mu > 0, \end{cases}$$

and define the temporary index set

$$\mathcal{I} = \{\, i_\mathbf{v}, i_\mathbf{v}+1, \ldots, n-1 \,\} \cup \bigcup_{\substack{i\in\mathcal{I}_\mathbf{v} \\ i<i_\mathbf{v}}} \{i\}.$$

1e) (Biorthogonalize $\mathbf{v}$ against these vectors.)
Compute the coefficients

$$t_{i,\mu} = \frac{\mathbf{w}_i^T \mathbf{v}}{\delta_i} \quad \text{for all} \quad i \in \mathcal{I}. \tag{4.1}$$

Set

$$\mathbf{v} = \mathbf{v} - \sum_{i \in \mathcal{I}} \mathbf{v}_i t_{i,\mu}. \tag{4.2}$$

1f) *Decide if* $\mathbf{v}$ *should be deflated, e.g., by checking if* $\|\mathbf{v}\| \leq$ `dtol`.
If yes, do the following:
(i) *If* $\mu > 0$ *and the deflated vector* $\mathbf{v}$ *is nonzero, then set* $\mathcal{I}_{\mathbf{w}} = \mathcal{I}_{\mathbf{w}} \cup \{\mu\}$ *and save the vectors* $\mathbf{v}_\mu$ *and* $\mathbf{w}_\mu$.
(The vector $\mathbf{v}$ is the μ-th column of the matrix $\mathbf{V}_\mu^{\mathrm{dl}}$ in (3.18).)
(ii) *Repeat all of Step* 1.

2) (Build the unnormalized right Lanczos vector $\mathbf{w}$.)
2a) *Set* $\phi = \phi + 1$.
2b) (Check if the left Krylov blocks are exhausted.)
If $\phi = n$, *then stop.* (There are no more left Krylov vectors.)
2c) (Advance the left block Krylov subspace.)
Set

$$\mathbf{w} = \begin{cases} \mathbf{l}_{\phi+p} & \text{if } \phi \leq 0, \\ \mathbf{A}^T \mathbf{w}_\phi & \text{if } \phi > 0. \end{cases}$$

2d) (Determine against which vectors $\mathbf{w}$ needs to be biorthogonalized.)
Set

$$i_{\mathbf{w}} = \begin{cases} 1 & \text{if } \phi \leq 0, \\ \max\{\, 1, \mu_\phi \,\} & \text{if } \phi > 0, \end{cases}$$

and define the temporary index set

$$\mathcal{I} = \{\, i_{\mathbf{w}}, i_{\mathbf{w}} + 1, \ldots, n-1 \,\} \cup \bigcup_{\substack{i \in \mathcal{I}_{\mathbf{w}} \\ i < i_{\mathbf{w}}}} \{i\}.$$

2e) (Biorthogonalize $\mathbf{w}$ against these vectors.)
Compute the coefficients

$$\tilde{t}_{i,\phi} = \frac{\mathbf{v}_i^T \mathbf{w}}{\delta_i} \quad \text{for all} \quad i \in \mathcal{I}. \tag{4.3}$$

Set

$$\mathbf{w} = \mathbf{w} - \sum_{i \in \mathcal{I}} \mathbf{w}_i \tilde{t}_{i,\phi}.$$

2f) *Decide if* $\mathbf{w}$ *should be deflated, e.g., by checking if* $\|\mathbf{w}\| \leq$ `dtol`.
If yes, do the following:
(i) *If* $\phi > 0$ *and the deflated vector* $\mathbf{w}$ *is nonzero, then set* $\mathcal{I}_{\mathbf{v}} = \mathcal{I}_{\mathbf{v}} \cup \{\phi\}$ *and save the vectors* $\mathbf{v}_\phi$ *and* $\mathbf{w}_\phi$.
(The vector $\mathbf{w}$ is the ϕ-th column of the matrix $\mathbf{W}_\phi^{\mathrm{dl}}$ in (3.19).)
(ii) *Repeat all of Step* 2.

3) (Normalize $\mathbf{v}$ and $\mathbf{w}$ to obtain the n-th pair of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$.)
Set

$$\mathbf{v}_n = \frac{\mathbf{v}}{t_{n,\mu}} \quad \text{and} \quad \mathbf{w}_n = \frac{\mathbf{w}}{\tilde{t}_{n,\phi}}, \tag{4.4}$$

where $t_{n,\mu}$ *and* $\tilde{t}_{n,\phi}$ *are suitable scaling factors, e.g.,*

$$t_{n,\mu} = \|\mathbf{v}\| \quad \text{and} \quad \tilde{t}_{n,\phi} = \|\mathbf{w}\|.$$

4) (Update the history indices.)
Set $\mu_n = \mu$ *and* $\phi_n = \phi$.
(This records that $\mathbf{v}_n$ was obtained from $\mathbf{r}_{\mu_n+m}$ if $\mu_n \leq 0$ or from $\mathbf{A}\mathbf{v}_{\mu_n}$ if $\mu_n > 0$, and that $\mathbf{w}_n$ was obtained from $\mathbf{l}_{\phi_n+m}$ if $\phi_n \leq 0$ or from $\mathbf{A}^T\mathbf{w}_{\phi_n}$ if $\phi_n > 0$.)
5) (Compute δ_n and check for breakdown.)
Set

$$\delta_n = \mathbf{w}_n^T \mathbf{v}_n .$$

If $\delta_n = 0$, *then stop.*

5. The general Lanczos-type algorithm with look-ahead

In this section, we present a statement of the general Lanczos-type algorithm with deflation and look-ahead.

5.1. **Keeping track of the look-ahead clusters.** Recall from (3.9)–(3.10) that, in the general algorithm, the Lanczos vectors are grouped into look-ahead clusters $\mathbf{V}^{(\gamma)}$ and $\mathbf{W}^{(\gamma)}$, $\gamma = 1, 2, \ldots, \gamma_{\max}$. To keep track of the sizes of these clusters, we use a sequence of *cluster indices*

$$\{\gamma(n)\}_{1 \leq n \leq n_{\max}}. \tag{5.1}$$

Here, for each n, we define $\gamma(n) = \gamma$ as the (unique) index of the clusters $\mathbf{V}^{(\gamma)}$ and $\mathbf{W}^{(\gamma)}$ that contain the n-th pair of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$. Note that the sequence of cluster indices is non-strictly increasing:

$$\gamma(1) \leq \gamma(2) \leq \cdots \gamma(n) \leq \gamma(n+1) \leq \cdots \leq \gamma(n_{\max}).$$

For the sequence (5.1), one can easily deduce any other required information on the clusters. For example, for each $\gamma = 1, 2, \ldots, \gamma_{\max}$, the set

$$\mathcal{C}_\gamma := \{\, i \mid \gamma(i) = \gamma \,\} \tag{5.2}$$

consists of all the indices of the Lanczos vectors $\mathbf{v}_i$ and $\mathbf{w}_i$ that are contained in the γ-th pair of clusters $\mathbf{V}^{(\gamma)}$ and $\mathbf{W}^{(\gamma)}$. Furthermore, we define the auxiliary sequence

$$\xi(n) := \min_{i \in \mathcal{C}_{\gamma(n)}} i \quad \text{for all} \quad n = 1, 2, \ldots, n_{\max}, \tag{5.3}$$

which records the indices of the first vector in each cluster. Clearly, the sequence $\{\xi(n)\}_{1 \leq n \leq n_{\max}}$ is non-strictly increasing, and is component-wise less than or equal to $\{\, 1, 2, \ldots, n_{\max} \,\}$. Note that, in view of (5.3), the $\gamma(n)$-th pair of clusters is given by

$$\mathbf{V}^{(\gamma(n))} = \begin{bmatrix} \mathbf{v}_{\xi(n)} & \mathbf{v}_{\xi(n)+1} & \cdots & \mathbf{v}_{\xi(n+1)-1} \end{bmatrix}$$

$$\text{and} \quad \mathbf{W}^{(\gamma(n))} = \begin{bmatrix} \mathbf{w}_{\xi(n)} & \mathbf{w}_{\xi(n)+1} & \cdots & \mathbf{w}_{\xi(n+1)-1} \end{bmatrix},$$

where, for $n = n_{\max}$, we set $\xi(n_{\max}+1) = n_{\max}+1$.

The sequences (5.2) and (5.3) clearly encode the same information as the cluster indices (5.1), and thus they are redundant. However, it turned out to be convenient to use all three quantities (5.1)–(5.3) in the statement of the general algorithm with look-ahead and in the proofs of its properties.

Example 5.1. Suppose that the look-ahead clusters start with

$$\mathbf{V}^{(1)} = \begin{bmatrix}\mathbf{v}_1\end{bmatrix}, \quad \mathbf{V}^{(2)} = \begin{bmatrix}\mathbf{v}_2 & \mathbf{v}_3\end{bmatrix}, \quad \mathbf{V}^{(3)} = \begin{bmatrix}\mathbf{v}_4\end{bmatrix}, \quad \ldots \quad ,$$

$$\mathbf{W}^{(1)} = \begin{bmatrix}\mathbf{w}_1\end{bmatrix}, \quad \mathbf{W}^{(2)} = \begin{bmatrix}\mathbf{w}_2 & \mathbf{w}_3\end{bmatrix}, \quad \mathbf{W}^{(3)} = \begin{bmatrix}\mathbf{w}_4\end{bmatrix}, \quad \ldots \quad .$$

The sequence of cluster indices then starts with

$$\{\gamma(n)\}_{n\geq 1} = \{\,1, 2, 2, 3, \ldots\}.$$

Moreover, we have $\mathcal{C}_1 = \{1\}$, $\mathcal{C}_2 = \{2, 3\}$, $\mathcal{C}_3 = \{4\}$, etc., and the sequence $\{\xi(n)\}_{n\geq 1}$ would start off as $\{\,1, 2, 2, 4, 5, \ldots\}$. The situation in this example would arise when $\delta_2 = \mathbf{w}_2^T \mathbf{v}_2 \approx 0$ triggers a look-ahead step, resulting in a pair of clusters starting with $\mathbf{v}_2$ and $\mathbf{w}_2$, and when $\mathbf{\Delta}^{(2)} = \begin{bmatrix}\mathbf{w}_2 & \mathbf{w}_3\end{bmatrix}^T \begin{bmatrix}\mathbf{v}_2 & \mathbf{v}_3\end{bmatrix}$ is well conditioned, allowing to terminate the clusters after adding only $\mathbf{v}_3$ and $\mathbf{w}_3$.

5.2. **Statement of the algorithm.** We now present a formal statement of the general algorithm with deflation and look-ahead. In particular, this statement shows exactly against which vectors one needs to biorthogonalize, and it gives explicit formulas for the recurrence coefficients.

Algorithm 5.2. (Lanczos-type method with deflation and look-ahead.)
INPUT: Matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$;
m right starting vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_m \in \mathbb{C}^N$;
p left starting vectors $\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_p \in \mathbb{C}^N$.

0) *Set* $\mu = -m$ *and* $\phi = -p$.
(μ is the index of the currently expanded vector in the $\mathbf{v}$ sequence; ϕ is the index of the currently expanded vector in the $\mathbf{w}$ sequence. A non-positive μ or ϕ means we are still fetching starting vectors.)
Set $\mathcal{I}_\mathbf{v} = \emptyset$ *and* $\mathcal{I}_\mathbf{w} = \emptyset$.
($\mathcal{I}_\mathbf{v}$ and $\mathcal{I}_\mathbf{w}$ record indices of clusters that must be preserved due to inexact deflation. If $k \in \mathcal{I}_\mathbf{v}$, respectively $k \in \mathcal{I}_\mathbf{w}$, then the currently constructed Lanczos vector $\mathbf{v}$, respectively $\mathbf{w}$, needs to be biorthogonalized against all vectors $\mathbf{w}_i$, respectively $\mathbf{v}_i$, with $i \in \mathcal{C}_k$.)
Set $\gamma(1) = 1$, $\gamma = 1$, $\mathcal{C}_1 = \emptyset$, *and* $\mathbf{V}^{(\gamma)} = \mathbf{W}^{(\gamma)} = \emptyset$.
($\gamma(n)$ is the index of the cluster containing the n-th Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$, γ is the number of the currently constructed look-ahead cluster, and $\mathcal{C}_\gamma$ is the set of indices of the already constructed Lanczos vectors in the γ-th clusters $\mathbf{V}^{(\gamma)}$ and $\mathbf{W}^{(\gamma)}$.)

For $n = 1, 2, \ldots,$ *do* (Build n-th pair of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$.) :

1) (Build the unnormalized right Lanczos vector $\mathbf{v}$.)
1a) *Set* $\mu = \mu + 1$.
1b) (Check if the right Krylov blocks are exhausted.)
If $\mu = n$, *then stop.* (There are no more right Krylov vectors.)
1c) (Advance the right block Krylov subspace.)
Set

$$\mathbf{v} = \begin{cases} \mathbf{r}_{\mu+m} & \text{if } \mu \leq 0, \\ \mathbf{A}\,\mathbf{v}_\mu & \text{if } \mu > 0. \end{cases}$$

If the current γ*-th cluster contains at least one vector, i.e.,* $\mathcal{C}_\gamma \neq \emptyset$, *we may add arbitrary combinations of the vectors in the* γ*-th cluster to* $\mathbf{v}$:

Set

$$\mathbf{v} = \mathbf{v} + \mathbf{V}^{(\gamma)} \left[t_{i,\mu}\right]_{i \in \mathcal{C}_\gamma}, \quad \textit{with arbitrary} \quad t_{i,\mu} \in \mathbb{C}.$$

1d) (Determine against which clusters $\mathbf{v}$ needs to be biorthogonalized.)
Set

$$\gamma_{\mathbf{v}} = \begin{cases} 1 & \textit{if } \mu \leq 0, \\ \max\{\, 1, \gamma(\phi_{\xi(\mu)}) \,\} & \textit{if } \mu > 0, \end{cases} \tag{5.4}$$

where, in the latter case, $\xi(\mu) = \min\limits_{i \in \mathcal{C}_{\gamma(\mu)}} i$, *and define the temporary index set*

$$\mathcal{I} = \{\, \gamma_{\mathbf{v}}, \gamma_{\mathbf{v}} + 1, \ldots, \gamma \,\} \cup \bigcup_{\substack{k \in \mathcal{I}_{\mathbf{v}} \\ k < \gamma_{\mathbf{v}}}} \{k\}.$$

1e) (Biorthogonalize $\mathbf{v}$ against these clusters.)
Compute the coefficient vectors

$$\left[t_{i,\mu}\right]_{i \in \mathcal{C}_k} = \left(\mathbf{\Delta}^{(k)}\right)^{-1} \left(\mathbf{W}^{(k)}\right)^T \mathbf{v} \quad \textit{for all} \quad k \in \mathcal{I}. \tag{5.5}$$

Set

$$\mathbf{v} = \mathbf{v} - \sum_{k \in \mathcal{I}} \mathbf{V}^{(k)} \left[t_{i,\mu}\right]_{i \in \mathcal{C}_k}.$$

1f) *Decide if* $\mathbf{v}$ *should be deflated; for example, this can be done by first orthogonalizing (in the ordinary one-sided sense)* $\mathbf{v}$ *against the vectors* $\mathbf{v}_i$, $i \in \mathcal{C}_\gamma$, *in the current* γ*-th cluster and then checking if the resulting vector* $\mathbf{v}$ *satisfies* $\|\mathbf{v}\| \leq$ `dtol`.
If yes, do the following:
(i) *If* $\mu > 0$ *and the deflated vector* $\mathbf{v}$ *is not an exact linear combination of the vectors* $\mathbf{v}_i$, $i \in \mathcal{C}_\gamma$, *then set* $\mathcal{I}_{\mathbf{w}} = \mathcal{I}_{\mathbf{w}} \cup \{\, \gamma(\mu) \,\}$, *and save the vectors* $\mathbf{v}_i$ *and* $\mathbf{w}_i$, $i \in \mathcal{C}_\gamma(\mu)$.
(The vector $\mathbf{v}$ is the μ-th column of the matrix $\mathbf{V}_\mu^{\mathrm{dl}}$ in (3.18).)
(ii) *Repeat all of Step* 1.

2) (Build the unnormalized right Lanczos vector $\mathbf{w}$.)
2a) *Set* $\phi = \phi + 1$.
2b) (Check if the left Krylov blocks are exhausted.)
If $\phi = n$, *then stop.* (There are no more left Krylov vectors.)
2c) (Advance the left block Krylov subspace.)
Set

$$\mathbf{w} = \begin{cases} \mathbf{l}_{\phi+p} & \textit{if } \phi \leq 0, \\ \mathbf{A}^T \mathbf{w}_\phi & \textit{if } \phi > 0. \end{cases}$$

If the current γ*-th cluster contains at least one vector, i.e.,* $\mathcal{C}_\gamma \neq \emptyset$, *we may add arbitrary combinations of the vectors in the* γ*-th cluster to* $\mathbf{v}$:
Set

$$\mathbf{w} = \mathbf{w} + \mathbf{W}^{(\gamma)} \left[\tilde{t}_{i,\phi}\right]_{i \in \mathcal{C}_\gamma}, \quad \textit{with arbitrary} \quad \tilde{t}_{i,\phi} \in \mathbb{C}.$$

2d) (Determine against which clusters $\mathbf{w}$ needs to be biorthogonalized.)
Set

$$\gamma_{\mathbf{w}} = \begin{cases} 1 & \textit{if } \phi \leq 0, \\ \max\{\, 1, \gamma(\mu_{\xi(\phi)}) \,\} & \textit{if } \phi > 0, \end{cases} \tag{5.6}$$

where, in the latter case, $\xi(\phi) = \min_{i \in \mathcal{C}_{\gamma(\phi)}} i$, *and define the temporary index set*

$$\mathcal{I} = \{ \gamma_{\mathbf{w}}, \gamma_{\mathbf{w}} + 1, \ldots, \gamma \} \cup \bigcup_{\substack{k \in \mathcal{I}_{\mathbf{w}} \\ k < \gamma_{\mathbf{w}}}} \{k\}.$$

2e) (Biorthogonalize $\mathbf{w}$ against these clusters.)
Compute the coefficient vectors

$$\left[\tilde{t}_{i,\phi}\right]_{i \in \mathcal{C}_k} = \left(\mathbf{\Delta}^{(k)}\right)^{-T} \left(\mathbf{V}^{(k)}\right)^T \mathbf{w} \quad \textit{for all} \quad k \in \mathcal{I}. \tag{5.7}$$

Set

$$\mathbf{w} = \mathbf{w} - \sum_{k \in \mathcal{I}} \mathbf{W}^{(k)} \left[\tilde{t}_{i,\phi}\right]_{i \in \mathcal{C}_k}.$$

2f) *Decide if* $\mathbf{w}$ *should be deflated; for example, this can be done by first orthogonalizing (in the ordinary one-sided sense)* $\mathbf{w}$ *against the vectors* $\mathbf{w}_i$, $i \in \mathcal{C}_\gamma$, *in the current* γ*-th cluster and then checking if the resulting vector* $\mathbf{w}$ *satisfies* $\|\mathbf{w}\| \leq$ `dtol`.
If yes, do the following:
(i) *If* $\phi > 0$ *and the deflated vector* $\mathbf{w}$ *is not an exact linear combination of* $\mathbf{w}_i$, $i \in \mathcal{C}_\gamma$, *then set* $\mathcal{I}_{\mathbf{v}} = \mathcal{I}_{\mathbf{v}} \cup \{ \gamma(\phi) \}$ *and save the vectors* $\mathbf{v}_i$ *and* $\mathbf{w}_i$, $i \in \mathcal{C}_\gamma(\phi)$.
(The vector $\mathbf{w}$ is the ϕ-th column of the matrix $\mathbf{W}_\phi^{\mathrm{dl}}$ in (3.19).)
(ii) *Repeat all of Step* 2.

3) (Normalize $\mathbf{v}$ and $\mathbf{w}$ to obtain the n-th pair of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$, and add them to current cluster.)
Set

$$\mathbf{v}_n = \frac{\mathbf{v}}{t_{n,\mu}} \quad \textit{and} \quad \mathbf{w}_n = \frac{\mathbf{w}}{\tilde{t}_{n,\phi}}, \tag{5.8}$$

where $t_{n,\mu}$ *and* $\tilde{t}_{n,\phi}$ *are suitable scaling factors, e.g.,*

$$t_{n,\mu} = \|\mathbf{v}\| \quad \textit{and} \quad \tilde{t}_{n,\phi} = \|\mathbf{w}\|.$$

Set $\mathbf{V}^{(\gamma)} = \mathbf{V}^{(\gamma)} \cup \{\mathbf{v}_n\}$ *and* $\mathbf{W}^{(\gamma)} = \mathbf{W}^{(\gamma)} \cup \{\mathbf{w}_n\}$.

4) (Update the history indices.)
Set $\mu_n = \mu$ *and* $\phi_n = \phi$.
(This records that $\mathbf{v}_n$ was obtained from $\mathbf{r}_{\mu_n+m}$ if $\mu_n \leq 0$ or from $\mathbf{A}\mathbf{v}_{\mu_n}$ if $\mu_n > 0$, and that $\mathbf{w}_n$ was obtained from $\mathbf{l}_{\phi_n+m}$ if $\phi_n \leq 0$ or from $\mathbf{A}^T\mathbf{w}_{\phi_n}$ if $\phi_n > 0$.)
Set $\mathcal{C}_\gamma = \mathcal{C}_\gamma \cup \{n\}$.
(This records that $\mathbf{v}_n$ and $\mathbf{w}_n$ are in the cluster with index $\gamma = \gamma(n)$.)

5) (Compute $\mathbf{\Delta}^{(\gamma)}$ and check for end of look-ahead cluster.)
Form

$$\mathbf{\Delta}^{(\gamma)} = \left(\mathbf{W}^{(\gamma)}\right)^T \mathbf{V}^{(\gamma)}.$$

If the matrix $\mathbf{\Delta}^{(\gamma)}$ *is "sufficiently" nonsingular, then*:
Increment the cluster counter $\gamma = \gamma + 1$;
Set $\gamma(n+1) = \gamma$, $\mathcal{C}_\gamma = \emptyset$, *and* $\mathbf{V}^{(\gamma)} = \mathbf{W}^{(\gamma)} = \emptyset$.
(The current cluster is complete, and so the $n+1$-th vectors constructed in the next iteration will start a new cluster.)
Otherwise:

Set $\gamma(n+1) = \gamma$.
(The current cluster is still incomplete, and so the $n+1$-th vectors constructed in the next iteration will still be added to the current cluster with index set $\mathcal{C}_{\gamma(n)}$.)

Remark 5.3. If no look-ahead steps occur, then

$$\gamma(n) = n, \quad \mathcal{C}_n = \{n\}, \quad \xi(n) = n, \quad , \mathbf{V}^{(n)} = \mathbf{v}_n, \quad \text{and} \quad \mathbf{W}^{(n)} = \mathbf{w}_n$$

for all $n = 1, 2, \ldots, n_{\max}$. Thus, in this case, the general Algorithm 5.2 just reduces to the generic Algorithm 4.1.

Remark 5.4. The optional orthogonalization in Steps 1f) and 2f) of Algorithm 5.2 only needs to be performed if the current cluster, $\mathcal{C}_\gamma$, is nonempty.

Remark 5.5. For the optional orthogonalization in Steps 1f) and 2f) of Algorithm 5.2, modified Gram-Schmidt should be used. Furthermore, the coefficients $t_{i,\mu}$, $i \in \mathcal{C}_\gamma$, respectively $\tilde{t}_{i,\phi}$, need to be updated to include the orthogonalization coefficients. Thus the orthogonalization in Step 1f) should be performed as follows:
For all $i \in \mathcal{C}_\gamma$, *set*:

$$\tau_{i,\mu} = \frac{\mathbf{v}_i^H \mathbf{v}}{\|\mathbf{v}_i\|^2}, \quad \mathbf{v} = \mathbf{v} - \mathbf{v}_i \tau_{i,\mu}, \quad \text{and} \quad t_{i,\mu} = t_{i,\mu} + \tau_{i,\mu}.$$

Similarly, the orthogonalization in Step 2f) is implemented as follows:
For all $i \in \mathcal{C}_\gamma$, *set*:

$$\tilde{\tau}_{i,\phi} = \frac{\mathbf{w}_i^H \mathbf{w}}{\|\mathbf{w}_i\|^2}, \quad \mathbf{w} = \mathbf{w} - \mathbf{w}_i \tilde{\tau}_{i,\phi}, \quad \text{and} \quad \tilde{t}_{i,\phi} = \tilde{t}_{i,\phi} + \tilde{\tau}_{i,\phi}.$$

Remark 5.6. The "cluster indices" $\gamma_{\mathbf{v}}$ and $\gamma_{\mathbf{w}}$ defined in (5.4) and (5.6) correspond to the "vector indices"

$$i_{\mathbf{v}} = \begin{cases} 1 & \text{if } \mu \le 0, \\ \max\{\, 1, \xi(\mu_{\xi(\mu)}) \,\} & \text{if } \mu > 0, \end{cases}$$

(5.9)

$$\text{and} \quad i_{\mathbf{w}} = \begin{cases} 1 & \text{if } \phi \le 0, \\ \max\{\, 1, \xi(\mu_{\xi(\phi)}) \,\} & \text{if } \phi > 0. \end{cases}$$

Indeed, using (5.3), the indices $\gamma_{\mathbf{v}}$ and $\gamma_{\mathbf{w}}$ directly translate into (5.9).

6. Properties of the Lanczos vectors

In Algorithm 5.2, true biorthogonality can always be achieved by explicitly biorthogonalizing against all previous vectors, i.e., by setting $\gamma_{\mathbf{v}} = \gamma_{\mathbf{w}} = 1$ in Steps 1d) and 2d). In this section, we prove two propositions to show that true biorthogonality can be achieved by explicitly biorthogonalizing only against the more recent clusters of vectors. We also discuss the case of *inexact deflation* for which the indices of the vectors involved must also be saved. We also show that in Steps 1f) and 2f), linear dependence against all previous vectors can be checked by examining only the vectors in the current cluster.

For the statements and the proofs of these results, we will use the vector indices (5.9) instead of $\gamma_{\mathbf{v}}$ and $\gamma_{\mathbf{w}}$.

Proposition 6.1. *In Step* 1e) *of Algorithm* 5.2 *at pass* n, *the vector* $\mathbf{v}$ *is already biorthogonal to* $\mathbf{w}_i$ *for all* $i < i_{\mathbf{v}}$ (*where* $i_{\mathbf{v}}$ *is given by* (5.9) *with* $\mu = \mu_n$), *as long as deflation occurs only when* $\mathbf{v}$ *and* $\mathbf{w}$ *are exact linear combinations of previous right and left vectors, respectively. Likewise in Step* 2e), *the vector* $\mathbf{w}$ *is already biorthogonal to* $\mathbf{v}_i$ *for all* $i < i_{\mathbf{w}}$ (*where* $i_{\mathbf{w}}$ *is given by* (5.9) *with* $\phi = \phi_n$), *under the same provision about deflation. Hence, in Steps* 1e) *and* 2e), *it is sufficient to biorthogonalize against just those more recent vectors starting with indices* $i_{\mathbf{v}}$ *and* $i_{\mathbf{w}}$. *With this limited biorthogonalization, the vectors* $\mathbf{v}_n$ *and* $\mathbf{w}_n$ *will be biorthogonal to all vectors* $\{\, \mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_{\xi(n)-1} \,\}$ *and* $\{\, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{\xi(n)-1} \,\}$, *respectively.*

Proof. The proof is by induction on the pass n through the main loop of the algorithm. At Step 1e) for a given pass n, we want to enforce the condition $\mathbf{w}_i^T \mathbf{v} = 0$ for $i < \xi(n)$. We would like to determine for which i is $\mathbf{w}_i^T \mathbf{v} = \mathbf{w}_i^T \mathbf{A} \mathbf{v}_{\mu_n}$ guaranteed to be already zero, relieving us of the necessity to biorthogonalize $\mathbf{v}$ against $\mathbf{w}_i$ explicitly. By induction, we know that $\mathbf{w}_j^T \mathbf{v}_{\mu_n} = 0$ for all $j < \xi(\mu_n)$, i.e. for all vectors in previous clusters $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(\gamma(n)-1)}$. Therefore, $\mathbf{w}_i^T \mathbf{A} \mathbf{v}_{\mu_n} = 0$ if $\mathbf{A}^T \mathbf{w}_i$ is a linear combination of the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_{\xi(\mu_n)-1}$. We consider two cases: I. there exists a j such that $\phi_j = i$, and II. there is no such j.

Case I. In this case, $\mathbf{w}_j$ was computed at the pass j through the main loop of the algorithm from $\mathbf{A}^T \mathbf{w}_i$ plus a linear combination of $\mathbf{w}_1, \ldots, \mathbf{w}_{j-1}$, hence $\mathbf{A}^T \mathbf{w}_i$ is a linear combination of $\mathbf{w}_1, \ldots, \mathbf{w}_j$. The strict monotonicity of the sequence $\{\phi_l\}_{l \geq 1}$ guarantees that for any i, $i = \phi_j < \phi_{\xi(\mu_n)}$ implies $j < \xi(\mu_n)$, which implies that $\mathbf{w}_i^T \mathbf{A}^T \mathbf{v}_{\mu_n} = 0$.

Case II. In this case, for any given i, choose j such that $\phi_j < i$, but $\phi_{j+1} > i$. This situation arises when at pass $j+1$ through the main loop, we were attempting to compute $\mathbf{w}_{j+1}$ by forming $\mathbf{A}^T \mathbf{w}_i$, but $\mathbf{A}^T \mathbf{w}_i$ was deflated out because it was already a linear combination of all the vectors computed to date: $\mathbf{w}_1, \ldots, \mathbf{w}_j$. Then $\mathbf{w}_i^T \mathbf{A}^T \mathbf{v}_{\mu_n} = 0$ is guaranteed as long as $j < \xi(\mu_n)$ as before, which in turn is guaranteed if $i < \phi_{\xi(\mu_n)}$.

Returning to pass n in the main loop, Step 1e), we are computing coefficients to enforce the condition

$$\mathbf{w}_i^T \mathbf{v} = \mathbf{w}_i^T \mathbf{A} \mathbf{v}_{\mu_n} - \mathbf{w}_i^T \mathbf{V}_{\xi(n)-1} \mathbf{\Delta}_{\xi(n)-1}^{-1} \mathbf{W}_{\xi(n)-1}^T \mathbf{v} = 0. \tag{6.1}$$

For $i < \xi(\phi_{\xi(\mu_n)}) \leq \phi_{\xi(\mu_n)}$, the preceding argument ensures that $\mathbf{w}_i^T \mathbf{A} \mathbf{v}_{\mu_n} = 0$, and thus the first term of the right-hand side of (6.1) is zero. Next we show that the second term,

$$\mathbf{w}_i^T \mathbf{V}_{\xi(n)-1} \mathbf{\Delta}^{-1} \mathbf{W}_{\xi(n)-1}^T \mathbf{v}, \tag{6.2}$$

of the right-hand side of (6.1) is also zero. Because of the clustering, $\mathbf{w}_i^T \mathbf{V}_{\xi(n)-1}$ is a vector with nonzeros in positions corresponding to the cluster containing vector $\mathbf{w}_i$, and $\mathbf{W}_{\xi(n)-1}^T \mathbf{v}$ is a vector with nonzeros in positions $\phi_{\xi(\mu_n)}, 1 + \phi_{\xi(\mu_n)}, \ldots, \xi(n) - 1$, by the preceding argument. Since, by (3.13), $\mathbf{\Delta}_{\xi(n)-1}$ is block diagonal with blocks corresponding to clusters defined by the sequence $\{\xi(n)\}$, the expression above is zero if $\mathbf{w}_i$ lies in a cluster earlier than the one containing vector number $\phi_{\xi(\mu_n)}$, hence, to guarantee that (6.2) is zero, i must be less than $\xi(\phi_{\xi(\mu_n)})$.

By swapping the roles of the left and right vectors and using the same arguments, we conclude that we do not need to biorthogonalize against all previous vectors, but only against the more recent ones. Specifically, we can amend Step 1d) and 1e) in the look-ahead algorithm to set $i_{\mathbf{v}}$ and $i_{\mathbf{w}}$ as given in (5.9). □

Remark 6.2. In the generic case, we have $\xi(i) = i$ for all i, and so the expressions (5.9) reduce to the integers $i_{\mathbf{v}}$ and $i_{\mathbf{w}}$ used in Steps 1d) and 2d) of the generic Algorithm 4.1.

Proposition 6.3. *In Step 2f), if the temporary vector* $\mathbf{v}$ *is a linear combination of* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n-1}$, *then it must be a linear combination of* $\mathbf{v}_{\xi(n)}, \mathbf{v}_{\xi(n)+1}, \ldots, \mathbf{v}_{n-1}$. *Thus in Step 2f), linear independence of the vector* $\mathbf{v}$ *with respect to* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n-1}$ *can be checked by just looking at the vectors* $\mathbf{v}_{\xi(n)}, \mathbf{v}_{\xi(n)+1}, \ldots, \mathbf{v}_{n-1}$, *i.e., those in the current cluster. A corresponding statement holds for the left Lanczos vectors.*

Proof. Decompose $\mathbf{v} = \mathbf{V}_{\xi(n)-1}\boldsymbol{\alpha} + \begin{bmatrix}\mathbf{v}_{\xi(n)} & \cdots & \mathbf{v}_{n-1}\end{bmatrix}\boldsymbol{\beta}$, in terms of the vectors in the current cluster, $\mathbf{v}_{\xi(n)}, \ldots, \mathbf{v}_{n-1}$ and in all the previous clusters, $\mathbf{V}_{\xi(n)-1}$. Then the biorthogonality condition (3.11) (just enforced in the preceding Step 1e)) yields

$$\begin{aligned} 0 &= \mathbf{W}_{\xi(n)-1}^T \mathbf{v} \\ &= \mathbf{W}_{\xi(n)-1}^T \mathbf{V}_{\xi(n)-1}\boldsymbol{\alpha} + \mathbf{W}_{\xi(n)-1}^T \begin{bmatrix}\mathbf{v}_{\xi(n)} & \cdots & \mathbf{v}_{n-1}\end{bmatrix}\boldsymbol{\beta} \\ &= \boldsymbol{\Delta}_{\xi(n)-1}\boldsymbol{\alpha}. \end{aligned}$$

Since $\boldsymbol{\Delta}_{\xi(n)-1}$ is nonsingular by construction, this implies that $\boldsymbol{\alpha} = 0$. □

The above discussion leads to the following theorem stating that Algorithm 5.2 actually generates the two cluster-wise biorthogonal bases for the appropriate block Krylov subspaces.

Theorem 6.4. *The general Lanczos Algorithm* 5.2 *generates a sequence of vectors* (3.1) *satisfying the conditions* (3.2), (3.3), *and the cluster-biorthogonality conditions* (3.11), *where the vectors are grouped into clusters* (3.9). *Algorithm* 5.2 *also generates two matrices of coefficients* $\mathbf{T}_\mu^{(e)}$ *and* $\widetilde{\mathbf{T}}_\mu^{(e)}$ *such that the recurrences* (3.18) *and* (3.19) *are satisfied.*

Proof. Conditions (3.2) and (3.3) follow directly from the formulas in Steps 1) and 2) of the algorithm by induction, since each new vector generated is equal to $\mathbf{A}$ or $\mathbf{A}^T$ times a previous vector, plus a linear combination of previous vectors. Condition (3.11) also follows by a rearrangement of the same formulas, just as in the generic algorithm. If deflation is carried out only when the vector to be deflated is an exact linear combination of previous vectors, then Proposition 6.1 applies to show that the biorthogonality condition (3.11) is satisfied.

What if we choose to deflate a vector that is not an exact linear combination of preceding vectors? Such a circumstance could occur if a vector were "numerically almost" a linear combination, for instance if the residue left after all biorthogonalization has been applied is very small. We consider the same situation as in the proof of Proposition 6.1. We are computing $\mathbf{v} = \mathbf{A}\mathbf{v}_{\mu_n}$, and given a $\mathbf{w}_i$, we would like to know if $\mathbf{w}_i$ is already orthogonal to $\mathbf{v}$. As in the proof of Proposition 6.1, we define j as the smallest index such that $i < \phi_{j+1}$, but $i > \phi_j$; $j+1$ is the index of the pass through the main loop when $\mathbf{A}^T\mathbf{w}_i$ was computed. But in this case, we assume that at the $j+1$-th pass, $\mathbf{A}^T\mathbf{w}_i$ had been thrown away even though it was not an exact linear combination of the vectors existing to date: $\mathbf{w}_1, \ldots, \mathbf{w}_j$. Then $\mathbf{A}^T\mathbf{w}_i$ is not in the space of vectors against which $\mathbf{v}_{\mu_n}$ has ever been biorthogonalized, and hence $\mathbf{w}_i^T\mathbf{A}\mathbf{v}_{\mu_n} \neq 0$. The expression (6.2) is also not zero, since $t_{i,\mu_n} \neq 0$. All the entries in the entire cluster containing the i-th entry are also nonzero due to the block-diagonal structure (3.13) of $\boldsymbol{\Delta}$. Therefore, in order to maintain the

conditions (3.11), it is necessary and sufficient to orthogonalize $\mathbf{v}$ explicitly against *the entire cluster* containing $\mathbf{w}_i$. This is encoded in the algorithm 5.2 by accumulating the index sets $\mathcal{I}_\mathbf{v}$ and $\mathcal{I}_\mathbf{w}$ containing the indices of each entire cluster to be so saved. □

7. Computational aspects

In this section, we discuss some implementation details for Algorithms 4.1 and 5.2. These details are necessary to produce a complete or efficient implementation, but have been left out of the descriptions of Algorithms 4.1 and 5.2 to keep the exposition as simple as possible.

Steps 1e) and 2e) implement a classical two-sided Gram-Schmidt biorthogonalization, but in practice a "modified" two-sided Gram-Schmidt process would be preferred [33]. For example, for the "modified" version of Step 1e) in Algorithm 4.1, one simply replaces (4.1) and (4.2) by the following update:
For all $i \in \mathcal{I}$, *set*:

$$t_{i,\mu} = \frac{\mathbf{w}_i^T \mathbf{v}}{\delta_i} \quad \text{and} \quad \mathbf{v} = \mathbf{v} - \mathbf{v}_i t_{i,\mu}.$$

In Step 2f), we must decide whether or not to deflate. Deflation must occur if the newly generated vector $\mathbf{w}$ is an exact linear combination of previous vectors $\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}$. We call this situation *exact deflation*. Proposition 6.3 shows that the biorthogonalization in Step 2e) has removed any component in the directions $\mathbf{w}_1, \ldots, \mathbf{w}_{\xi(n)-1}$, leaving only the component in the span of vectors in the current cluster: $\mathbf{w}_{\xi(n)}, \ldots, \mathbf{w}_{n-1}$. If not already zero, $\mathbf{w}$ must be orthogonalized against the vectors $\mathbf{w}_{\xi(n)}, \ldots, \mathbf{w}_{n-1}$ in order to determine whether it is linearly independent. If it is exactly linearly dependent, then it must be deflated. If it is linearly independent, then it is the user's choice whether to save the $\mathbf{w}$ before or after this last orthogonalization. In any case, the coefficients $\left[\tilde{t}_{i,\phi}\right]_{\xi(n) \le i \le n-1}$ of $\tilde{\mathbf{T}}_\phi^{(e)}$ must be filled in, unless $\mathbf{w}$ is not deflated and the unorthogonalized version is saved. In exact deflation, Proposition 6.1 applies, and there is no need to accumulate any indices in $\mathcal{I}_\mathbf{v}$. Corresponding statements hold for exact deflation of a right Lanczos vector in Step 1f).

If the $\mathbf{w}$ in Step 2f) is not an exact linear combination of previous vectors, the user may decide to deflate anyway. This situation is called *inexact deflation*. This may happen if, for example, $\mathbf{w}$ is "almost" a linear combination, according to some criteria set in advance. Whenever inexact deflation occurs, Theorem 6.4 states that it is necessary to save the indices of vectors in $\mathcal{I}_\mathbf{v}$ in order to accomplish true biorthogonalization against older Lanczos vectors in later passes. It may happen that throwing away these indices will have only a small effect on the resulting Lanczos vectors, but we will reserve discussion of these numerical issues for a later paper. In any case, in Step 1d), we need only consider those indices less than $\xi(n)$, since we cannot biorthogonalize against vectors in the same cluster as $\mathbf{w}_n$. Corresponding statements apply to the right vectors in Step 2f).

We emphasize that the index sets $\mathcal{I}_\mathbf{w}$ and $\mathcal{I}_\mathbf{v}$ do not contain indices of vectors that have been deflated out—these vectors have no index since they do not appear among the Lanczos vectors at all. Rather, in the case of the generic Algorithm 4.1 without look-ahead, $\mathcal{I}_\mathbf{w}$, respectively $\mathcal{I}_\mathbf{v}$, contain the indices of the vectors already among the right, respectively left, Lanczos vectors from previous passes such that when these vectors are expanded in Step 1c), respectively Step 2c), the expanded

result almost lies within the space of existing vectors. Similarly, in the case of the general Algorithm 5.2 with look-ahead, $\mathcal{I}_{\mathbf{w}}$, respectively $\mathcal{I}_{\mathbf{v}}$ contain the indices of those look-ahead clusters that contain at least one vector that lead to an almost linearly dependent vector when expanded in Step 1c), respectively Step 2c).

In Step 5) of the general Algorithm 5.2, one needs to decide if the current look-ahead cluster can be closed. In view of (3.12), a necessary condition for closing the look-ahead cluster is that the matrix $\mathbf{\Delta}^{(\gamma)}$ is nonsingular. It is thus tempting to base the look-ahead strategy solely on a measure of singularity of $\mathbf{\Delta}^{(\gamma)}$, such as the smallest singular value. However, as was illustrated in [17] for the Lanczos algorithm with single starting vectors, such a look-ahead strategy is not appropriate and does not lead to a robust algorithm. Instead, a reliable look-ahead strategy needs to check the singularity of $\mathbf{\Delta}^{(\gamma)}$, as well as the sizes of the recurrence coefficients $t_{i,\mu}$ and $\tilde{t}_{i,\phi}$ in (5.5) and (5.7) relative to some estimate, $n(\mathbf{A})$, for $\|\mathbf{A}\|$. More precisely, the second check states that the current look-ahead cluster should only be closed if

$$|t_{i,\mu}|\,,\ \left|\tilde{t}_{i,\phi}\right| \leq \texttt{fac} \times n(\mathbf{A}) \quad \text{for all} \quad i. \tag{7.1}$$

Here, `fac` is an appropriate constant, typically $\texttt{fac} = 10$, and if not available a priori, the norm estimate $n(\mathbf{A})$ can be obtained easily during the first few iterations. This check guarantees (7.1) that the component $\mathbf{A}\mathbf{v}_\mu$, respectively $\mathbf{A}^T\mathbf{w}_\phi$, of the new Lanczos vector is not dominated by the previous Lanczos vectors. Note that $\mathbf{A}\mathbf{v}_\mu$ and $\mathbf{A}^T\mathbf{w}_\phi$ are the only parts of $\mathbf{v}$ and $\mathbf{w}$, respectively, that advance the block Krylov subspaces.

8. Concluding remarks

We presented a Lanczos-type algorithm for the construction of biorthogonal basis vectors for the right and left block Krylov subspaces induced by a given square matrix and two blocks of m right and p left starting vectors. Our algorithm can handle the most general case of arbitrary, and not necessarily identical, initial block sizes m and p, while all previously proposed Lanczos-type algorithms for multiple starting vectors are restricted to the special case $m = p$. Another feature of our algorithm is a built-in deflation procedure to detect and delete linearly dependent or almost linearly dependent vectors in the underlying block Krylov subspaces. We also showed how look-ahead can be incorporated into our algorithm in order to remedy the potential breakdowns and near-breakdowns that can occur in Lanczos-type algorithms for nonsymmetric matrices.

The objective of this paper is to describe our Lanczos-type algorithm and to prove some of its key properties. In order to keep the length of the paper reasonable, we decided not to include numerical examples. Applications of the algorithm to the problems mentioned in Section 1.2 and results of numerical experiments will be reported elsewhere.

Freund and Malhotra [18] already developed a block version of QMR, called BL-QMR, for the solution of multiple linear systems (1.1) that uses the generic Lanczos-type Algorithm 4.1 with deflation, but without look-ahead. The BL-QMR algorithm has been tested extensively and numerical results are presented in [18, 30]. The results in [18, 30] clearly illustrate the importance of deflation in the underlying Lanczos-type algorithm. More precisely, basis vectors do become almost linearly dependent in several of the numerical examples in [18, 30], yet BL-QMR converges

as long as these vectors are deflated properly. However, as soon as the deflation procedure is turned off, BL-QMR fails to converge.

In [13], Feldmann and Freund describe the application of an early version (without deflation and without look-ahead) of our Lanczos-type algorithm to the problem of computing matrix Padé approximants to matrix-valued transfer functions (1.5). Some numerical results for problems of this type arising in circuit simulation are also presented in [13].

Finally, we stress that we are well aware of the connections between the problem treated in this paper, namely the construction of suitable basis vectors for block Krylov subspaces, and the related problems of solving block Hankel systems and constructing matrix Padé approximations; see, e.g., [7, 8, 35, 41, 42] and the references given therein. The connections between the proposed Lanczos-type algorithm and these related problems should be explored further. A first such result on the connection of our algorithm to matrix Padé approximation was given by Freund [15].

Appendix

In this appendix, we illustrate the effects of *exact deflation*, *inexact deflation*, and *look-ahead clusters* with a specific example. In order to simplify the presentation of the example, we just give the overall structure of the matrices computed without giving actual numerical values. In the case of inexact deflation and/or look-ahead clusters, it is easy to construct an artificial example that will exhibit the indicated structure by just starting with a completely generic Lanczos expansion. It is then always possible to force an inexact deflation and/or a look-ahead cluster at any point in the expansion, even if not called for by any heuristic such as a norm bound. However, to generate an example exhibiting exact deflation, it is necessary to start with blocks of starting vectors, $\mathbf{R}$ and $\mathbf{L}$, such that the block Krylov subspaces they induce intersect low-order invariant subspaces of $\mathbf{A}$ and $\mathbf{A}^T$, respectively.

Consider a matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$, a matrix of m right starting vectors $\mathbf{R}$, and a matrix of p left starting vectors $\mathbf{L}$, where $N = 20$, $m = 3$, $p = 2$, and suppose that, in Algorithm 5.2, we encounter a deflation on a $\mathbf{v}$ vector at iteration $n = 11$ and on a $\mathbf{w}$ vector at iteration $n = 14$. For $n \leq 10$ (i.e., before any deflation), each new $\mathbf{v}_n$ is generated from $\mathbf{v} = \mathbf{A}\,\mathbf{v}_\mu$ where $\mu = n - 3$, and each new $\mathbf{w}_n$ is generated from $\mathbf{w} = \mathbf{A}^T\mathbf{w}_\phi$ where $\phi = n - 2$. Now suppose that at iteration $n = 11$ we find that $\mathbf{A}\mathbf{v}_8$ is a linear combination of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{10}$, and that at iteration $n = 14$ we find that $\mathbf{A}^T\mathbf{w}_{12}$ is a linear combination of $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{13}$. Hence in both cases, we decide to *deflate* those vectors out so that $\mathbf{v}_{11}$ and $\mathbf{w}_{14}$ end up being generated from $\mathbf{A}\mathbf{v}_9$ and $\mathbf{A}^T\mathbf{w}_{13}$, respectively.

In the situation described above, the indices computed by Algorithm 4.1 are shown in Table 1. The iteration number n is placed in parentheses when a deflation occurs, since then the computation of the new n-th $\mathbf{v}$ or $\mathbf{w}$ vector is postponed.

The matrices of coefficients generated have the banded nonzero structure given in Figures 2 and 3. Here and in the sequel, we use "+" to denote a diagonal entry, "`x`" to denote a nonzero off-diagonal entry, and "`d`" to denote an entry that is nonzero only if the deflations are inexact. Furthermore, we use "·" to mark zero entries.

In this example, $n_{\max} = N = 20$, and the matrices of recurrences coefficients at termination of the algorithm are $\mathbf{T}_{20}^{(e)}$ and $\widetilde{\mathbf{T}}_{20}^{(e)}$. The structure of $\mathbf{T}_{20}^{(e)}$ can be

TABLE 1. Table of indices for example with deflation but no look-ahead.

n	μ_n	ϕ_n	ϕ_{μ_n}	μ_{ϕ_n}	Remarks
1	-2	-1	-	-	
2	-1	0	-	-	
3	0	1	-	-	
4	1	2	-	-	
5	2	3	-	-	
6	3	4	1	1	
7	4	5	2	2	
8	5	6	3	3	
9	6	7	4	4	
10	7	8	5	5	
(11)	8		6		$\mathbf{A}\mathbf{v}_8$ deflated
11	9	9	7	6	
12	10	10	8	7	
13	11	11	9	9	
(14)		12		10	$\mathbf{A}^T\mathbf{w}_{12}$ deflated
14	12	13	10	11	
15	13	14	11	12	
16	14	15	13	13	
17	15	16	14	14	
18	16	17	15	15	
19	17	18	16	16	
20	18	19	17	17	
(21)	19				$\mathbf{A}\mathbf{v}_{19}$ deflated
(21)	20				$\mathbf{A}\mathbf{v}_{20}$ deflated
(21)		20		18	$\mathbf{A}^T\mathbf{w}_{20}$ deflated

read off Table 1 by reading down the table column labeled μ_n as follows. To find out the nonzeros in matrix column 9, read the line in Table 1 where a 9 appears in table column μ_n. Then the nonzeros in the 9-th matrix column lie in the rows between the indices in the table columns marked ϕ_{μ_n} and n, namely 7 through 11. When a deflation occurs, as in $\mathbf{A}\mathbf{v}_8$ while in iteration $n = 11$, the table entry is flagged by the parentheses. In this case the nonzeros lie the range from the index in table column ϕ_{μ_n} up to *one less* than the entry in table column n, namely 6 through 10. These limits do not include the entries arising from inexact deflation. A dual procedure can be used for the structure of $\widetilde{\mathbf{T}}_{20}^{(e)}$, using table columns ϕ_n, n, μ_{ϕ_n} in place of μ_n, n, ϕ_{μ_n}, respectively.

In this example, the matrix $\mathbf{\Delta}_{20} = \mathbf{W}_{20}^T\mathbf{V}_{20}$ is diagonal, since no look-ahead steps have been performed.

We now modify this example to add look-ahead clusters to illustrate the effect of such clusters on the structure of the coefficient matrices. We form two consecutive pairs of look-ahead clusters of length 3 each, one with $\begin{bmatrix}\mathbf{v}_5 & \mathbf{v}_6 & \mathbf{v}_7\end{bmatrix}$, $\begin{bmatrix}\mathbf{w}_5 & \mathbf{w}_6 & \mathbf{w}_7\end{bmatrix}$ and one with $\begin{bmatrix}\mathbf{v}_8 & \mathbf{v}_9 & \mathbf{v}_{10}\end{bmatrix}$, $\begin{bmatrix}\mathbf{w}_8 & \mathbf{w}_9 & \mathbf{w}_{10}\end{bmatrix}$. The μ and ϕ indices are identical to the above, but the indices marking the vectors against which we must orthogonalize will also incorporate the look-ahead structure, according to the formulas $\gamma_{\mathbf{v}} = \max\{\,1, \xi(\phi_{\xi(\mu)})\,\}$, $\gamma_{\mathbf{w}} = \max\{\,1, \xi(\mu_{\xi(\phi)})\,\}$. The indices are shown

$$\mathbf{T}_{20}^{(e)} = \begin{bmatrix}
+ & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & d & d & d & d & d & d & d & d \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & +
\end{bmatrix}$$

FIGURE 2. Zero structure of right-hand coefficient matrix for example with deflation but no look-ahead.

$$\widetilde{\mathbf{T}}_{20}^{(e)} = \begin{bmatrix}
+ & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & d & d & d & d & d & d & d & d & d & d \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & +
\end{bmatrix}$$

FIGURE 3. Zero structure of left-hand coefficient matrix for example with deflation but no look-ahead.

in Table 2. The resulting cluster index sets are as follows:

$$\begin{array}{llll}
\mathcal{C}_1 = \{1\} & \mathcal{C}_5 = \{5,6,7\} & \mathcal{C}_9 = \{13\} & \mathcal{C}_{13} = \{17\} \\
\mathcal{C}_2 = \{2\} & \mathcal{C}_6 = \{8,9,10\} & \mathcal{C}_{10} = \{14\} & \mathcal{C}_{14} = \{18\} \\
\mathcal{C}_3 = \{3\} & \mathcal{C}_7 = \{11\} & \mathcal{C}_{11} = \{15\} & \mathcal{C}_{15} = \{19\} \\
\mathcal{C}_4 = \{4\} & \mathcal{C}_8 = \{12\} & \mathcal{C}_{12} = \{16\} & \mathcal{C}_{16} = \{20\}
\end{array}$$

TABLE 2. Table of indices for example with both deflation and look-ahead.

n	μ_n	ϕ_n	$\gamma(n)$	$\xi(n)$	$\xi(\phi_{\xi(\mu_n)})$	$\xi(\mu_{\xi(\phi_n)})$	Remarks
1	-2	-1	1	1	-	-	
2	-1	0	2	2	-	-	
3	0	1	3	3	-	-	
4	1	2	4	4	-	-	
5	2	3	5	5	-	-	
6	3	4	5	5	1	1	
7	4	5	5	5	2	2	
8	5	6	6	8	3	2	
9	6	7	6	8	3	2	
10	7	8	6	8	3	5	
(11)	8				5		$\mathbf{A}\mathbf{v}_8$ deflated
11	9	9	7	11	5	5	
12	10	10	8	12	5	5	
13	11	11	9	13	8	8	
(14)		12				8	$\mathbf{A}^T\mathbf{w}_{12}$ deflated
14	12	13	10	14	8	11	
15	13	14	11	15	11	12	
16	14	15	12	16	13	13	
17	15	16	13	17	14	14	
18	16	17	14	18	15	15	
19	17	18	15	19	16	16	
20	18	19	16	20	17	17	
(21)	19				18		$\mathbf{A}\mathbf{v}_{19}$ deflated
(21)	20				19		$\mathbf{A}\mathbf{v}_{20}$ deflated
(21)		20				18	$\mathbf{A}^T\mathbf{w}_{20}$ deflated

The structure of the coefficient matrices are shown in Figures 4 and 5, using the same notation as before. The zero structures in this case can be read off Table 2 in a manner similar to the previous. For example, the nonzeros in column μ_n of $\mathbf{T}_{20}^{(e)}$ lie in the rows with indices ranging from the corresponding "$\xi(\phi_{\xi(\mu_n)})$" entry through the "n" entry (one less if deflation occurs) in Table 2.

$$\mathbf{T}_{20}^{(e)} = \begin{bmatrix}
+ & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & + & x & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & x & x & + & x & x & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & x & x & + & x & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & x & x & + & x & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & d & d & d & d & d & d & d \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & +
\end{bmatrix}$$

FIGURE 4. Zero structure of right-hand coefficient matrix for example with both deflation and look-ahead.

$$\tilde{\mathbf{T}}_{20}^{(e)} = \begin{bmatrix}
+ & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & + & x & x & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & + & x & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & x & + & x & x & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & x & + & x & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & x & + & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & x & x & d & d & d & d & d & d & d & d \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & x & d & d & d & d & d & d & d & d \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & d & d & d & d & d & d & d & d \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & + & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & +
\end{bmatrix}$$

FIGURE 5. Zero structure of left-hand coefficient matrix for example with both deflation and look-ahead.

In the look-ahead case, the matrix $\mathbf{\Delta}_{20}$ is only block diagonal, as illustrated in Figure 6.

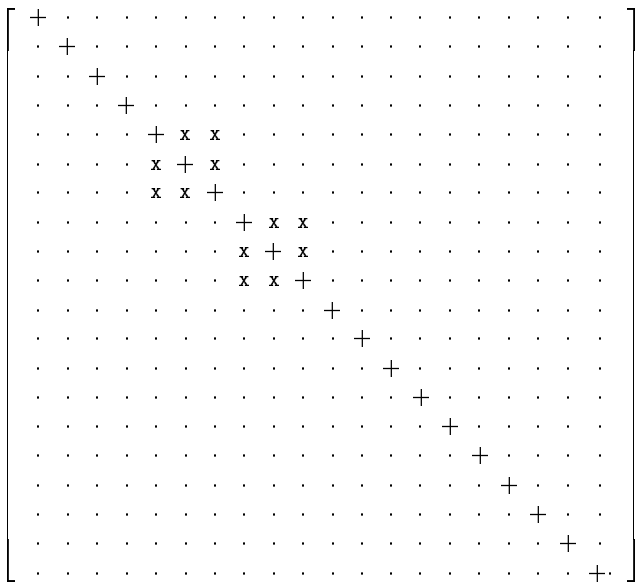
$\boldsymbol{\Delta}_{20} =$ 

FIGURE 6. Zero structure of block-diagonal matrix $\boldsymbol{\Delta}_{20}$ for example with deflation and look-ahead.

ACKNOWLEDGMENTS

The third author would like to thank Jane Cullum for pointing out the deflation procedure in [9]. He is also grateful to Susanne Freund for proof-reading various versions of this paper.

References

[1] J. I. Aliaga, *Algoritmos paralelos basados en el método de Lanczos. Aplicaciones en problemas de control*, Doctoral Thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain, 1995.

[2] J. I. Aliaga, D. L. Boley, and V. Hernández, *A block clustered Lanczos algorithm*, Presentation at the workshop on "Numerical Linear Algebra with Applications", Oberwolfach, Germany, April 1994.

[3] Z. Bai, D. Day, and Q. Ye, *ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems*, Research Report 95-04, Department of Mathematics, University of Kentucky, Lexington, KY, 1995.

[4] D. L. Boley, *Krylov space methods on state-space control models*, Circuits Systems Signal Process. **13** (1994), 733–758.

[5] D. L. Boley, S. Elhay, G. H. Golub, and M. H. Gutknecht, *Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights*, Numer. Algorithms **1** (1991), 21–44.

[6] D. L. Boley and G. H. Golub, *The nonsymmetric Lanczos algorithm and controllability*, Systems Control Lett. **16** (1991), 97–105.

[7] O. H. Bosgra and A. J. J. Van der Weiden, *Input-out invariants for linear multivariable systems*, IEEE Trans. Automat. Control **AC-25** (1980), 20–36.

[8] A. Bultheel, *Recursive algorithms for the matrix Padé problem*, Math. Comp. **35** (1980), 875–892.

[9] J. K. Cullum and W. E. Donath, *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace for large, sparse symmetric matrices*, Proc. 1974 IEEE Conference on Decision and Control, IEEE Press, New York, 1974, pp. 505–509.

[10] J. K. Cullum and R. A. Willoughby, *Lanczos algorithms for large symmetric eigenvalue computations, Volume 1, Theory*, Birkhäuser, Basel, 1985.

[11] ———, *A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices*, Large Scale Eigenvalue Problems (J. Cullum and R. A. Willoughby, eds.), North-Holland, Amsterdam, The Netherlands, 1986, pp. 193–240.

[12] P. Feldmann and R. W. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. Computer-Aided Design **14** (1995), 639–649.

[13] ———, *Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm*, Proc. 32nd Design Automation Conference, ACM, New York, 1995, pp. 474–479.

[14] R. W. Freund, *The look-ahead Lanczos process for nonsymmetric matrices and its applications*, Proceedings of the Cornelius Lanczos International Centenary Conference (J. D. Brown, M. T. Chu, D. C. Ellison, and R. J. Plemmons, eds.), SIAM, Philadelphia, 1994, pp. 33–47.

[15] ———, *Computation of matrix Padé approximations of transfer functions via a Lanczos-type process*, Approximation Theory VIII, Vol. 1: Approximation and Interpolation (C. K. Chui and L. L. Schumaker, eds.), World Scientific Publishing Co., Inc., Singapore, 1995, pp. 215–222.

[16] R. W. Freund and P. Feldmann, *Efficient circuit analysis by Padé approximation via the Lanczos process*, Presentation at the workshop on "Numerical Linear Algebra with Applications", Oberwolfach, Germany, April 1994.

[17] 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** (1993), 137–158.

[18] R. W. Freund and M. Malhotra, *A block-QMR algorithm for non-Hermitian linear systems with multiple right-hand sides*, Numerical Analysis Manuscript No. 95–09, AT&T Bell Laboratories, Murray Hill, NJ, 1995. (Available on WWW at `http://cm.bell-labs.com/cs/doc/95`)

[19] R. W. Freund and N. M. Nachtigal, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math. **60** (1991), 315–339.

[20] ———, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput. **15** (1994), 313–337.

[21] G. H. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, Mathematical Software III (J. R. Rice, ed.), Academic Press, New York, 1977, pp. 361–377.

[22] W. B. Gragg, *Matrix interpretations and applications of the continued fraction algorithm*, Rocky Mountain J. Math. **4** (1974), 213–225.

[23] W. B. Gragg and A. Lindquist, *On the partial realization problem*, Linear Algebra Appl. **50** (1983), 277–319.

[24] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, part I*, SIAM J. Matrix Anal. Appl. **13** (1992), 594–639.

[25] ———, *A completed theory of the unsymmetric Lanczos process and related algorithms, part II*, SIAM J. Matrix Anal. Appl. **15** (1994), 15–58.

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

[27] ———, *Computational enhancement of an unsymmetric block Lanczos algorithm*, Internat. J. Numer. Methods Engrg. **30** (1990), 1083–1089.

[28] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards **45** (1950), 255–282.

[29] ———, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards **49** (1952), 33–53.

[30] M. Malhotra, R. W. Freund, and P. M. Pinsky, *Iterative solution of multiple radiation and scattering problems in structural acoustics using a block quasi-minimal residual algorithm*, Numerical Analysis Manuscript No. 96–11, Bell Laboratories, Murray Hill, NJ, 1996. (Available on WWW at `http://cm.bell-labs.com/cs/doc/96`)

[31] A. A. Nikishin and A. Yu. Yeremin, *Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: general iterative scheme*, SIAM J. Matrix Anal. Appl. **16** (1995), 1135–1153.

[32] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl. **29** (1980), 293–322.

[33] B. N. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl. **13** (1992), 567–593.

[34] B. N. Parlett, D. R. Taylor, and Z. A. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp. **44** (1985), 105–124.

[35] J. Rissanen, *Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomials*, Math. Comp. **27** (1973), 147–154.

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

[37] T.-J. Su, *A decentralized linear quadratic control design method for flexible structures*, Ph.D. Thesis, Department of Aerospace and Engineering Mechanics, The University of Texas at Austin, Austin, TX, 1989.

[38] T.-J. Su and R. R. Craig, Jr., *Model reduction and control of flexible structures using Krylov vectors*, J. Guidance Control Dynamics **14** (1991), 260–267.

[39] D. R. Taylor, *Analysis of the look ahead Lanczos algorithm*, Ph.D. Thesis, Department of Mathematics, University of California, Berkeley, CA, 1982.

[40] R. Underwood, *An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems*, Ph.D. Thesis, Computer Science Department, Stanford University, Stanford, CA, 1975.

[41] A. J. J. Van der Weiden and O. H. Bosgra, *The determination of structural properties of a linear multivariable system by operations of system similarity*, Internat. J. Control **32** (1980), 489–537.

[42] G.-L. Xu and A. Bultheel, *Matrix Padé approximation: definitions and properties*, Linear Algebra Appl. **137/138** (1990), 67–136.

(J. I. Aliaga) Departamento de Informática, Universidad Jaume I, Campus de Penyeta Roja, 12071 Castellón, Spain
E-mail address: aliaga@inf.uji.es

(D. L. Boley) Computer Science Department, University of Minnesota, 4-192 EE/CSci Building, 200 Union Street S.E., Minneapolis, Minnesota 55455–0159
E-mail address: boley@cs.umn.edu

(R. W. Freund) Bell Laboratories, Lucent Technologies, Room 2C–420, 700 Mountain Avenue, Murray Hill, New Jersey 07974–0636
E-mail address: freund@research.bell-labs.com

(V. Hernández) Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Apartado de Correos 22012, 46071 Valencia, Spain
E-mail address: vhernand@dsic.upv.es