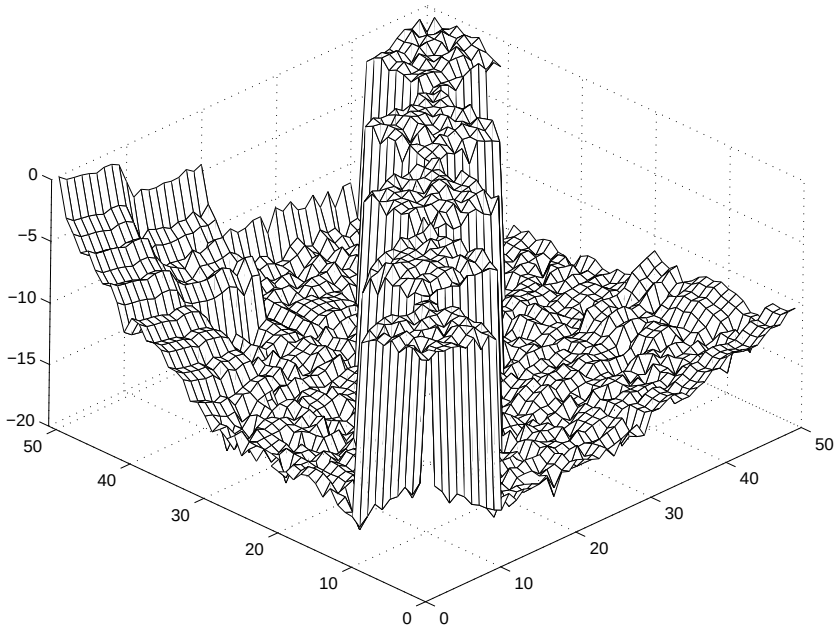
Diss. ETH No. 16337

Reliable Nonsymmetric Block Lanczos Algorithms

Damian Loher



Diss. ETH No. 16337

Reliable Nonsymmetric Block Lanczos Algorithms

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Mathematics

presented by

DAMIAN LOHER
dipl. phys. ETH
born 25.07.1968

citizen of
St. Gallen and Oberriet SG

accepted on the recommendation of

Prof. M. Gutknecht, examiner
Prof. R. Jeltsch, co-examiner

2006

Abstract

In this thesis we consider a new approach to the iterative solution of linear systems

$$\mathbf{AX} = \mathbf{B}$$

with a nonhermitian, nonsingular coefficient matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$ and multiple right-hand sides stored in $\mathbf{B} \in \mathbb{C}^{N \times s}$ that are all known in advance. We intend to generalize methods based on the unsymmetric Lanczos process, such as QMR and BiCG, to the block case, thereby including look-ahead and deflation (i.e. the elimination of (nearly) linearly dependent right and left Lanczos vectors). We propose to make use of block operations whenever possible, in contrast to the algorithm developed by Aliaga, Boley, Freund and Hernandez [1], where a vectorwise construction of the basis vectors for the block Krylov spaces is used. We point out, however, that with the algorithm from [1], the matrix vector products could still be computed blockwise, but other operations like orthogonalizations of new vectors have to be done vectorwise. In our algorithm, all operations are done blockwise, and this leads to the additional possibility of permuting columns to avoid look-ahead in certain situations. Additionally, a completely blockwise approach offers more potential for program optimization.

From the above it follows that the first step is to develop a suitable block version of the unsymmetric Lanczos process. This is achieved by decoupling the sizes of the right and left clusters (defined by the diagonal blocks of the block diagonal inner product matrix of the left and right Lanczos vectors) from the sizes of the right and left blocks. Consequently, the cluster size is not related to the sizes of the blocks, and we may choose it according to aspects like performance and numerical stability. We will formulate our block Lanczos algorithm in such a way that the look-ahead stratagy (which also determines the cluster size) can be varied by changing just a few statements of the algorithm so that different variants can be tried out easily. The result of the first step are two algorithms which are generalizations of the look-ahead Lanczos biorthogonalization method (LABiO) and the look-ahead Lanczos biorthogonalization and biconjugation method (LABiOC) to the block case.

The second step is then to construct iterative solvers on the basis of our block Lanczos process. This will be done for QMR and BiCG in the second part of this work. In the case of QMR, only the variant based on our block version of LABiO will be developed, and for BiCG, we will consider block variants of BiORes and BiOMin. At the end of this part, numerical experiments are carried out to test the new algorithms.

Zusammenfassung

In dieser Arbeit betrachten wir einen neuen Weg, um lineare Gleichungssysteme

$$\mathbf{AX} = \mathbf{B}$$

mit nichthermitescher, regulärer Koeffizientenmatrix $\mathbf{A} \in \mathbb{C}^{N \times N}$ und mehreren rechten Seiten in $\mathbf{B} \in \mathbb{C}^{N \times s}$ iterativ zu lösen, wobei alle rechten Seiten schon im voraus bekannt sein müssen. Wir wollen Methoden, die auf dem unsymmetrischen Lanczos-Prozess aufbauen, wie z. B. QMR und BiCG, auf den Blockfall verallgemeinern und dabei auch Look-ahead und Deflation (also die Elimination von (beinahe) linear abhängigen rechten und linken Lanczos-Vektoren) berücksichtigen. Wir schlagen vor, dass Blockoperationen verwendet werden, wann immer dies möglich ist, im Gegensatz zum Algorithmus von Aliaga, Boley, Freund und Hernandez [1], wo die Basisvektoren für die Block-Krylovräume einzeln erzeugt werden. Zwar könnten beim Algorithmus aus [1] die Matrix-Vektor-Produkte ebenfalls blockweise berechnet werden, aber andere Operationen wie die Orthogonalisierung neuer Vektoren müssen für jeden Vektor einzeln durchgeführt werden. Bei unserem Algorithmus werden alle Operationen blockweise gemacht, und dies führt zur zusätzlichen Möglichkeit, Spalten zu vertauschen, damit in gewissen Situationen Look-ahead vermieden werden kann. Weiter bietet der blockweise Zugang mehr Potential, Programme zu optimieren.

Aus dem Obigen folgt, dass der erste Schritt darin besteht, eine geeignete Blockversion des unsymmetrischen Lanczos-Prozesses zu entwickeln. Dies wird dadurch erreicht, dass die Grösse der

linken und rechten Cluster (die durch die diagonalen Blöcke der blockdiagonalen Skalarproduktmatrix der linken und rechten Lanczos-Vektoren definiert ist) entkoppelt wird von den Grössen der rechten und linken Blöcke. Die Clustergrösse ist daher unabhängig von den Grössen der rechten und linken Blöcke, und wir können sie gemäss anderen Aspekten wie Geschwindigkeit oder numerischer Stabilität wählen. Wir werden unseren Block-Lanczos-Prozess so formulieren, dass die Look-ahead-Strategie (die auch die Grösse der Cluster bestimmt) durch Ändern von wenigen Anweisungen des Algorithmus variiert werden kann, so dass verschiedene Varianten leicht ausprobiert werden können. Das Resultat des ersten Schrittes sind zwei Algorithmen, die die Biorthogonalisationsmethode von Lanczos mit Look-ahead (LABiO) und die Biorthogonalisations- und Bikonjugationsmethode von Lanczos mit Look-ahead (LABiOC) auf den Blockfall verallgemeinern.

Im zweiten Schritt werden dann iterative Löser konstruiert, die auf unserem Block-Lanczos-Prozess aufbauen. Dies wird für QMR und BiCG im zweiten Teil der Arbeit durchgeführt. Im Fall von QMR wird nur eine Version entwickelt, die auf unserer Blockversion von LABiO aufbaut, und bei BiCG werden wir Blockvarianten von BiORes und BiOMin betrachten. Am Schluss dieses Teils werden numerische Experimente durchgeführt, um die neuen Algorithmen zu testen.

Dank

Diese Arbeit wurde unter Aufsicht von Prof. Martin Gutknecht am Seminar für angewandte Mathematik der ETH Zürich ausgeführt. Ihm möchte ich für die angenehme Zusammenarbeit ebenso danken wie für die Ideen, die er zu dieser Arbeit beigesteuert hat. Prof. R. Jeltsch danke ich für die Übernahme des Korreferates. Schliesslich möchte ich mich auch bei meinen Eltern bedanken, dass sie mir die Ausbildung ermöglicht haben.

Contents

0 Introduction

Assume that we are given a set of linear equations

$$\mathbf{AX} = \mathbf{B}$$

with a nonhermitian, nonsingular coefficient matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$ and multiple right-hand sides stored in $\mathbf{B} \in \mathbb{C}^{N \times s}$ that are all known in advance. If the matrix $\mathbf{A}$ is large and sparse, we may choose to use an iterative method to solve these equations. In this work, we try to combine several known ideas in a new way in order to obtain fast and reliable block iterative methods. To describe the ideas behind our work, we take a short look at the evolution of block Lanczos algorithms, thereby concentrating on points related to our task.

Block Lanczos algorithms were first considered in the 1970s for the computation of eigenvalues of Hermitian matrices (see [1, Section 1.3] and the references given there). In 1979 Ruhe [18] published his band Lanczos algorithm which includes a vectorwise computation of the Lanczos vectors as well as a proper deflation checking procedure. In the nonhermitian case there is not only the need for deflation but also the need to treat (near) breakdowns. Also, deflation may occur independently of each other in the left and right block Krylov spaces. Therefore it is not surprising that the development of algorithms which address these problems took much more time. The ABLE (adaptive block Lanczos method for nonhermitian eigenvalues) method by Bai, Day and Ye [2] is an important contribution in this context. In this algorithm, the blocksize is adapted to be at least equal to the order of multiple or clustered eigenvalues and it is increased if a (near) breakdown occurs. The ABLE method uses block operations throughout and therefore achieves one of our most important goals. On the other hand, its handling of deflation and breakdowns requires that all the Lanczos vectors be stored and is therefore not suited for the solution of linear equations.

The first block method for solving linear equations with nonhermitian matrix seems to be O'Leary's block biconjugate gradients algorithm [17] where deflations lead to a restart and breakdowns were not treated. A major progress was achieved by Aliaga, Boley, Freund and Hernandez by developing a block version of the unsymmetric Lanczos process [1] which includes proper deflation and look-ahead procedures. Additionally, the number of right and left starting vectors does not have to be the same, and there is no need to store all the Lanczos vectors. On the basis of this algorithm, Freund and Malhotra subsequently constructed a block QMR method [8]. The drawback of the algorithm from [1], however, is that it uses a vectorwise computation of the bases of the block Krylov spaces (as in Ruhe's algorithm), and although the matrix vector products could still be computed blockwise, other operations like orthogonalizations of new vectors have to be done vectorwise. This becomes even more pronounced in the block QMR method from [8] where new solutions are obtained after every extension of the block Krylov spaces so that the QR factorization has to be updated vectorwise. Consequently, these algorithms lead to worse performance compared with a completely blockwise algorithm.

So the question arises whether it is possible to develop a completely blockwise algorithm which is still reliable, i.e. handles deflation and look-ahead in a satisfactory way and does not involve long recursions. It is the goal of this thesis to give a positive answer to this question. To this end, we first develop a suitable block version of the unsymmetric Lanczos process. Instead of aiming at vectorwise biorthogonality as in [1], we opt for blockwise biorthogonality so that the inner product matrix will usually be a blockdiagonal matrix even if no look-ahead occurs. The sizes of the right and left clusters are determined by the sizes of the blocks in the inner product matrix and are not related in any way to the sizes of the right and left blocks. We are therefore able to formulate our block Lanczos algorithm in such a way that the cluster size can be chosen according to aspects like performance and numerical stability. In an implementation, a parameter can be introduced which controls the way the cluster sizes are determined so that different variants can be tried out easily. The result of the first part are two algorithms which are generalizations of the look-ahead Lanczos biorthogonalization method (LABiO) and the look-ahead Lanczos biorthogonalization and biconjugation method (LABiOC) to the block case.

The second part of this work is devoted to the construction of three Lanczos process based block Krylov space methods. We first deal with QMR and develop a block variant based on our block

version of LABIO. Then we consider block BICG and develop block variants of BLBIORES and BLBIOMIN. The question of how to avoid the pivot breakdown is also addressed. At the end of this part, some numerical experiments are presented.

The sections of this thesis reflect the scheme outlined above: Section 1 describes those algorithms which will later be generalized to the block case. We only consider the simplest case of one single pair of starting vectors and without look-ahead. Section 2 is devoted to the construction of our block versions of LABIO and LABIOC, and Section 3 to the construction of the three block Krylov space methods mentioned above.

1 Overview of some Lanczos process based Krylov space methods

In this section, we describe in their simplest form those algorithms which will appear later in this work, i.e. in the nonblock case and without look-ahead. We closely follow [13], where more details may be found. The standard inner product on $\mathbb{C}^N$ will be denoted by $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\star \mathbf{y}$ and the hermitian conjugate of $\mathbf{M} \in \mathbb{C}^{n \times m}$ will be written as $\mathbf{M}^\star = \overline{\mathbf{M}^\top}$. We also introduce the convention that the statement "line $n.i$" refers to line i of Algorithm n.

1.1 The BIO algorithm

The *Lanczos biorthogonalization* (BIO) *algorithm* [14], often referred to as the *unsymmetric* or *two-sided Lanczos algorithm*, is a process for generating two finite sequences $\{\mathbf{y}_n\}_{n=0}^{\nu-1}$ and $\{\widetilde{\mathbf{y}}_n\}_{n=0}^{\nu-1}$, whose length ν depends on $\mathbf{A}$, $\mathbf{b}$, $\mathbf{y}_0$, and $\widetilde{\mathbf{y}}_0$, such that, for $m, n = 0, 1, \ldots, \nu - 1$,

$$\begin{aligned} \mathbf{y}_n &\in \mathcal{K}_{n+1} := \operatorname{span}(\mathbf{y}_0, \mathbf{A}\mathbf{y}_0, \ldots, \mathbf{A}^n \mathbf{y}_0), \\ \widetilde{\mathbf{y}}_m &\in \widetilde{\mathcal{K}}_{m+1} := \operatorname{span}(\widetilde{\mathbf{y}}_0, \mathbf{A}^\star \widetilde{\mathbf{y}}_0, \ldots, (\mathbf{A}^\star)^m \widetilde{\mathbf{y}}_0) \end{aligned} \tag{1.1}$$

and

$$\langle \widetilde{\mathbf{y}}_m, \mathbf{y}_n \rangle = \begin{cases} 0 & \text{if} \quad m \neq n, \\ \delta_n \neq 0 & \text{if} \quad m = n. \end{cases} \tag{1.2}$$

$\mathcal{K}_n$ and $\widetilde{\mathcal{K}}_m$ are *Krylov (sub)spaces* of $\mathbf{A}$ and $\mathbf{A}^\star$, respectively. The condition (1.2) means that the sequences are *biorthogonal*. Their elements $\mathbf{y}_n$ and $\widetilde{\mathbf{y}}_m$ are called *right* and *left Lanczos vectors*, respectively.

The biorthogonal sequences of Lanczos vectors are constructed by a two-sided version of the well-known Gram–Schmidt process, but the latter is not applied to the bases used in the definition (1.1), since these are normally close to linearly dependent. Instead, the vectors that are orthogonalized are of the form $\mathbf{A}\mathbf{y}_n$ and $\mathbf{A}^\star \widetilde{\mathbf{y}}_n$; i.e. they are created in each step from the most recently constructed pair by multiplication with $\mathbf{A}$ and $\mathbf{A}^\star$, respectively. It follows that

$$\mathbf{y}_n \in \mathcal{K}_{n+1} \ominus \mathcal{K}_n\,, \qquad \widetilde{\mathbf{y}}_n \in \widetilde{\mathcal{K}}_{n+1} \ominus \widetilde{\mathcal{K}}_n, \tag{1.3}$$

and thus

$$\mathcal{K}_{n+1} = \operatorname{span}(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n)\,, \qquad \widetilde{\mathcal{K}}_{n+1} = \operatorname{span}(\widetilde{\mathbf{y}}_0, \widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_n)\,.$$

The length ν of the sequences is determined by the impossibility of extending them such that the conditions (1.1) and (1.2) still hold with $\delta_n \neq 0$.

Algorithm 1 gives a detailed statement of the BIO algorithm. The coefficients α_n, $\widetilde{\alpha}_n$, β_{n-1}, $\widetilde{\beta}_{n-1}$, γ_n, $\widetilde{\gamma}_n$, and δ_{n+1} are defined uniquely except for the freedom in line 1.16 for choosing two of the three quantities γ_n, $\widetilde{\gamma}_n$, and δ_{n+1}. A popular choice is

$$\gamma_n := \|\mathbf{y}_{\text{tmp}}\|\,, \qquad \widetilde{\gamma}_n := \|\widetilde{\mathbf{y}}_{\text{tmp}}\|\,, \qquad \delta_{n+1} := \delta_{\text{tmp}}/(\gamma_n \widetilde{\gamma}_n), \tag{1.4}$$

which leads to normalized Lanczos vectors at the cost of two additional inner products. The recursions for $\mathbf{y}_{n+1}$ and $\widetilde{\mathbf{y}}_{n+1}$ both have three terms:

$$\begin{aligned} \mathbf{y}_{n+1}\gamma_n &= \mathbf{A}\mathbf{y}_n - \mathbf{y}_n \alpha_n - \mathbf{y}_{n-1}\beta_{n-1}\,, \\ \widetilde{\mathbf{y}}_{n+1}\widetilde{\gamma}_n &= \mathbf{A}^\star \widetilde{\mathbf{y}}_n - \widetilde{\mathbf{y}}_n \widetilde{\alpha}_n - \widetilde{\mathbf{y}}_{n-1}\widetilde{\beta}_{n-1}\,. \end{aligned} \tag{1.5}$$

Algorithm 1 Biorthogonalization (BiO) algorithm

1: Choose $\mathbf{y}_0$, $\widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle \neq 0$.
2: $\mathbf{y}_{-1} := \mathbf{0}$, $\widetilde{\mathbf{y}}_{-1} := \mathbf{0}$, $\beta_{-1} := 0$, $\widetilde{\beta}_{-1} := 0$
3: **for** $n = 0, 1, \ldots$ **do**
4: $\quad \alpha_n := \langle \widetilde{\mathbf{y}}_n, \mathbf{A}\mathbf{y}_n \rangle / \delta_n$
5: $\quad \widetilde{\alpha}_n := \overline{\alpha_n}$
6: $\quad \beta_{n-1} := \overline{\widetilde{\gamma}_{n-1}} \delta_n / \delta_{n-1}$ $\qquad$ (if $\quad n > 0$)
7: $\quad \widetilde{\beta}_{n-1} := \gamma_{n-1} \delta_n / \delta_{n-1} = \beta_{n-1} \gamma_{n-1} / \widetilde{\gamma}_{n-1}$ $\qquad$ (if $\quad n > 0$)
8: $\quad \mathbf{y}_{\mathrm{tmp}} := \mathbf{A}\mathbf{y}_n - \mathbf{y}_n \alpha_n - \mathbf{y}_{n-1} \beta_{n-1}$
9: $\quad \widetilde{\mathbf{y}}_{\mathrm{tmp}} := \mathbf{A}^\star \widetilde{\mathbf{y}}_n - \widetilde{\mathbf{y}}_n \widetilde{\alpha}_n - \widetilde{\mathbf{y}}_{n-1} \widetilde{\beta}_{n-1}$
10: $\quad \delta_{\mathrm{tmp}} := \langle \widetilde{\mathbf{y}}_{\mathrm{tmp}}, \mathbf{y}_{\mathrm{tmp}} \rangle$
11: $\quad$ **if** $\delta_{\mathrm{tmp}} = 0$ **then**
12: $\quad\quad$ Choose $\gamma_n \neq 0$ and $\widetilde{\gamma}_n \neq 0$
13: $\quad\quad$ Set $\nu := n + 1$, $y_\nu := \mathbf{y}_{\mathrm{tmp}} / \gamma_n$, $\widetilde{y}_\nu := \widetilde{\mathbf{y}}_{\mathrm{tmp}} / \widetilde{\gamma}_n$ and $\delta_{n+1} := 0$.
14: $\quad\quad$ **stop**
15: $\quad$ **end if**
16: $\quad$ Choose $\gamma_n \neq 0$, $\widetilde{\gamma}_n \neq 0$, and δ_{n+1} such that $\gamma_n \overline{\widetilde{\gamma}_n} \delta_{n+1} = \delta_{\mathrm{tmp}}$.
17: $\quad$ Set $\mathbf{y}_{n+1} := \mathbf{y}_{\mathrm{tmp}} / \gamma_n$, $\widetilde{\mathbf{y}}_{n+1} := \widetilde{\mathbf{y}}_{\mathrm{tmp}} / \widetilde{\gamma}_n$.
18: **end for**

The BiO algorithm stops in line 1.14 if $\delta_{\mathrm{tmp}} = 0$. If $\mathbf{y}_{\mathrm{tmp}} = \mathbf{0}$ or $\widetilde{\mathbf{y}}_{\mathrm{tmp}} = \mathbf{0}$ or both, then we have a *right*, *left* or *full termination*, respectively. It may also happen, however, that $\delta_{\mathrm{tmp}} = 0$ but $\mathbf{y}_{\mathrm{tmp}} \neq \mathbf{0}$ and $\widetilde{\mathbf{y}}_{\mathrm{tmp}} \neq \mathbf{0}$; this is referred to as a *Lanczos breakdown*.

For $n \leq \nu$, let us introduce the $N \times n$ matrices

$$\mathbf{Y}_n := \begin{bmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_{n-1} \end{bmatrix}, \qquad \widetilde{\mathbf{Y}}_n := \begin{bmatrix} \widetilde{\mathbf{y}}_0 & \widetilde{\mathbf{y}}_1 & \cdots & \widetilde{\mathbf{y}}_{n-1} \end{bmatrix}, \tag{1.6}$$

the diagonal matrices

$$\mathbf{D}_{\delta;n} := \mathrm{diag}(\delta_0, \delta_1, \ldots, \delta_{n-1}),$$

and the $n \times n$ tridiagonal matrices

$$\underline{\mathbf{T}}_n := \begin{bmatrix} \mathbf{T}_n \\ \hline \gamma_{n-1} \mathbf{l}_n^\top \end{bmatrix} = \begin{bmatrix} \alpha_0 & \beta_0 & & & \\ \gamma_0 & \alpha_1 & \beta_1 & & \\ & \gamma_1 & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \beta_{n-2} \\ & & & \gamma_{n-2} & \alpha_{n-1} \\ & & & & \gamma_{n-1} \end{bmatrix},$$

where $\mathbf{l}_n = (0, \ldots, 0, 1)^\top \in \mathbb{R}^n$, and the analogous matrices with tildes. Then we have altogether

$$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1} \underline{\mathbf{T}}_n, \qquad \mathbf{A}^\star \widetilde{\mathbf{Y}}_n = \widetilde{\mathbf{Y}}_{n+1} \underline{\widetilde{\mathbf{T}}}_n \qquad (n \leq \nu), \tag{1.7}$$

$$\widetilde{\mathbf{Y}}_n^\star \mathbf{Y}_n = \mathbf{D}_{\delta;n}, \qquad \widetilde{\mathbf{Y}}_n^\star \mathbf{A} \mathbf{Y}_n = \mathbf{D}_{\delta;n} \mathbf{T}_n \qquad (n \leq \nu),$$

$$\mathbf{D}_{\delta;n} \mathbf{T}_n = \widetilde{\mathbf{T}}_n^\star \mathbf{D}_{\delta;n} \qquad (n \leq \nu).$$

Another popular algorithm for computing nested bases of $\mathcal{K}_n$ for $n = 1, 2, \ldots$ is the *Arnoldi algorithm*. It differs from the BiO algorithm in that it computes an orthogonal basis for $\mathcal{K}_n$ using the modified Gram-Schmidt procedure. Consequently, it does not need left Krylov spaces and only one matrix vector product is needed at each step while BiO requires an additional product with $\mathbf{A}^\star$. On the other hand, the Arnoldi recurrence for $\mathbf{y}_n$ generally involves all previous vectors so that the construction of an orthogonal basis of the Krylov space becomes expensive and memory-intensive. The Arnoldi process is the basis of the GMRes algorithm.

1.2 The BIORES form of the BICG method

Using general principles [11] it is easy to see how the Lanczos BIO algorithm can be applied to the problem of solving linear systems of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{C}^{N\times N}$ is nonsingular.

Here we discuss first an algorithm introduced by Young and Jea [22]. It was originally called Lanczos/Orthores and renamed BIORES in [12]. It is one of several forms of the biconjugate gradients (BICG) method, which is defined by requiring that iterates $\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K}_n$ satisfy the *Petrov–Galerkin condition*

$$\langle \widetilde{\mathbf{y}}, \mathbf{r}_n \rangle = 0 \quad (\forall \widetilde{\mathbf{y}} \in \widetilde{\mathcal{K}}_n), \qquad \text{i.e.} \quad \widetilde{\mathcal{K}}_n \perp \mathbf{r}_n \,. \tag{1.8}$$

The most straightforward way of taking into account the two conditions $\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K}_n$ and $\mathbf{r}_n \perp \widetilde{\mathcal{K}}_n$ is the following one. Representing $\mathbf{x}_n - \mathbf{x}_0$ in terms of the Lanczos vectors we can write

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Y}_n \mathbf{k}_n \,, \qquad \mathbf{r}_n = \mathbf{r}_0 - \mathbf{A}\mathbf{Y}_n \mathbf{k}_n \,, \tag{1.9}$$

with some coordinate vector $\mathbf{k}_n$. Using $\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n$, see (1.7), and

$$\mathbf{r}_0 = \mathbf{y}_0 \rho_0 = \mathbf{Y}_{n+1} \underline{\mathbf{e}}_1 \rho_0 \,,$$

with $\underline{\mathbf{e}}_1 := \begin{bmatrix} 1 & 0 & 0 & \cdots \end{bmatrix} \in \mathbb{R}^{n+1}$ and $\rho_0 := \|\mathbf{r}_0\|$ (assuming $\|\mathbf{y}_0\| = 1$ here), we find that

$$\mathbf{r}_n = \mathbf{Y}_{n+1} \left(\underline{\mathbf{e}}_1 \rho_0 - \underline{\mathbf{T}}_n \mathbf{k}_n \right). \tag{1.10}$$

In view of $\widetilde{\mathbf{Y}}_n^\star \mathbf{Y}_{n+1} = \begin{bmatrix} \mathbf{D}_{\delta;n} & \mathbf{0} \end{bmatrix}$, the Petrov–Galerkin condition (1.8), which may be written as $\widetilde{\mathbf{Y}}_n^\star \mathbf{r}_n = \mathbf{0}$, finally yields the square tridiagonal linear system

$$\mathbf{T}_n \mathbf{k}_n = \mathbf{e}_1 \rho_0 \,,$$

where now $\mathbf{e}_1 \in \mathbb{R}^n$. By solving it for $\mathbf{k}_n$ and inserting the solution into (1.9) we could compute $\mathbf{x}_n$. However, this approach is very memory-intensive, as one has to store all right Lanczos vectors for evaluating $\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Y}_n \mathbf{k}_n$. We note that this approach is analogous to the full orthogonalization method (FOM). Fortunately, there are more efficient versions of the BICG method that generate not only the residuals (essentially the right Lanczos vectors) but also the iterates with short recurrences. One such algorithm is BIORES, see Algorithm 2. Note that this is the so-called *consistent* version of BIORES, which means that $\mathbf{y}_n$ is the nth residual: $\mathbf{y}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$.

Algorithm 2 BIORES form of the BICG method

Choose an initial approximation $\mathbf{x}_0$, set $\mathbf{y}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, and choose $\widetilde{\mathbf{y}}_0$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle \neq 0$. Apply Algorithm 1 (BIO) with $\gamma_n := -\alpha_n - \beta_{n-1}$ and some $\widetilde{\gamma}_n \neq 0$, so that 1.8–1.17 simplify to:

$$\begin{aligned} \mathbf{y}_{n+1} &:= (\mathbf{A}\mathbf{y}_n - \mathbf{y}_n \alpha_n - \mathbf{y}_{n-1}\beta_{n-1})/\gamma_n \\ \widetilde{\mathbf{y}}_{n+1} &:= (\mathbf{A}^\star \widetilde{\mathbf{y}}_n - \widetilde{\mathbf{y}}_n \widetilde{\alpha}_n - \widetilde{\mathbf{y}}_{n-1}\widetilde{\beta}_{n-1})/\widetilde{\gamma}_n \\ \delta_{n+1} &:= \langle \widetilde{\mathbf{y}}_{n+1}, \mathbf{y}_{n+1} \rangle \end{aligned}$$

Additionally, compute $\mathbf{x}_{n+1} := -(\mathbf{y}_n + \mathbf{x}_n \alpha_n + \mathbf{x}_{n-1}\beta_{n-1})/\gamma_n$.
If $\gamma_n = 0$, the algorithm breaks down ("pivot breakdown"), and we set $\dot{\nu} := n$. If $\mathbf{y}_{n+1} = 0$, it terminates and $\mathbf{x}_{n+1}$ is the solution; if $\mathbf{y}_{n+1} \neq 0$, but $\delta_{n+1} = 0$, the algorithm breaks also down ("Lanczos breakdown" if $\widetilde{\mathbf{y}}_{n+1} \neq 0$, "left termination" if $\widetilde{\mathbf{y}}_{n+1} = 0$). In these two cases we set $\dot{\nu} := n + 1$.

1.3 The QMR method

The basic version of the QMR (quasi minimal residual) method of Freund and Nachtigal [9] without look-ahead is obtained as follows: the BIO algorithm with normalized Lanczos vectors (i.e. with normalization (1.4)) is applied to build up bases of the growing Krylov spaces $\mathcal{K}_n$ and $\widetilde{\mathcal{K}}_n$. The right

initial vector $\mathbf{y}_0 := \mathbf{r}_0/\|\mathbf{r}_0\|$ is the normalized initial residual, while the left one, $\widetilde{\mathbf{y}}_0$ can be chosen arbitrarily. The relations (1.9)–(1.10) remain valid, but since finding the minimum residual becomes too expensive, the so-called *quasi-residual*

$$\mathbf{q}_n := \underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}_n\mathbf{k}_n\,, \qquad \text{satisfying} \quad \mathbf{r}_n = \mathbf{Y}_{n+1}\mathbf{q}_n\,, \tag{1.11}$$

(see (1.10)) is minimized instead. Let $\underline{\mathbf{T}}_n = \mathbf{Q}_n\underline{\mathbf{R}}_n^{\mathrm{MR}}$ be a QR decomposition of $\underline{\mathbf{T}}_n$. The last row of the upper triangular $(n+1)\times n$ matrix $\underline{\mathbf{R}}_n^{\mathrm{MR}}$ is zero. If we denote its upper square $n\times n$ submatrix by $\mathbf{R}_n^{\mathrm{MR}}$ and let

$$\underline{\mathbf{h}}_n := \left[\begin{array}{c}\mathbf{h}_n\\ \hline \widetilde{\eta}_{n+1}\end{array}\right] := \mathbf{Q}_n^\star\underline{\mathbf{e}}_1\rho_0\,, \tag{1.12}$$

we see that

$$\mathbf{k}_n := (\mathbf{R}_n^{\mathrm{MR}})^{-1}\mathbf{h}_n \tag{1.13}$$

is the solution of our least squares problem since

$$\begin{aligned}\|\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}_n\mathbf{k}_n\|^2 &= \|\mathbf{Q}_n^\star\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{R}}_n^{\mathrm{MR}}\mathbf{k}_n\|^2\\ &= \|\underline{\mathbf{h}}_n - \underline{\mathbf{R}}_n^{\mathrm{MR}}\mathbf{k}_n\|^2 \qquad (1.14)\\ &= \|\mathbf{h}_n - \mathbf{R}_n^{\mathrm{MR}}\mathbf{k}_n\|^2 + |\widetilde{\eta}_{n+1}|^2\\ &= |\widetilde{\eta}_{n+1}|^2. \qquad (1.15)\end{aligned}$$

In fact, multiplying the least squares problem (1.11) by the unitary matrix $\mathbf{Q}_n^\star$ turns it into one with an upper triangular matrix, see (1.14), where the choice of $\mathbf{k}_n$ no longer influences the defect of the last equation, and thus the problem is solved by choosing $\mathbf{k}_n$ such that the first n equations are fulfilled.

From (1.11) and (1.15) we see in particular that the minimum norm of the quasi-residual is equal to $|\widetilde{\eta}_{n+1}|$ and hence can be found without computing $\mathbf{k}_n$ or the quasi residual. The unitary matrix $\mathbf{Q}_n$ is only determined in its factored form, as the product of n Givens rotations that are chosen to annihilate the subdiagonal elements of the tridiagonal (or Hessenberg) matrix:

$$\mathbf{Q}_n := \left[\begin{array}{c|c}\mathbf{Q}_{n-1} & \mathbf{0}\\ \hline \mathbf{0}^\top & 1\end{array}\right]\mathbf{G}_n \qquad \text{with} \quad \mathbf{G}_n := \left[\begin{array}{c|cc}\mathbb{1}_{n-1} & \mathbf{0} & \mathbf{0}\\ \hline \mathbf{0}^\top & c_n & -s_n\\ \mathbf{0}^\top & \overline{s_n} & c_n\end{array}\right], \tag{1.16}$$

where $c_n \geq 0$ and $s_n \in \mathbb{C}^n$ satisfying $c_n^2 + |s_n^2| = 1$ are chosen such that

$$\mathbf{G}_n^\star\begin{bmatrix}\star\\ \vdots\\ \star\\ \mu_n\\ \nu_n\end{bmatrix} = \begin{bmatrix}\star\\ \vdots\\ \star\\ c_n\mu_n + s_n\nu_n\\ 0\end{bmatrix} \quad \text{with} \quad \begin{bmatrix}\star\\ \vdots\\ \star\\ \mu_n\\ \nu_n\end{bmatrix} := \left[\begin{array}{c|c}\mathbf{Q}_{n-1}^\star & \mathbf{0}\\ \hline \mathbf{0}^\top & 1\end{array}\right]\underline{\mathbf{T}}_n\begin{bmatrix}0\\ \vdots\\ 0\\ 1\end{bmatrix},$$

which means that

$$c_n := \frac{|\mu_n|}{\sqrt{|\mu_n|^2 + |\nu_n|^2}}, \quad s_n := \overline{c_n\frac{\nu_n}{\mu_n}}, \qquad \text{if } \mu_n \neq 0,$$

$$c_n := 0, \qquad\qquad s_n := 1, \qquad\qquad \text{if } \mu_n = 0.$$

If $\underline{\mathbf{T}}_n$ is real, c_n and s_n are the cosine and sine of the rotation angle.

The formula for updating $\underline{\mathbf{h}}_n$ is therefore very simple:

$$\left[\begin{array}{c}\mathbf{h}_n\\ \hline \widetilde{\eta}_{n+1}\end{array}\right] = \underline{\mathbf{h}}_n = \mathbf{G}_n^\star\left[\begin{array}{c}\underline{\mathbf{h}}_{n-1}\\ \hline 0\end{array}\right] = \mathbf{G}_n^\star\left[\begin{array}{c}\mathbf{h}_{n-1}\\ \hline \widetilde{\eta}_n\\ 0\end{array}\right] = \left[\begin{array}{c}\mathbf{h}_{n-1}\\ \hline c_n\,\widetilde{\eta}_n\\ -\overline{s_n}\,\widetilde{\eta}_n\end{array}\right]. \tag{1.17}$$

In particular, it follows that

$$\|\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}_n\mathbf{k}_n\| = |\widetilde{\eta}_{n+1}| = |s_n\widetilde{\eta}_n| = |s_1\, s_2 \cdots s_n|\, \|\mathbf{r}_0\|,$$

since $\widetilde{\eta}_1 = \|\mathbf{r}_0\|$. Even more important is the fact that $\mathbf{h}_n \in \mathbb{C}^n$ emerges from $\mathbf{h}_{n-1} \in \mathbb{C}^{n-1}$ by just appending an additional component $c_n\widetilde{\eta}_n$. By rewriting the first equation in (1.9) using (1.13) as

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Z}_n\mathbf{h}_n\,, \qquad \text{where} \quad \mathbf{Z}_n := [\ \mathbf{z}_0 \quad \ldots \quad \mathbf{z}_{n-1}\ ] := \mathbf{Y}_n(\mathbf{R}_n^{\mathrm{MR}})^{-1}$$

contains the QMR direction vectors, we can conclude that

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{z}_{n-1}c_n\widetilde{\eta}_n\,. \tag{1.18}$$

Finally, since $\mathbf{R}_n^{\mathrm{MR}}$ is a banded upper tridiagonal matrix with bandwidth three, the relation

$$\mathbf{Y}_n = \mathbf{R}_n^{\mathrm{MR}}\mathbf{Z}_n \tag{1.19}$$

can be viewed as the matrix representation of a three-term recurrence for generating the vectors $\{\mathbf{z}_k\}_{k=0}^{n-1}$.

Multiplying (1.18) by $\mathbf{A}$ we could find an analogous recurrence for the residuals, but since it would require an extra matrix-vector product, it is of no interest. There is another, cheaper way of updating the residual. First, inserting $\underline{\mathbf{T}}_n = \mathbf{Q}_n\underline{\mathbf{R}}_n^{\mathrm{MR}}$ and (1.13) into (1.10) and taking (1.12) into account we get

$$\begin{aligned}\mathbf{r}_n &= \mathbf{Y}_{n+1}\left(\underline{\mathbf{e}}_1\rho_0 - \mathbf{Q}_n\underline{\mathbf{R}}_n^{\mathrm{MR}}(\mathbf{R}_n^{\mathrm{MR}})^{-1}\mathbf{h}_n\right) = \mathbf{Y}_{n+1}\left(\underline{\mathbf{e}}_1\rho_0 - \mathbf{Q}_n\begin{bmatrix}\mathbf{h}_n\\ 0\end{bmatrix}\right)\\ &= \mathbf{Y}_{n+1}\mathbf{Q}_n\,\mathbf{l}_{n+1}\widetilde{\eta}_{n+1}\,, \qquad \text{where} \quad \mathbf{l}_{n+1} = [\ 0 \quad \ldots \quad 0 \quad 1\ ]^\top \in \mathbb{R}^{n+1}\end{aligned} \tag{1.20}$$

as before. Using (1.16) we conclude further that

$$\begin{aligned}\mathbf{r}_n &= \left[\ \mathbf{Y}_n \,\middle|\, \mathbf{y}_n\ \right]\left[\begin{array}{c|c}\mathbf{Q}_{n-1} & \mathbf{0}\\ \hline \mathbf{0}^\top & 1\end{array}\right]\mathbf{G}_n\begin{bmatrix}\mathbf{0}\\ 1\end{bmatrix}\widetilde{\eta}_{n+1}\\ &= -\mathbf{Y}_n\mathbf{Q}_{n-1}\mathbf{l}_n s_n\widetilde{\eta}_{n+1} + \mathbf{y}_n c_n\widetilde{\eta}_{n+1}\,.\end{aligned}$$

Finally, using (1.20) and $\widetilde{\eta}_{n+1} = -\overline{s_n}\,\widetilde{\eta}_n$ (see (1.17)) to simplify the first term on the right-hand side, we get the recursion

$$\mathbf{r}_n = \mathbf{r}_{n-1}|s_n|^2 + \mathbf{y}_n c_n\widetilde{\eta}_{n+1}\,. \tag{1.21}$$

In general, $\|\mathbf{r}_n\| = \|\mathbf{q}_n\|$ does not hold, instead we just have

$$\|\mathbf{r}_n\| \le \|\mathbf{Y}_{n+1}\|\,\|\mathbf{q}_n\| \le \sqrt{n+1}\,|\widetilde{\eta}_{n+1}|\,,$$

since $\mathbf{Y}_{n+1}$ has columns of length 1, and $\|\mathbf{q}_n\| = |\widetilde{\eta}_{n+1}|$ as before; see (1.15). The factor $\sqrt{n+1}$ normally leads to a large overestimate, so that the bound is of limited value. However, the relationship

---

Algorithm 3 BIOQMR version of the QMR method

For solving $\mathbf{Ax} = \mathbf{b}$, choose an initial approximation $\mathbf{x}_0 \in \mathbb{C}^N$.
Let $\mathbf{r}_0 := (\mathbf{b} - \mathbf{Ax}_0)$ and $\mathbf{y}_0 := \mathbf{r}_0/\|\mathbf{r}_0\|$, choose $\widetilde{\mathbf{y}}_0$ of unit length.
Apply Algorithm 1 (BIO) with the option (1.4) producing normalized Lanczos vectors.
Within step $n-1$ of the main loop, after generating $\mathbf{y}_n$ and $\widetilde{\mathbf{y}}_n$,

(i) update the QR factorization $\underline{\mathbf{T}}_n = \mathbf{Q}_n\underline{\mathbf{R}}_n^{\mathrm{MR}}$.
(ii) compute the coefficient vector $\mathbf{h}_n$ by appending the component $c_n\widetilde{\eta}_n$ to $\mathbf{h}_{n-1}$, and compute the new last component $\widetilde{\eta}_{n+1} := -\overline{s_n}\,\widetilde{\eta}_n$ of $\underline{\mathbf{h}}_n$,
(iii) compute $\mathbf{z}_{n-1}$ according to the three-term recurrence implied by (1.19),
(iv) compute $\mathbf{x}_n$ and $\mathbf{r}_n$ according to (1.18) and (1.21), respectively,
(v) stop if $\|\mathbf{r}_n\|/\|\mathbf{r}_0\|$ is sufficiently small.

---

between the residual and the quasi-residual may suggest sparing the work for updating the residual and computing its norm until the norm $|\widetilde{\eta}_{n+1}|$ of the quasi-residual has dropped below a certain tolerance. In Algorithm 3, which gives a (simplified) version of the QMR method, we nevertheless assume that the residual is updated. We choose to call it BIOQMR for distinction, to indicate that it is based on the BIO algorithm without look-ahead, whose results are then piped into the QMR least squares process.

1.4 The BIOC algorithm

While the BIO algorithm for the nonsymmetric Lanczos process described in subsection 1.1 is based on a three-term recurrence we turn now to another algorithm based on a coupled pair of two-term recurrences for the same process. In addition to the pair of biorthogonal Krylov space bases, a second pair of biconjugate (i.e. $\mathbf{A}$-biorthogonal) bases for the same Krylov space is now generated. That is why we use here the acronym BIOC for this algorithm.

The formulas in lines 4.7–4.8 and 4.12–4.13 are known as *coupled two-term recurrences*. By eliminating $\mathbf{v}_n$ and $\widetilde{\mathbf{v}}_m$ from them we may get back to the three-term recurrences (1.5) of the BIO algorithm.

Algorithm 4 BIOC

1: Choose $\mathbf{y}_0, \widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle \neq 0$ and $\delta'_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\mathbf{y}_0 \rangle \neq 0$.
2: $\mathbf{v}_0 := \mathbf{y}_0$, $\widetilde{\mathbf{v}}_0 := \widetilde{\mathbf{y}}_0$.
3: **for** $n = 0, 1, \ldots$ **do**
4: Choose $\gamma_n \neq 0$, $\widetilde{\gamma}_n \neq 0$.
5: $\varphi_n := \delta'_n / \delta_n$
6: $\widetilde{\varphi}_n := \overline{\varphi_n}$
7: $\mathbf{y}_{n+1} := (\mathbf{A}\mathbf{v}_n - \mathbf{y}_n \varphi_n)/\gamma_n$
8: $\widetilde{\mathbf{y}}_{n+1} := (\mathbf{A}^\star \widetilde{\mathbf{v}}_n - \widetilde{\mathbf{y}}_n \widetilde{\varphi}_n)/\widetilde{\gamma}_n$
9: $\delta_{n+1} := \overline{\langle \widetilde{\mathbf{y}}_{n+1}, \mathbf{y}_{n+1} \rangle}$
10: $\psi_n := \overline{\widetilde{\gamma}_n \delta_{n+1} / \delta'_n}$
11: $\widetilde{\psi}_n := \gamma_n \delta_{n+1} / \delta'_n$
12: $\mathbf{v}_{n+1} := \mathbf{y}_{n+1} - \mathbf{v}_n \psi_n$
13: $\widetilde{\mathbf{v}}_{n+1} := \widetilde{\mathbf{y}}_{n+1} - \widetilde{\mathbf{v}}_n \widetilde{\psi}_n$
14: $\delta'_{n+1} := \langle \widetilde{\mathbf{v}}_{n+1}, \mathbf{A}\mathbf{v}_{n+1} \rangle$
15: **if** $\delta_{n+1} = 0$ or $\delta'_{n+1} = 0$ **then**
16: $\dot{\nu} := n + 1$
17: **stop**
18: **end if**
19: **end for**

It can be shown (see [13, Theorem 7.1]) that the sequences $\{\mathbf{y}_n\}_{n=0}^{\dot{\nu}}$ and $\{\widetilde{\mathbf{y}}_n\}_{n=0}^{\dot{\nu}}$ generated by the BIOC algorithm are biorthogonal, and the sequences $\{\mathbf{v}_n\}_{n=0}^{\dot{\nu}}$ and $\{\widetilde{\mathbf{v}}_n\}_{n=0}^{\dot{\nu}}$ are biconjugate (with respect to $\mathbf{A}$) except that $\langle \widetilde{\mathbf{y}}_{\dot{\nu}}, \mathbf{y}_{\dot{\nu}} \rangle = 0$ or $\langle \widetilde{\mathbf{v}}_{\dot{\nu}}, \mathbf{A}\mathbf{v}_{\dot{\nu}} \rangle = 0$. That is, for $m, n = 0, 1, \ldots, \dot{\nu}$,

$$\langle \widetilde{\mathbf{y}}_m, \mathbf{y}_n \rangle = \begin{cases} 0\,, & m \neq n, \\ \delta_n\,, & m = n, \end{cases}$$

$$\langle \widetilde{\mathbf{v}}_m, \mathbf{A}\mathbf{v}_n \rangle = \begin{cases} 0\,, & m \neq n, \\ \delta'_n = \delta_n \varphi_n\,, & m = n, \end{cases}$$

where $\delta_n \neq 0$ and $\delta'_n \neq 0$ for $0 \leq n \leq \dot{\nu} - 1$, but $\delta_{\dot{\nu}} = 0$ or $\delta'_{\dot{\nu}} = 0$. Moreover, for $n = 1, \ldots, \dot{\nu} - 1$ holds in addition to (1.3)

$$\mathbf{v}_n \in \mathcal{K}_{n+1} \backslash \mathcal{K}_n\,, \qquad \widetilde{\mathbf{v}}_n \in \widetilde{\mathcal{K}}_{n+1} \backslash \widetilde{\mathcal{K}}_n\,.$$

Algorithm 4 may break down if $\delta_{n+1} = 0$ (Lanczos breakdown) or if $\delta'_{n+1} = 0$ (pivot breakdown).

Next, we turn to the matrix relations related to the BIOC algorithm. In addition to $\mathbf{Y}_n$ and $\widetilde{\mathbf{Y}}_n$ of (1.6) we need the $N \times n$ matrices

$$\mathbf{V}_n := \begin{bmatrix} \mathbf{v}_0 & \mathbf{v}_1 & \cdots & \mathbf{v}_{n-1} \end{bmatrix} \qquad \widetilde{\mathbf{V}}_n := \begin{bmatrix} \widetilde{\mathbf{v}}_0 & \widetilde{\mathbf{v}}_1 & \cdots & \widetilde{\mathbf{v}}_{n-1} \end{bmatrix}$$

the $n \times n$ matrices

$$\mathbf{L}_n := \begin{bmatrix} \varphi_0 & & & & \\ \gamma_0 & \varphi_1 & & & \\ & \gamma_1 & \varphi_2 & & \\ & & \ddots & \ddots & \\ & & & \gamma_{n-2} & \varphi_{n-1} \end{bmatrix}, \qquad \mathbf{U}_n := \begin{bmatrix} 1 & \psi_0 & & & \\ & 1 & \psi_1 & & \\ & & 1 & \ddots & \\ & & & \ddots & \psi_{n-2} \\ & & & & 1 \end{bmatrix},$$

which are lower and upper bidiagonal, respectively, and the *extended* bidiagonal matrices

$$\underline{\mathbf{L}}_n := \left[\begin{array}{c} \mathbf{L}_n \\ \hline \gamma_{n-1}\mathbf{l}_n^\top \end{array}\right] = \begin{bmatrix} \varphi_0 & & & & \\ \gamma_0 & \varphi_1 & & & \\ & \gamma_1 & \varphi_2 & & \\ & & \ddots & \ddots & \\ & & & \gamma_{n-2} & \varphi_{n-1} \\ & & & & \gamma_{n-1} \end{bmatrix}$$

and

$$\underline{\mathbf{U}}_n := \left[\begin{array}{c|c} \mathbf{U}_n & \mathbf{l}_n\psi_{n-1} \end{array}\right] = \begin{bmatrix} 1 & \psi_0 & & & \\ & 1 & \psi_1 & & \\ & & 1 & \ddots & \\ & & & \ddots & \psi_{n-2} \\ & & & & 1 & \psi_{n-1} \end{bmatrix},$$

with an additional row and column, respectively. Analogously, we define $\widetilde{\mathbf{L}}_n$, $\widetilde{\mathbf{U}}_n$, $\underline{\widetilde{\mathbf{L}}}_n$, and $\underline{\widetilde{\mathbf{U}}}_n$ in the obvious way. Then, according to 4.12, 4.13, 4.7 and 4.8,

$$\mathbf{Y}_n = \mathbf{V}_n\mathbf{U}_n\,, \qquad \widetilde{\mathbf{Y}}_n = \widetilde{\mathbf{V}}_n\widetilde{\mathbf{U}}_n \qquad (n \le \dot{\nu})$$

and

$$\mathbf{A}\mathbf{V}_n = \mathbf{Y}_{n+1}\underline{\mathbf{L}}_n\,, \qquad \mathbf{A}^\star\widetilde{\mathbf{V}}_n = \widetilde{\mathbf{Y}}_{n+1}\underline{\widetilde{\mathbf{L}}}_n \qquad (n \le \dot{\nu}).$$

1.5 The BIOMIN form of the BICG method

This algorithm is often just called "the BICG method". It is essentially due to Lanczos [15] but it has been worked out in a more useful form only by Fletcher [6]. We apply Algorithm 4 (BIOC) with $\gamma_n := -\varphi_n$ and $\widetilde{\gamma}_n := -\overline{\varphi_n}$, so that after substituting $\omega_n := 1/\varphi_n$ we obtain Algorithm 5. Here, the vector $\mathbf{y}_n$ is again identical to the nth residual $\mathbf{r}_n$.

2 A block Lanczos process

The basis of this work consists of a version of the unsymmetric Lanczos process which includes deflation and look-ahead. As in the nonblock case, there are several possibilities to realize the idea of the two-sided block Gram-Schmidt process. We will consider the analogues of LABIO and LABIOC, which we call BLBIO and BLBIOC, respectively. In Subsection 2.1, we outline the goal and general properties of the block Lanczos algorithm, more precisely, of the BLBIO variant. Subsection 2.2 then covers BLBIO and its basic properties in detail. The algorithm described in this subsection does contain look-ahead but the choice of a look-ahead strategy is not yet specified. Different realizations based on different choices of look-ahead strategies are possible and will be discussed in Subsection 2.3. The next step is then to consider BLBIOC, see Subsection 2.4, and finally, we present some numerical examples in Subsection 2.5.

Algorithm 5 BIOMIN form of the BICG method

For solving $\mathbf{Ax} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0$ and set $\mathbf{v}_0 := \mathbf{y}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$.
Choose $\widetilde{\mathbf{y}}_0$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle \neq 0$ and $\delta_0' := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\mathbf{v}_0 \rangle \neq 0$.
$\widetilde{\mathbf{v}}_0 := \widetilde{\mathbf{y}}_0$
for $n = 0, 1, \ldots$ **do**
  $\omega_n := \delta_n / \delta_n'$
  $\mathbf{y}_{n+1} := \mathbf{y}_n - \mathbf{A}\mathbf{v}_n \omega_n$
  $\widetilde{\mathbf{y}}_{n+1} := \widetilde{\mathbf{y}}_n - \mathbf{A}^\star \widetilde{\mathbf{v}}_n \overline{\omega_n}$
  $\mathbf{x}_{n+1} := \mathbf{x}_n + \mathbf{v}_n \omega_n$
  $\delta_{n+1} := \langle \widetilde{\mathbf{y}}_{n+1}, \mathbf{y}_{n+1} \rangle$
  $\psi_n := -\delta_{n+1} / \delta_n$
  $\mathbf{v}_{n+1} := \mathbf{y}_{n+1} - \mathbf{v}_n \psi_n$
  $\widetilde{\mathbf{v}}_{n+1} := \widetilde{\mathbf{y}}_{n+1} - \widetilde{\mathbf{v}}_n \overline{\psi_n}$
  $\delta_{n+1}' := \langle \widetilde{\mathbf{v}}_{n+1}, \mathbf{A}\mathbf{v}_{n+1} \rangle$
  If $\mathbf{y}_{n+1} = 0$ the process terminates and $\mathbf{x}_{n+1}$ is the solution.
  If $\delta_{n+1} = 0$ (and hence $\psi_n = 0$) or $\delta_{n+1}' = 0$, but $\mathbf{y}_{n+1} \neq 0$, the algorithm breaks down.
end for
In all cases we set $\dot{\nu} := n + 1$.

2.1 Basic idea and goals

Let $\mathbf{y}_0 \in \mathbb{C}^{N \times s}$ and $\widetilde{\mathbf{y}}_0 \in \mathbb{C}^{N \times \widetilde{s}}$ be two given blocks of vectors. They generate a pair of block Krylov subspaces which we denote by

$$\begin{aligned} \mathcal{B}_n^{\square}(\mathbf{A}, \mathbf{y}_0) &:\equiv \text{block span}\,(\mathbf{y}_0, \mathbf{A}\mathbf{y}_0, \ldots, \mathbf{A}^{n-1}\mathbf{y}_0) \ \subset \mathbb{C}^{N \times s}, \\ \widetilde{\mathcal{B}}_n^{\square}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0) &:\equiv \text{block span}\,(\widetilde{\mathbf{y}}_0, \mathbf{A}^\star\widetilde{\mathbf{y}}_0, \ldots, (\mathbf{A}^\star)^{n-1}\widetilde{\mathbf{y}}_0) \ \subset \mathbb{C}^{N \times \widetilde{s}} \end{aligned}$$

for $n = 1, 2, \ldots$. This means that

$$\mathcal{B}_n^{\square}(\mathbf{A}, \mathbf{y}_0) = \left\{ \sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{y}_0 \boldsymbol{\gamma}_k \;\middle|\; \boldsymbol{\gamma}_k \in \mathbb{C}^{s \times s} \right\},$$

and an analogous statement holds for $\widetilde{\mathcal{B}}_n^{\square}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$. Let

$$\mathbf{z} = [\, z^{(1)}, \ldots, z^{(s)} \,] = \sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{y}_0 \boldsymbol{\gamma}_k \in \mathcal{B}_n^{\square}(\mathbf{A}, \mathbf{y}_0).$$

This condition is equivalent to $z^{(j)} \in \mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ for $j = 1, \ldots, s$ if we define $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ to be the subspace of $\mathbb{C}^N$ spanned by the columns of $\mathbf{y}_0, \mathbf{A}\mathbf{y}_0, \ldots, \mathbf{A}^{n-1}\mathbf{y}_0$:

$$\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0) :\equiv \left\{ \sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{y}_0 \boldsymbol{\beta}_k \;\middle|\; \boldsymbol{\beta}_k \in \mathbb{C}^{s \times 1} \right\} = \sum_{i=1}^{s} K_n(\mathbf{A}, y_0^{(i)}) \ \subset \mathbb{C}^N,$$

so that $\mathcal{B}_n^{\square}(\mathbf{A}, \mathbf{y}_0) = \prod_{i=1}^{s} \mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$, i.e. $\mathcal{B}_n^{\square}(\mathbf{A}, \mathbf{y}_0)$ is the direct product of s copies of $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$. The dimensions of $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ and $\widetilde{\mathcal{B}}_n(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$ are defined as follows:

$$\begin{aligned} \dim \mathcal{B}_n(\mathbf{A}, \mathbf{y}_0) &\equiv: \nu(n) \leq ns, \\ \dim \widetilde{\mathcal{B}}_n(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0) &\equiv: \widetilde{\nu}(n) \leq n\widetilde{s}. \end{aligned}$$

Let $b \in \mathbb{C}^N$ be a column vector. As usual, we denote the grade of b with respect to $\mathbf{A}$ by $\bar{\nu}(b, \mathbf{A}) = \min\{\, n \mid \dim \mathcal{K}_n = \dim \mathcal{K}_{n+1} \,\}$. Now we define the block grade of $\mathbf{y}_0$ with respect to $\mathbf{A}$ by $\bar{\nu}(\mathbf{y}_0, \mathbf{A}) :\equiv \min\{\, n \mid \dim \mathcal{B}_n = \dim \mathcal{B}_{n+1} \,\}$, which implies

$$\bar{\nu}(\mathbf{y}_0, \mathbf{A}) \leq \max\{\, \bar{\nu}(y_0^{(i)}, \mathbf{A}) \mid 1 \leq i \leq s \,\}.$$

Our goal is to construct two nested sequences of normalized bases $\mathbf{Y}_n$ and $\widetilde{\mathbf{Y}}_{\widetilde{n}}$ of $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ and $\widetilde{\mathcal{B}}_{\widetilde{n}}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$, respectively, which we write columnwise as

$$\begin{aligned} \mathbf{Y}_n &= [\, y_0, \ldots, y_{\nu(n)-1} \,], \qquad \|y_i\|_2 = 1 \\ \widetilde{\mathbf{Y}}_{\widetilde{n}} &= [\, \widetilde{y}_0, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1} \,], \qquad \|\widetilde{y}_i\|_2 = 1 \end{aligned} \tag{2.1}$$

for $n = 1, 2, \ldots$ and a sequence of indices $0 \equiv: k_0 < k_1 < k_2 < \ldots$ with the property that

$$\mathbf{D}_{\widetilde{n},n} :\equiv \widetilde{\mathbf{Y}}_{\widetilde{n}}^\star \mathbf{Y}_n = \text{block diag}(\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{l-1}, \boldsymbol{\delta}_{l;\widetilde{n},n}) \ \in \mathbb{C}^{\widetilde{\nu}(\widetilde{n}) \times \nu(n)} \tag{2.2}$$

if $k_l + 1 \leq \min\{\nu(n), \widetilde{\nu}(\widetilde{n})\}$, where $\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{l-1}$ are nonsingular matrices of size $r_i :\equiv k_{i+1} - k_i$ for $i = 0, \ldots, l-1$. So the $k_l \times k_l$ leading principal submatrix of $\mathbf{D}_{\widetilde{n},n}$ decomposes into l nonsingular blocks. In (2.2) the block denoted by $\boldsymbol{\delta}_{l;\widetilde{n},n}$ is of size $(\widetilde{\nu}(\widetilde{n}) - k_l) \times (\nu(n) - k_l)$ (which may amount to an empty matrix), reflecting the fact that the two spaces $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ and $\widetilde{\mathcal{B}}_{\widetilde{n}}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$ usually have different dimensions. From now on, the indices $k_0, k_1, \ldots$ will be referred to as *regular* indices. To complete our notation, we set $\nu(0) :\equiv \widetilde{\nu}(0) :\equiv 0$ and $\mathcal{B}_0(\mathbf{A}, \mathbf{y}_0) :\equiv \widetilde{\mathcal{B}}_0(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0) :\equiv \{0\}$ which allows us to define

$$\begin{aligned} s_i &:\equiv \nu(i+1) - \nu(i) = \dim\big(\mathcal{B}_{i+1}(\mathbf{A}, \mathbf{y}_0) \ominus \mathcal{B}_i(\mathbf{A}, \mathbf{y}_0)\big) \\ \widetilde{s}_i &:\equiv \widetilde{\nu}(i+1) - \widetilde{\nu}(i) = \dim\big(\widetilde{\mathcal{B}}_{i+1}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0) \ominus \widetilde{\mathcal{B}}_i(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)\big) \\ \mathbf{y}_i &:\equiv [\, y_{\nu(i)}, \ldots, y_{\nu(i+1)-1} \,] \in \mathbb{C}^{N \times s_i} \\ \widetilde{\mathbf{y}}_i &:\equiv [\, \widetilde{y}_{\widetilde{\nu}(i)}, \ldots, \widetilde{y}_{\widetilde{\nu}(i+1)-1} \,] \in \mathbb{C}^{N \times \widetilde{s}_i} \end{aligned}$$

where $i = 0, 1, 2, \ldots$. Now we can rewrite (2.1) as $\mathbf{Y}_n = [\, \mathbf{y}_0, \ldots, \mathbf{y}_{n-1} \,]$[1].

The structure of the matrix $\mathbf{D}_{\widetilde{n},n}$ suggests another way of partitioning the two vector sequences $\{y_i\}_{i \geq 0}$ and $\{\widetilde{y}_i\}_{i \geq 0}$ which is oriented towards the block biorthogonality. We define

$$\begin{aligned} \mathbf{z}_i &:\equiv [\, y_{k_i}, \ldots, y_{k_{i+1}-1} \,] \in \mathbb{C}^{N \times r_i}, \qquad \mathbf{Z}_l :\equiv [\, \mathbf{z}_0, \ldots, \mathbf{z}_{l-1} \,] \\ \widetilde{\mathbf{z}}_i &:\equiv [\, \widetilde{y}_{k_i}, \ldots, \widetilde{y}_{k_{i+1}-1} \,] \in \mathbb{C}^{N \times r_i}, \qquad \widetilde{\mathbf{Z}}_l :\equiv [\, \widetilde{\mathbf{z}}_0, \ldots, \widetilde{\mathbf{z}}_{l-1} \,] \end{aligned}$$

for $i = 0, \ldots, l-1$ so that the biorthogonality can be written as

$$\widetilde{\mathbf{z}}_i^\star \mathbf{z}_j = \begin{cases} \mathbf{0} & i \neq j \\ \boldsymbol{\delta}_i & i = j \end{cases} \qquad \text{or} \quad \widetilde{\mathbf{Z}}_l^\star \mathbf{Z}_l = \text{block diag}(\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{l-1}). \tag{2.3}$$

The blocks $\{\mathbf{y}_n\}$ and $\{\widetilde{\mathbf{y}}_n\}$ will be referred to as *right* and *left blocks*, while $\{\mathbf{z}_l\}$ and $\{\widetilde{\mathbf{z}}_l\}$ will be called *right* and *left clusters*, respectively.

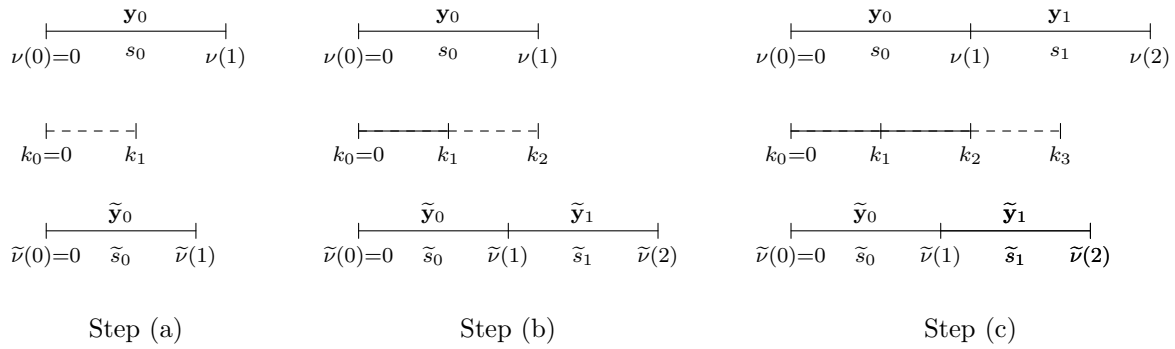
We note that from the definitions of s_i and r_i we obtain for $i = 0, 1, 2, \ldots$

$$k_{i+1} = \sum_{j=0}^{i} r_j \qquad \text{and} \qquad \nu(i+1) = \sum_{j=0}^{i} s_j.$$

In order to give an idea of the algorithm, we will now go through a simple example. It is clear that this simple case does not illustrate all the possible situations that can arise when BLBIO is run but the reader may still get an idea of the algorithm.

First of all, we need two starting blocks $\mathbf{y}$ for the right block Krylov space and $\widetilde{\mathbf{y}}$ for the left one. Clearly the columns of the starting blocks must be linearly independent, so we first have to check $\mathbf{y}$ and $\widetilde{\mathbf{y}}$ for deflation. We denote the resulting blocks by $\mathbf{y}_0$ and $\widetilde{\mathbf{y}}_0$. Consequently, the ranges $\mathcal{R}(\mathbf{y}_0)$ and $\mathcal{R}(\mathbf{y})$ are identical, e. g. the columns of $\mathbf{y}_0$ may form an orthonormal basis of $\mathcal{R}(\mathbf{y})$. Next, a value for k_1, the first nontrivial regular index, has to be fixed. We want our algorithm to be flexible with respect to the cluster size, i.e. in theory, it should work with any cluster size. We do not see, however, a reason to choose k_1 larger than s_0 so that $1 \leq k_1 \leq s_0$. At this point, we do not know yet whether our choice for k_1 works, consequently we say that we have fixed an *initial* value for k_1. So we now have arrived at step (a).

[1] From now on we sometimes only write the formulas for quantities related to the right block Krylov spaces. Similar formulas hold with $\mathbf{A}$, $\mathbf{y}_i$, s_i, etc. replaced by $\mathbf{A}^\star$, $\widetilde{\mathbf{y}}_i$, $\widetilde{s}_i$, etc.



Step (a) Step (b) Step (c)

Here, the dashed line means that the choice of k_1 is not final yet. Our next task is to check whether the current value for k_1 works, which means that we build preliminary clusters $\mathbf{z}_0 = [\, y_0, \ldots, y_{k_1-1}\,]$ and $\widetilde{\mathbf{z}}_0 = [\, \widetilde{y}_0, \ldots, \widetilde{y}_{k_1-1}\,]$ and check the nonsingularity of $\boldsymbol{\delta}_0 = \widetilde{\mathbf{z}}_0^\star \mathbf{z}_0$. Assume that $\boldsymbol{\delta}_0$ turns out to be nonsingular (in finite precision arithmetic: well-conditioned). Then we declare k_1, $\mathbf{z}_0$ and $\widetilde{\mathbf{z}}_0$ as final and thus have completed the first clusters. Now it has to be noted that since $k_1 < \nu(1) = s_0$ there are right vectors which are not part of $\mathbf{z}_0$; we call them the *ungrouped* right vectors. Since the ungrouped right vectors will later become part of a different right cluster, they must be orthogonal to $\widetilde{\mathbf{z}}_0$, and we have to orthogonalize $y_{k_1}, \ldots, y_{\nu(1)-1}$ against $\widetilde{\mathbf{z}}_0$. It is not hard to show that this orthogonalization preserves the linear independency of all the right vectors. Similarly, we can see from the above figure that $k_1 < \widetilde{\nu}(1)$ so that the ungrouped left vectors $\widetilde{y}_{k_1}, \ldots, \widetilde{y}_{\widetilde{\nu}(1)-1}$ have to be orthogonalized against $\mathbf{z}_0$, and the linear independency of the left vectors is thereby not destroyed.

Here we would like to point out that the ungrouped right and left vectors might still be subject to change when more clusters are completed. Since the next right cluster will contain at least one vector, the vectors $y_{k_1+1}, \ldots, y_{\nu(1)-1}$ are preliminary at this point, and an analogous statement holds for the left vectors. So, stricly speaking, the vectors $y_{k_1+1}, \ldots, y_{\nu(1)-1}$ cannot be called right vectors yet but in this work, for simplicity we will call them so.

Now we turn to the next regular index k_2 and fix a preliminary value for it, see step (b) of the above figure. Here k_2 happens to be equal to $\nu(1)$ so that there are enough right vectors to form the preliminary right cluster $\mathbf{z}_1$. But we see that $k_2 > \widetilde{\nu}(1)$, whence the left block Krylov space has to be extended before the preliminary left cluster $\widetilde{\mathbf{z}}_1$ can be built. Thus we compute $\mathbf{A}^\star \widetilde{\mathbf{y}}_0$ and orthogonalize it against $\mathbf{z}_0$, the only right cluster completed so far. We denote the resulting block by $\widetilde{\mathbf{y}}_{\text{tmp}}$. Now the columns of $\widetilde{\mathbf{y}}_{\text{tmp}}$ may be linearly dependent so that we have to check them for deflation. But here, a little care is needed: it is not enough to check the columns of $\widetilde{\mathbf{y}}_{\text{tmp}}$ alone, we need to check the columns of $\widetilde{\mathbf{y}}_{\text{tmp}}$ *plus the currently ungrouped left vectors*, i.e. the block $[\, \widetilde{y}_{k_1}, \ldots, \widetilde{y}_{\widetilde{\nu}(1)-1} \;\; \widetilde{\mathbf{y}}_{\text{tmp}}\,]$ (note that, at this point, the columns of $\widetilde{\mathbf{y}}_{\text{tmp}}$ are not yet left vectors). Since $\widetilde{y}_{k_1}, \ldots, \widetilde{y}_{\widetilde{\nu}(1)-1}$ are already linearly independent, they will "survive" the deflation checking process and the remaining columns of $\widetilde{\mathbf{y}}_{\text{tmp}}$ form the block $\widetilde{\mathbf{y}}_1$. From now on, we consider the columns of this new block as left vectors, and they are now added to the set of ungrouped left vectors, which thus consists of $\widetilde{y}_{k_1}, \ldots, \widetilde{y}_{\widetilde{\nu}(2)-1}$. Again we incorrectly call all of these vectors "left vectors" despite the possibility that they might still be subject to change. So at this point, we know that the ungrouped left vectors are linearly independent and orthogonal to all the completed right clusters, and an analogous statement holds for the right ungrouped vectors.

To continue, we assume that no columns of $\widetilde{\mathbf{y}}_{\text{tmp}}$ need to be deflated so that $\widetilde{\nu}(2) > k_2$ and we now can form the preliminary cluster $\widetilde{\mathbf{z}}_1$ (now, step (b) shows the current status). As described above for the case of $\boldsymbol{\delta}_0$, it is necessary to check the nonsingularity (or well-definedness) of $\boldsymbol{\delta}_1 = \widetilde{\mathbf{z}}_1^\star \mathbf{z}_1$. Let us assume that $\boldsymbol{\delta}_1$ is nonsingular (well-defined) whence the clusters $\mathbf{z}_1$ and $\widetilde{\mathbf{z}}_1$ can be completed. At this point, there are no ungrouped right vectors due to $k_1 = \nu(1)$, but some left vectors are left over, and they now have to be orthogonalized against $\mathbf{z}_1$.

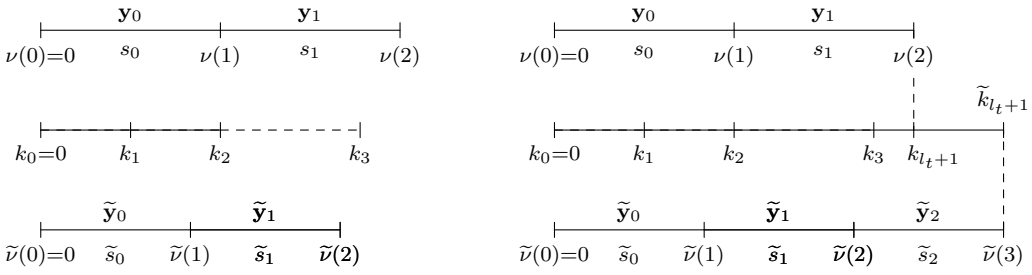
By now, the reader has certainly guessed that the following properties always hold:

- The completed right and left clusters satisfy the biorthogonality (2.3) with nonsingular $\boldsymbol{\delta}_l$.
- The ungrouped right vectors are linearly independent and orthogonal to all the completed left clusters.

- The ungrouped left vectors are linearly independent and orthogonal to all the completed right clusters.

From these three properties the linear independency of the right and left vectors follows easily and therefore it is not surprising that they will appear again later (see Lemma 5).

We return to our example and choose a preliminary k_3, and since $k_3 > \nu(1)$, it is necessary that the right block Krylov space be extended. This is analogous to the extension of the left block Krylov space outlined above, and we will not dwell on it. For simplicity, we assume that one such extension yields enough right vectors, i.e. $k_3 < \nu(2)$, thus arriving at step (c). This time, however, we consider the case where our current value for k_3 leads to a singular matrix $\boldsymbol{\delta}_3$. The first thing we can try now is to replace a vector in the preliminary cluster $\widetilde{\mathbf{z}}_3$ by a column of $\widetilde{\mathbf{y}}_1$ which is not already in $\widetilde{\mathbf{z}}_3$ (it might be necessary to replace several vectors, depending on the rank deficiency of $\boldsymbol{\delta}_3$). If we are lucky, the new $\boldsymbol{\delta}_3$ will be nonsingular. If this does not help, it is necessary to change our preliminary value for k_3. In the example, we assume that we have to change the value for k_3 so that the situation depicted in step (d) arises.



Step (d) Step (e)

We see from the picture that the left block Krylov space has to be extended and assume that after the extension the clusters $\mathbf{z}_2$ and $\widetilde{\mathbf{z}}_2$ can be completed. Now an initial value for k_4 is fixed, and we assume that $k_4 > \nu(2)$ and that it turns out to be impossible to extend the right block Krylov space so that the process terminates. Now we set $l_t = 3$, $k_{l_t+1} = \nu(2)$ and $\widetilde{k}_{l_t+1} = \widetilde{\nu}(3)$. In other words, all those right vectors which were ungrouped at termination are put into a final cluster $\mathbf{z}_{l_t} = [\, y_{k_3}, \ldots, y_{\nu(2)-1} \,]$ and similarly for the left vectors: $\widetilde{\mathbf{z}}_{l_t} = [\, \widetilde{y}_{k_3}, \ldots, \widetilde{y}_{\widetilde{\nu}(3)-1} \,]$. It follows that these last clusters contain a different number of vectors, in contrast to all other clusters obtained so far. Step (e) shows the state after termination.

From this short example, a few conclusions can be drawn: First there are three basic steps which occur again and again:

- The extension of the left block Krylov space, including deflation of linearly dependent left vectors.
- The extension of the right block Krylov space, including deflation of linearly dependent right vectors.
- Checking whether our current value of k_{l+1} works, including a procedure to be followed if the current value of k_{l+1} fails (so that look-ahead is included here).

Formulation of these three steps in such a way that the three properties listed earlier are invariants lead to the version of BLBIO described in detail in Section 2.2. There is, however, one more important point to keep in mind: As we ultimately want to solve block systems of linear equations, we also want recurrences analogous to the nonblock case. More precisely, we must be able to show the existence of matrices $\underline{\mathbf{T}}_i \in \mathbb{C}^{\nu(i+1)\times\nu(i)}$ and $\underline{\widetilde{\mathbf{T}}}_i \in \mathbb{C}^{\widetilde{\nu}(i+1)\times\widetilde{\nu}(i)}$ (which should have a suitable structure) such that

$$\mathbf{A}\mathbf{Y}_i = \mathbf{Y}_{i+1}\underline{\mathbf{T}}_i$$
$$\mathbf{A}^\star\widetilde{\mathbf{Y}}_i = \widetilde{\mathbf{Y}}_{i+1}\underline{\widetilde{\mathbf{T}}}_i$$

holds in exact arithmetic for $i = 1, 2, \ldots$.

2.2 BLBIO and its fundamental properties

The following definitions will be useful for the description of BLBIO. First we set $n(i) :\equiv \min\{j \mid y_{i-1} \in \mathcal{B}_j(\mathbf{A}, \mathbf{y}_0)\}$, so that $n(i) \geq 1$ and

$$\begin{aligned} n = n(i) \quad &\iff \quad y_{i-1} \text{ is a basis vector of } \mathcal{B}_n(\mathbf{A}, \mathbf{y}_0) \ominus \mathcal{B}_{n-1}(\mathbf{A}, \mathbf{y}_0) \\ &\iff \quad y_{i-1} \text{ is a column of } \mathbf{y}_{n-1} \quad \iff \quad \nu(n-1) \leq i-1 < \nu(n). \end{aligned}$$

Furthermore, we define $\ell(i)$ to be the index of the cluster which contains y_{i-1}. Consequently $\ell(i) \geq 0$ and

$$l = \ell(i) \quad \iff \quad y_{i-1} \text{ is a column of } \mathbf{z}_l \quad \iff \quad k_l \leq i-1 < k_{l+1}.$$

Thus $\ell(i)$ and $n(i)$ are defined for $i = 1, 2, \ldots$.

Now we begin our detailed description of BLBIO, see Algorithms 6 to 8. It consists of three major steps:

1) Extension of the left block Krylov space (Algorithm 7).
2) Checking whether the next right and left clusters can be completed (Algorithm 6).
3) Extension of the right block Krylov space (Algorithm 8).

The overall structure is covered in Algorithm 6. The first thing to do is to check the input blocks $\mathbf{y}$ and $\widetilde{\mathbf{y}}$ for deflation; this leads to the starting blocks $\mathbf{y}_0$ and $\widetilde{\mathbf{y}}_0$, respectively. We compute a high rank revealing QR (HRRQR) factorization to do this check, e. g. for the left starting vectors:

$$\widetilde{\mathbf{y}} = [\,\widetilde{\mathbf{y}}_0 \;\; \widetilde{\mathbf{y}}_0^{\triangle}\,] \begin{bmatrix} \widetilde{\boldsymbol{\rho}}_0 & \widetilde{\boldsymbol{\rho}}_0^{\square} \\ \mathbf{0} & \widetilde{\boldsymbol{\rho}}_0^{\triangle} \end{bmatrix} \widetilde{\pi}_0^{\top} =: [\,\widetilde{\mathbf{y}}_0 \;\; \widetilde{\mathbf{y}}_0^{\triangle}\,] \begin{bmatrix} \widetilde{\boldsymbol{\eta}}_0 \\ \widetilde{\boldsymbol{\eta}}_0^{\triangle} \end{bmatrix}. \tag{2.4}$$

Let $\widetilde{s}_0 = \widetilde{\nu}(1)$ be the numerical rank of $\widetilde{\mathbf{y}}$ and $\widetilde{s}_0^{\triangle} :\equiv \widetilde{s} - \widetilde{s}_0$. Table 1 (with $\widetilde{n}$ replaced by 0 and $\widetilde{s}_{-1}$ replaced by $\widetilde{s}$) gives an overview over the matrices appearing in this decomposition. The columns of $\widetilde{\mathbf{y}}_0^{\triangle}$ are discarded. If s_0 or $\widetilde{s}_0$ turns out to be zero, we cannot run the algorithm and have to stop immediately. This manner of terminating is abnormal, and therefore, when we later use the term "termination", we do not refer to this abnormal termination. Next we need to fix the initial choice for k_1 subject to the condition $1 \leq k_1 \leq \nu(1)$ (step LA1). This completes the initializations, now the main loop is entered, and first it is necessary that the number of available left vectors be at least equal to k_{l+1}. When program execution reaches line 10, we already have computed the blocks $\mathbf{y}_0, \ldots, \mathbf{y}_{n-1}$ from the right block Krylov space, $\widetilde{\mathbf{y}}_0, \ldots, \widetilde{\mathbf{y}}_{\widetilde{n}-1}$ from the left and the regular indices $k_0, \ldots, k_l$ (this will be proved later, see Proposition 6).

Algorithm 7 shows the extension of the left block Krylov space in more detail. We first compute $\mathbf{A}^{\star}\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ and biorthogonalize against $\mathbf{z}_0, \ldots, \mathbf{z}_{l-1}$:

$$\widetilde{\mathbf{y}}_{\mathrm{tmp}} = \mathbf{A}^{\star}\widetilde{\mathbf{y}}_{\widetilde{n}-1} - \widetilde{\mathbf{z}}_{l-1}\widetilde{\boldsymbol{\tau}}^z_{l-1,\widetilde{n}-1} - \cdots - \widetilde{\mathbf{z}}_0\widetilde{\boldsymbol{\tau}}^z_{0,\widetilde{n}-1} \tag{2.5}$$

where, for $j = 0, \ldots, \widetilde{n}-1$, the matrices $\widetilde{\boldsymbol{\tau}}^z_{i,j} \in \mathbb{C}^{r_i \times \widetilde{s}_j}$ are given by $\widetilde{\boldsymbol{\tau}}^z_{i,j} = \boldsymbol{\delta}_i^{-\star}\mathbf{z}_i^{\star}\mathbf{A}^{\star}\widetilde{\mathbf{y}}_j$ for $i = 0, \ldots, l-1$. Lines 2 to 4 take care of this biorthogonalization. Then, in line 5, we need to check for deflation all the ungrouped left vectors, i.e. the columns of $\widetilde{\mathbf{y}}_{\mathrm{tmp}}$ and of $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}} :\equiv [\,\widetilde{y}_{k_l}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}\,]$. We do this by first orthogonalizing $\widetilde{\mathbf{y}}_{\mathrm{tmp}}$ against $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}}$, which means that we compute $\widetilde{\mathbf{y}}_{\mathrm{tmp}} - \widetilde{\mathbf{y}}_{l-1,\widetilde{n}}\boldsymbol{\alpha}$ with

$$\boldsymbol{\alpha} = (\widetilde{\mathbf{y}}^{\star}_{l-1,\widetilde{n}}\widetilde{\mathbf{y}}_{l-1,\widetilde{n}})^{-1}\widetilde{\mathbf{y}}^{\star}_{l-1,\widetilde{n}}\widetilde{\mathbf{y}}_{\mathrm{tmp}}. \tag{2.6}$$

In Lemma 5, the columns of $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}}$ will turn out to be linearly independent, therefore the inverse in this formula exists. With this orthogonalized block $\widetilde{\mathbf{y}}_{\mathrm{tmp}} - \widetilde{\mathbf{y}}_{l-1,\widetilde{n}}\boldsymbol{\alpha}$, we now do a rank revealing QR factorization and obtain a decomposition as in (2.4) with index 0 replaced by $\widetilde{n}$. We define $\widetilde{s}_{\widetilde{n}}$ to be the number of columns of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ and $\widetilde{s}_{\widetilde{n}}^{\triangle} :\equiv \widetilde{s}_{\widetilde{n}-1} - \widetilde{s}_{\widetilde{n}}$. See again Table 1. The columns of $[\,\widetilde{\mathbf{y}}_{\widetilde{n}} \;\; \widetilde{\mathbf{y}}_{\widetilde{n}}^{\triangle}\,]$ are orthonormal and $\widetilde{\boldsymbol{\rho}}_{\widetilde{n}}$ is upper triangular and nonsingular. It should be noted that in exact arithmetic $\widetilde{\boldsymbol{\rho}}_{\widetilde{n}}^{\triangle}$ would be zero, so that the columns of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ then form an orthonormal basis of the column space of $[\,\widetilde{\mathbf{y}}_{\widetilde{n}} \;\; \widetilde{\mathbf{y}}_{\widetilde{n}}^{\triangle}\,]$, but in finite precision arithmetic we only have bounds for $\|\widetilde{\boldsymbol{\rho}}_{\widetilde{n}}^{\triangle}\|_2$ and $\|\widetilde{\boldsymbol{\eta}}_{\widetilde{n}}^{\triangle}\|_2$ which depend

Algorithm 6 BLBIO

Input: Matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$, block $\mathbf{y} \in \mathbb{C}^{N\times s}$ of right Lanczos vectors, block $\widetilde{\mathbf{y}} \in \mathbb{C}^{N\times \widetilde{s}}$ of left Lanczos vectors

1: /* *Initializations, including:* */
2: Check $\mathbf{y}$ and $\widetilde{\mathbf{y}}$ for deflation $\rightarrow \mathbf{y}_0$, $\widetilde{\mathbf{y}}_0$, $\nu(1)$, $\widetilde{\nu}(1)$, s_0, $\widetilde{s}_0$.
3: **if** $s_0 = 0$ **or** $\widetilde{s}_0 = 0$ **then**
4: **stop** /* *abnormal termination* */
5: **end if**
6: Set $l = 0$, $n = 1$, $\widetilde{n} = 1$, $k_0 = 0$.
7: LA1 Fix initial choice for k_1 under the condition $1 \leq k_1 \leq \nu(1)$.
8: **loop**
9: /* *1) Compute enough $\widetilde{y}$ vectors:* */
10: Extend left block Krylov space according to Algorithm 7.
11: /* *2) Test whether current value of k_{l+1} leads to a nonsingular $\boldsymbol{\delta}_l$:* */
12: Set $\mathbf{z}_l = [\, y_{k_l}, \ldots, y_{k_{l+1}-1} \,]$, $\widetilde{\mathbf{z}}_l = [\, \widetilde{y}_{k_l}, \ldots, \widetilde{y}_{k_{l+1}-1} \,]$ and $r_l = k_{l+1} - k_l$.
13: Compute $\boldsymbol{\delta}_l = \widetilde{\mathbf{z}}_l^\star \mathbf{z}_l$ and $\Delta_l = r_l - \operatorname{rank}(\boldsymbol{\delta}_l)$.
14: **if** $\Delta_l > 0$ **then**
15: Check whether there is a permutation P of the columns of $\mathbf{y}_{n-1}$ which might cure the rank deficiency of $\boldsymbol{\delta}_l$.
16: **if** there is such a P **then**
17: Apply P to the columns of $\mathbf{y}_{n-1}$.
18: Adapt the matrix $\boldsymbol{\eta}_{n-1}$ by applying the inverse permutation $P^\star$ to its rows.
19: Set $\mathbf{z}_l = [\, y_{k_l}, \ldots, y_{k_{l+1}-1} \,]$.
20: Compute $\boldsymbol{\delta}_l = \widetilde{\mathbf{z}}_l^\star \mathbf{z}_l$ and $\Delta_l = r_l - \operatorname{rank}(\boldsymbol{\delta}_l)$.
21: **end if**
22: **end if**
23: **if** $\Delta_l > 0$ **then**
24: Repeat lines 15 to 20 for the left vectors.
25: **end if**
26: **if** $\Delta_l > 0$ **then**
27: LA2 Increase k_{l+1}.
28: **else**
29: Orthogonalize $[\, y_{k_{l+1}}, \ldots, y_{\nu(n)-1} \,]$ against $\widetilde{\mathbf{z}}_l$ and $[\, \widetilde{y}_{k_{l+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1} \,]$ against $\mathbf{z}_l$. Renormalize the orthogonalized vectors and adapt $\underline{\mathbf{T}}_n$ and $\underline{\widetilde{\mathbf{T}}}_{\widetilde{n}}$.
30: LA3 Fix new initial choice for k_{l+2}.
31: Set $l = l + 1$.
32: **end if**
33: /* *3) Compute enough y vectors:* */
34: LA4 Fix minimum number of right vectors $\rightarrow$ minnum.
35: Extend right block Krylov space according to Algorithm 8.
36: **end loop**

Algorithm 7 Extension of left block Krylov space for BLBIO

while $\widetilde{\nu}(\widetilde{n}) < k_{l+1}$ **do**
  Extend the left Krylov space $\rightarrow \mathbf{A}^\star\widetilde{\mathbf{y}}_{\widetilde{n}-1}$.
  Determine against which right clusters $\mathbf{A}^\star\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ needs to be orthogonalized.
  Perform orthogonalization of $\mathbf{A}^\star\widetilde{\mathbf{y}}_{\widetilde{n}-1} \rightarrow \widetilde{\mathbf{y}}_{\mathrm{tmp}}$.
  Check the vectors $[\,\widetilde{y}_{k_l}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}, \widetilde{\mathbf{y}}_{\mathrm{tmp}}\,]$ for deflation $\rightarrow \widetilde{s}_{\widetilde{n}}$, $\widetilde{\mathbf{y}}_{\widetilde{n}}$, $\widetilde{\nu}(\widetilde{n}+1)$, $\widetilde{s}^{\triangle}_{\widetilde{n}}$, $\widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}$, $\widetilde{\nu}^{\triangle}(\widetilde{n}+1)$
  if $\widetilde{s}_{\widetilde{n}} = 0$ **then**
    Set $l_t = l$, $n_t = n$, $\widetilde{n}_t = \widetilde{n}$, $k_{l_t+1} = \nu(n_t)$, $\widetilde{k}_{l_t+1} = \widetilde{\nu}(\widetilde{n}_t)$, $l = l+1$.
    Set $\mathbf{z}_{l_t} = [\,y_{k_{l_t}}, \ldots, y_{\nu(n_t)-1}\,]$, $\widetilde{\mathbf{z}}_{l_t} = [\,\widetilde{y}_{k_{l_t}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n}_t)-1}\,]$ and $\boldsymbol{\delta}_{l_t} = \widetilde{\mathbf{z}}^\star_{l_t}\widetilde{\mathbf{z}}_{l_t}$.
    stop
  end if
  Set $\widetilde{n} = \widetilde{n}+1$.
end while

Matrix	Size	Remarks
$[\,\widetilde{\mathbf{y}}_{\widetilde{n}}\ \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\,]$	$N \times \widetilde{s}_{\widetilde{n}-1}$	columns orthonormal
$\widetilde{\mathbf{y}}_{\widetilde{n}}$	$N \times \widetilde{s}_{\widetilde{n}}$	new left block
$\widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}$	$N \times \widetilde{s}^{\triangle}_{\widetilde{n}}$	orthogonal to $\widetilde{\mathbf{y}}_{\widetilde{n}}$
$\widetilde{\pi}_{\widetilde{n}}$	$\widetilde{s}_{\widetilde{n}-1} \times \widetilde{s}_{\widetilde{n}-1}$	permutation matrix
$\widetilde{\boldsymbol{\rho}}_{\widetilde{n}}$	$\widetilde{s}_{\widetilde{n}} \times \widetilde{s}_{\widetilde{n}}$	upper triangular, nonsingular
$\widetilde{\boldsymbol{\rho}}^{\square}_{\widetilde{n}}$	$\widetilde{s}_{\widetilde{n}} \times \widetilde{s}^{\triangle}_{\widetilde{n}}$	
$\widetilde{\boldsymbol{\rho}}^{\triangle}_{\widetilde{n}}$	$\widetilde{s}^{\triangle}_{\widetilde{n}} \times \widetilde{s}^{\triangle}_{\widetilde{n}}$	almost zero
$\widetilde{\boldsymbol{\eta}}_{\widetilde{n}}$	$\widetilde{s}_{\widetilde{n}} \times \widetilde{s}_{\widetilde{n}-1}$	$\widetilde{\boldsymbol{\eta}}_{\widetilde{n}} :\equiv [\,\widetilde{\boldsymbol{\rho}}_{\widetilde{n}}\ \widetilde{\boldsymbol{\rho}}^{\square}_{\widetilde{n}}\,]\,\widetilde{\pi}^{\top}_{\widetilde{n}}$
$\widetilde{\boldsymbol{\eta}}^{\triangle}_{\widetilde{n}}$	$\widetilde{s}^{\triangle}_{\widetilde{n}} \times \widetilde{s}_{\widetilde{n}-1}$	$\widetilde{\boldsymbol{\eta}}^{\triangle}_{\widetilde{n}} :\equiv [\,\mathbf{0}\ \widetilde{\boldsymbol{\rho}}^{\triangle}_{\widetilde{n}}\,]\,\widetilde{\pi}^{\top}_{\widetilde{n}}$

Table 1: Matrices involved in HRRQR

on the particular algorithm used to compute the HRRQR factorization (see [3] for more details). This implies

$$\widetilde{\mathbf{y}}_{\mathrm{tmp}} - \widetilde{\mathbf{y}}_{l-1,\widetilde{n}}\boldsymbol{\alpha} = \widetilde{\mathbf{y}}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}_{\widetilde{n}} + \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}^{\triangle}_{\widetilde{n}}.$$

In the case where no deflation occurs, we have $\widetilde{s}_{\widetilde{n}} = \widetilde{s}_{\widetilde{n}-1}$ and so the matrices $\widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}$, $\widetilde{\boldsymbol{\rho}}^{\square}_{\widetilde{n}}$, $\widetilde{\boldsymbol{\rho}}^{\triangle}_{\widetilde{n}}$, $\widetilde{\boldsymbol{\eta}}^{\triangle}_{\widetilde{n}}$ are all empty. On the other hand it may also happen that $\widetilde{s}_{\widetilde{n}} = 0$, which means that $\widetilde{\mathbf{y}}_{\widetilde{n}} \approx \mathbf{0}$ and consequently the left block Krylov space is exhausted.

By using (2.5) and setting $\widetilde{\boldsymbol{\tau}}^{y}_{l-1,\widetilde{n}-1} :\equiv \boldsymbol{\alpha}$, we arrive at the following recursion:

$$\widetilde{\mathbf{y}}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}_{\widetilde{n}} = \mathbf{A}^\star\widetilde{\mathbf{y}}_{\widetilde{n}-1} - \widetilde{\mathbf{y}}_{l-1,\widetilde{n}}\widetilde{\boldsymbol{\tau}}^{y}_{l-1,\widetilde{n}-1} - \widetilde{\mathbf{z}}_{l-1}\widetilde{\boldsymbol{\tau}}^{z}_{l-1,\widetilde{n}-1} - \cdots - \widetilde{\mathbf{z}}_0\widetilde{\boldsymbol{\tau}}^{z}_{0,\widetilde{n}-1} - \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}^{\triangle}_{\widetilde{n}}.$$

For later reference, it is useful to rewrite this recursion in slightly different form by observing that the inequality $k_l \le \widetilde{\nu}(\widetilde{n}) < k_{l+1}$ holds. (If the first inequality was false, we would have $k_l > \widetilde{\nu}(\widetilde{n})$ or $k_l - 1 \ge \widetilde{\nu}(\widetilde{n})$ so that the block $\widetilde{\mathbf{y}}_{\widetilde{n}}$ would have been computed before the completion of $\widetilde{\mathbf{z}}_{l-1}$. The second inequality is just the condition in the **while** loop.) This implies that $\widetilde{y}_{\widetilde{\nu}(\widetilde{n})}$ is a column of $\widetilde{\mathbf{z}}_l$ so that $l = \ell(\widetilde{\nu}(\widetilde{n})+1)$ according to the definition of the function $\ell(i)$. If we define

$$\ell_1(\widetilde{n}) :\equiv \ell(\widetilde{\nu}(\widetilde{n})+1) - 1 = l - 1,$$

we have $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}} = \widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}),\widetilde{n}} = [\,\widetilde{y}_{k_{\ell_1(\widetilde{n})+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}\,]$ and write the recursion as

$$\widetilde{\mathbf{y}}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}_{\widetilde{n}} = \mathbf{A}^\star\widetilde{\mathbf{y}}_{\widetilde{n}-1} - \widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}),\widetilde{n}}\widetilde{\boldsymbol{\tau}}^{y}_{\ell_1(\widetilde{n}),\widetilde{n}-1} - \widetilde{\mathbf{z}}_{\ell_1(\widetilde{n})}\widetilde{\boldsymbol{\tau}}^{z}_{\ell_1(\widetilde{n}),\widetilde{n}-1} - \cdots - \widetilde{\mathbf{z}}_0\widetilde{\boldsymbol{\tau}}^{z}_{0,\widetilde{n}-1} - \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}^{\triangle 2}_{\widetilde{n}} \qquad (2.7)$$

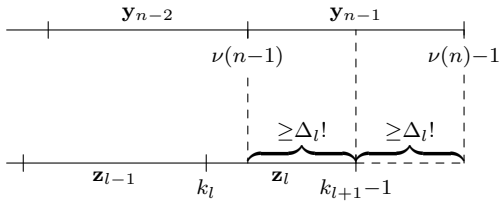
By now the columns of $[\,\widetilde{\mathbf{y}}_{l-1,\widetilde{n}}\,]$ are the ungrouped left vectors that can be used for the next cluster. Here it must be noted that the block $\widetilde{\mathbf{y}}_{\widetilde{n}}$ may be changed later as more clusters are completed and

[2] If $\ell_1(\widetilde{n}) = -1$, no clusters have been completed yet. In this case $\widetilde{\mathbf{y}}_{-1,\widetilde{n}} = [\,\widetilde{y}_{k_0}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}\,]$ and there are no terms $\widetilde{\mathbf{z}}_i$ in this recursion. In the future, we will not mention this or similar exceptional cases any more.

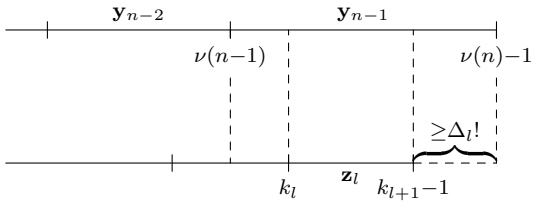
some part of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ is orthogonalized against them. The number $\widetilde{s}_{\widetilde{n}}$, however, will remain unchanged because the later orthogonalizations will retain the linear independency of the columns of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ (see Lemma 5). Strictly speaking, the columns of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ are not yet left vectors, but as we pointed out in Section 2.1, for simplicity we will not distinguish between final and preliminary left vectors and call them all "left vectors". The final form of the recurrence (2.7) will be obtained in equation (2.14).

If this does not yield enough left vectors to build the next preliminary cluster, we have to cycle again through the **while** loop on line 7.1. Continuing this way the algorithm either terminates because some $\mathbf{A}^\star$-invariant subspace has been found or generates enough left vectors. In the latter case, the variable $\widetilde{n}$ has the value $\widetilde{n}(k_{l+1})$ (so that $\widetilde{y}_{k_{l+1}-1}$, the last left vector which will become part of the temporary cluster $\widetilde{\mathbf{z}}_l$, is a column of $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$) and the left blocks $\widetilde{\mathbf{y}}_0, \ldots, \widetilde{\mathbf{y}}_{\widetilde{n}-1}$ have been computed.

Now we return to Algorithm 6 and continue with step 2). This step consists of checking whether our current temporary k_{l+1} works. Since at this point $n = n(k_{l+1})$ (this follows from Definition 3 below) and $\widetilde{n} = \widetilde{n}(k_{l+1})$, we know that $y_{k_{l+1}-1}$ is a column of $\mathbf{y}_{n-1}$ and $\widetilde{y}_{k_{l+1}-1}$ is a column of $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$. We are now able to build temporary clusters $\mathbf{z}_l$ and $\widetilde{\mathbf{z}}_l$ as in line 6.12 and need to compute the rank of $\boldsymbol{\delta}_l$ obtained in line 6.13. If $\boldsymbol{\delta}_l$ turns out to be singular, we can first try to apply a permutation to the right or left vectors to obtain a new $\boldsymbol{\delta}_l$. Since both cases are largely analogous, we will describe the process only for the right vectors. Thus we want to replace columns of $\mathbf{y}_{n-1}$ which are part of $\mathbf{z}_l$ by other columns of $\mathbf{y}_{n-1}$ which are not contained in $\mathbf{z}_l$. It would be necessary to exchange Δ_l of them, and to check whether this is possible, the following two cases have to be considered (see the following Figure and recall that the columns of $\mathbf{z}_l$ are $y_{k_l}, \ldots, y_{k_{l+1}-1}$): In case A, the first vector in $\mathbf{z}_l$ is either identical to the first vector of $\mathbf{y}_{n-1}$ or "to the left" of it. So the vectors numbered $\nu(n-1), \ldots, k_{l+1}-1$ are columns of both $\mathbf{z}_l$ and $\mathbf{y}_{n-1}$, while those with indices $k_{l+1}, \ldots, \nu(n)-1$ (if there are any) are in $\mathbf{y}_{n-1}$ but not in $\mathbf{z}_l$. In case B, however, index k_l lies to the right of index $\nu(n-1)$ so that we are free to replace any vector of $\mathbf{z}_l$.



Case A: $k_l \leq \nu(n-1)$



Case B: $k_l > \nu(n-1)$

Now we formulate the conditions for the existence of a permutation P as in line 6.15.

- There must be at least Δ_l columns of $\mathbf{z}_l$ which are also columns of $\mathbf{y}_{n-1}$. In case A, this implies

$$\Delta_l \leq (k_{l+1}-1) - \nu(n-1) + 1 = k_{l+1} - \nu(n-1),$$

  while it is always satisfied in case B. We note that in both cases, this condition may be formulated as

$$\Delta_l \leq k_{l+1} - \max\{k_l, \nu(n-1)\}. \tag{2.8}$$

- There must be at least Δ_l columns of $\mathbf{y}_{n-1}$ which are not already contained in $\mathbf{z}_l$. Thus we need

$$\Delta_l \leq \big(\nu(n)-1\big) - (k_{l+1}-1) = \nu(n) - k_{l+1}. \tag{2.9}$$

If conditions (2.8) and (2.9) are satisfied, we can go on to find a permutation P. For example, an HRRQR decomposition might be used to check the nonsingularity of $\boldsymbol{\delta}_l$ in line 6.12, and this information can be used to find a permutation of the right vectors as follows. Define $\boldsymbol{\pi}_{\delta,l}$ to be the column permutation computed by the HRRQR algorithm so that $\boldsymbol{\delta}_l\boldsymbol{\pi}_{\delta,l} = \widetilde{\mathbf{z}}_l^\star(\mathbf{z}_l\boldsymbol{\pi}_{\delta,l})$. We know that the last Δ_l columns of $\boldsymbol{\delta}_l\boldsymbol{\pi}_{\delta,l}$ are linearly dependent on the earlier ones whence we set $p_i :\equiv \pi_{\delta,l}^{-1}(r_l - \Delta_l + i) + k_l - 1$ for $i = 1, \ldots, \Delta_l$, where $\pi_{\delta,l}$ denotes the permutation represented by the matrix $\boldsymbol{\pi}_{\delta,l}$. At this point, we are led to another condition (apart from (2.8) and (2.9)) which must

be satisfied: All the indices $p_1, \ldots, p_{\Delta_l}$ must lie in the part of $\mathbf{z}_l$ which may be removed from $\mathbf{z}_l$, or as a formula:

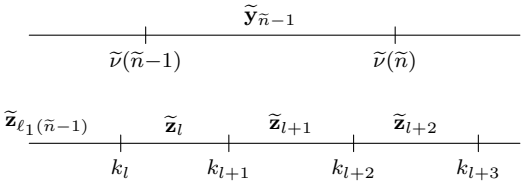
$$\min\{\, p_1, \ldots, p_{\Delta_l} \,\} \geq \max\{k_l, \nu(n-1)\}. \tag{2.10}$$

As can be seen from the preceding figures, this condition is always satisfied in case B but not necessarily in case A. If (2.10) is also fulfilled, we may choose for P any permutation which exchanges the right vectors $y_{p_1}, \ldots, y_{p_{\Delta_l}}$ with any Δ_l columns of $\mathbf{y}_{n(k_l)-1}$ which are not simultaneously columns of $\mathbf{z}_l$. For example, P might permute $y_{k_{l+1}}, \ldots, y_{k_{l+1}+\Delta_l-1}$ into the cluster $\mathbf{z}_l$. The application of this permutation P changes the recursion of the block $\mathbf{y}_{n-1}$ only in a trivial way (the rows of $\boldsymbol{\eta}_{n-1}$ get permuted by the inverse permutation $P^\star$, see line 6.18) and may overcome the problem of singular $\boldsymbol{\delta}_l$. If the problem still persists, we may repeat the same procedure with $\boldsymbol{\delta}_l^\star$ in place of $\boldsymbol{\delta}_l$, which possibly yields a permutation $\widetilde{P}$ of the left vectors to be tried out.

In case the above conditions are not fulfilled or if the singularity cannot be removed, then in line 6.26 Δ_l will be positive, and the choices of k_{l+1}, $\mathbf{z}_l$ and $\widetilde{\mathbf{z}}_l$ have to be changed. So in this case, a look-ahead procedure has to be started. Different look-ahead procedures are possible and characterize different variants of the algorithm.

Remark 1. At this point, we stress again that, in theory, the need for look-ahead only arises when $\boldsymbol{\delta}_l$ is singular *and there is no permutation of basis vectors of* $\mathbf{y}_{n-1}$ *or* $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ *leading to a nonsingular matrix* $\boldsymbol{\delta}_l$. In practice, however, it is cumbersome to try out several permutations so we may choose to start a look-ahead procedure if one permutation fails. Nevertheless we expect that look-ahead will occur less often than with the vectorwise approach as used in [1]. ○

Assuming that we have found a regular index k_{l+1}, the set of ungrouped left vectors is set to $\{\widetilde{y}_{k_{l+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}\}$. Now it is necessary to orthogonalize the ungrouped vectors against $\mathbf{z}_l$ in order to enlarge the zero matrices in (2.2) (and similarly for the right vectors, we will describe the process only for the left vectors because the extension of the left block Krylov space has already been described). For the following considerations, it might be helpful to keep a picture like the following in mind.



So the left cluster $\widetilde{\mathbf{z}}_l$ has just been completed. Consequently we can write $\widetilde{\mathbf{y}}_{\widetilde{n}-1} = [\,\widetilde{\mathbf{y}}_{\text{in}}\ \widetilde{\mathbf{y}}_{l,\widetilde{n}}\,]$ where $\widetilde{\mathbf{y}}_{\text{in}} :\equiv [\,\widetilde{y}_{\widetilde{\nu}(\widetilde{n}-1)}, \ldots, \widetilde{y}_{k_{l+1}-1}\,]$ (these vectors will not change any more) and $\widetilde{\mathbf{y}}_{l,\widetilde{n}} = [\,\widetilde{y}_{k_{l+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}\,]$. If $\widetilde{\mathbf{y}}_{l,\widetilde{n}}$ is nonempty, then its columns have to be orthogonalized against the recently completed right cluster $\mathbf{z}_l$. So we compute $\widetilde{\mathbf{y}}_{l,\widetilde{n}} - \widetilde{\mathbf{z}}_l \boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}}$ and this implies the following update of $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$

$$\begin{aligned}\widetilde{\mathbf{y}}_{\widetilde{n}-1} &= [\,\widetilde{\mathbf{y}}_{\text{in}}\ \ \widetilde{\mathbf{y}}_{l,\widetilde{n}} - \widetilde{\mathbf{z}}_l \boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}}\,] = \widetilde{\mathbf{y}}_{\widetilde{n}-1} - [\,\mathbf{0}_{N\times(k_{l+1}-\widetilde{\nu}(\widetilde{n}-1))}\ \ \widetilde{\mathbf{z}}_l \boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}}\,] \\ &= \widetilde{\mathbf{y}}_{\widetilde{n}-1} - \widetilde{\mathbf{z}}_l\,[\,\mathbf{0}_{r_l\times(k_{l+1}-\widetilde{\nu}(\widetilde{n}-1))}\ \ \boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}}\,].\end{aligned} \tag{2.11}$$

Now if $\widetilde{n} > 1$ we have to investigate the influence of this orthogonalization on the recurrence (2.7). To this end we multiply both sides of the above equation from the right by $\widetilde{\boldsymbol{\eta}}_{\widetilde{n}-1}$ and use (2.7) with $\widetilde{n}$ replaced by $\widetilde{n}-1$:

$$\begin{aligned}\widetilde{\mathbf{y}}_{\widetilde{n}-1}\widetilde{\boldsymbol{\eta}}_{\widetilde{n}-1} = \mathbf{A}^\star \widetilde{\mathbf{y}}_{\widetilde{n}-2} - \widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}-1),\widetilde{n}-1}\widetilde{\boldsymbol{\tau}}^y_{\ell_1(\widetilde{n}-1),\widetilde{n}-2} - \widetilde{\mathbf{z}}_{\ell_1(\widetilde{n}-1)}\widetilde{\boldsymbol{\tau}}^z_{\ell_1(\widetilde{n}-1),\widetilde{n}-2} - \cdots - \widetilde{\mathbf{z}}_0\widetilde{\boldsymbol{\tau}}^z_{0,\widetilde{n}-2} \\ - \widetilde{\mathbf{y}}^\triangle_{\widetilde{n}-1}\widetilde{\boldsymbol{\eta}}^\triangle_{\widetilde{n}-1} - \widetilde{\mathbf{z}}_l\,[\,\mathbf{0}_{r_l\times(k_{l+1}-\widetilde{\nu}(\widetilde{n}-1))}\ \ \boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}}\,]\,\widetilde{\boldsymbol{\eta}}_{\widetilde{n}-1}.\end{aligned}$$

Recall that $\widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}-1),\widetilde{n}-1} = [\,\widetilde{y}_{k_{\ell_1(\widetilde{n}-1)+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n}-1)-1}\,]$ are those left vectors which were ungrouped *when the block* $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ *was obtained first*, and at that point, the left clusters $\widetilde{\mathbf{z}}_0, \ldots, \widetilde{\mathbf{z}}_{\ell_1(\widetilde{n}-1)}$ were completed. By setting

$$\widetilde{\boldsymbol{\tau}}^z_{l,\widetilde{n}-2} :\equiv [\,\mathbf{0}_{r_l\times(k_{l+1}-\widetilde{\nu}(\widetilde{n}-1))}\ \ \boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}}\,]\,\widetilde{\boldsymbol{\eta}}_{\widetilde{n}-1} \tag{2.12}$$

we obtain

$$\widetilde{\mathbf{y}}_{\widetilde{n}-1}\widetilde{\boldsymbol{\eta}}_{\widetilde{n}-1} = \mathbf{A}^{\star}\widetilde{\mathbf{y}}_{\widetilde{n}-2} - \widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}-1),\widetilde{n}-1}\widetilde{\boldsymbol{\tau}}^{y}_{\ell_1(\widetilde{n}-1),\widetilde{n}-2} - \widetilde{\mathbf{z}}_{l}\widetilde{\boldsymbol{\tau}}^{z}_{l,\widetilde{n}-2} - \cdots - \widetilde{\mathbf{z}}_0\widetilde{\boldsymbol{\tau}}^{z}_{0,\widetilde{n}-2} - \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}-1}\widetilde{\boldsymbol{\eta}}^{\triangle}_{\widetilde{n}-1}. \tag{2.13}$$

At this point, we also have to renormalize those vectors which have been changed by the orthogonalization, which means that the corresponding rows of $\widetilde{\boldsymbol{\eta}}_{\widetilde{n}-1}$ have to be scaled.

As more clusters are completed, the block $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ may be updated again. Consider the example shown in the preceding figure. Here, when cluster $\widetilde{\mathbf{z}}_{l+1}$ is completed, the block $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ is still not exhausted because $k_{l+2} < \widetilde{\nu}(\widetilde{n})$ so that at least one column of $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ has to be orthogonalized against $\mathbf{z}_{l+1}$. By repeating the preceding argument we see that this orthogonalization leads to an additional term $-\widetilde{\mathbf{z}}_{l+1}\widetilde{\boldsymbol{\tau}}^{z}_{l+1,\widetilde{n}-2}$ in (2.13). But for the next regular index we have $k_{l+3} \geq \widetilde{\nu}(\widetilde{n})$ so that no column of $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ needs to be orthogonalized against $\mathbf{z}_{l+2}$. Since $\widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}$ is a column of $\widetilde{\mathbf{z}}_{l+2}$, it follows that $l+2 = \ell(\widetilde{\nu}(\widetilde{n}))$, and the last update of $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ happens when cluster $\ell(\widetilde{\nu}(\widetilde{n})) - 1$ is completed. Consequently, if we define

$$\ell_2(\widetilde{n}) :\equiv \ell(\widetilde{\nu}(\widetilde{n}+1)) - 1,$$

the final recurrence may be written as follows (as in (2.7), we write $\widetilde{n}$ instead of $\widetilde{n}-1$):

$$\widetilde{\mathbf{y}}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}_{\widetilde{n}} = \mathbf{A}^{\star}\widetilde{\mathbf{y}}_{\widetilde{n}-1} - \widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}),\widetilde{n}}\widetilde{\boldsymbol{\tau}}^{y}_{\ell_1(\widetilde{n}),\widetilde{n}-1} - \widetilde{\mathbf{z}}_{\ell_2(\widetilde{n})}\widetilde{\boldsymbol{\tau}}^{z}_{\ell_2(\widetilde{n}),\widetilde{n}-1} - \cdots - \widetilde{\mathbf{z}}_0\widetilde{\boldsymbol{\tau}}^{z}_{0,\widetilde{n}-1} - \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}^{\triangle}_{\widetilde{n}}. \tag{2.14}$$

So far, we have considered the recursion for a block which has to be updated according to (2.11) at least once. But what if there is no such update and (2.7) is already the final recurrence? This happens if and only if $k_{l+1} \geq \widetilde{\nu}(\widetilde{n})$ for the final choice of k_{l+1}. In this case, (2.14) is still valid since the vectors $\widetilde{y}_{\widetilde{\nu}(\widetilde{n}-1)}$ and $\widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}$ both belong to cluster $\widetilde{\mathbf{z}}_l$ so that $\ell_1(\widetilde{n}-1) = \ell_2(\widetilde{n}-1)$.

Remark 2. Equation (2.14) gives the final form of the recursion for any left block. But we actually have shown more: Assume that at some point l clusters have been completed and $\widetilde{s}_0, \ldots, \widetilde{s}_{\widetilde{n}-1}$ have been computed. Then for $0 \leq j \leq \widetilde{n}-1$ there are two cases:

- $\widetilde{\nu}(j+1) \leq k_l$: Then all columns of the block $\widetilde{\mathbf{y}}_j$ are part of a cluster. Since the completion of the next and all subsequent clusters will not affect its columns it will not change any more. So if $j > 0$ the recursion for $\widetilde{\mathbf{y}}_j$ is given by (2.14).
- $\widetilde{\nu}(j+1) > k_l$: In this case we only have a preliminary block $\widetilde{\mathbf{y}}_j$ at our disposal. As more clusters are completed, some of its columns may be changed due to the necessary orthogonalization of the ungrouped vectors. For the sake of simplicity, this preliminary block is also denoted by $\widetilde{\mathbf{y}}_j$, and if $j > 0$ then its preliminary recursion is given by

$$\widetilde{\mathbf{y}}_j\widetilde{\boldsymbol{\eta}}_j = \mathbf{A}^{\star}\widetilde{\mathbf{y}}_{j-1} - \widetilde{\mathbf{y}}_{\ell_1(j),j}\widetilde{\boldsymbol{\tau}}^{y}_{\ell_1(j),j-1} - \widetilde{\mathbf{z}}_{l-1}\widetilde{\boldsymbol{\tau}}^{z}_{l-1,j-1} - \cdots - \widetilde{\mathbf{z}}_0\widetilde{\boldsymbol{\tau}}^{z}_{0,j-1} - \widetilde{\mathbf{y}}^{\triangle}_{j}\widetilde{\boldsymbol{\eta}}^{\triangle}_{j}. \qquad \bigcirc$$

At step 3) the right block Krylov space is to be extended, but it first has to be decided how many vectors we need, which depends on the look-ahead strategy. Therefore we have to choose a value `minnum` at step LA4 (line 6.34). After that we proceed as described for the left blocks (see Algorithm 8) and end up with a (final) recurrence analogous to (2.14).

$$\mathbf{y}_n\boldsymbol{\eta}_n = \mathbf{A}\mathbf{y}_{n-1} - \mathbf{y}_{\ell_3(n),n}\boldsymbol{\tau}^{y}_{\ell_3(n),n-1} - \mathbf{z}_{\ell_4(n)}\boldsymbol{\tau}^{z}_{\ell_4(n),n-1} - \cdots - \mathbf{z}_0\boldsymbol{\tau}^{z}_{0,n-1} - \mathbf{y}^{\triangle}_{n}\boldsymbol{\eta}^{\triangle}_{n}, \tag{2.15}$$

where

$$\ell_3(n) :\equiv \ell(\nu(n)+1) - 1,$$
$$\ell_4(n) :\equiv \ell(\nu(n+1)) - 1.$$

Furthermore, if the first proposition for a look-ahead strategy is used (Section 2.3), then we can fix a value for k_{l+1} only at step LA5.

Definition 3. A *look-ahead strategy* for BLBIO consists of

- A specification of the steps LA1 to LA5 in Algorithms 6 and 8. The rule for choosing the variable `minnum` in step LA4 must ensure that the variable n has the value $n(k_{l+1})$ if the condition of **while** loop in line 8.1 becomes false.

Algorithm 8 Extension of right block Krylov space for BLBIO

1: **while** $\nu(n) <$ minnum **do**
2: Extend the right Krylov space $\rightarrow \mathbf{A}\mathbf{y}_{n-1}$.
3: Determine against which left clusters $\mathbf{A}\mathbf{y}_{n-1}$ needs to be orthogonalized.
4: Perform orthogonalization of $\mathbf{A}\mathbf{y}_{n-1} \rightarrow \mathbf{y}_{\mathrm{tmp}}$.
5: Check the vectors $[\, y_{k_l}, \ldots, y_{\nu(n)-1}, \mathbf{y}_{\mathrm{tmp}} \,]$ for deflation $\rightarrow s_n$, $\mathbf{y}_n$, $\nu(n+1)$, $s_n^{\triangle}$, $\mathbf{y}_n^{\triangle}$, $\nu^{\triangle}(n+1)$.
6: **if** $s_n = 0$ **then**
7: Set $l_t = l$, $n_t = n$, $\widetilde{n}_t = \widetilde{n}$, $k_{l_t+1} = \nu(n_t)$, $\widetilde{k}_{l_t+1} = \widetilde{\nu}(\widetilde{n}_t)$, $l = l + 1$.
8: Set $\mathbf{z}_{l_t} = [\, y_{k_{l_t}}, \ldots, y_{\nu(n_t)-1} \,]$, $\widetilde{\mathbf{z}}_{l_t} = [\, \widetilde{y}_{k_{l_t}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n}_t)-1} \,]$ and $\boldsymbol{\delta}_{l_t} = \widetilde{\mathbf{z}}_{l_t}^{\star} \widetilde{\mathbf{z}}_{l_t}$.
9: **stop**
10: **end if**
11: Set $n = n + 1$.
12: LA5 Fix value for k_{l+1} (changed or initial value).
13: **end while**

- An algorithm for computing the rank of the matrices $\boldsymbol{\delta}_l$ (to be used in lines 6.13 and 6.20).

The restriction on the rule for step LA4 in the first item of this definition just ensures that we only compute as many new right blocks as necessary. For the left blocks the analogue is automatically true, as was pointed out on page 16.

The process may terminate at lines 7.9 or 8.9 if one of the Krylov spaces is exhausted. We define the values of the indices n, $\widetilde{n}$ and l at termination as n_t, $\widetilde{n}_t$ and l_t, respectively. This definition implies that the last right and left blocks are given by $\mathbf{y}_{n_t-1}$ and $\widetilde{\mathbf{y}}_{\widetilde{n}_t-1}$, respectively. On the other hand, the remaining y and $\widetilde{y}$ vectors are put into clusters $\mathbf{z}_{l_t}$ and $\widetilde{\mathbf{z}}_{l_t}$, and if both of these final clusters are nonempty, then a final delta matrix $\boldsymbol{\delta}_{l_t}$ can be computed (lines 7.8 and 8.8). In contrast to all the other right and left clusters, these last clusters $\mathbf{z}_{l_t}$ and $\widetilde{\mathbf{z}}_{l_t}$ do not necessarily contain the same number of vectors. We denote the number of columns of $\mathbf{z}_{l_t}$ by r_{l_t} and the number of columns of $\widetilde{\mathbf{z}}_{l_t}$ by $\widetilde{r}_{l_t}$ so that the matrix $\boldsymbol{\delta}_{l_t}$ is of size $\widetilde{r}_{l_t} \times r_{l_t}$.

Remark 4. It may be worth mentioning that the matrix $\mathbf{D}_{\widetilde{n}_t, n_t} = \text{block diag}(\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{l_t-1}, \boldsymbol{\delta}_{l_t}) \in \mathbb{C}^{\widetilde{\nu}(\widetilde{n}_t) \times \nu(n_t)}$ is of slightly varying form, depdending on whether r_{l_t} and $\widetilde{r}_{l_t}$ are zero or not. We consider the following cases:

- $r_{l_t} = 0$ and $\widetilde{r}_{l_t} > 0$: Then we have

$$\begin{aligned} \nu(n_t) &= k_{l_t} = k_{l_t+1}, \\ \widetilde{\nu}(\widetilde{n}_t) &= \widetilde{k}_{l_t+1} = k_{l_t} + \widetilde{r}_{l_t} \end{aligned}$$

and

$$\mathbf{D}_{\widetilde{n}_t, n_t} = \left[\begin{array}{ccc} \boldsymbol{\delta}_0 & & \\ & \ddots & \\ & & \boldsymbol{\delta}_{l_t-1} \\ \hline & \mathbf{0}_{\widetilde{r}_{l_t} \times \nu(n_t)} & \end{array}\right].$$

- $r_{l_t} > 0$ and $\widetilde{r}_{l_t} = 0$: Then we have

$$\begin{aligned} \nu(n_t) &= k_{l_t} + r_{l_t}, \\ \widetilde{\nu}(\widetilde{n}_t) &= \widetilde{k}_{l_t+1} = k_{l_t} \end{aligned}$$

and

$$\mathbf{D}_{\widetilde{n}_t, n_t} = \left[\begin{array}{ccc|c} \boldsymbol{\delta}_0 & & & \\ & \ddots & & \mathbf{0}_{\widetilde{\nu}(\widetilde{n}_t) \times r_{l_t}} \\ & & \boldsymbol{\delta}_{l_t-1} & \end{array}\right].$$

- $r_{l_t} = 0$ and $\widetilde{r}_{l_t} = 0$: Then we have $\nu(n_t) = \widetilde{\nu}(\widetilde{n}_t) = k_{l_t}$ and $\mathbf{D}_{\widetilde{n}_t, n_t} = \text{block diag}(\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{l_t-1})$. ○

At this point, a few fundamental properties of BLBIO which do not depend on the particular look-ahead strategy can be stated. We first prove a key property of BLBIO, namely the three properties listed on page 12.

Lemma 5. *Assume that* BLBIO *is run with an arbitrary look-ahead procedure according to Definition 3. Then the following three properties always hold:*

$$\text{For } 0 \le i, j \le l-1\text{:} \quad \widetilde{\mathbf{z}}_i^\star \mathbf{z}_j = \begin{cases} \mathbf{0}, & i \ne j \\ \boldsymbol{\delta}_i, & i = j \end{cases} \tag{2.16a}$$

$$\text{For } i = 1, \ldots, l-1\text{:} \quad \begin{cases} \{y_j\}_{j \ge k_l} \perp \widetilde{\mathbf{z}}_i \\ \{\widetilde{y}_j\}_{j \ge \widetilde{k}_l} \perp \mathbf{z}_i \end{cases} \tag{2.16b}$$

$$\left.\begin{array}{l} \text{The vectors } \{y_j\}_{j \ge k_l} \text{ are linearly independent.} \\ \text{The vectors } \{\widetilde{y}_j\}_{j \ge \widetilde{k}_l} \text{ are linearly independent.} \end{array}\right\} \tag{2.16c}$$

Here it is understood that $\widetilde{k}_i :\equiv k_i$ *for* $i = 0, \ldots, l_t$. *After termination,* (2.16) *holds with* $l - 1 = l_t$.

So property (2.16a) means that all clusters which have been completed so far obey the biorthogonality, while (2.16b) states that those right and left vectors which have not yet been grouped into a cluster are already orthogonal to all the available left and right clusters, respectively. The last item (2.16c) states that all the ungrouped right vectors are linearly independent and the same for the left ones.

Proof. It is clear that these three properties hold when the main loop in line 6.8 is reached for the first time since (2.16a) and (2.16b) are empty at this point and the columns of $\mathbf{y}_0$ and $\widetilde{\mathbf{y}}_0$ are linearly independent.

Next we have to show that (2.16) forms an invariant of the loop starting at line 6.8. Assume that (2.16) holds at the beginning of the main loop. If Algorithm 7 is entered, the **while** loop on line 7.1 is executed. We assume that the three properties are valid at the beginning of the **while** loop. The block $\widetilde{\mathbf{y}}_{\text{tmp}}$ is orthogonalized against all available right clusters so that (2.16b) still holds when line 7.6 is reached, while (2.16a) is not touched so far. In line 7.5, $\widetilde{\mathbf{y}}_{\text{tmp}}$ is first orthogonalized against $\widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}),\widetilde{n}} = [\, \widetilde{y}_{k_{\ell_1(\widetilde{n})+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1} \,]$. As the columns of $\widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}),\widetilde{n}}$ are just the ungrouped left vectors, we know by (2.16c) that they are linearly independent, and so the inverse in (2.6) exists. Now, after checking the columns of $\widetilde{\mathbf{y}}_{\text{tmp}}$ for deflation, new left vectors (namely the columns of $\widetilde{\mathbf{y}}_{\widetilde{n}}$) are obtained and added to the set of ungrouped vectors, which now consists of $\widetilde{y}_{k_{\ell_1(\widetilde{n})+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n}+1)-1}$ (i.e. the columns of $\widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}),\widetilde{n}}$ and $\widetilde{\mathbf{y}}_{\widetilde{n}}$). But since the blocks $\widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}),\widetilde{n}}$ and $\widetilde{\mathbf{y}}_{\widetilde{n}}$ are orthogonal, the ungrouped vectors are still linearly independent and (2.16c) remains valid. We conclude that all three properties are still valid when line 7.6 is reached.

If $\widetilde{s}_{\widetilde{n}}$ is zero, we exit the algorithm by creating the two final clusters $\mathbf{z}_{l_t}$ and $\widetilde{\mathbf{z}}_{l_t}$. Consequently, at line 7.9, (2.16a) holds with $l - 2 = l_t - 1$ in place of $l - 1$, but due to (2.16b) and the definition of $\boldsymbol{\delta}_{l_t}$, (2.16a) also holds for $l - 1 = l_t$. Due to the definitions of the last regular indices k_{l_t+1} and $\widetilde{k}_{l_t+1}$, the sets $\{y_j\}_{j \ge k_l}$ and $\{\widetilde{y}_j\}_{j \ge \widetilde{k}_l}$ are empty. So we have shown that if BLBIO stops at line 7.9, then (2.16) holds with $l = l_t + 1$.

Assuming that BLBIO continues, it follows that (2.16a) to (2.16c) are still valid at the end of the **while** loop 7.1 and therefore also at the end of Algorithm 7. Next, lines 6.12 to 6.22 do not affect the three properties since the clusters $\mathbf{z}_l$ and $\widetilde{\mathbf{z}}_l$ obtained in line 6.12 are preliminary and application of a permutation P in line 6.17 does not influence the linear independency of the right vectors. In the **if** statement on line 6.26, new clusters are completed in the **else** branch, but due to line 6.29 (the remaining vectors are orthogonalized against the new clusters), both (2.16a) and (2.16b) are still valid at line 6.32. Thus we have to show that (2.16c) also remains valid at line 6.32, which amounts to showing that it is not destroyed at line 6.29. Here, for the left vectors, an update of the

form (2.11) is done, and if we first set $\widetilde{\mathbf{y}}'_\perp :\equiv \widetilde{\mathbf{y}}_{l,\widetilde{n}} - \widetilde{\mathbf{z}}_l \boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}}$, the assertion is that the columns of $\widetilde{\mathbf{y}}'_\perp$ are linearly independent. Let $t_\perp \in \mathbb{C}^{\widetilde{\nu}(\widetilde{n})-k_{l+1}}$ and consider

$$0 = \widetilde{\mathbf{y}}'_\perp t_\perp = \widetilde{\mathbf{y}}_{l,\widetilde{n}} t_\perp - \widetilde{\mathbf{z}}_l \boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}} t_\perp = \widetilde{\mathbf{y}}_{l,\widetilde{n}} t_\perp + \widetilde{\mathbf{z}}_l \overline{t}_\perp = [\, \widetilde{y}_{k_l}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1} \,] \begin{bmatrix} \overline{t}_\perp \\ t_\perp \end{bmatrix}$$

where $\overline{t}_\perp :\equiv -\boldsymbol{\delta}_l^{-\star} \mathbf{z}_l^\star \widetilde{\mathbf{y}}_{l,\widetilde{n}} t_\perp$. Now we note that $\widetilde{y}_{k_l}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}$ are just the ungrouped vectors before completion of $\widetilde{\mathbf{z}}_l$, so we may apply (2.16c) to obtain that they are linearly independent, whence $t_\perp = 0$, which proves the assertion. The normalization of the columns of $\widetilde{\mathbf{y}}'_\perp$ is harmless and therefore (2.16c) is also valid after line 6.29.

A similar argument as above shows that the extension of the right block Krylov space (if done) does not change the validity of (2.16). Therefore, the properties (2.16) are valid at line 6.36 if the algorithm does not terminate and hold with $l = l_t + 1$ after termination of BLBIO. □

As a corollary of this Lemma, we obtain the assertion made in the paragraph after equation (2.7). But the Lemma also enables us to establish some more basic properties of BLBIO:

Proposition 6. *Assume that* BLBIO *is run with an arbitrary look-ahead procedure according to Definition 3. Then we have*

(i) All y and $\widetilde{y}$ vectors are of unit length.

(ii) Outside of step 3), the number of (possibly preliminary) y vectors computed so far is given by $\nu(n)$, while outside of step 1), the number of (possibly preliminary) $\widetilde{y}$ vectors computed so far is $\widetilde{\nu}(\widetilde{n})$. In other words, the right blocks $\mathbf{y}_0, \ldots, \mathbf{y}_{n-1}$ are available outside of step 3) and the left blocks $\widetilde{\mathbf{y}}_0, \ldots, \widetilde{\mathbf{y}}_{\widetilde{n}-1}$ are available outside of step 1), and the blocks $\mathbf{y}_{n-1}$ and $\widetilde{\mathbf{y}}_{\widetilde{n}-1}$ may be preliminary.

(iii) Outside of step 2), the right clusters $\mathbf{z}_0, \ldots, \mathbf{z}_{l-1}$ and the left clusters $\widetilde{\mathbf{z}}_0, \ldots, \widetilde{\mathbf{z}}_{l-1}$ have been completed.

(iv) After termination, (ii) and (iii) hold with $n = n_t$, $\widetilde{n} = \widetilde{n}_t$ and $l = l_t + 1$, and all computed right and left vectors are grouped into a cluster.

(v) The right and left clusters $\mathbf{z}_i$ and $\widetilde{\mathbf{z}}_i$ both have r_i columns for $i = 0, \ldots, l_t - 1$. The last right and left clusters contain r_{l_t} and $\widetilde{r}_{l_t}$ vectors, respectively. Additionally, the biorthogonality (2.3) *holds for $0 \le i, j \le l_t$.*

(vi) In exact arithmetic an with a zero deflation tolerance, the right and left vector sequences form nested bases of $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ and $\widetilde{\mathcal{B}}_n(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$, respectively. In particular, the columns of $\mathbf{y}_n$ span $\mathcal{B}_{n+1}(\mathbf{A}, \mathbf{y}_0) \ominus \mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ $(n = 0, \ldots, n_t - 1)$, while the columns of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ span $\widetilde{\mathcal{B}}_{\widetilde{n}+1}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0) \ominus \widetilde{\mathcal{B}}_{\widetilde{n}}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$ $(\widetilde{n} = 0, \ldots, \widetilde{n}_t - 1)$.

(vii) When clusters $\mathbf{z}_{l-1}$ and $\widetilde{\mathbf{z}}_{l-1}$ are completed, the variables n and $\widetilde{n}$ have the values $n(k_l)$ and $\widetilde{n}(k_l)$, respectively. In other words, BLBIO *only computes as many right and left blocks as necessary for testing regular indices.*

Proof. (i) follows from the fact that a QR decomposition is used as deflation checking procedure so that normalized vectors are generated and that in line 6.29, the orthogonalized vectors are again normalized. The assertions (ii) and (iii) follow readily from Algorithms 6 to 8. (iv) and (v) follow easily from Lemma 5, observing that after termination, the sets $\{y_j\}_{j \ge k_l}$ and $\{\widetilde{y}_j\}_{j \ge k_l}$ are empty.

Next, we show assertion (vi) only for the right vectors. From the construction of the right vectors, it follows that $\mathcal{B}_i(\mathbf{A}, \mathbf{y}_0) = \operatorname{span}(y_0, \ldots, y_{\nu(i)-1})$ for $i = 1, \ldots, n_t$. To show that the vectors $\{y_0, \ldots, y_{\nu(i)-1}\}$ are linearly independent, we temporarily define $l :\equiv \ell(\nu(i))$ and then take a linear combination of the zero vector with complex coefficients $\lambda_0, \ldots, \lambda_{\nu(i)-1}$:

$$0 = \sum_{k=0}^{\nu(i)-1} \lambda_k y_k = \sum_{k=0}^{k_l-1} \lambda_k y_k + \sum_{k=k_l}^{\nu(i)-1} \lambda_k y_k = \sum_{n=0}^{l-1} \mathbf{z}_n t_n + \sum_{k=k_l}^{\nu(i)-1} \lambda_k y_k$$

where $t_n :\equiv (\lambda_{k_n}, \ldots, \lambda_{k_{n+1}-1})^\top$. Using the biorthogonality relations (2.16a) and (2.16b), we get for $j = 0, \ldots, l-1$:

$$0 = \widetilde{\mathbf{z}}_j^\star \sum_{k=0}^{\nu(i)-1} \lambda_k y_k = \boldsymbol{\delta}_j t_j,$$

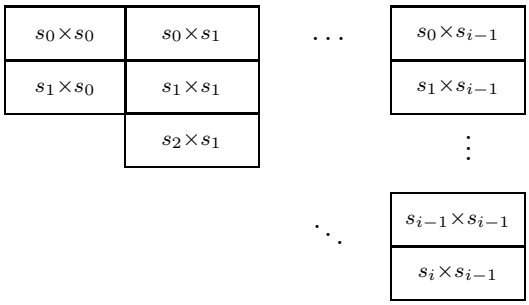


Figure 1: Block Hessenberg structure of $\underline{\mathbf{T}}_i$

which implies $t_j = 0$. Now we look at the vectors $y_{k_l}, \ldots, y_{\nu(i)-1}$ and note that they are precisely the ungrouped vectors after cluster $\mathbf{z}_{l-1}$ has been obtained because the definition of l above implies that they are not changed any more when clusters $\mathbf{z}_l$ and $\widetilde{\mathbf{z}}_l$ are completed. So we may apply (2.16c), which yields that these vectors are linearly independent and consequently the remaining coefficients $\lambda_{k_l}, \ldots, \lambda_{\nu(i)-1}$ also must be zero. This completes the proof of (vi). Finally, (vii) follows from Algorithm 7 and Definition 3. □

As a corollary of this proposition, we also obtain the recursions (2.14) and (2.15), and Remark 2. Our next goal is to write the recursions in matrix form, which we carry out only for the right vectors. An analogous statement holds for the left vectors.

Proposition 7 (Recursions). *Assume that the right vectors $y_0, \ldots, y_{\nu(p+1)-1}$ (that is, the blocks $\mathbf{y}_0, \ldots, \mathbf{y}_p$) have been grouped into a cluster[3]. The recursions* (2.15) *lead to the matrix relation*

$$\mathbf{A}\mathbf{Y}_i = \mathbf{Y}_{i+1}\underline{\mathbf{T}}_i + \mathbf{Y}_{i+1}^{\triangle}\underline{\mathbf{T}}_i^{\triangle} \qquad i = 1, \ldots, p, \tag{2.17}$$

where the matrix $\underline{\mathbf{T}}_i \in \mathbb{C}^{\nu(i+1)\times\nu(i)}$ is of upper block Hessenberg structure (see Figure 1), and we set

$$\nu^{\triangle}(i+1) :\equiv \sum_{j=0}^{i} s_i^{\triangle},$$

$$\mathbf{Y}_i^{\triangle} :\equiv [\,\mathbf{y}_0^{\triangle}, \ldots, \mathbf{y}_{i-1}^{\triangle}\,] \in \mathbb{C}^{N\times\nu^{\triangle}(i)},$$

$$\underline{\mathbf{T}}_i^{\triangle} :\equiv \begin{bmatrix} \mathbf{0}_{s_0^{\triangle}\times s_0} & & & \\ \boldsymbol{\eta}_1^{\triangle} & \mathbf{0}_{s_1^{\triangle}\times s_1} & & \\ & \ddots & \ddots & \\ & & \ddots & \mathbf{0}_{s_{i-1}^{\triangle}\times s_{i-1}} \\ & & & \boldsymbol{\eta}_i^{\triangle} \end{bmatrix} \in \mathbb{C}^{\nu^{\triangle}(i+1)\times\nu(i)}.$$

After termination of BlBiO, (2.17) *holds with $p = n_t - 1$. In the case of exact arithmetic and with the deflation tolerance set to zero, we get*

$$\mathbf{A}\mathbf{Y}_i = \mathbf{Y}_{i+1}\underline{\mathbf{T}}_i \qquad i = 1, \ldots, p. \tag{2.18}$$

Proof. Let $0 < j \le p$ and consider the (final) recursion for the block $\mathbf{y}_j$, which is given by (2.15) with n replaced by j. We can easily rewrite the right-hand side of (2.15) without explicit reference to the right clusters. To this end we set for $0 \le i \le \ell_4(j)$

$$\mathbf{S}_i^z :\equiv \begin{bmatrix} \mathbf{0}_{k_i\times r_i} \\ \mathbb{1}_{r_i} \\ \mathbf{0}_{(\nu(j+1)-k_{i+1})\times r_i} \end{bmatrix} \in \mathbb{R}^{\nu(j+1)\times r_i}$$

[3]This assumption is not necessary to show the assertion, but it makes the proof a little easier, and only this case will be needed later.

so that $\mathbf{z}_i = \mathbf{Y}_{j+1}\mathbf{S}_i^z$ for $0 \le i \le \ell_4(j)$. Furthermore, we set

$$\mathbf{S}_{\ell_3(j),j} :\equiv \begin{bmatrix} \mathbf{0}_{k_{\ell_3(j)+1}\times(\nu(j)-k_{\ell_3(j)+1})} \\ \mathbb{1}_{\nu(j)-k_{\ell_3(j)+1}} \\ \mathbf{0}_{s_j\times(\nu(j)-k_{\ell_3(j)+1})} \end{bmatrix} \in \mathbb{R}^{\nu(j+1)\times(\nu(j)-k_{\ell_3(j)+1})}$$

so that $\mathbf{y}_{\ell_3(j),j} = [\, y_{k_{\ell_3(j)+1}}, \ldots, y_{\nu(j)-1}\,] = \mathbf{Y}_{j+1}\mathbf{S}_{\ell_3(j),j}$. And finally

$$\mathbf{S}_j :\equiv \begin{bmatrix} \mathbf{0}_{\nu(j)\times s_j} \\ \mathbb{1}_{s_j} \end{bmatrix} \in \mathbb{R}^{\nu(j+1)\times s_j}$$

so that $\mathbf{y}_j = \mathbf{Y}_{j+1}\mathbf{S}_j$. Substituting these definitions into the recursion (2.15) yields $\mathbf{A}\mathbf{y}_{j-1} = \mathbf{Y}_{j+1}\mathbf{F}_j + \mathbf{y}_j^{\triangle}\boldsymbol{\eta}_j^{\triangle}$ with

$$\mathbf{F}_j :\equiv \mathbf{S}_j\boldsymbol{\eta}_j + \mathbf{S}_{\ell_3(j),j}\boldsymbol{\tau}^y_{\ell_3(j),j} + \mathbf{S}^z_{\ell_4(j)}\boldsymbol{\tau}^z_{\ell_4(j),j-1} + \cdots + \mathbf{S}^z_0\boldsymbol{\tau}^z_{0,j-1} \in \mathbb{C}^{\nu(j+1)\times s_{j-1}}.$$

If we denote the rows $\nu(i)+1, \ldots, \nu(i+1)$ of $\mathbf{F}_j$ by $\boldsymbol{\tau}_{i,j-1}$ for $i = 0, \ldots, j$, we obtain

$$\mathbf{A}\mathbf{y}_{j-1} = \mathbf{y}_j\boldsymbol{\tau}_{j,j-1} + \mathbf{y}_{j-1}\boldsymbol{\tau}_{j-1,j-1} + \cdots + \mathbf{y}_0\boldsymbol{\tau}_{0,j-1} + \mathbf{y}_j^{\triangle}\boldsymbol{\eta}_j^{\triangle}.$$

The recursion for $\mathbf{y}_j$ determines columns $\nu(j-1)+1, \ldots, \nu(j)$ of $\underline{\mathbf{T}}_j$ and the nonzero elements in these columns are given by $\underline{\mathbf{T}}_j(1:\nu(j+1), \nu(j-1)+1:\nu(j)) :\equiv [\,\boldsymbol{\tau}^\star_{0,j-1}, \ldots, \boldsymbol{\tau}^\star_{j,j-1}\,]^\star$. Consequently we now obtain (2.17). After termination, the same argument works for any $j \in \{1, \ldots, n_t - 1\}$. □

The following proposition shows that, at least in the case of exact arithmetic, these recurrences do not involve all the previously computed right and left clusters. It also gives a criterion for the matrix coefficients $\boldsymbol{\tau}^z_{i,n-1}$ and $\widetilde{\boldsymbol{\tau}}^z_{i,\widetilde{n}-1}$ in (2.14) and (2.15) to be zero. For the sake of brevity, we only give the formulation for the right blocks.

Proposition 8. *Assume exact arithmetic and a zero deflation tolerance and set* $l :\equiv \ell\big(\min\{\nu(n_t - 1), \widetilde{\nu}(\widetilde{n}_t - 1)\}\big)$. *Now we define* $\widetilde{\psi}(j) :\equiv k_{\ell(\widetilde{\nu}(j))+1}$ *and compute sequences* $\{b_i\}_{i\ge1}$, $\{i_j\}_{j\ge2}$ *by Algorithm 9. Finally, we set*

$$\phi(i) :\equiv k_{b_q} \qquad \text{if } \widetilde{\psi}(i_q) < i \le \widetilde{\psi}(i_{q+1}).$$

Then from the relations (2.18) *and the analogue for the left vectors it follows*

(i) In column i *of* $\underline{\mathbf{T}}_{n_t-1}$ *the elements* $1, \ldots, \phi(i)$ *and* $\nu(n(i)+2)+1, \ldots, \nu(n_t)$ *are zero. If* $\phi(i) = 0$, *then there may not be any zeros above the main diagonal in column* i.

(ii) In (2.15), *for* $n > 0$, *the matrix coefficient* $\boldsymbol{\tau}^z_{i,n-1}$ *is zero if* $k_{i+1} \le \phi(\nu(n-1)+1)$.

Algorithm 9 Auxiliary indices for ϕ

```
j = 1, a_1 = 2, b_1 = 0
for k = 2, 3, ... do
    a_k = max{ i | ψ̃(i) = ψ̃(a_{k-1}) }
    b = max{ l | k_l ≤ ν̃(a_k − 1) }
    if b > b_j then
        j = j + 1
        b_j = b, i_j = a_k
    end if
end for
```

Proof. It follows from the block Hessenberg structure of $\underline{\mathbf{T}}_n$ that elements $\nu(n(i)+2)+1, \ldots, \nu(n_t)$ of column i are zero. Now we set $p :\equiv n_t - 1$ and $q :\equiv \widetilde{n}_t - 1$ and multiply (2.18) from the left by $\widetilde{\mathbf{Y}}_q^\star$ and the recursion for the left vectors by $\mathbf{Y}_p^\star$, which yields

$$\widetilde{\mathbf{Y}}_q^\star\mathbf{A}\mathbf{Y}_p = \widetilde{\mathbf{Y}}_q^\star\mathbf{Y}_{p+1}\underline{\mathbf{T}}_p = \widetilde{\mathbf{Y}}_q^\star\left[\mathbf{Y}_p,\, \mathbf{y}_p\right]\underline{\mathbf{T}}_p = \left[\mathbf{D}_{q,p}\,|\,\star_{\widetilde{\nu}(q)\times s_p}\right]\underline{\mathbf{T}}_p$$
$$\mathbf{Y}_p^\star\mathbf{A}^\star\widetilde{\mathbf{Y}}_q = \mathbf{Y}_p^\star\widetilde{\mathbf{Y}}_{q+1}\underline{\widetilde{\mathbf{T}}}_q = \mathbf{Y}_p^\star\left[\widetilde{\mathbf{Y}}_q,\, \widetilde{\mathbf{y}}_q\right]\underline{\widetilde{\mathbf{T}}}_q = \left[\mathbf{D}^\star_{q,p}\,|\,\star_{\nu(p)\times\widetilde{s}_q}\right]\underline{\widetilde{\mathbf{T}}}_q$$

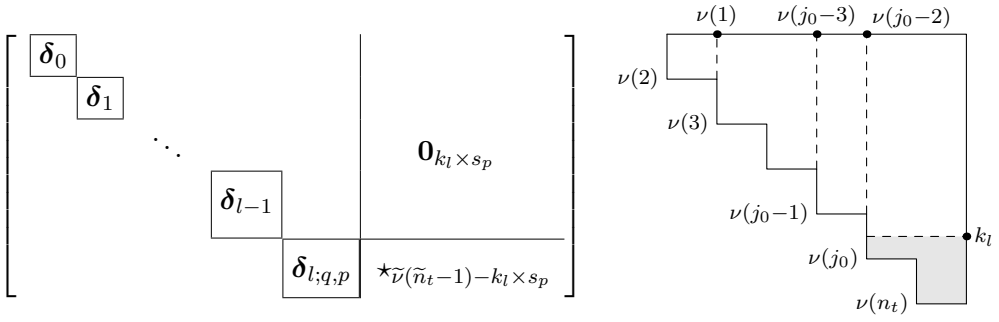


Figure 2: Left hand side of (2.19)

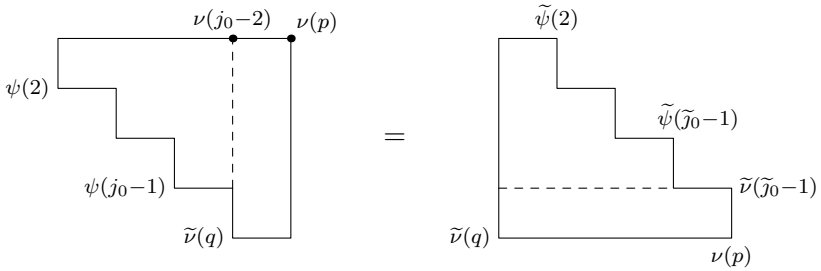
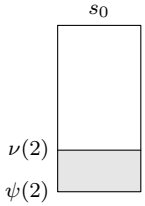


Figure 3: Nonzero patterns in (2.19)

and therefore

$$\left[\mathbf{D}_{q,p} \,|\, \star_{\widetilde{\nu}(q)\times s_p}\right] \underline{\mathbf{T}}_p = \widetilde{\underline{\mathbf{T}}}_q^{\star} \left[\begin{array}{c} \mathbf{D}_{q,p} \\ \hline \star_{\widetilde{s}_q \times \nu(p)} \end{array}\right]. \tag{2.19}$$

We first consider the left hand side of this equation, see Figure 2. The first k_l rows of $\underline{\mathbf{T}}_p$ are multiplied by $\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{l-1}$ while the rest of of $\underline{\mathbf{T}}_p$ is multiplied by $\boldsymbol{\delta}_{l;q,p}$ and the starred part of the left factor. First we consider the columns $1, \ldots, \nu(1)$ of the product matrix. In the corresponding columns of $\underline{\mathbf{T}}_p$, the rows $\nu(2)+1, \ldots, \nu(n_t)$ are zero, and we see that the matrices $\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{\ell(\nu(2))}$ contribute to the product so that the nonzero pattern is extended as shown in the following figure, where the shaded area marks the additional nonzero elements.



In an analogous way, we consider columns $\nu(i)+1, \ldots, \nu(i+1) \le \nu(j_0-2)$ of the product, where $j_0 = n(k_l)$. Now the nonzero elements are extended until row $\psi(i+2)$. Finally we turn to the remaining columns, i.e. $\nu(j_0-2)+1, \ldots, \nu(p)$. Since the shaded area of $\underline{\mathbf{T}}_p$ is multiplied by $\boldsymbol{\delta}_{l;q,p}$ and the starred part of the left factor, we get in general fill in until the last row. Therefore the left hand side of (2.19) has a nonzero pattern as shown in the left part of Figure 3. The same argument as above shows that the nonzero pattern of the right-hand side of (2.19) is given by the right part of Figure 3. Now we know that $l-1 < l_t$ so that the matrices $\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{l-1}$ are all nonsingular, and from this we can derive that $\underline{\mathbf{T}}_{n_t-1}$ has a nonzero pattern as illustrated by Figure 4. Writing the number of leading zeros in column i as a formula leads to the step function $\phi(i)$ as defined above. Algorithm 9 computes a sequence $i_2, i_3, \ldots$ such that the number of leading zeros of $\underline{\mathbf{T}}_n$ increases at column $\widetilde{\psi}(i_j)+1$. To this end, we first have to find out where $\widetilde{\psi}(i)$ jumps, and the sequence $a_1, a_2, \ldots$ stores this information. But even if $\widetilde{\psi}(i)$ increases, this does not necessarily mean that the number of leading zeros of $\underline{\mathbf{T}}_n$ also increases. Instead, in line 9.4, we know that there are at most $\widetilde{\nu}(a_k-1)$ zeros in column $\widetilde{\psi}(a_k)+1$, and we need to check whether more clusters fit into rows $1, \ldots, \widetilde{\nu}(a_k-1)$. If the condition $b > b_j$ is true, we have a step in the matrix $\underline{\mathbf{T}}_n$ at column $\widetilde{\psi}(a_k)+1$, with the new number of zeros being given by k_{b_j}.

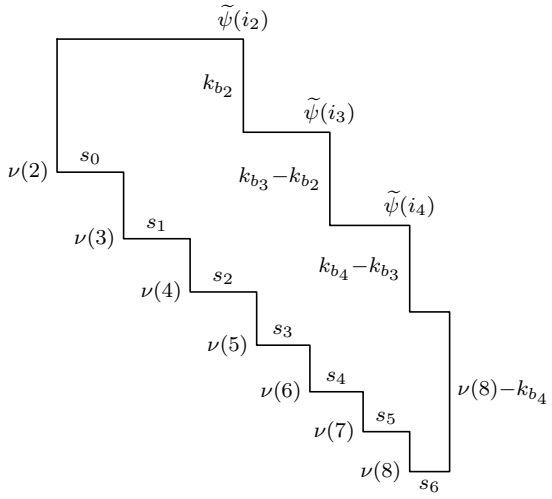


Figure 4: Nonzero structure of $\underline{\mathbf{T}}_{n_t-1}$

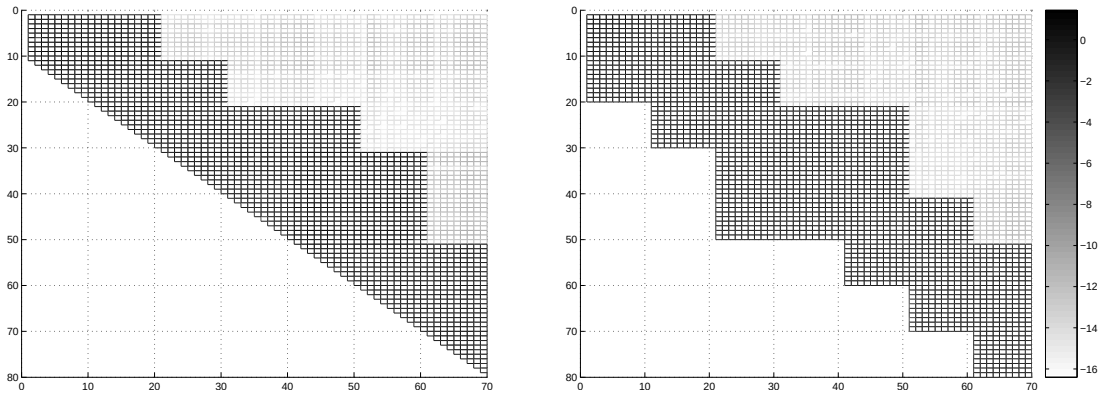


Figure 5: Absolute values of entries of $\underline{\mathbf{T}}_7$ (left) and of right-hand side of (2.19) (right)

This shows assertion (i) of the proposition. We note that in practice, we cannot compute the maximum in line 9.3 but need to estimate it. So if really large clusters extending over several blocks occur, it may happen that we do superfluous orthogonalizations.

The last vector of the cluster $\mathbf{z}_i$ is $y_{k_{i+1}-1}$ while the first columns of $\boldsymbol{\tau}_{0,n-1}, \ldots, \boldsymbol{\tau}_{n,n-1}$ are part of column $\nu(n-1)+1$ of $\underline{\mathbf{T}}_n$. So in equation (2.15) the coefficient $\boldsymbol{\tau}^z_{i,n-1}$ is zero if the index k_{i+1} falls into the upper band of zeros of column $\nu(n-1)+1$ of $\underline{\mathbf{T}}_{n_t-1}$, and we get the inequality $k_{i+1} \leq \phi(\nu(n-1)+1)$ as a condition for $\boldsymbol{\tau}^z_{i,n-1}$ to be zero. □

Example 9. It may be a good point here to take a look at two examples. We take a 100×100 matrix of random entries and use the first variant of look-ahead strategy discussed in Section 2.3. Furthermore, full orthogonalization is performed whenever a new block has to be orthogonalized against the available clusters. The right and left starting blocks contain ten random vectors each, and one look-ahead step is forced artificially when regular index k_4 is preliminary. Since neither further look-ahead nor deflation occurred until completion of $\mathbf{z}_6$ and $\widetilde{\mathbf{z}}_6$, $\mathbf{z}_3$ and $\widetilde{\mathbf{z}}_3$ contain 20 vectors while all the other clusters contain 10. The structure of $\underline{\mathbf{T}}_7$ is shown in the left part of Figure 5. The absolute values of the matrix elements are plotted according to the colorbar on the right, with a logarithmic scale being used. In this case, the right-hand side of (2.19) looks as in the right part of Figure 5.

If deflation occurs, then the matrix $\underline{\mathbf{T}}$ in general does not exhibit such a regular lower band structure as in the previous case. For example, take the same matrix $\mathbf{A}$ but this time, the right starting block is

$$\mathbf{y}_0 = [\, \mathbf{A}y_7 \; y_1 \; \ldots \; y_7 \,]$$

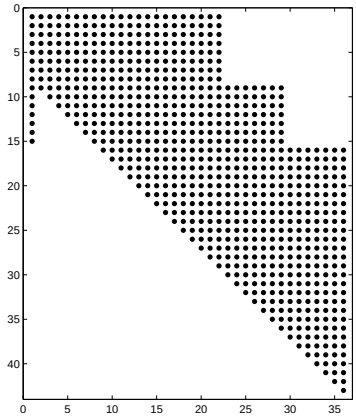
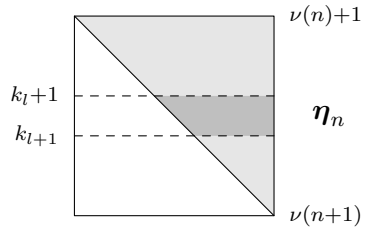


Figure 6: Elements of $\underline{\mathbf{T}}_5$ with absolute value larger than 10^{-8}

where $y_1, \ldots, y_7 \in \mathbb{C}^{100}$ are random vectors. As in the previous example, the left starting block contains ten random vectors, and full orthogonalization was done. Figure 6 shows those elements t_{ij} of $\underline{\mathbf{T}}_5$ for which $|t_{ij}| > 10^{-8}$.

This example illustrates that as long as neither deflation nor permutations occur, the matrix $\underline{\mathbf{T}}_n$ has lower bandwidth s_0+1 (an analogous statement holds for $\underline{\widetilde{\mathbf{T}}}_{\widetilde{n}}$). To prove this, we consider matrix $\boldsymbol{\eta}_n$ and show that the band structure is not destroyed when some columns of the preliminary block $\mathbf{y}_n$ have to be orthogonalized against some left cluster $\widetilde{\mathbf{z}}_l$. Consider the analogue of (2.12) for the right blocks. Since $\boldsymbol{\eta}_n$ is upper triangular, the first $k_{l+1} - \nu(n)$ columns of $\boldsymbol{\tau}^z_{l,n-1}$ are zero. Now we note that the first row of $\boldsymbol{\eta}_n$ affected by $\boldsymbol{\tau}^z_{n-1}$ is $k_l + 1 - \nu(n)$ (see the following figure), and in this row, the first $k_l - \nu(n)$ elements are zero. As $k_l - \nu(n) < k_{l+1} - \nu(n)$, we get the assertion.



Remark 10. It should be emphasized that we do not make any assumptions about the matrices $\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{l_t-1}$ except that they are nonsingular. In particular, they need not be diagonal. It is in general not possible to find vector sequences $\{w_i\}_{i=0}^{\nu(n_t)-1}$ and $\{\widetilde{w}_j\}_{j=0}^{\widetilde{\nu}(\widetilde{n}_t)-1}$ with the following properties:

1) The columns of $\mathbf{W}_n :\equiv [\, w_0, \ldots, w_{\nu(n)-1} \,]$ span $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$, $n = 1, \ldots, n_t$ while the columns of $\widetilde{\mathbf{W}}_{\widetilde{n}} :\equiv [\, \widetilde{w}_0, \ldots, \widetilde{w}_{\widetilde{\nu}(\widetilde{n})-1} \,]$ span $\widetilde{\mathcal{B}}_{\widetilde{n}}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$ for $\widetilde{n} = 1, \ldots, \widetilde{n}_t$.
2) If we set $\mathbf{w}_l :\equiv [\, w_{k_l}, \ldots, w_{k_{l+1}-1} \,]$ (with an analogous definition for the $\widetilde{w}$ vectors) for $l = 0, \ldots, l_t$, then the biorthogonality becomes

$$\widetilde{\mathbf{w}}_i^\star \mathbf{w}_j = \begin{cases} \mathbf{0} & i \neq j \\ \boldsymbol{\delta}_i^w & i = j \end{cases}$$

with *diagonal* $r_i \times r_i$ matrices $\boldsymbol{\delta}_i^w$ if $0 \leq i \leq l_t - 1$. The matrix $\boldsymbol{\delta}_{l_t}^w$ is of size $\widetilde{r}_{l_t} \times r_{l_t}$ and if it is nonempty, then it has nonzeros only on the main diagonal.

To prove this we derive from 1) that $\mathbf{Y}_n = \mathbf{W}_n \mathbf{U}_n$ with some block upper triangular matrix $\mathbf{U}_n$ with blocks of size $s_i \times s_i$ on the diagonal for $i = 0, \ldots, n-1$. This implies

$$\mathbf{D}_{\widetilde{n}_t, n_t} = \widetilde{\mathbf{Y}}^\star_{\widetilde{n}_t} \mathbf{Y}_{n_t} = \widetilde{\mathbf{U}}^\star_{\widetilde{n}_t} \widetilde{\mathbf{W}}^\star_{\widetilde{n}_t} \mathbf{W}_{n_t} \mathbf{U}_{n_t}. \tag{2.20}$$

Such a decomposition of $\mathbf{D}_{\widetilde{n}_t, n_t}$ may not exist, however, as can be seen from the following simple example: Assume that we only have one right and left starting vector, i.e. $s_0 = \widetilde{s}_0 = 1$. Then we

have $s_i = 1$ for $i = 0, \ldots, n_t - 1$ and $\widetilde{s}_j = 1$ for $j = 0, \ldots, \widetilde{n}_t - 1$. Thus $\mathbf{U}_{n_t}$ as well as $\widetilde{\mathbf{U}}_{\widetilde{n}_t}$ are upper triangular matrices so that (2.20) amounts to a (possibly rectangular) LDU decomposition of $\mathbf{D}_{\widetilde{n}_t, n_t}$. But if look-ahead occurs, then we may get a matrix $\boldsymbol{\delta}_l$ which does not have an LU decomposition without pivoting (independently of the look-ahead strategy chosen), and therefore the rectangular LDU decomposition of $\mathbf{D}_{\widetilde{n}_t, n_t}$ may not exist.

Thus the conditions 1) and 2) can not always be fulfilled. With our definition of the right and left clusters, we have chosen to give up condition 2) above as is common if look-ahead is used. But in contrast to the algorithm presented in [1], our algorithm may generate nondiagonal $\boldsymbol{\delta}_l$ matrices even if no look-ahead occurs (this depends on the look-ahead strategy used, see Section 2.3). ○

2.3 Look-ahead strategies

The BLBIO algorithm as outlined in the previous section does not involve a choice of a look-ahead strategy. This allows us to consider various possibilities and then to test which of them leads to the best performance. In this subsection we deal with the question whether we should aim for clusters of size one if possible (as the authors do in [1]) or whether larger clusters should be preferred. Two variations with differring emphasis are discussed.

2.3.1 Rank computations

According to Definition 3, every look-ahead strategy for BLBIO incorporates an algorithm for computing the rank of the matrices $\boldsymbol{\delta}_l$. We always use an HRRQR decomposition for these rank computations because it also yields some useful information for doing permutations in case a rank deficient $\boldsymbol{\delta}_l$ matrix occurs.

2.3.2 First variant: large clusters

One possibility of a look-ahead procedure is to proceed in a way analogous to the case of one right and left starting vector [13, Section 19.1]. That is, we intend to choose $k_l = \nu(l)$ for $l = 0, 1, 2, \ldots$ if possible. If this was always possible we would have $\mathbf{z}_i = \mathbf{y}_i$ (but not necessarily $\widetilde{\mathbf{z}}_i = \widetilde{\mathbf{y}}_i$). So we construct a sequence $0 \equiv: n_0 < n_1 < \ldots$ with the property $k_l = \nu(n_l)$. If we assume that an index n_l has already been fixed, the goal is to find an index $n_{l+1} \in \mathbb{N}$ as small as possible such that

$$\mathbf{z}_l = [\, \mathbf{y}_{n_l}, \ldots, \mathbf{y}_{n_{l+1}-1} \,] = [\, y_{\nu(n_l)}, \ldots, y_{\nu(n_{l+1})-1} \,],$$
$$\widetilde{\mathbf{z}}_l = [\, \widetilde{y}_{\nu(n_l)}, \ldots, \widetilde{y}_{\nu(n_{l+1})-1} \,].$$

The indices $\{n_i\}_{i \geq 0}$ will be called *cluster indices*. Clearly, the choice of the cluster indices also implies a choice for the regular indices. Steps LA1 to LA5 of BLBIO now can be formulated more precisely for this look-ahead procedure as outlined in Algorithm 10. So at step LA1, the value $\nu(1)$

Algorithm 10 First variant of look-ahead strategy for BLBIO

Choose an algorithm for computing rank($\boldsymbol{\delta}_l$).
LA1) Set $n_0 = 0$, $n_1 = 1$ and $k_1 = \nu(1)$.
LA2) Set $n_{l+1} = n_{l+1} + 1$.
LA3) Set $n_{l+2} = n_{l+1} + 1$.
LA4) Set minnum $= \nu(n) + 1$.
LA5) Set $k_{l+1} = \nu(n)$.

is assigned as the first choice of k_1. Step LA2 specifies how a new candidate for k_{l+1} is found if the current value of k_{l+1} has to be changed, which here just amounts to increasing the current value of n_{l+1} by one. Consequently, a new right block is appended to $\mathbf{z}_l$. Then in step LA3, a first choice for n_{l+2} is fixed in the obvious way. Step LA4 consists of making sure that the **while** loop in line 8.1 will be executed exactly once. Finally, at step LA5 the value of k_{l+1} is known.

In theory, this variant of the Lanczos process only breaks down in the case that there exists $n_l \in \mathbb{N}$ such that for every $n > n_l$ the choice $n_{l+1} = n$ leads to a singular matrix $\boldsymbol{\delta}_l$. This situation

is called an *incurable breakdown*. In practice, however, this look-ahead strategy may fail due to limited storage because it allows only a limited control over the cluster size r_l. Since the cluster size may become relatively large and the matrices $\boldsymbol{\delta}_l$ may be far from diagonal, the work involved in the singularity checks in Algorithm 6 may increase a lot.

On the other side, this look-ahead strategy leads to a form of the block Lanczos process which incorporates block operations whenever possible and therefore achieves one of our goals set out in Section 0. Furthermore, we also expect less reorthogonalizations of type (2.11) to happen, and so the risk that the matrix $\widetilde{\mathbf{y}}^{\star}_{l-1,\widetilde{n}}\widetilde{\mathbf{y}}_{l-1,\widetilde{n}}$ in (2.6) might get ill-conditioned is reduced.

2.3.3 Second variant: small clusters

Now the goal is to form clusters containing only one vector, if possible. So the first choice for k_1 is one (step LA1), and if a new value for k_{l+1} is needed, it is incremented only by Δ_l (step LA2), which is the minimum needed to obtain a nonsingular larger $\boldsymbol{\delta}_l$. Step LA3 now consists of fixing the smallest possible initial choice for k_{l+2} while in step LA4 we make sure that at least k_{l+1} right vectors will be computed by the **while** loop on line 8.1. With this look-ahead strategy, step LA5 is empty.

Algorithm 11 Second variant of look-ahead strategy for BLBIO

Choose an algorithm for computing rank($\boldsymbol{\delta}_l$).
LA1) Set $k_1 = 1$.
LA2) Set $k_{l+1} = k_{l+1} + \Delta_l$.
LA3) Set $k_{l+2} = k_{l+1} + 1$.
LA4) Set minnum $= k_{l+1}$.

If this algorithm is combined with BLBIO, we expect the matrix $\mathbf{D}_{k_{l+1},k_{l+1}} = \widetilde{\mathbf{Z}}^{\star}_l\mathbf{Z}_l$ to be almost diagonal, similar to [1, Figure 2], and it is indeed diagonal if no look-ahead occurs. So the cluster sizes are likely to be much smaller than with Algorithm 10. On the other side, although the singularity checks may involve much less work, orthogonalizations against clusters are now vectorwise operations, in contrast to the first variant. Additionally, the risk that the matrix $\boldsymbol{\alpha}$ in (2.6) might get ill-conditioned increases, a problem which could be eliminated by reorthogonalizing the ungrouped vectors after each operation of type (2.11). In our implementation, however, we did not do this.

Although we can hardly expect satisfactory results with this look-ahead strategy, in Subsection 3.4, we will compute several examples with both variants to see the difference in performance.

Remark 11. It is clear that Algorithms 10 and 11 are indeed look-ahead strategies in the sense of Definition 3. Furthermore, in the nonblock case $s_0 = \widetilde{s}_0 = 1$ they amount to the same. It schould be noted, however, that our look-ahead strategy may not optimal in the nonblock case, see [13, p. 381] for more information. ○

2.4 BLBIOC

We introduce a new pair of vector sequences $\{v_i\}_{i\geq 0}$ and $\{\widetilde{v}_i\}_{i\geq 0}$, which we want to be linked to the sequences of the right and left vectors as follows: Let

$$\mathbf{V}_n :\equiv [\, v_0, \ldots, v_{\nu(n)-1}\,],$$
$$\widetilde{\mathbf{V}}_{\widetilde{n}} :\equiv [\, \widetilde{v}_0, \ldots, \widetilde{v}_{\widetilde{\nu}(\widetilde{n})-1}\,].$$

Then we request that there be nonsingular upper triangular matrices $\mathbf{U}_n \in \mathbb{C}^{\nu(n)\times\nu(n)}$ and $\widetilde{\mathbf{U}}_{\widetilde{n}} \in \mathbb{C}^{\widetilde{\nu}(\widetilde{n})\times\widetilde{\nu}(\widetilde{n})}$ such that

$$\mathbf{Y}_n = \mathbf{V}_n\mathbf{U}_n \tag{2.21a}$$
$$\widetilde{\mathbf{Y}}_{\widetilde{n}} = \widetilde{\mathbf{V}}_{\widetilde{n}}\widetilde{\mathbf{U}}_{\widetilde{n}} \tag{2.21b}$$

for every $n = 1, \ldots, n_t$ and $\widetilde{n} = 1, \ldots, \widetilde{n}_t$. Note that $\mathbf{U}_n$ and $\widetilde{\mathbf{U}}_{\widetilde{n}}$ are not allowed to be *block* upper triangular. The reason for this restriction is that the above equations allow us to express $\mathbf{z}_l$ as a linear combination of $v_0, \ldots, v_{k_{l+1}-1}$ (and similarly for the left vectors), which will be used in the derivation of orthogonality relations below and in some proofs. It will turn out later that we may require $\mathbf{U}_n$ and $\widetilde{\mathbf{U}}_{\widetilde{n}}$ to be upper triangular and still do permutations to avoid immediate look-ahead.

It also follows from (2.21) that the columns of $\mathbf{V}_n$ form a basis of $\mathcal{B}_n$ for $n = 1, \ldots, n_t$, and similarly for $\widetilde{\mathbf{V}}_{\widetilde{n}}$. The regular indices $\{k_i\}_{i\geq 0}$ will be chosen such they also allow us to partition the sequences $\{v_i\}_{i\geq 0}$ and $\{\widetilde{v}_i\}_{i\geq 0}$ in a way which shows the $\mathbf{A}$ biorthogonality: Let

$$\begin{aligned} \mathbf{w}_l &:\equiv [\, v_{k_l}, \ldots, v_{k_{l+1}-1} \,], \qquad & \mathbf{W}_l &:\equiv [\, \mathbf{w}_0, \ldots, \mathbf{w}_{l-1} \,], \\ \widetilde{\mathbf{w}}_l &:\equiv [\, \widetilde{v}_{k_l}, \ldots, \widetilde{v}_{k_{l+1}-1} \,], \qquad & \widetilde{\mathbf{W}}_l &:\equiv [\, \widetilde{\mathbf{w}}_0, \ldots, \widetilde{\mathbf{w}}_{l-1} \,]. \end{aligned}$$

The blocks $\mathbf{v}_i :\equiv [v_{\nu(i)}, \ldots, v_{\nu(i+1)-1}]$ and $\widetilde{\mathbf{v}}_i :\equiv [\widetilde{v}_{\widetilde{\nu}(i)}, \ldots, \widetilde{v}_{\widetilde{\nu}(i+1)-1}]$ will be referred to as *right* and *left direction blocks*, while $\{\mathbf{w}_j\}$ and $\{\widetilde{\mathbf{w}}_j\}$ will be called *right* and *left direction clusters*, respectively. We will continue to refer to $\{\mathbf{z}_l\}$ and $\{\widetilde{\mathbf{z}}_l\}$ as right and left clusters. As in the case of the right and left clusters, the last direction clusters are special, and we adopt the analogous definitions for $\mathbf{w}_{l_t}$ and $\widetilde{\mathbf{w}}_{l_t}$ as for $\mathbf{z}_{l_t}$ and $\widetilde{\mathbf{z}}_{l_t}$.

Now the right and left direction blocks are $\mathbf{A}$ orthogonal:

$$\widetilde{\mathbf{w}}_i^\star \mathbf{A} \mathbf{w}_j = \begin{cases} \mathbf{0} & \text{if } i \neq j \\ \boldsymbol{\delta}'_i & \text{if } i = j \end{cases}, \tag{2.22}$$

where $\boldsymbol{\delta}'_j$ is of size $r_j \times r_j$ and nonsingular if $0 \leq j < l_t$, while $\boldsymbol{\delta}'_{l_t}$ is of size $\widetilde{r}_{l_t} \times r_{l_t}$[4].

It follows from (2.21a) that there is an upper block Hessenberg matrix $\underline{\mathbf{L}}_n \in \mathbb{C}^{\nu(n+1)\times\nu(n)}$ (with block structure as shown in Figure 1) such that

$$\mathbf{A}\mathbf{V}_n = \mathbf{Y}_{n+1}\underline{\mathbf{L}}_n. \tag{2.23}$$

Now we consider the last block column of (2.21a) and the nth block column of (2.23)

$$\mathbf{y}_n = \mathbf{W}_l \mathbf{g}_n + \mathbf{v}_{l-1,n}\mathbf{g}_{n;l} + \mathbf{v}_n \mathbf{u}_{n,n}, \tag{2.24}$$

$$\mathbf{A}\mathbf{v}_{n-1} = \mathbf{Z}_l \mathbf{f}_{n-1} + \mathbf{y}_{l-1,n}\mathbf{f}_{n-1;l} + \mathbf{y}_n \boldsymbol{\gamma}_n. \tag{2.25}$$

Here, the following definitions are understood:

$$\begin{aligned} l &= \ell(\nu(n)+1) - 1 = \ell_3(n), \\ \mathbf{v}_{l-1,n} &= [\, v_{k_l}, \ldots, v_{\nu(n)-1} \,], \end{aligned}$$

and $\mathbf{u}_{n,n}$ is a nonsingular upper triangular matrix of size $s_n \times s_n$. Recall further that $\mathbf{y}_{l,n} = [\, y_{k_{l+1}}, \ldots, y_{\nu(n)-1} \,]$.

Next we derive orthogonality relations which follow from (2.21), (2.22) and (2.23). First, it follows from the definition of $\ell_3(n)$ that

$$\mathbf{A}\mathbf{v}_n \perp \widetilde{\mathbf{w}}_0, \ldots, \widetilde{\mathbf{w}}_{\ell_3(n)}, \tag{2.26}$$

and now (2.21b) implies

$$\mathbf{A}\mathbf{v}_n \perp \widetilde{\mathbf{z}}_0, \ldots, \widetilde{\mathbf{z}}_{\ell_3(n)}. \tag{2.27}$$

The analogous relation for the left vectors is $\mathbf{A}^\star \widetilde{\mathbf{v}}_{\widetilde{n}} \perp \mathbf{z}_0, \ldots, \mathbf{z}_{\ell_1(\widetilde{n})}$. Another orthogonality relation can be established by starting from the fact that $\mathbf{y}_n$ is orthogonal to the left clusters $\widetilde{\mathbf{z}}_0, \ldots, \widetilde{\mathbf{z}}_{\ell_3(n)}$ (note that this orthogonality relation holds even if we do not have the final block $\mathbf{y}_n$ yet; the first version of $\mathbf{y}_n$ already satisfies it). See also Figure 7. Since the last $\widetilde{y}$ vector of $\widetilde{\mathbf{z}}_{\ell_3(n)}$ has number $k_{\ell_3(n)+1} - 1$, we consider $\widetilde{y}_{k_{\ell_3(n)+1}}$, which is a column of $\widetilde{\mathbf{y}}_{\widetilde{n}(k_{\ell_3(n)+1}+1)-1}$. It follows that

$$\mathbf{y}_n \perp \widetilde{\mathbf{y}}_0, \ldots, \widetilde{\mathbf{y}}_{\widetilde{n}(k_{\ell_3(n)+1}+1)-2}.$$

[4]From now on, we will usually write the formulas only for the right vectors if the ones for the left vectors are obvious.

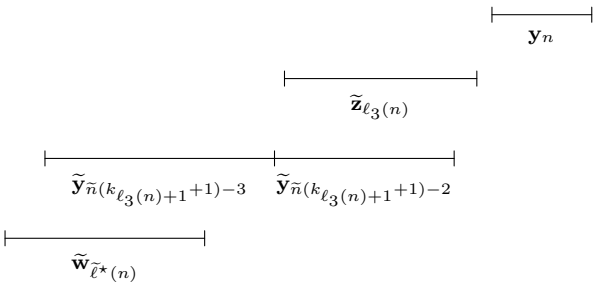


Figure 7: Orthogonality relation obeyed by $\mathbf{y}_n$

Now we consider $\widetilde{\mathbf{y}}_{\widetilde{n}(k_{\ell_3(n)+1}+1)-3}$. The last column of this block has number $\widetilde{\nu}(\widetilde{n}(k_{\ell_3(n)+1}+1)-2)-1$, so we define

$$\widetilde{\ell}^\star(n) :\equiv \ell(\widetilde{\nu}(\widetilde{n}(k_{\ell_3(n)+1}+1)-2)+1)-1.$$

Then

$$\mathbf{y}_n \perp \mathbf{A}^\star \widetilde{\mathbf{w}}_i \qquad \text{for } i = 0, \ldots, \widetilde{\ell}^\star(n). \tag{2.28}$$

An analogous argument shows

$$\begin{aligned} \widetilde{\mathbf{y}}_{\widetilde{n}} &\perp \mathbf{A}\mathbf{w}_i \qquad \text{for } i = 0, \ldots, \ell^\star(\widetilde{n}), \\ \ell^\star(\widetilde{n}) &:\equiv \ell(\nu(n(k_{\ell_1(\widetilde{n})+1}+1)-2)+1)-1. \end{aligned} \tag{2.29}$$

Note that by a similar argument as above we can show

$$\begin{aligned} \widetilde{\mathbf{z}}_l &\perp \mathbf{A}\mathbf{w}_0, \ldots, \mathbf{A}\mathbf{w}_{\ell'(l)} \\ \ell'(l) &:\equiv \ell(\nu(n(k_l+1)-2)+1)-1. \end{aligned} \tag{2.30}$$

In the sequel, we describe an algorithm which yields (2.21), (2.22) and (2.23). Then we can use the above orthogonality relations to show that the recursions (2.24) and (2.25) are short. It has to be pointed out, however, that our algorithm will only lead to these recursions in exact arithmetic and with a zero deflation tolerance. In particular, it will turn out that if inexact deflation is done, then in (2.23), an additional term resulting from deflation will appear.

The main part of BLBIOC is given in Algorithm 12, while the extensions of the two block Krylov spaces are again formulated separately, see Algorithms 13 and 14. We will focus on those aspects of BLBIOC which are different from BLBIO.

During the initializations, we also have to fix (preliminary) blocks $\mathbf{v}_0$ and $\widetilde{\mathbf{v}}_0$ and compute their products with $\mathbf{A}$ and $\mathbf{A}^\star$, respectively, so that regular index k_1 can be tested if necessary. The main loop starts at line 12.10 and is similar to the main loop of BLBIO. So the first issue is the extension of the left block Krylov space (if necessary), see Algorithm 13. At line 13.2, the block $\mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}-1}$ is available, and according to the analogue of (2.27) for the left vectors, it is already orthogonal to $\mathbf{z}_0, \ldots, \mathbf{z}_{\ell_1(\widetilde{n}-1)}$, which means that we only need to orthogonalize it against $\mathbf{z}_{\ell_1(\widetilde{n}-1)+1}, \ldots, \mathbf{z}_{\ell_1(\widetilde{n})}$. The block resulting from these orthogonalizations is denoted by $\widetilde{\mathbf{y}}_{\text{tmp}}$. At this point, the following recursion holds:

$$\widetilde{\mathbf{y}}_{\text{tmp}} = \mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}-1} - \sum_{j=\ell_1(\widetilde{n}-1)+1}^{\ell_1(\widetilde{n})} \widetilde{\mathbf{z}}_j \widetilde{\mathbf{f}}_{\widetilde{n}-1,j}, \tag{2.31}$$

where $\widetilde{\mathbf{f}}_{\widetilde{n}-1,j} = \boldsymbol{\delta}_j^{-\star}\mathbf{z}_j^\star\mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}-1}$ for $j = \ell_1(\widetilde{n}-1)+1, \ldots, \ell_1(\widetilde{n})$. Note that, at this moment, $\ell_1(\widetilde{n}) = l-1$ is the number of completed right and left clusters. Next we need to check for deflation the ungrouped left vectors $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}} = [\,\widetilde{y}_{k_l}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}\,]$ together with the columns of $\widetilde{\mathbf{y}}_{\text{tmp}}$, which we do again by first orthogonalizing its columns against the columns of $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}}$ and then doing a rank revealing QR factorization. This leads to

$$\widetilde{\mathbf{y}}_{\text{tmp}} - \widetilde{\mathbf{y}}_{l-1,\widetilde{n}}\boldsymbol{\alpha}' = \widetilde{\mathbf{y}}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'_{\widetilde{n}} + \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'^{\triangle}_{\widetilde{n}}$$

Algorithm 12 BlBiOC

Input: Matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$, block $\mathbf{y} \in \mathbb{C}^{N\times s}$ of right Lanczos vectors, block $\widetilde{\mathbf{y}} \in \mathbb{C}^{N\times \widetilde{s}}$ of left Lanczos vectors

1: /* *Initializations, including:* */
2: Check $\mathbf{y}$ and $\widetilde{\mathbf{y}}$ for deflation $\rightarrow \mathbf{y}_0$, $\widetilde{\mathbf{y}}_0$, $\nu(1)$, $\widetilde{\nu}(1)$, s_0, $\widetilde{s}_0$.
3: **if** $s_0 = 0$ **or** $\widetilde{s}_0 = 0$ **then**
4: **stop** /* *abnormal termination* */
5: **end if**
6: Set $\mathbf{v}_0 = \mathbf{y}_0$, $\widetilde{\mathbf{v}}_0 = \widetilde{\mathbf{y}}_0$.
7: Compute $\mathbf{A}\mathbf{v}_0$ and $\mathbf{A}^\star\widetilde{\mathbf{v}}_0$.
8: Set $l = 0$, $j = 0$, $n = 1$, $\widetilde{n} = 1$, $k_0 = 0$.
9: LA1 Fix initial choice for k_1 under the condition $1 \le k_1 \le \nu(1)$.
10: **loop**
11: /* *1) Compute enough $\widetilde{y}$ vectors:* */
12: Extend left block Krylov space according to Algorithm 13.
13: /* *2) Now we test the current value of k_{l+1}:* */
14: Execute lines 12 to 25 of Algorithm 6.
15: **if** $\Delta_l > 0$ **then**
16: LA2 Increase k_{l+1}.
17: **else**
18: /* *First test passed, now do second test:* */
19: Set $\mathbf{w}_l = [\, v_{k_l}, \ldots, v_{k_{l+1}-1}\,]$, $\widetilde{\mathbf{w}}_l = [\, \widetilde{v}_{k_l}, \ldots, \widetilde{v}_{k_{l+1}-1}\,]$.
20: Compute $\boldsymbol{\delta}'_l = \widetilde{\mathbf{w}}_l^\star \mathbf{A}\mathbf{w}_l$ and $\Delta'_l = r_l - \mathrm{rank}(\boldsymbol{\delta}'_l)$.
21: **if** $\Delta'_l > 0$ **then**
22: Check whether there is a permutation P of the columns of $\mathbf{v}_{n-1}$ which might cure the rank deficiency of $\boldsymbol{\delta}'_l$.
23: **if** there is such a P **then**
24: Apply P to the columns of $\mathbf{y}_{n-1}$ and $\mathbf{v}_{n-1}$ and adapt the recursions for the right vectors.
25: Set $\mathbf{w}_l = [\, v_{k_l}, \ldots, v_{k_{l+1}-1}\,]$.
26: Compute $\boldsymbol{\delta}'_l = \widetilde{\mathbf{w}}_l^\star \mathbf{A}\mathbf{w}_l$ and $\Delta'_l = r_l - \mathrm{rank}(\boldsymbol{\delta}'_l)$.
27: **end if**
28: **end if**
29: **if** $\Delta'_l > 0$ **then**
30: Repeat lines 22 to 26 for the left direction vectors.
31: **end if**
32: **if** $\Delta'_l > 0$ **then**
33: LA2' Increase k_{l+1}.
34: **end if**
35: **if** $\Delta_l = 0$ **and** $\Delta'_l = 0$ **then**
36: Orthogonalize $[\, v_{k_{l+1}}, \ldots, v_{\nu(n)-1}\,]$ against $\mathbf{A}^\star\widetilde{\mathbf{w}}_l$ and $[\, \widetilde{v}_{k_{l+1}}, \ldots, \widetilde{v}_{\widetilde{\nu}(\widetilde{n})-1}\,]$ against $\mathbf{A}\mathbf{w}_l$. Renormalize the orthogonalized vectors and adapt the recursions for the right vectors.
37: Orthogonalize $[\, y_{k_{l+1}}, \ldots, y_{\nu(n)-1}\,]$ against $\widetilde{\mathbf{z}}_l$ and $[\, \widetilde{y}_{k_{l+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1}\,]$ against $\mathbf{z}_l$. Renormalize the orthogonalized vectors and adapt the recursions for the left vectors.
38: LA3 Fix new initial choice for k_{l+2}.
39: Set $l = l + 1$.
40: **end if**
41: **end if**
42: /* *3) Compute enough y vectors:* */
43: LA4 Fix minimum number of right vectors $\rightarrow$ minnum.
44: Extend right block Krylov space according to Algorithm 14.
45: **end loop**

Algorithm 13 Extension of left block Krylov space for BLBIOC

1: **while** $\widetilde{\nu}(\widetilde{n}) < k_{l+1}$ **do**
2: Orthogonalize $\mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}-1}$ against $\mathbf{z}_{\ell_1(\widetilde{n}-1)+1},\dots,\mathbf{z}_{l-1} \to \widetilde{\mathbf{y}}_{\mathrm{tmp}}$.
3: Orthogonalize $\widetilde{\mathbf{y}}_{\mathrm{tmp}}$ against $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}}$.
4: Check $\widetilde{\mathbf{y}}_{\mathrm{tmp}}$ for deflation $\to \widetilde{s}_{\widetilde{n}}$, $\widetilde{\mathbf{y}}_{\widetilde{n}}$, $\widetilde{\nu}(\widetilde{n}+1)$, $\widetilde{s}^{\triangle}_{\widetilde{n}}$, $\widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}$, $\widetilde{\nu}^{\triangle}(\widetilde{n}+1)$
5: **if** $\widetilde{s}_{\widetilde{n}} = 0$ **then**
6: Set $l_t = l$, $n_t = n$, $\widetilde{n}_t = \widetilde{n}$.
7: Set $k_{l_t+1} = \nu(n_t)$, $\widetilde{k}_{l_t+1} = \widetilde{\nu}(\widetilde{n}_t)$, $l = l+1$.
8: Set $\mathbf{z}_{l_t} = [\, y_{k_{l_t}},\dots,y_{\nu(n_t)-1}\,]$, $\widetilde{\mathbf{z}}_{l_t} = [\, \widetilde{y}_{k_{l_t}},\dots,\widetilde{y}_{\widetilde{\nu}(\widetilde{n}_t)-1}\,]$ and $\boldsymbol{\delta}_{l_t} = \widetilde{\mathbf{z}}^\star_{l_t}\widetilde{\mathbf{z}}_{l_t}$.
9: Set $\mathbf{w}_{l_t} = [\, v_{k_{l_t}},\dots,v_{\nu(n_t)-1}\,]$, $\widetilde{\mathbf{w}}_{l_t} = [\, \widetilde{v}_{k_{l_t}},\dots,\widetilde{v}_{\widetilde{\nu}(\widetilde{n}_t)-1}\,]$ and $\boldsymbol{\delta}'_{l_t} = \widetilde{\mathbf{w}}^\star_{l_t}\mathbf{A}\mathbf{w}_{l_t}$.
10: **stop**
11: **end if**
12: Set $\widetilde{\mathbf{v}}_{\widetilde{n}} = \widetilde{\mathbf{y}}_{\widetilde{n}}$ and orthogonalize it against $\mathbf{A}\mathbf{w}_{\ell^\star(\widetilde{n})+1},\dots,\mathbf{A}\mathbf{w}_{l-1}$.
13: Normalize columns of $\widetilde{\mathbf{v}}_{\widetilde{n}}$.
14: Compute $\mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}}$.
15: Set $\widetilde{n} = \widetilde{n}+1$.
16: **end while**

with $\boldsymbol{\alpha}' = (\widetilde{\mathbf{y}}^\star_{l-1,\widetilde{n}}\widetilde{\mathbf{y}}_{l-1,\widetilde{n}})^{-1}\widetilde{\mathbf{y}}^\star_{l-1,\widetilde{n}}\widetilde{\mathbf{y}}_{\mathrm{tmp}}$ (it will follow from Lemma 14 that the inverse in this formula exists). Substituting (2.31) yields

$$\widetilde{\mathbf{y}}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'_{\widetilde{n}} = \mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}-1} - \sum_{j=\ell_1(\widetilde{n}-1)+1}^{\ell_1(\widetilde{n})} \widetilde{\mathbf{z}}_j\widetilde{\mathbf{f}}_{\widetilde{n}-1,j} - \widetilde{\mathbf{y}}_{l-1,\widetilde{n}}\boldsymbol{\alpha}' - \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'^{\triangle}_{\widetilde{n}}.$$

At this point, the columns of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ are preliminary versions of new left vectors. As in the case of BLBIO, when more clusters are completed, we will have to orthogonalize some part of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ against recently completed right clusters, see line 12.37, and the final version of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ is obtained when clusters $\ell_2(\widetilde{n})$ are completed. Therefore its final recursion is

$$\widetilde{\mathbf{y}}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'_{\widetilde{n}} = \mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}-1} - \sum_{j=\ell_1(\widetilde{n}-1)+1}^{\ell_2(\widetilde{n})} \widetilde{\mathbf{z}}_j\widetilde{\mathbf{f}}_{\widetilde{n}-1,j} - \widetilde{\mathbf{y}}_{\ell_1(\widetilde{n}),\widetilde{n}}\boldsymbol{\alpha}' - \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'^{\triangle}_{\widetilde{n}}, \tag{2.32}$$

where

$$\widetilde{\mathbf{f}}_{\widetilde{n}-1,j} = \begin{cases} \boldsymbol{\delta}_j^{-\star}\mathbf{z}_j^\star\mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}-1} & \text{for } j = \ell_1(\widetilde{n}-1)+1,\dots,\ell_1(\widetilde{n}) \\ [\,\mathbf{0}_{r_j\times(k_{j+1}-\widetilde{\nu}(\widetilde{n}))} \quad \boldsymbol{\delta}_j^{-\star}\mathbf{z}_j^\star\widetilde{\mathbf{y}}_{j,\widetilde{n}+1}\,]\,\widetilde{\boldsymbol{\eta}}'_{\widetilde{n}} & \text{for } j = \ell_1(\widetilde{n})+1,\dots,\ell_2(\widetilde{n}), \end{cases}$$

and $\widetilde{\mathbf{y}}_{j,\widetilde{n}+1} = [\, \widetilde{y}_{k_{j+1}},\dots,\widetilde{y}_{\widetilde{\nu}(\widetilde{n}+1)-1}\,]$ denotes the part of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ which has to be modified. So we can write

$$\begin{aligned}\mathbf{A}^\star\widetilde{\mathbf{v}}_{\widetilde{n}-1} &= \widetilde{\mathbf{Y}}_{\widetilde{n}+1}\left(\widetilde{\mathbf{S}}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'_{\widetilde{n}} + \sum_{j=\ell_1(\widetilde{n}-1)+1}^{\ell_2(\widetilde{n})} \widetilde{\mathbf{S}}^z_j\widetilde{\mathbf{f}}_{\widetilde{n}-1,j} + \widetilde{\mathbf{S}}_{\ell_1(\widetilde{n}),\widetilde{n}}\boldsymbol{\alpha}'\right) + \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'^{\triangle}_{\widetilde{n}} \\ &=: \sum_{i=0}^{\widetilde{n}} \widetilde{\mathbf{y}}_i\widetilde{\mathbf{l}}_{i,\widetilde{n}-1} + \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\widetilde{\boldsymbol{\eta}}'^{\triangle}_{\widetilde{n}}.\end{aligned}$$

Here, we denote by $\widetilde{\mathbf{l}}_{i,\widetilde{n}-1}$ rows $\widetilde{\nu}(i)+1,\dots,\widetilde{\nu}(i+1)$ of the matrix in the braces ($i = 0,\dots,\widetilde{n}$), and $\widetilde{\mathbf{S}}_{\widetilde{n}}$, $\widetilde{\mathbf{S}}^z_j$, $\widetilde{\mathbf{S}}_{\ell_1(\widetilde{n}),\widetilde{n}}$ are the analogues of the matrices defined in the proof of Proposition 7. From this we conclude that

$$\mathbf{A}^\star\widetilde{\mathbf{V}}_{\widetilde{n}} = \widetilde{\mathbf{Y}}_{\widetilde{n}+1}\underline{\widetilde{\mathbf{L}}}_{\widetilde{n}} + \widetilde{\mathbf{Y}}^{\triangle}_{\widetilde{n}+1}\underline{\mathbf{T}}'^{\triangle}_{\widetilde{n}} \tag{2.33}$$

with $\underline{\widetilde{\mathbf{L}}}_{\widetilde{n}} = (\widetilde{\mathbf{l}}_{ij}) \in \mathbb{C}^{\widetilde{\nu}(\widetilde{n}+1)\times\widetilde{\nu}(\widetilde{n})}$ and

$$\widetilde{\mathbf{Y}}^{\triangle}_{\widetilde{n}+1} :\equiv [\,\widetilde{\mathbf{y}}^{\triangle}_0, \ldots, \widetilde{\mathbf{y}}^{\triangle}_{\widetilde{n}}\,] \in \mathbb{C}^{N\times\nu^{\triangle}(i)}$$

$$\underline{\mathbf{T}}'^{\triangle}_{\widetilde{n}} :\equiv \begin{bmatrix} \mathbf{0}_{s_0^{\triangle}\times s_0} & & & \\ \boldsymbol{\eta}_1'^{\triangle} & \mathbf{0}_{s_1^{\triangle}\times s_1} & & \\ & \ddots & \ddots & \\ & & \ddots & \mathbf{0}_{s_{i-1}^{\triangle}\times s_{i-1}} \\ & & & \boldsymbol{\eta}_i'^{\triangle} \end{bmatrix} \in \mathbb{C}^{\nu^{\triangle}(i+1)\times\nu(i)}.$$

So (2.33) leads to (2.23) as long as no deflation occurs or if only exact deflation is done.

Now we assume that $\widetilde{s}_{\widetilde{n}} > 0$ so that the left block Krylov space has successfully been extended. Then we immeditately compute a first version of $\widetilde{\mathbf{v}}_{\widetilde{n}}$ (line 13.12) by orthogonalizing $\widetilde{\mathbf{y}}_{\widetilde{n}}$ against $\mathbf{A}\mathbf{w}_0, \ldots, \mathbf{A}\mathbf{w}_{l-1}$. According to (2.29), $\widetilde{\mathbf{y}}_{\widetilde{n}}$ is already orthogonal to $\mathbf{A}\mathbf{w}_0, \ldots, \mathbf{A}\mathbf{w}_{\ell^\star(\widetilde{n})}$ so we have the following recursion

$$\widetilde{\mathbf{v}}_{\widetilde{n}}\widetilde{\mathbf{u}}^{i}_{\widetilde{n},\widetilde{n}} = \widetilde{\mathbf{y}}_{\widetilde{n}} - \sum_{j=\ell^\star(\widetilde{n})+1}^{i} \widetilde{\mathbf{w}}_j \widetilde{\mathbf{g}}^{i}_{\widetilde{n},j} \tag{2.34}$$

with $i = \ell_1(\widetilde{n}) = l-1$ and $\widetilde{\mathbf{g}}^{i}_{\widetilde{n},j} = \boldsymbol{\delta}_j'^{-\star}\mathbf{w}_j^\star\mathbf{A}^\star\widetilde{\mathbf{y}}_{\widetilde{n}}$ for $j = \ell^\star(\widetilde{n})+1, \ldots, i$. The matrix $\widetilde{\mathbf{u}}^{i}_{\widetilde{n},\widetilde{n}}$ is diagonal and nonsingular and ensures that the columns of $\widetilde{\mathbf{v}}_{\widetilde{n}}$ have unit norm (it follows from Lemma 14 below that the normalization is always possible). Whenever a new regular index is fixed, however, both $\widetilde{\mathbf{y}}_{\widetilde{n}}$ and $\widetilde{\mathbf{v}}_{\widetilde{n}}$ possibly need to be changed due to the additional orthogonality constraints. Our next task is therefore to investigate the influence of new orthogonality conditions on (2.34).

Proposition 12. *Assume that $\ell_1(\widetilde{n}) < \ell_2(\widetilde{n})$. Then (2.34) holds for all $\ell_1(\widetilde{n}) \le i \le \ell_2(\widetilde{n})$ with $\widetilde{\mathbf{u}}^{i}_{\widetilde{n},\widetilde{n}}$ diagonal and nonsingular. The blocks $\widetilde{\mathbf{y}}_{\widetilde{n}}$ and $\widetilde{\mathbf{v}}_{\widetilde{n}}$ are in their final form if and only if $i = \ell_2(\widetilde{n})$.*

Proof. We use induction on i. First, we already have shown the assertion for $i = \ell_1(\widetilde{n})$. So we assume that the assertion holds for $i = \ell_1(\widetilde{n}), \ldots, p-1 < \ell_2(\widetilde{n})$ with $p > \ell_1(\widetilde{n})$ (induction hypothesis). Consequently, we can write

$$\widetilde{\mathbf{v}}_{\widetilde{n}}\widetilde{\mathbf{u}}^{p-1}_{\widetilde{n},\widetilde{n}} = \widetilde{\mathbf{y}}_{\widetilde{n}} - \sum_{j=\ell^\star(\widetilde{n})+1}^{p-1} \widetilde{\mathbf{w}}_j \widetilde{\mathbf{g}}^{p-1}_{\widetilde{n},j}. \tag{2.35}$$

Now assume that clusters $\mathbf{z}_p$ and $\mathbf{w}_p$ are completed. On one hand, some part of $\widetilde{\mathbf{y}}_{\widetilde{n}}$ (namely $\widetilde{\mathbf{y}}_{p,\widetilde{n}+1}$) has to be orthogonalized against $\mathbf{z}_p$ and then the changed columns have to be renormalized. These two operations lead to a new block which we temporarily denote by $\widetilde{\mathbf{y}}'_{\widetilde{n}}$:

$$\widetilde{\mathbf{y}}'_{\widetilde{n}}\widetilde{\boldsymbol{\gamma}}_{\widetilde{n},p} = \widetilde{\mathbf{y}}_{\widetilde{n}} - \widetilde{\mathbf{z}}_p\widetilde{\boldsymbol{\tau}}^{z}_{p,\widetilde{n}} \tag{2.36}$$

with $\widetilde{\boldsymbol{\tau}}^{z}_{p,\widetilde{n}} = [\,\mathbf{0}_{r_p\times(k_{p+1}-\widetilde{\nu}(\widetilde{n}))}\ \boldsymbol{\delta}_p^{-\star}\mathbf{z}_p^\star\widetilde{\mathbf{y}}_{p,\widetilde{n}+1}\,]$. The matrix $\widetilde{\boldsymbol{\gamma}}_{\widetilde{n},p}$ is due to the renormalization and is therefore diagonal and nonsingular. On the other hand, some part of $\widetilde{\mathbf{v}}_{\widetilde{n}}$ (namely $\widetilde{\mathbf{v}}_{p,\widetilde{n}+1}$) has to be orthogonalized against $\mathbf{A}\mathbf{w}_p$ and then the changed columns have to be renormalized. These two operations lead to a new block which we temporarily denote by $\widetilde{\mathbf{v}}'_{\widetilde{n}}$:

$$\widetilde{\mathbf{v}}'_{\widetilde{n}}\widetilde{\boldsymbol{\gamma}}'_{\widetilde{n},p} = \widetilde{\mathbf{v}}_{\widetilde{n}} - \widetilde{\mathbf{w}}_p\widetilde{\boldsymbol{\beta}}_{p,\widetilde{n}} \tag{2.37}$$

with $\widetilde{\boldsymbol{\beta}}_{p,\widetilde{n}} = [\,\mathbf{0}_{r_p\times(k_{p+1}-\widetilde{\nu}(\widetilde{n}))}\ \boldsymbol{\delta}_p'^{-\star}(\mathbf{A}\mathbf{w}_p)^\star\widetilde{\mathbf{v}}_{p,\widetilde{n}+1}\,]$. The matrix $\widetilde{\boldsymbol{\gamma}}'_{\widetilde{n},p}$ is diagonal and nonsingular. Now we substitute (2.36) and (2.37) into (2.35) and obtain

$$(\widetilde{\mathbf{v}}'_{\widetilde{n}}\widetilde{\boldsymbol{\gamma}}'_{\widetilde{n},p} + \widetilde{\mathbf{w}}_p\widetilde{\boldsymbol{\beta}}_{p,\widetilde{n}})\widetilde{\mathbf{u}}^{p-1}_{\widetilde{n},\widetilde{n}} = \widetilde{\mathbf{y}}'_{\widetilde{n}}\widetilde{\boldsymbol{\gamma}}_{\widetilde{n},p} + \widetilde{\mathbf{z}}_p\widetilde{\boldsymbol{\tau}}^{z}_{p,\widetilde{n}} - \sum_{j=\ell^\star(\widetilde{n})+1}^{p-1} \widetilde{\mathbf{w}}_j \widetilde{\mathbf{g}}^{p-1}_{\widetilde{n},j}.$$

The next step is to eliminate $\widetilde{\mathbf{z}}_p$ from this recursion. To this end, we write $\widetilde{\mathbf{z}}_p = \widetilde{\mathbf{w}}_{\ell^\star(\widetilde{n})+1}\widetilde{\mathbf{u}}^{z}_{\ell^\star(\widetilde{n})+1,p} + \cdots + \widetilde{\mathbf{w}}_p\widetilde{\mathbf{u}}^{z}_{p,p}$ (according to (2.30) we have

$$\widetilde{\mathbf{z}}_p \perp \mathbf{A}\mathbf{w}_0, \ldots, \mathbf{A}\mathbf{w}_{\ell'(p)}$$

and therefore $\widetilde{\mathbf{z}}_p \perp \mathbf{A}\mathbf{w}_0, \ldots, \mathbf{A}\mathbf{w}_{\ell^\star(\widetilde{n})}$ since $k_p \geq k_{\ell_1(\widetilde{n})+1}$). Now we define

$$\widetilde{\mathbf{u}}^p_{\widetilde{n},\widetilde{n}} :\equiv \widetilde{\boldsymbol{\gamma}}'_{\widetilde{n},p}\widetilde{\mathbf{u}}^{p-1}_{\widetilde{n},\widetilde{n}}\widetilde{\boldsymbol{\gamma}}^{-1}_{\widetilde{n},p} \tag{2.38}$$

$$\widetilde{\mathbf{g}}^p_{\widetilde{n},j} :\equiv (\widetilde{\mathbf{g}}^{p-1}_{\widetilde{n},j} - \widetilde{\mathbf{u}}^z_{j,p}\widetilde{\boldsymbol{\tau}}^z_{p,\widetilde{n}})\widetilde{\boldsymbol{\gamma}}^{-1}_{\widetilde{n},p} \qquad \text{for } \ell^\star(\widetilde{n})+1 \leq j \leq p-1$$

$$\widetilde{\mathbf{g}}^p_{\widetilde{n},p} :\equiv (\widetilde{\boldsymbol{\beta}}_{\widetilde{n},p}\widetilde{\mathbf{u}}^{p-1}_{\widetilde{n},\widetilde{n}} - \widetilde{\mathbf{u}}^z_{p,p}\widetilde{\boldsymbol{\tau}}^z_{p,\widetilde{n}})\widetilde{\boldsymbol{\gamma}}^{-1}_{\widetilde{n},p} \tag{2.39}$$

and drop the primes in $\widetilde{\mathbf{y}}'_{\widetilde{n}}$ and $\widetilde{\mathbf{v}}'_{\widetilde{n}}$. Then we obtain

$$\widetilde{\mathbf{v}}_{\widetilde{n}}\widetilde{\mathbf{u}}^p_{\widetilde{n},\widetilde{n}} = \widetilde{\mathbf{y}}_{\widetilde{n}} - \sum_{j=\ell^\star(\widetilde{n})+1}^{p} \widetilde{\mathbf{w}}_j\widetilde{\mathbf{g}}^p_{\widetilde{n},j}.$$

By the induction hypothesis, $\widetilde{\mathbf{u}}^{p-1}_{\widetilde{n},\widetilde{n}}$ is diagonal and nonsingular so that due to (2.38), the same holds for $\widetilde{\mathbf{u}}^p_{\widetilde{n},\widetilde{n}}$. □

From this proposition, we conclude that the length of the recursion (2.34) does not increase when new clusters are completed.

Proposition 13. *If no permutations occur in* BLBIOC*, then relation* (2.21b) *holds with a nonsingular, upper triangular matrix* $\widetilde{\mathbf{U}}_{\widetilde{n}} \in \mathbb{C}^{\widetilde{\nu}(\widetilde{n})\times\widetilde{\nu}(\widetilde{n})}$.

It will be shown below that permutations do not destroy the upper triangular structure of $\widetilde{\mathbf{U}}_{\widetilde{n}}$.

Proof. The assertion is clearly true when the main loop of BLBIOC is reached (line 12.10). If the left block Krylov space has to be extended, then only the strict upper triangular part of $\widetilde{\mathbf{U}}_{\widetilde{n}}$ is modified since we orthogonalize at most against $\mathbf{z}_{\ell_1(\widetilde{n})}$. The next modification of $\widetilde{\mathbf{U}}_{\widetilde{n}}$ happens when new clusters have been completed, and the only contributions to the lower triangular part of $\widetilde{\mathbf{U}}_{\widetilde{n}}$ might come from (2.38) and (2.39). But $\widetilde{\mathbf{u}}^p_{\widetilde{n},\widetilde{n}}$ is diagonal, and the first $k_{p+1} - \widetilde{\nu}(\widetilde{n})$ columns of $\widetilde{\boldsymbol{\tau}}^z_{p,\widetilde{n}}$ as well as $\widetilde{\boldsymbol{\beta}}_{p,\widetilde{n}}$ are zero, and therefore, the upper triangular structure of $\widetilde{\mathbf{U}}_{\widetilde{n}}$ is preserved (this is the same type of argument which was used in Example 9; see the figure there). If no permutations occur, then the matrix does not change any more. □

For the orthogonalization (2.34), no matrix vector products have to be computed because $\mathbf{A}\mathbf{w}_i$ and $\mathbf{A}^\star\widetilde{\mathbf{w}}_i$ are already available (all the products $\mathbf{A}^\star\widetilde{v}_i$ are computed during the extension of the left block Krylov space, see line 13.14, and are updated whenever the ungrouped left direction vectors change, and similarly for the right vectors, see below).

After the extension of the left block Krylov space, we are in a position to test the current value of k_{l+1}. In contrast to BLBIO, there are now two tests because both $\boldsymbol{\delta}_l$ and $\boldsymbol{\delta}'_l$ have to be well-conditioned. The first test is the same as for BLBIO, and if it is passed, then the second test starts at line 12.19. Since we always construct temporary direction blocks whenever a new preliminary left or right block is available, we are able to form preliminary versions of $\mathbf{w}_l$ and $\widetilde{\mathbf{w}}_l$ so that a temporary $\boldsymbol{\delta}'_l$ can be computed in line 12.20. If this matrix turns out to be numerically singular, we go on to check whether a permutation of the right vectors is possible and if so, a new $\boldsymbol{\delta}'_l$ can be tested in line 12.26. The same process is repeated for the left vectors with $\boldsymbol{\delta}'^\star_l$ in place of $\boldsymbol{\delta}'_l$ if the numerical singularity still persists. In contrast to the procedure for the right vectors, we now also have to compute the HRRQR decomposition of $\boldsymbol{\delta}'^\star_l$ to get some information about how to choose a permutation.

Since we always want the recursions (2.21) and (2.23) to be valid, we have to show that they are not changed when a permutation occurs. Assume that we have a permutation P which should be applied to the columns of $\mathbf{y}_{n-1}$. As every permutation can be written as a product of transpositions (i.e. permutations which only interchange two elements), we may assume without loss of generality that P is a transposition. Let $\mathbf{P}$ denote the matrix of order $\nu(n)$ representing P. Then we want to multiply (2.21a) by $\mathbf{P}$ from the right, which yields

$$\mathbf{Y}_n\mathbf{P} = \mathbf{V}_n\mathbf{U}_n\mathbf{P} = \mathbf{V}_n\mathbf{P}\mathbf{P}^\star\mathbf{U}_n\mathbf{P} = \mathbf{V}_n\mathbf{P}\mathbf{U}'_n$$

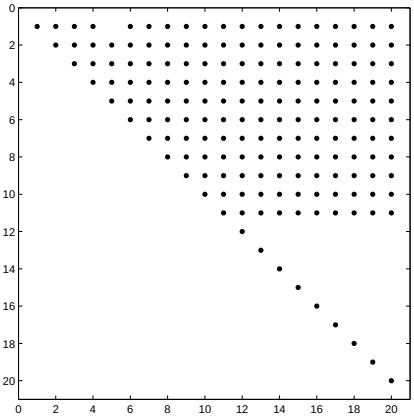


Figure 8: Example nonzero pattern of $\mathbf{U}_n$

with $\mathbf{U}'_n :\equiv \mathbf{P}^\star \mathbf{U}_n \mathbf{P} = \mathbf{P}\mathbf{U}_n\mathbf{P}$ since $\mathbf{P}^\star = \mathbf{P}^{-1} = \mathbf{P}$. So we have to show that $\mathbf{U}'_n$ is also upper triangular. Consider the structure of $\mathbf{U}_n$ at this point, see Figure 8 for an example. Now, if $\mathbf{P}$ interchanges columns i and j, then $i, j \geq k_l$ because we can only permute ungrouped vectors. In the example shown in Figure 8, we have $k_l = 11$. Consequently, the matrix $\mathbf{P}$ acting from the left on $\mathbf{U}_n$ interchanges two rows whose only nonzero is the diagonal element, and it is easy to verify that by $\mathbf{P}$ acting from the right, the two diagonal elements are again moved to the main diagonal (u_{ii} is now the jth diagonal element and vice versa). We conclude that $\mathbf{U}'_n$ is again upper triangular.

It follows that in order to preserve (2.21a), it is necessary to permute both $\mathbf{y}_{n-1}$ and $\mathbf{v}_{n-1}$ even if we originally only intended to permute $\mathbf{y}_{n-1}$. On the other hand, the above argument also works if we originally wanted to permute $\mathbf{v}_{n-1}$, and therefore we may conclude that (2.21a) is not destroyed by a permutation of the right vectors. Together with Proposition 13, we obtain that (2.21) holds with $\mathbf{U}_n$ and $\widetilde{\mathbf{U}}_{\widetilde{n}}$ upper triangular.

Next, we consider the other recursion, equation (2.23). Here, we only need to apply the inverse of our permutation to the rows of $\mathbf{l}_{n-1,n-1}$ since $\mathbf{A}\mathbf{v}_{n-1}$ will only be needed when $\mathbf{y}_n$ is computed. So (2.23) is also preserved by a permutation. The argument for the left vectors is the same.

In case no permutations are possible or they are not successful, a new value has to be assigned to k_{l+1} (step LA2'). It is supposed that the same rule for finding a new value is applied at step LA2' as at step LA2. If, on the other hand, $\boldsymbol{\delta}'_l$ is numerically nonsingular, then the current value of k_{l+1} is final, and we turn to the possible orthogonalizations of ungrouped vectors (lines 12.36 and 12.37). The effects of these operations on the recursions have already been described earlier in this subsection.

Termination of BlBiOC is the same as for BlBiO except that we also have to form the last direction clusters (lines 14.9 and 13.9).

Now we fill in the remaining gaps in the proof of Proposition 12.

Lemma 14. *Assume that* BlBiOC *is run with an arbitrary look-ahead procedure according to Definition 3 (and with the same rule applied in step LA2' as in step LA2). Then properties* (2.16) *are invariants as well as the following:*

$$\textit{For } 0 \leq i, j \leq l-1\textit{:} \quad \widetilde{\mathbf{w}}_i^\star \mathbf{A} \mathbf{w}_j = \begin{cases} \mathbf{0}, & i \neq j \\ \boldsymbol{\delta}'_i, & i = j \end{cases} \tag{2.40a}$$

$$\textit{For } i = 1, \ldots, l-1\textit{:} \quad \begin{cases} \{v_j\}_{j \geq k_l} \perp \mathbf{A}^\star \widetilde{\mathbf{w}}_i \\ \{\widetilde{v}_j\}_{j \geq \widetilde{k}_l} \perp \mathbf{A}\mathbf{w}_i \end{cases} \tag{2.40b}$$

$$\left.\begin{array}{l} \textit{The vectors } \{v_j\}_{j \geq k_l} \textit{ are linearly independent.} \\ \textit{The vectors } \{\widetilde{v}_j\}_{j \geq \widetilde{k}_l} \textit{ are linearly independent.} \end{array}\right\} \tag{2.40c}$$

Recall that $\widetilde{k}_i :\equiv k_i$ *for* $i = 0, \ldots, l_t$ *and* $\widetilde{k}_{l_t+1} = \widetilde{\nu}(\widetilde{n}_t)$. *After termination,* (2.16) *and* (2.40) *hold with* $l - 1 = l_t$.

Algorithm 14 Extension of right block Krylov space for BLBIOC

1: **while** $\nu(n) <$ minnum **do**
2: Orthogonalize $\mathbf{A}\mathbf{v}_{n-1}$ against $\widetilde{\mathbf{z}}_{\ell_3(n-1)+1}, \ldots, \widetilde{\mathbf{z}}_{l-1} \rightarrow \mathbf{y}_{\mathrm{tmp}}$.
3: Orthogonalize $\mathbf{y}_{\mathrm{tmp}}$ against $\mathbf{y}_{l-1,n}$.
4: Check $\mathbf{y}_{\mathrm{tmp}}$ for deflation $\rightarrow s_n$, $\mathbf{y}_n$, $\nu(n+1)$, $s_n^{\triangle}$, $\mathbf{y}_n^{\triangle}$, $\nu^{\triangle}(n+1)$
5: **if** $s_n = 0$ **then**
6: Set $l_t = l$, $n_t = n$, $\widetilde{n}_t = \widetilde{n}$.
7: Set $k_{l_t+1} = \nu(n_t)$, $\widetilde{k}_{l_t+1} = \widetilde{\nu}(\widetilde{n}_t)$, $l = l+1$.
8: Set $\mathbf{z}_{l_t} = [\, y_{k_{l_t}}, \ldots, y_{\nu(n_t)-1}\,]$, $\widetilde{\mathbf{z}}_{l_t} = [\, \widetilde{y}_{k_{l_t}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n}_t)-1}\,]$ and $\boldsymbol{\delta}_{l_t} = \widetilde{\mathbf{z}}_{l_t}^{\star}\widetilde{\mathbf{z}}_{l_t}$.
9: Set $\mathbf{w}_{l_t} = [\, v_{k_{l_t}}, \ldots, v_{\nu(n_t)-1}\,]$, $\widetilde{\mathbf{w}}_{l_t} = [\, \widetilde{v}_{k_{l_t}}, \ldots, \widetilde{v}_{\widetilde{\nu}(\widetilde{n}_t)-1}\,]$ and $\boldsymbol{\delta}'_{l_t} = \widetilde{\mathbf{w}}_{l_t}^{\star}\mathbf{A}\mathbf{w}_{l_t}$.
10: **stop**
11: **end if**
12: Set $\mathbf{v}_n = \mathbf{y}_n$ and orthogonalize it against $\mathbf{A}^{\star}\widetilde{\mathbf{w}}_{\widetilde{\ell}^{\star}(n)+1}, \ldots, \mathbf{A}^{\star}\widetilde{\mathbf{w}}_{l-1}$.
13: Normalize columns of $\mathbf{v}_n$.
14: Compute $\mathbf{A}\mathbf{v}_n$.
15: Set $n = n+1$.
16: LA5 Fix value for k_{l+1} (changed or initial value).
17: **end while**

Proof. It is clear that (2.16) and (2.40) hold when the main loop in line 12.10 is reached for the first time since (2.16a), (2.16b), (2.40a) and (2.40b) are all empty at this point and the columns of $\mathbf{y}_0$, $\widetilde{\mathbf{y}}_0$, $\mathbf{v}_0$ and $\widetilde{\mathbf{v}}_0$ are linearly independent.

Next we have to show that (2.16) and (2.40) form an invariant of the loop starting at line 12.10. Assume that (2.16) and (2.40) hold at the beginning of the main loop. If Algorithm 13 is entered, the **while** loop on line 13.1 is executed. We assume that the six properties are valid at the beginning of the **while** loop. As in the case of BLBIO, we can show that the inverse in the expression for $\boldsymbol{\alpha}'$ above exists and that all six properties are still valid when line 13.5 is reached.

If $\widetilde{s}_{\widetilde{n}}$ is zero, we exit the algorithm by creating the two final clusters $\mathbf{z}_{l_t}$ and $\widetilde{\mathbf{z}}_{l_t}$ and also the two final direction clusters $\mathbf{w}_{l_t}$ and $\widetilde{\mathbf{w}}_{l_t}$. Consequently, at line 13.10, (2.16a) and (2.40a) hold with $l-2 = l_t - 1$ in place of $l-1$, but due to (2.16b), (2.40b) and the definitions of $\boldsymbol{\delta}_{l_t}$ and $\boldsymbol{\delta}'_{l_t}$, (2.16a) and (2.40a) also hold for $l-1 = l_t$. The definitions of the last regular indices k_{l_t+1} and $\widetilde{k}_{l_t+1}$ imply that the sets $\{y_j\}_{j\geq k_l}$, $\{\widetilde{y}_j\}_{j\geq \widetilde{k}_l}$, $\{v_j\}_{j\geq k_l}$ and $\{\widetilde{v}_j\}_{j\geq \widetilde{k}_l}$ are empty. So we have shown that if BLBIOC stops at line 13.10, then (2.16) and (2.40) hold with $l = l_t + 1$.

In line 13.12, a first version of $\widetilde{\mathbf{v}}_{\widetilde{n}}$ is computed by first orthogonalizing $\widetilde{\mathbf{y}}_{\widetilde{n}}$ against $\mathbf{A}\mathbf{w}_0, \ldots, \mathbf{A}\mathbf{w}_{l-1}$, and these vectors are added to the set of ungrouped left direction vectors. We temporarily denote by $\widetilde{\mathbf{v}}'_{l-1,\widetilde{n}} :\equiv [\, \widetilde{v}'_{k_l}, \ldots, \widetilde{v}'_{\widetilde{\nu}(\widetilde{n})-1}\,]$ the ungrouped left direction vectors after line 13.12. Assume now that the columns of $\widetilde{\mathbf{v}}'_{l-1,\widetilde{n}}$ are linearly dependent. Then there is a vector $t \in \mathbb{C}^{\widetilde{\nu}(\widetilde{n})-k_l} \setminus \{0\}$ such that $\widetilde{\mathbf{v}}'_{l-1,\widetilde{n}}t = 0$. But all the columns of $\widetilde{\mathbf{v}}'_{l-1,\widetilde{n}}$ were obtained from the currently ungrouped left vectors $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}}$ by orthogonalization against $\mathbf{A}\mathbf{w}_0, \ldots, \mathbf{A}\mathbf{w}_{l-1}$ so that there are matrices $\widetilde{\mathbf{h}}_{\widetilde{n},j}^{l-1}$ for $j = 0, \ldots, l-1$ such that

$$\widetilde{\mathbf{v}}'_{l-1,\widetilde{n}} = \widetilde{\mathbf{y}}_{l-1,\widetilde{n}} - \sum_{j=0}^{l-1} \widetilde{\mathbf{w}}_j \widetilde{\mathbf{h}}_{\widetilde{n},j}^{l-1},$$

which according to (2.21b) also leads to a relation of the form

$$\widetilde{\mathbf{v}}'_{l-1,\widetilde{n}} = \widetilde{\mathbf{y}}_{l-1,\widetilde{n}} - \sum_{j=0}^{l-1} \widetilde{\mathbf{z}}_j \widetilde{\mathbf{h}}'^{l-1}_{\widetilde{n},j},$$

and we conclude that $0 = \widetilde{\mathbf{y}}_{l-1,\widetilde{n}}t - \sum_{j=0}^{l-1} \widetilde{\mathbf{z}}_j t_j$ (where $t_j :\equiv \widetilde{\mathbf{h}}'^{l-1}_{\widetilde{n},j}t$). By (2.16b), the block $\widetilde{\mathbf{y}}_{l-1,\widetilde{n}}$ is orthogonal to $\mathbf{z}_0, \ldots, \mathbf{z}_{l-1}$ whence $t_j = 0$ for $j = 0, \ldots, l-1$. Then we derive from (2.16c) that $t = 0$, in contradiction to the above assumption. Consequently, the ungrouped left direction vectors are linearly independent when line 13.12 has been executed. This also means that the normalization of

the columns of $\widetilde{\mathbf{v}}_{\widetilde{n}}$ is possible (compare (2.34) and line 13.13). We conclude that all six properties are still valid after the left block Krylov space has been extended.

Next, lines 6.12 to 6.22 are executed, which does not influence the validity of our assertion, and neither do lines 12.15 to 12.33. As in the case of BLBIO, it is quite easy to show that the orthogonalizations in lines 12.36 and 12.37 preserve linear independency of the corresponding vector sequences and therefore, all the six assertions are still valid when the right block Krylov space is extended. A similar argument as outlined above for the left block Krylov space shows that the assertions are still true at the end of the main loop of BLBIOC. □

Some properties of BLBIOC which follow immediately from the previous lemma are summarized in the following proposition.

Proposition 15. *Assume that* BLBIOC *is run with an arbitrary look-ahead procedure according to Definition 3 (and with the same rule applied in step LA2' as in step LA2). Then in addition to the statements of Proposition 6, the following holds:*

(i) All right and left direction vectors are of unit length.

(ii) Outside of step 3), the number of (possibly preliminary) v vectors computed so far is given by $\nu(n)$, while outside of step 1), the number of (possibly preliminary) $\widetilde{v}$ vectors computed so far is $\widetilde{\nu}(\widetilde{n})$. In other words, the right direction blocks $\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}$ are available outside of step 3) and the left blocks $\widetilde{\mathbf{v}}_0, \ldots, \widetilde{\mathbf{v}}_{\widetilde{n}-1}$ are available outside of step 1), and the blocks $\mathbf{v}_{n-1}$ and $\widetilde{\mathbf{v}}_{\widetilde{n}-1}$ may be preliminary.

(iii) Outside of step 2), the right direction clusters $\mathbf{w}_0, \ldots, \mathbf{w}_{l-1}$ and the left direction clusters $\widetilde{\mathbf{w}}_0, \ldots, \widetilde{\mathbf{w}}_{l-1}$ have been completed.

(iv) After termination, (ii) and (iii) hold with $n = n_t$, $\widetilde{n} = \widetilde{n}_t$ and $l = l_t + 1$, and all computed right and left direction vectors are grouped into a direction cluster.

(v) The right and left direction clusters $\mathbf{w}_i$ and $\widetilde{\mathbf{w}}_i$ both have r_i columns for $i = 0, \ldots, l_t - 1$. The last right and left direction clusters contain r_{l_t} and $\widetilde{r}_{l_t}$ vectors, respectively. Additionally, the biorthogonality (2.22) holds for $0 \leq i, j \leq l_t$.

(vi) In exact arithmetic an with a zero deflation tolerance, the right and left direction vectors form nested sequences of bases of $\mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ and $\widetilde{\mathcal{B}}_n(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$, respectively. In particular, the columns of $\mathbf{v}_n$ span $\mathcal{B}_{n+1}(\mathbf{A}, \mathbf{y}_0) \ominus \mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$ $(n = 0, \ldots, n_t - 1)$, while the columns of $\widetilde{\mathbf{v}}_{\widetilde{n}}$ span $\widetilde{\mathcal{B}}_{\widetilde{n}+1}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0) \ominus \widetilde{\mathcal{B}}_{\widetilde{n}}(\mathbf{A}^\star, \widetilde{\mathbf{y}}_0)$ $(\widetilde{n} = 0, \ldots, \widetilde{n}_t - 1)$.

(vii) When direction clusters $\mathbf{w}_{l-1}$ and $\widetilde{\mathbf{w}}_{l-1}$ are completed, the variables n and $\widetilde{n}$ have the values $n(k_l)$ and $\widetilde{n}(k_l)$, respectively. In other words, BLBIOC *only computes as many right and left direction blocks as necessary for testing regular indices.*

The proof of this proposition is very similar to the case of BLBIO and is therefore omitted. The next proposition gives some information about the structure of $\underline{\mathbf{L}}_n$ which will be useful for the construction of block Krylov space methods.

Proposition 16. *Assume exact arithmetic and a zero deflation tolerance. Then the matrix $\underline{\mathbf{L}}_n$ is lower block triangular, see Figure 9.*

Proof. From the definition of $\underline{\mathbf{L}}_n$, we know that it is block upper Hessenberg. Now we multiply (2.23) by $\widetilde{\mathbf{Y}}_{\widetilde{n}}^\star$ from the left and get

$$\widetilde{\mathbf{Y}}_{\widetilde{n}}^\star \mathbf{A}\mathbf{V}_n = \widetilde{\mathbf{Y}}_{\widetilde{n}}^\star \mathbf{Y}_{n+1} \underline{\mathbf{L}}_n = \widetilde{\mathbf{Y}}_{\widetilde{n}}^\star \big[\, \mathbf{Y}_n \; \mathbf{y}_n \,\big] \underline{\mathbf{L}}_n = \big[\, \mathbf{D}_{\widetilde{n},n} \; \star_{\widetilde{\nu}(\widetilde{n}) \times s_n} \,\big] \underline{\mathbf{L}}_n.$$

On the other hand,

$$\big(\mathbf{A}\mathbf{V}_n\big)^\star \widetilde{\mathbf{Y}}_{\widetilde{n}} = \big(\mathbf{A}\mathbf{V}_n\big)^\star \widetilde{\mathbf{V}}_{\widetilde{n}} \widetilde{\mathbf{U}}_{\widetilde{n}} = (\mathbf{D}'_{\widetilde{n},n})^\star \widetilde{\mathbf{U}}_{\widetilde{n}},$$

where

$$\mathbf{D}'_{\widetilde{n},n} :\equiv \widetilde{\mathbf{V}}_{\widetilde{n}}^\star \mathbf{A}\mathbf{V}_n = \text{block diag}(\boldsymbol{\delta}'_0, \ldots, \boldsymbol{\delta}'_{l-1}, \boldsymbol{\delta}'_{l;\widetilde{n},n}) \; \in \mathbb{C}^{\widetilde{\nu}(\widetilde{n}) \times \nu(n)},$$

where $k_l + 1 \leq \min\{\nu(n), \widetilde{\nu}(\widetilde{n})\}$, so that

$$\big[\, \mathbf{D}_{\widetilde{n},n} \; \star_{\widetilde{\nu}(\widetilde{n}) \times s_n} \,\big] \underline{\mathbf{L}}_n = \widetilde{\mathbf{U}}_{\widetilde{n}}^\star \mathbf{D}'_{\widetilde{n},n},$$

where the matrix on the right-hand side is block lower triangular with diagonal blocks of size $r_i \times r_i$. This shows the assertion. □

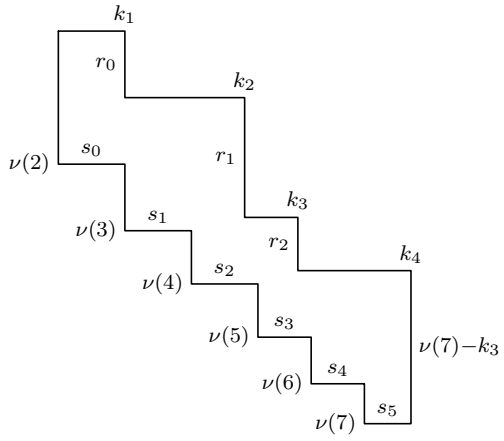


Figure 9: Nonzero pattern of $\underline{\mathbf{L}}_n$

		large clusters		small clusters	
BLBIO	Orthogonalization	full	local	full	local
	l_t+1, $\nu(n_t)$, $\widetilde{\nu}(\widetilde{n}_t)$	5, 50, 50	5, 50, 51	50, 50, 50	60, 60, 60
	N	$2.80 \cdot 10^{-14}$	$2.79 \cdot 10^{-14}$	$2.49 \cdot 10^{-14}$	$2.52 \cdot 10^{-14}$
	$\widetilde{N}$	$5.79 \cdot 10^{-15}$	$9.02 \cdot 10^{-15}$	$1.47 \cdot 10^{-14}$	$1.48 \cdot 10^{-14}$
BLBIOC	l_t+1, $\nu(n_t)$, $\widetilde{\nu}(\widetilde{n}_t)$	5, 50, 50	5, 50, 51	50, 50, 50	58, 58, 60
	N_1	$1.71 \cdot 10^{-14}$	$1.70 \cdot 10^{-14}$	$6.57 \cdot 10^{-14}$	$9.64 \cdot 10^{-14}$
	N_2	$3.31 \cdot 10^{-14}$	$1.31 \cdot 10^{-14}$	$1.88 \cdot 10^{-14}$	$3.25 \cdot 10^{-14}$
	$\widetilde{N}_1$	$5.42 \cdot 10^{-15}$	$5.71 \cdot 10^{-15}$	$8.15 \cdot 10^{-14}$	$7.06 \cdot 10^{-15}$
	$\widetilde{N}_2$	$1.17 \cdot 10^{-13}$	$4.03 \cdot 10^{-7}$	$2.80 \cdot 10^{-14}$	$1.85 \cdot 10^{-14}$

Table 2: Results for 50×50 random matrix

2.5 Numerical examples

As described on page 13, a check for deflation in Algorithm 6 involves a (high) rank revealing QR factorization. In our Matlab implementations of BLBIO and BLBIOC, we use the algorithm which was developed by Chan [3] based on earlier work by Foster [7]. A reference implementation of the Chan-Foster HRRQR Algorithm can be found in [5]. For this algorithm, the quantity $\|\boldsymbol{\rho}_{\widetilde{n}}^{\triangle}\|_2$ can be bounded as in [3, Theorem 3.1].

Now we present a few test examples. They all have been carried out with Matlab 7 on a PC with a 2.8GHz Pentium IV processor. First we choose the tolerance for the singularity tests $\mathrm{tol}_{\mathrm{sing}} = \mathrm{tol}_{\mathrm{defl}} = 10^{-6}$. Now take as matrix $\mathbf{A}$ a 50×50 random matrix generated by the Matlab commands `rand('seed',0)` (to reset the pseudo random number generator to its initial state) and `rand(50,50)`. As right input block 10 random vectors were taken and three random vectors formed the left input block. Now BLBIO and BLBIOC were run with different options (large/small clusters, full/local orthogonalization). Look-ahead never occurred in this experiment and therefore no permutations either. With small clusters and full orthogonalization, BLBIO stopped at line 8.9 (i.e. an $\mathbf{A}$-invariant subspace was found) when $\mathbf{z}_{49}$ and $\widetilde{\mathbf{z}}_{49}$ were completed. When local orthogonalization was used, the iteration continued instead until the maximum number of right and left vectors (which here was 60) was reached. On the other hand, with large clusters and local orthogonalization, we obtained 50 right and 51 left vectors which were grouped into 5 clusters containing 10 vectors. Consequently, one left vector remained ungrouped at the end. For checking the BLBIO recursions, Table 2 gives the two norms

$$N := \equiv \|\mathbf{A}\mathbf{Y}_{n_t-1} - \mathbf{Y}_{n_t}\underline{\mathbf{T}}_{n_t-1} - \mathbf{Y}_{n_t}^{\triangle}\underline{\mathbf{T}}_{n_t-1}^{\triangle}\|_2$$
$$\widetilde{N} := \equiv \|\mathbf{A}^{\star}\widetilde{\mathbf{Y}}_{\widetilde{n}_t-1} - \widetilde{\mathbf{Y}}_{\widetilde{n}_t}\underline{\widetilde{\mathbf{T}}}_{\widetilde{n}_t-1} - \widetilde{\mathbf{Y}}_{\widetilde{n}_t}^{\triangle}\underline{\widetilde{\mathbf{T}}}_{\widetilde{n}_t-1}^{\triangle}\|_2,$$

and we see that the BLBIO recursions are satisfied up to rounding errors. The BLBIOC recursions

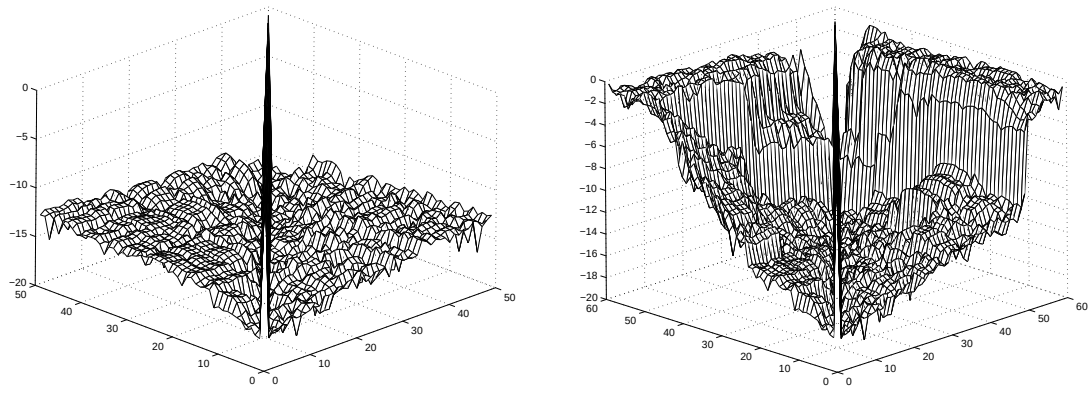


Figure 10: Loss of biorthogonality with small clusters and with full (left) and local orthogonalization

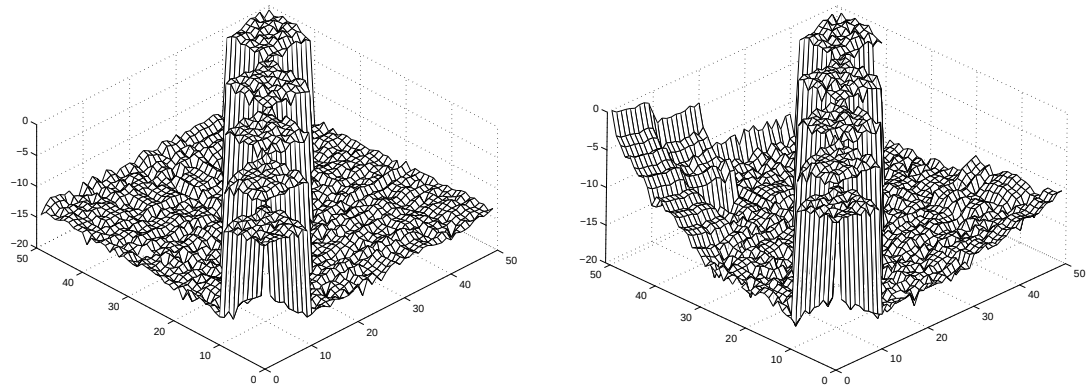


Figure 11: Loss of biorthogonality with larger clusters and with full (left) and local orthogonalization

are verified by computing the norms

$$
\begin{aligned}
N_1 &:\equiv \|\mathbf{A}\mathbf{Y}_{n_t-1} - \mathbf{Y}_{n_t}\underline{\mathbf{L}}_{n_t-1} - \mathbf{Y}^{\triangle}_{n_t}\underline{\mathbf{T}}^{\triangle}_{n_t-1}\|_2 \\
N_2 &:\equiv \|\mathbf{Y}_{n_t} - \mathbf{V}_{n_t}\mathbf{U}_{n_t}\|_2 \\
\widetilde{N}_1 &:\equiv \|\mathbf{A}^{\star}\widetilde{\mathbf{Y}}_{\widetilde{n}_t-1} - \widetilde{\mathbf{Y}}_{\widetilde{n}_t}\underline{\widetilde{\mathbf{L}}}_{\widetilde{n}_t-1} - \widetilde{\mathbf{Y}}^{\triangle}_{\widetilde{n}_t}\underline{\widetilde{\mathbf{T}}}^{\triangle}_{\widetilde{n}_t-1}\|_2 \\
\widetilde{N}_2 &:\equiv \|\widetilde{\mathbf{Y}}_{\widetilde{n}_t} - \widetilde{\mathbf{V}}_{\widetilde{n}_t}\widetilde{\mathbf{U}}_{\widetilde{n}_t}\|_2.
\end{aligned}
$$

Concerning the relatively large value of $\widetilde{N}_2$ in the case of large clusters and local orthogonalization, we remark that cancellation occurred when the relation $\widetilde{\mathbf{Y}}_{\widetilde{n}} = \widetilde{\mathbf{V}}_{\widetilde{n}}\widetilde{\mathbf{U}}_{\widetilde{n}}$ was adapted after completion of $\mathbf{z}_4$ and $\widetilde{\mathbf{z}}_4$. So we conclude that the BLBIOC recursions are also satisfied up to rounding errors.

To check the biorthogonality we show the absolute values of the inner products $\widetilde{\mathbf{Y}}^{\star}_{\widetilde{n}_t}\mathbf{Y}_{n_t}$ in Figure 10 for small and in Figure 11 for large clusters. Here the scale on the z axis is logarithmic with base 10. As was to be expected, in Figure 10 local orthogonalization leads to global biorthogonality being lost quickly but there is still fairly good local biorthogonality. A similar statement also holds for the $\mathbf{A}$ biorthogonality in the case of BLBIOC, see Figure 12.

If we take the same matrix but only one right and left starting vector (which again are random vectors) then there is no difference between the two look-ahead strategies and since no look-ahead occurred, the matrices $\underline{\mathbf{T}}_{n_t-1}$ and $\underline{\widetilde{\mathbf{T}}}_{\widetilde{n}_t-1}$ computed by BLBIO are tridiagonal. To check this, we set the elements of $\underline{\mathbf{T}}_{n_t-1}$ on the main and on the first upper and lower codiagonal to zero, and the resulting matrix (which should be zero) had a 2-norm of $3.9\cdot 10^{-10}$, and the norm of the corresponding matrix resulting from $\underline{\widetilde{\mathbf{T}}}_{\widetilde{n}_t-1}$ was $4.0\cdot 10^{-11}$. The following table gives some additional information about this experiment:

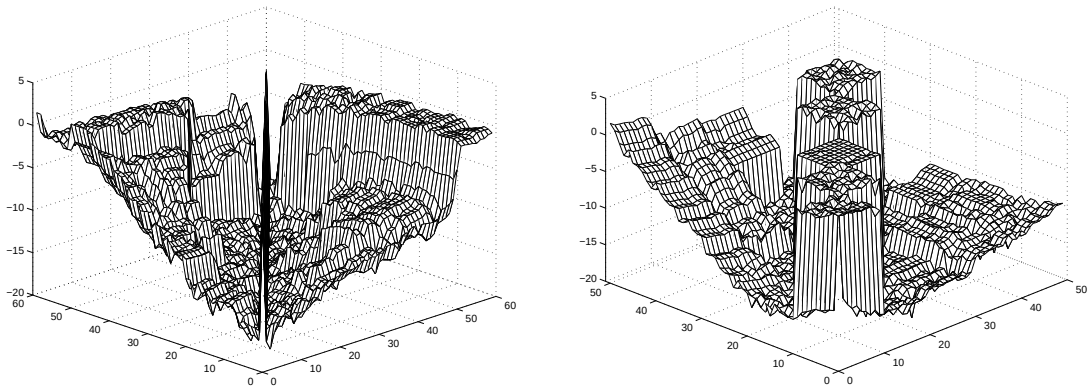


Figure 12: Loss of $\mathbf{A}$ biorthogonality in BLBIOC for small (left) and large (right) clusters (local orthog.)

	Orthogonalization	full	local
BLBIO	l_t+1, $\nu(n_t)$, $\widetilde{\nu}(\widetilde{n}_t)$	50, 50, 50	60, 60, 60
	N	$2.61 \cdot 10^{-15}$	$1.94 \cdot 10^{-14}$
	$\widetilde{N}$	$5.92 \cdot 10^{-15}$	$1.79 \cdot 10^{-14}$
BLBIOC	l_t+1, $\nu(n_t)$, $\widetilde{\nu}(\widetilde{n}_t)$	50, 50, 50	60, 60, 60
	N_1	$2.85 \cdot 10^{-15}$	$3.67 \cdot 10^{-15}$
	N_2	$1.49 \cdot 10^{-15}$	$3.16 \cdot 10^{-14}$
	$\widetilde{N}_1$	$5.94 \cdot 10^{-15}$	$5.91 \cdot 10^{-15}$
	$\widetilde{N}_2$	$2.45 \cdot 10^{-15}$	$2.29 \cdot 10^{-14}$

In all the examples described so far, look-ahead never occurred, whence we now turn to a more realistic problem. Our matrix is now Arc130 from the Original Harwell sparse matrix test collection [16]. This matrix has order 130 and is fairly ill-conditioned (its 2-norm condition number is of order 10^{10}). The starting blocks contained 5 right and 3 left random vectors. For the small clusters we get

	Orthogonalization	full	local
BLBIO	l_t+1, $\nu(n_t)$, $\widetilde{\nu}(\widetilde{n}_t)$	24, 133, 133	13, 123, 123
	N	$2.76 \cdot 10^{-10}$	$2.67 \cdot 10^{-10}$
	$\widetilde{N}$	$5.45 \cdot 10^{-11}$	$5.45 \cdot 10^{-11}$
BLBIOC	l_t+1, $\nu(n_t)$, $\widetilde{\nu}(\widetilde{n}_t)$	7, 68, 69	7, 58, 60
	N_1	$6.00 \cdot 10^{-11}$	$5.98 \cdot 10^{-11}$
	N_2	$6.69 \cdot 10^{-12}$	$1.10 \cdot 10^{-11}$
	$\widetilde{N}_1$	$3.37 \cdot 10^{-11}$	$3.37 \cdot 10^{-11}$
	$\widetilde{N}_2$	$5.08 \cdot 10^{-15}$	$3.68 \cdot 10^{-15}$

So we see that with full orthogonalization BLBIO only managed to complete 24 clusters although 133 right and left vectors were computed, and similarly with local orthogonalization. The values of the regular indices were

BLBIO	$k_0, \ldots, k_8$	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	$k_{15}, \ldots, k_{24}$	$\widetilde{k}_{l_t+1}$
full orth.	$0, \ldots, 8$	113	114	115	116	117	123	$124, \ldots, 133$	133
local orth.	$0, \ldots, 8$	113	114	115	116	117	123		123

We see that the clusters $\mathbf{z}_8$ and $\widetilde{\mathbf{z}}_8$ both contain 105 vectors irrespective of whether full or local orthogonalization was used. In both cases, the singularity of $\boldsymbol{\delta}_8$ was removed by a permutation. With local orthogonalization, 123 right and left vectors were computed and 117 of them were grouped into a cluster when the algorithm terminated. On the other hand, with full orthogonalization, 133 right and left vectors were computed and all of them were grouped into clusters at termination. The

biorthogonality is lost quickly, even with full orthogonalization. We note that with this matrix the ordinary Lanczos process with look-ahead also has troubles: If random vectors are used as starting vectors it only manages to complete two pairs of Lanczos vectors and then starts a cluster which keeps growing until the iteration limit of 200 steps is reached.

3 Three Lanczos process based block Krylov space methods

In this section we turn to the task of solving multiple systems of linear equations,

$$\mathbf{A}x^{(j)} = b^{(j)}, \qquad j = 1, \ldots, s \tag{3.1}$$

with the same nonsingular coefficient matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$ but different right-hand sides $b^{(j)} \in \mathbb{C}^N$. It is always assumed that all right-hand side vectors $b^{(j)}$ are available simultaneously so that (3.1) is equivalent to the block system

$$\mathbf{AX} = \mathbf{B} \tag{3.2}$$

where $\mathbf{B} = [\, b^{(1)}, \ldots, b^{(s)} \,]$ and $\mathbf{X} = [\, x^{(1)}, \ldots, x^{(s)} \,]$. The goal of the present section consists in constructing block Krylov space methods on the basis of the BLBIO and BLBIOC algorithms outlined in the previous section. To this end, we will proceed in a way similar to [13] and [4]. Since the analysis of effects resulting from finite precision arithmetic is anyway far beyond the scope of this work, we usually assume **exact arithmetic** and for simplicity also a **zero deflation tolerance** for BLBIO(C).

3.1 Fundamentals of block Krylov space methods

For any block $\mathbf{X}_0$ of initial guesses for the exact solution $\mathbf{X}$ of (3.2) we can form the block of corresponding initial residuals $\mathbf{R}_0 = \mathbf{B} - \mathbf{AX}_0$. Now we define

Definition 17. An iterative procedure is called a *block Krylov space method* if it generates a sequence of block iterates $\mathbf{X}_n = [\, x_n^{(1)}, \ldots, x_n^{(s)} \,]$ for $n = 0, 1, \ldots$ such that $\mathbf{X}_n \in \mathbf{X}_0 + \mathcal{B}_n^{\square}(\mathbf{A}, \mathbf{R}_0) \subset \mathbb{C}^{N\times s}$ or equivalently

$$x_n^{(i)} \in x_0^{(i)} + \mathcal{B}_n(\mathbf{A}, \mathbf{R}_0), \qquad i = 1, \ldots, s. \tag{3.3}$$

Let $\mathbf{R}_n :\equiv \mathbf{B} - \mathbf{AX}_n = [\, r_n^{(1)}, \ldots, r_n^{(s)} \,]$ be the block of residuals corresponding to $\mathbf{X}_n$. From (3.3) we conclude

$$r_n^{(i)} \in r_0^{(i)} + \mathbf{A}\mathcal{B}_n(\mathbf{A}, \mathbf{R}_0), \qquad i = 1, \ldots, s.$$

So we use the BLBIO(C) algorithm with $\mathbf{y} = \mathbf{R}_0$ to get a basis of $\mathcal{B}_n(\mathbf{A}, \mathbf{R}_0)$. In line 6.2 or 12.2, the block $\mathbf{R}_0$ is checked for deflation, and s_0 is defined to be its numerical rank:

$$\mathbf{R}_0 = [\, \mathbf{y}_0 \, \mathbf{y}_0^{\triangle} \,] \begin{bmatrix} \boldsymbol{\eta}_0 \\ \mathbf{0}_{s-s_0\times s} \end{bmatrix} = \mathbf{y}_0\boldsymbol{\eta}_0. \tag{3.4}$$

From this, it follows that $\mathcal{B}_n(\mathbf{A}, \mathbf{R}_0) = \mathcal{B}_n(\mathbf{A}, \mathbf{y}_0)$, whence (3.3) may be replaced by

$$x_n^{(i)} \in x_0^{(i)} + \mathcal{B}_n(\mathbf{A}, \mathbf{y}_0), \qquad i = 1, \ldots, s.$$

From (3.4), we can also find a basis $\gamma_1, \ldots, \gamma_{s-s_0}$ of $\ker(\boldsymbol{\eta}_0)$ (where $\gamma_i = (\gamma_{i,1}, \ldots, \gamma_{i,s})^\top \in \mathbb{C}^s$). Since $\mathbf{R}_0\gamma_i = 0$, we know the solutions of the systems $\mathbf{A}x_{\gamma_i} = \mathbf{B}\gamma_i$ where $x_{\gamma_i} :\equiv \mathbf{X}_0\gamma_i$ for $i = 1, \ldots, s$. By multiplying (3.2) by γ_1 we get $\mathbf{AX}\gamma_i = \mathbf{B}\gamma_i = \mathbf{A}x_{\gamma_i}$, and if j_i is an index such that $\gamma_{i,j_i} \neq 0$, then at the end of the iteration, we can compute the solution vector $x^{(j_i)}$ via

$$x^{(j_i)} = \frac{1}{\gamma_{i,j_i}} \left(x_{\gamma_i} - \sum_{\substack{j=1 \\ j\neq j_i}}^{s} x^{(j)}\gamma_{i,j} \right). \tag{3.5}$$

So it is, in general, not necessary to keep all the iterate vectors for all iterations of the block Krylov method, and every block Krylov method should eliminate the superfluous iterate vectors. We first introduce some convenient notation and then turn to an algorithm for this elimination.

As in [8], we denote by $\mathbf{X}_n^{\mathrm{cr}} \in \mathbb{C}^{N\times m_{\mathrm{cr}}}$ the block of iterates which are retained at step n, after deflation, for subsequent iterations. Let $J_n :\equiv \{i_1,\ldots,i_{m_{\mathrm{cr}}}\}$ be the set containing the column indices of those linear systems in (3.1) that are present in $\mathbf{X}_n^{\mathrm{cr}}$ so that $\mathbf{X}_n^{\mathrm{cr}} = [\,x_n^{(i_1)},\ldots,x_n^{(i_{m_{\mathrm{cr}}})}\,]$. The subset of linear systems in (3.1) that correspond to the columns of the currently active block $\mathbf{X}_n^{\mathrm{cr}}$ can be written as $\mathbf{A}\mathbf{X}^{\mathrm{cr}} = \mathbf{B}^{\mathrm{cr}}$, where $\mathbf{X}^{\mathrm{cr}} = [\,x^{(i_1)},\ldots,x^{(i_{m_{\mathrm{cr}}})}\,]$ and similarly for $\mathbf{B}^{\mathrm{cr}}$ and any other matrix with superscript $^{\mathrm{cr}}$. In particular, we have $\mathbf{R}_j^{\mathrm{cr}} = \mathbf{B}^{\mathrm{cr}} - \mathbf{A}\mathbf{X}_j^{\mathrm{cr}} = [\,r_j^{(i_1)},\ldots,r_j^{(i_{m_{\mathrm{cr}}})}\,]$ for $j = 0,1,\ldots$.

Now we derive an algorithm which eliminates $s - s_0$ linear systems from the iteration. We first choose $j_1 = \max_{1\le j\le s}|\gamma_{1,j}|$ and remove column j_1 from all matrices superscripted $^{\mathrm{cr}}$. Assuming $s - s_0 > 1$ and using the fact that

$$r_0^{(j_1)} = -\frac{1}{\gamma_{1,j_1}}\sum_{\substack{j=1\\ j\neq j_1}}^{s} r_0^{(j)}\gamma_{1,j},$$

we get for $2 \le i \le s - s_0$

$$0 = \mathbf{R}_0^{\mathrm{cr}}\gamma_i = \sum_{\substack{j=1\\ j\neq j_1}}^{s} r_0^{(j)}\left(\gamma_{i,j} - \frac{\gamma_{i,j_1}}{\gamma_{1,j_1}}\gamma_{1,j}\right) = \mathbf{R}_0^{\mathrm{cr}}\widetilde{\gamma}_i,$$

where

$$\begin{aligned}\mathbf{R}_0^{\mathrm{cr}} &:\equiv [\,r_0^{(1)},\ldots,r_0^{(j_1-1)},r_0^{(j_1+1)},\ldots,r_0^{(s)}\,]\\ \widetilde{\gamma}_i &:\equiv \left(\gamma_{i,l} - \frac{\gamma_{i,j_1}}{\gamma_{1,j_1}}\gamma_{1,l}\right)_{l=1,\ldots,j_1-1,j_1+1,\ldots,s}.\end{aligned}$$

So we have eliminated column j_1, and $\ker(\mathbf{R}_0^{\mathrm{cr}})$ is of dimension $s - s_0 - 1$. Furthermore, the vectors $\widetilde{\gamma}_2,\ldots,\widetilde{\gamma}_{s-s_0}$ are elements of $\ker(\mathbf{R}_0^{\mathrm{cr}})$. Now we show that they are linearly independent. To this end, assume the contrary so that $0_{s-1} = \sum_{i=2}^{s-s_0}\lambda_i\widetilde{\gamma}_i$ with complex numbers $\lambda_2,\ldots,\lambda_{s-s_0}$ which are not all zero. Now define

$$\widetilde{\gamma}_i' :\equiv \begin{pmatrix}\gamma_{i,1} - \frac{\gamma_{i,j_1}}{\gamma_{1,j_1}}\gamma_{1,1}\\ \vdots\\ \gamma_{i,s} - \frac{\gamma_{i,j_1}}{\gamma_{1,j_1}}\gamma_{1,s}\end{pmatrix} = \gamma_i - \frac{\gamma_{i,j_1}}{\gamma_{1,j_1}}\gamma_1 \in \mathbb{C}^s$$

so that $\widetilde{\gamma}_i'$ is the same as $\widetilde{\gamma}_i$ except that at position j_1, a zero has been inserted. Therefore we obtain

$$0_s = \sum_{i=2}^{s-s_0}\lambda_i\widetilde{\gamma}_i' = \sum_{i=2}^{s-s_0}\lambda_i\gamma_i - \left(\sum_{i=2}^{s-s_0}\lambda_i\frac{\gamma_{i,j_1}}{\gamma_{1,j_1}}\right)\gamma_1,$$

but as $\gamma_1,\ldots,\gamma_s$ are linearly independent, it follows that $\lambda_i = 0$ for $i = 2,\ldots,s - s_0$. We conclude that the vectors $\widetilde{\gamma}_2,\ldots,\widetilde{\gamma}_{s-s_0}$ form a basis of $\ker(\mathbf{R}_0^{\mathrm{cr}})$.

This whole process may be repeated if $s - s_0 - 1 > 1$ until we arrive at a block $\mathbf{R}_0^{\mathrm{cr}} \in \mathbb{C}^{N\times s_0}$ with full rank s_0. From (3.4), it follows that we can write $\mathbf{R}_0^{\mathrm{cr}} = \mathbf{y}_0\boldsymbol{\eta}_0^{\mathrm{cr}}$ where $\boldsymbol{\eta}_0^{\mathrm{cr}}$ is nonsingular.

In a similar way, if $n > 0$, we may determine a basis of $\ker(\mathbf{R}_n^{\mathrm{cr}})$ which enables us to eliminate further linear systems by using an analogue procedure as described above. As a result, we obtain $m_{\mathrm{cr}} \le s_n$. The details of the deflation of $\mathbf{R}_n^{\mathrm{cr}}$ for $n > 0$ depend on the block Krylov method used and so we will come back to this point later.

BLBIO(C) also requires that we choose some left starting block $\widetilde{\mathbf{y}}$, which might be a block of random vectors with the same number of columns as $\mathbf{R}_0$. Now we know from Proposition 6, part (vi), that for $n > 0$, the columns of $\mathbf{Y}_n$ form a basis of the space $\mathcal{B}_n(\mathbf{A},\mathbf{R}_0)$ and consequently (3.3)

implies the existence of a vector $k_n^{(i)} \in \mathbb{C}^{\nu(n)}$ such that $x_n^{(i)} = x_0^{(i)} + \mathbf{Y}_n k_n^{(i)}$ $(i \in J_n)$, and by setting $\mathbf{k}_n^{\mathrm{cr}} :\equiv [\, k_n^{(i_1)}, \ldots, k_n^{(i_{m_{\mathrm{cr}}})} \,] \in \mathbb{C}^{\nu(n) \times m_{\mathrm{cr}}}$ we obtain

$$\mathbf{X}_n^{\mathrm{cr}} = \mathbf{X}_0^{\mathrm{cr}} + \mathbf{Y}_n \mathbf{k}_n^{\mathrm{cr}}, \qquad n = 1, 2, \ldots, \tag{3.6}$$

so that together with Proposition 7 it follows for the corresponding residual blocks

$$\mathbf{R}_n^{\mathrm{cr}} = \mathbf{R}_0^{\mathrm{cr}} - \mathbf{A}\mathbf{Y}_n \mathbf{k}_n^{\mathrm{cr}} = \mathbf{R}_0^{\mathrm{cr}} - \mathbf{Y}_{n+1} \underline{\mathbf{T}}_n \mathbf{k}_n^{\mathrm{cr}}. \tag{3.7}$$

Next we define $\underline{\mathbf{e}}_0 :\equiv [\, \mathbb{1}_{s_0} \; \mathbf{0}_{s_0 \times (\nu(n+1) - s_0)} \,]^\top \in \mathbb{C}^{\nu(n+1) \times s_0}$, which yields $\mathbf{R}_0^{\mathrm{cr}} = \mathbf{Y}_{n+1} \underline{\mathbf{e}}_0 \boldsymbol{\eta}_0^{\mathrm{cr}}$, and (3.7) becomes

$$\mathbf{R}_n^{\mathrm{cr}} = \mathbf{Y}_{n+1} (\underline{\mathbf{e}}_0 \boldsymbol{\eta}_0^{\mathrm{cr}} - \underline{\mathbf{T}}_n \mathbf{k}_n^{\mathrm{cr}}). \tag{3.8}$$

Different block Krylov methods differ in the way the matrix $\mathbf{k}_n^{\mathrm{cr}} \in \mathbb{C}^{\nu(n+1) \times m_{\mathrm{cr}}}$ is determined. We first turn to BLQMR and then to BLBICG.

3.2 BLQMR

The idea behind BLQMR is very similar to the nonblock case considered in Subsection 1.3 We would like to choose the matrix $\mathbf{k}_n^{\mathrm{cr}}$ in such a way that $\|\mathbf{R}_n^{\mathrm{cr}}\|_2$ is minimized. In general, however, the columns of $\mathbf{Y}_{n+1}$ are not orthonormal, so this would be too expensive. We choose to minimize the *quasi-residual*, that is

$$\|\underline{\mathbf{e}}_0 \boldsymbol{\eta}_0^{\mathrm{cr}} - \underline{\mathbf{T}}_n \mathbf{k}_n^{\mathrm{cr}}\|_2 = \min_{\mathbf{k}^{\mathrm{cr}} \in \mathbb{C}^{\nu(n) \times m_{\mathrm{cr}}}} \|\underline{\mathbf{e}}_0 \boldsymbol{\eta}_0^{\mathrm{cr}} - \underline{\mathbf{T}}_n \mathbf{k}^{\mathrm{cr}}\|_2. \tag{3.9}$$

Recall that according to Proposition 6, part (i), the columns of $\mathbf{Y}_{n+1}$ are normalized to one so that (3.9) means that all vectors $y_0, \ldots, y_{\nu(n+1)-1}$ are treated with equal weight in the quasi-minimization.

To perform this quasi-minimization we first need to prove the following fact.

Proposition 18. *The matrix $\underline{\mathbf{T}}_n$ generated by* BLBIO *has full column rank, i. e.* $\operatorname{rank}(\underline{\mathbf{T}}_n) = \nu(n)$.

Proof. Assume that $\operatorname{rank}(\underline{\mathbf{T}}_n) < \nu(n)$. Then there is a vector $x \in \mathbb{C}^{\nu(n)}$, not equal to the zero vector, such that $\underline{\mathbf{T}}_n x = 0$. Using Proposition 7 we obtain $\mathbf{A}\mathbf{Y}_n x = \mathbf{Y}_{n+1} \underline{\mathbf{T}}_n x = 0$, but since $\mathbf{A}$ is nonsingular, this yields $\mathbf{Y}_n x = 0$. Now from Proposition 6, part (vi), we know that the columns of $\mathbf{Y}_n$ are linearly independent, so that we get $x = 0$, a contradiction to the assumption made above. □

Let $n > 0$ and

$$\underline{\mathbf{T}}_n = \mathbf{Q}_{n+1} \underline{\mathbf{R}}_n^{\mathrm{QR}} = \mathbf{Q}_{n+1} \begin{bmatrix} \mathbf{R}_n^{\mathrm{QR}} \\ \mathbf{0}_{s_n \times \nu(n)} \end{bmatrix} \tag{3.10}$$

be a QR decomposition of $\underline{\mathbf{T}}_n$ where $\mathbf{Q}_{n+1} \in \mathbb{C}^{\nu(n+1) \times \nu(n+1)}$ is unitary and $\mathbf{R}_n^{\mathrm{QR}} \in \mathbb{C}^{\nu(n) \times \nu(n)}$ upper triangular. Proposition 18 implies that $\mathbf{R}_n^{\mathrm{QR}}$ is nonsingular, whence the least squares problem (3.9) has a unique solution. If we define

$$\begin{bmatrix} \mathbf{t}_1^{\mathrm{cr}} \\ \mathbf{t}_2^{\mathrm{cr}} \end{bmatrix} :\equiv \mathbf{Q}_{n+1}^\star \underline{\mathbf{e}}_0 \boldsymbol{\eta}_0^{\mathrm{cr}}, \qquad \mathbf{t}_1^{\mathrm{cr}} \in \mathbb{C}^{\nu(n) \times m_{\mathrm{cr}}}, \quad \mathbf{t}_2^{\mathrm{cr}} \in \mathbb{C}^{s_n \times m_{\mathrm{cr}}}$$

it follows that $\mathbf{k}_n^{\mathrm{cr}} = (\mathbf{R}_n^{\mathrm{QR}})^{-1} \mathbf{t}_1^{\mathrm{cr}}$, and equations (3.6) and (3.8) become

$$\mathbf{X}_n^{\mathrm{cr}} = \mathbf{X}_0^{\mathrm{cr}} + \mathbf{Y}_n (\mathbf{R}_n^{\mathrm{QR}})^{-1} \mathbf{t}_1^{\mathrm{cr}} \tag{3.11}$$

$$\mathbf{R}_n^{\mathrm{cr}} = \mathbf{Y}_{n+1} \mathbf{Q}_{n+1} \begin{bmatrix} \mathbf{0}_{\nu(n) \times m_{\mathrm{cr}}} \\ \mathbf{t}_2^{\mathrm{cr}} \end{bmatrix}. \tag{3.12}$$

Equation (3.11) still needs to be transformed into a recursion. To do this, we first have to introduce an updating scheme for the QR decomposition of $\underline{\mathbf{T}}_n$ (see next subsection). Using the fact that $\|\mathbf{Y}_n\|_2 \leq \sqrt{\nu(n)}$ we obtain from (3.12)

$$\|\mathbf{R}_n^{\mathrm{cr}}\|_2 \leq \sqrt{\nu(n+1)}\, \|\mathbf{t}_2^{\mathrm{cr}}\|_2.$$

A program implementing our BLQMR algorithm might thus monitor the quantity on the right-hand side of this estimate until it becomes comparable to the desired convergence tolerance and then start computing true residuals for the last few iterations.

Another consequence of (3.12) is that we may determine a basis of $\ker(\mathbf{R}_n^{\mathrm{cr}})$ by computing a basis of $\ker(\mathbf{t}_2^{\mathrm{cr}})$. This allows us to give a complete statement of the deflation of $\mathbf{R}_n^{\mathrm{cr}}$ in the case of BLQMR, see Algorithm 15. It is formulated in such a way that the case $n > 0$ is also covered. In this case,

Algorithm 15 Deflation of block $\mathbf{X}$ of iterates

Input: Index n, block $\mathbf{X}$ of iterates
  if $n = 0$ **then** /* $\mathbf{X} = \mathbf{X}_0$. */
    Initialize all matrices superscripted $^{\mathrm{cr}}$, including $\mathbf{X}_0^{\mathrm{cr}} = \mathbf{X}$, $\mathbf{R}_0^{\mathrm{cr}} = \mathbf{R}_0$.
    $m_{\mathrm{cr}} = s$, $d = s - s_0$
    Find basis $\gamma_1, \ldots, \gamma_d$ of $\ker(\boldsymbol{\eta}_0)$.
  else /* *Now* $\mathbf{X} = \mathbf{X}_n^{cr}$. */
    $d = \dim(\ker(\mathbf{t}_2^{\mathrm{cr}}))$
    Find basis $\gamma_1, \ldots, \gamma_d$ of $\ker(\mathbf{t}_2^{\mathrm{cr}})$.
  end if
  for $i = 1, \ldots, d$ **do**
    Determine j_i such that $|\gamma_{i,j_i}| = \max_{1 \le k \le m_{\mathrm{cr}}} |\gamma_{i,k}|$.
    for $k = i+1, \ldots, d$ **do**
      $g_l = \gamma_{k,l} - \gamma_{i,l}\gamma_{k,j_i}/\gamma_{i,j_i}$ for $l = 1, \ldots, j_i - 1, j_i + 1, \ldots, m_{\mathrm{cr}}$
      $\gamma_k = [\, g_1, \ldots, g_{j_i-1}, g_{j_i+1}, \ldots, g_{m_{\mathrm{cr}}} \,]$
    end for
    Eliminate column j_i from all matrices superscripted $^{\mathrm{cr}}$.
    $m_{\mathrm{cr}} = m_{\mathrm{cr}} - 1$
  end for

the block $\mathbf{X}_n^{\mathrm{cr}}$ is passed in place of $\mathbf{X}$, and in view of (3.12), we have to determine a basis of $\ker(\mathbf{t}_2^{\mathrm{cr}})$. In any case, after termination we have $m_{\mathrm{cr}} \le s_n$.

3.2.1 Block QR decomposition

An important feature of BLQMR is the fact that the QR decomposition appearing in (3.10) can be updated as n increases. In this subsection we present an algorithm based on Householder reflections which was outlined in [19] for the case where the coefficient matrix $\mathbf{A}$ is hermitian.

We first write down the matrix $\underline{\mathbf{T}}_n$ in a form more convenient for our current task. To this end we note that when the right block $\mathbf{y}_j$ (where $1 \le j \le n$) is computed, the coefficients $\boldsymbol{\tau}_{0,j-1}, \ldots, \boldsymbol{\tau}_{j,j-1}$ form columns $\nu(j-1)+1, \ldots, \nu(j)$ of $\underline{\mathbf{T}}_n$. So in the first of these columns the elements $1, \ldots, \phi(\nu(j-1)+1)$ are zero according to Proposition 8. Hence element $\phi(\nu(j-1)+1)+1$ in column $\nu(j-1)+1$ is the first to lie within the nonzero pattern of $\underline{\mathbf{T}}_n$ as shown in Figure 4, and if we define $\mathsf{m}(p) :\equiv n(\phi(\nu(p)+1)+1)-1$ for $0 \le p \le n_t - 2$ (since $\underline{\mathbf{T}}_{n_t-1}$ has $n_t - 1$ block columns), we obtain $\boldsymbol{\tau}_{i,j-1} = \mathbf{0}$ for $i = 0, \ldots, \mathsf{m}(j-1) - 1$. Now we set $n_0 :\equiv \min\{n \mid \mathsf{m}(n) > 0\}$, so that we have

$$\underline{\mathbf{T}}_n = \begin{bmatrix} \boldsymbol{\tau}_{0,0} & \cdots & \boldsymbol{\tau}_{0,n_0-1} & \mathbf{0}_{\nu(\mathsf{m}(n_0))\times s_{n_0}} & \cdots & \mathbf{0}_{\nu(\mathsf{m}(n-1))\times s_{n-1}} \\ \boldsymbol{\tau}_{1,0} & & \vdots & \boldsymbol{\tau}_{\mathsf{m}(n_0),n_0} & & \boldsymbol{\tau}_{\mathsf{m}(n-1),n-1} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & \boldsymbol{\tau}_{n_0,n_0-1} & \vdots & & \vdots \\ & & & \boldsymbol{\tau}_{n_0+1,n_0} & & \boldsymbol{\tau}_{n-1,n-1} \\ & & & & & \boldsymbol{\tau}_{n,n-1} \end{bmatrix}$$

Our goal is to determine the unitary matrix $\mathbf{Q}_{n+1}$ in its factored form. With $\mathbf{Q}_1 :\equiv \mathbb{1}_{s_0}$ we introduce the recurrence relation

$$\mathbf{Q}_{n+1} :\equiv \begin{bmatrix} \mathbf{Q}_n & \mathbf{0}_{\nu(n)\times s_n} \\ \mathbf{0}_{s_n\times\nu(n)} & \mathbb{1}_{s_n} \end{bmatrix} \mathbf{U}_n$$

for $n = 1, 2, \ldots$, where

$$\mathbf{U}_n :\equiv \begin{bmatrix} \mathbb{1}_{\nu(n-1)} & \mathbf{0}_{\nu(n-1)\times s_{n-1}+s_n} \\ \mathbf{0}_{s_{n-1}+s_n\times\nu(n-1)} & \mathbf{U}'_n \end{bmatrix} = \text{block diag}(\mathbb{1}_{\nu(n-1)}, \mathbf{U}'_n).$$

Here $\mathbf{U}'_n$ is a unitary matrix of size $s_{n-1} + s_n$. Consequently we can write

$$\mathbf{Q}_{n+1} = \begin{bmatrix} \mathbf{Q}_n(:, 1{:}\nu(n-1)) & \mathbf{Q}_n(:, \nu(n-1)+1{:}\nu(n))\mathbf{U}'_n(1{:}s_{n-1}, :) \\ \mathbf{0}_{s_n\times\nu(n-1)} & \mathbf{U}'_n(s_{n-1}+1{:}s_{n-1}+s_n, :) \end{bmatrix}. \tag{3.13}$$

Assume for the moment that we are given a sequence of such unitary transformations $\mathbf{U}'_1, \ldots, \mathbf{U}'_n$. Then $\mathbf{Q}_{n+1}$ can be computed by the scheme shown in Algorithm 16. Now the effect of $\mathbf{Q}_n^\star$ is to

Algorithm 16 Computation of $\mathbf{Q}_{n+1}$

Input: Unitary matrices $\mathbf{U}'_1, \ldots, \mathbf{U}'_n$, where $\mathbf{U}'_i$ is of order $s_{i-1} + s_i$.

$\mathbf{Q}_{n+1} = \mathbb{1}_{\nu(n+1)}$

for $i = 1, \ldots, n$ **do**

$\mathbf{Q}_{n+1}(1{:}\nu(i), \nu(i-1)+1{:}\nu(i+1)) = \mathbf{Q}_{n+1}(1{:}\nu(i), \nu(i-1)+1{:}\nu(i))\mathbf{U}'_i(1{:}s_{i-1}, :)$

$\mathbf{Q}_{n+1}(\nu(i)+1{:}\nu(i+1), \nu(i-1)+1{:}\nu(i+1)) = \mathbf{U}'_i(s_{i-1}+1{:}s_{i-1}+s_i, :)$

end for

annihilate all subdiagonal elements of $\underline{\mathbf{T}}_n$ except those in the last s_{n-1} columns:

$$\text{block diag}(\mathbf{Q}_n^\star, \mathbb{1}_{s_n})\underline{\mathbf{T}}_n = \begin{bmatrix} \boldsymbol{\theta}_{0,0} & \cdots & \boldsymbol{\theta}_{0,n_0-1} & \mathbf{0}_{\nu(\mathsf{m}(n_0)-1)\times s_{n_0}} & \cdots & \mathbf{0}_{\nu(\mathsf{m}(n-2)-1)\times s_{n-2}} & \mathbf{0}_{\nu(\mathsf{m}(n-1)-1)\times s_{n-1}} \\ & \ddots & \vdots & \boldsymbol{\theta}_{\mathsf{m}(n_0)-1,n_0} & & \boldsymbol{\theta}_{\mathsf{m}(n-2)-1,n-2} & \boldsymbol{\theta}_{\mathsf{m}(n-1)-1,n-1} \\ & & \boldsymbol{\theta}_{n_0-1,n_0-1} & \vdots & & \vdots & \vdots \\ & & & \boldsymbol{\theta}_{n_0,n_0} & & \vdots & \vdots \\ & & & & & \boldsymbol{\theta}_{n-2,n-2} & \boldsymbol{\theta}_{n-2,n-1} \\ & & & & & & \boldsymbol{\tau}'_{n-1,n-1} \\ & & & & & & \boldsymbol{\tau}_{n,n-1} \end{bmatrix}$$

Here $\boldsymbol{\theta}_{i,j} \in \mathbb{C}^{s_i\times s_j}$, the blocks $\boldsymbol{\theta}_{i,i}$ are upper triangular and some (or even all) of the zero matrices in the first row may be empty. The last block column of this matrix is obtained as follows:

$$\begin{bmatrix} \boldsymbol{\theta}_{\mathsf{m}(n-1)-1,n-1} \\ \vdots \\ \boldsymbol{\theta}_{n-2,n-1} \\ \boldsymbol{\tau}'_{n-1,n-1} \end{bmatrix} = \text{block diag}(\mathbb{1}_{s_{\mathsf{m}(n-1)-1}+s_{\mathsf{m}(n-1)}+\cdots+s_{n-3}}, \mathbf{U}'^{\star}_{n-1})$$

$$\text{block diag}(\mathbb{1}_{s_{\mathsf{m}(n-1)-1}+\cdots+s_{n-4}}, \mathbf{U}'^{\star}_{n-2}, \mathbb{1}_{s_{n-1}}) \ldots$$

$$\text{block diag}(\mathbf{U}'^{\star}_{\mathsf{m}(n-1)}, \mathbb{1}_{s_{\mathsf{m}(n-1)+1}+\cdots+s_{n-1}}) \begin{bmatrix} \mathbf{0}_{s_{\mathsf{m}(n-1)-1}\times s_{n-1}} \\ \boldsymbol{\tau}_{\mathsf{m}(n-1),n-1} \\ \vdots \\ \boldsymbol{\tau}_{n-1,n-1} \end{bmatrix}.$$

Now we have to construct $\mathbf{U}'_n$ such that

$$\begin{bmatrix} \boldsymbol{\tau}'_{n-1,n-1} \\ \boldsymbol{\tau}_{n,n-1} \end{bmatrix} = \mathbf{U}'_n \begin{bmatrix} \boldsymbol{\theta}_{n-1,n-1} \\ \mathbf{0}_{s_n\times s_{n-1}} \end{bmatrix} \tag{3.14}$$

where $\boldsymbol{\theta}_{n-1,n-1} \in \mathbb{C}^{s_{n-1}\times s_{n-1}}$ is upper triangular. Before describing the construction of $\mathbf{U}'_n$, however, we state the algorithm for updating the QR factorization of $\underline{\mathbf{T}}_n$, see Algorithm 17.

Algorithm 17 Updating the QR factorization of $\underline{\mathbf{T}}_n$

Input: QR decomposition of $\underline{\mathbf{T}}_{n-1}$: $\underline{\mathbf{T}}_{n-1} = \mathbf{Q}_n \underline{\mathbf{R}}_{n-1}^{\mathrm{QR}}$.
Output: QR decomposition of $\underline{\mathbf{T}}_n$: $\underline{\mathbf{T}}_n = \mathbf{Q}_{n+1} \underline{\mathbf{R}}_n^{\mathrm{QR}}$.

```
/* Initializations: */
for i = m(n-1), ..., n-1 do
    θ_{i,n-1} = τ_{i,n-1}
end for
if m(n-1) > 0 then
    θ_{m(n-1)-1,n-1} = 0_{s_{m(n-1)-1} × s_{n-1}}
    first = m(n-1) - 1
else
    first = 0
end if
/* Apply U'_{first}, ..., U'_{n-1} to the last block column of T_n: */
for i = first, ..., n-2 do
    [θ_{i,n-1}; θ_{i+1,n-1}] = U'^⋆_{i+1} [θ_{i,n-1}; θ_{i+1,n-1}]
end for
τ'_{n-1,n-1} = θ_{n-1,n-1}
Compute U'_n and the new θ_{n-1,n-1} according to Algorithm 19.
Compute Q_{n+1} according to Algorithm 16.
r = [0^⊤_{ν(first)×s_{n-1}}  θ^⊤_{m(n-1)-1,n-1}  ...  θ^⊤_{n-2,n-1}]^⊤
R_n^QR = [R_{n-1}^QR  r ; 0_{s_{n-1}×ν(n-1)}  θ_{n-1,n-1}]
```

Our next task is to find a recursion for $\mathbf{X}_n^{\mathrm{cr}}$ which is to be used instead of (3.11). First we set

$$\begin{bmatrix} \mathbf{t}_1'^{\mathrm{cr}} \\ \mathbf{t}_2'^{\mathrm{cr}} \end{bmatrix} :\equiv \mathbf{Q}_n^\star \underline{\mathbf{e}}_0 \boldsymbol{\eta}_0^{\mathrm{cr}}, \qquad \mathbf{t}_1'^{\mathrm{cr}} \in \mathbb{C}^{\nu(n-1)\times m_{\mathrm{cr}}}, \quad \mathbf{t}_2'^{\mathrm{cr}} \in \mathbb{C}^{s_{n-1}\times m_{\mathrm{cr}}}. \tag{3.15}$$

Here $\underline{\mathbf{e}}_0 = [\,\mathbb{1}_{s_0}\, \mathbf{0}_{s_0\times\nu(n)-s_0}\,]^\top$. The reader will have noted that the symbol $\underline{\mathbf{e}}_0$ was previously defined to have $\nu(n+1)$ rows rather than $\nu(n)$, but since it is clear from the context how many rows the matrix $\underline{\mathbf{e}}_0$ has, we will use the same symbol for both cases. Now, using relation (3.13), we get

$$\begin{aligned} \mathbf{Q}_{n+1}^\star \underline{\mathbf{e}}_0 \boldsymbol{\eta}_0^{\mathrm{cr}} &= \begin{bmatrix} \mathbf{Q}_n^\star(1{:}\nu(n-1),:) & \mathbf{0}_{\nu(n-1)\times s_n} \\ \mathbf{U}_n'^\star(:,1{:}s_{n-1})\mathbf{Q}_n^\star(\nu(n-1)+1{:}\nu(n),:) & \mathbf{U}_n'^\star(:,s_{n-1}+1{:}s_{n-1}+s_n) \end{bmatrix} \underline{\mathbf{e}}_0 \boldsymbol{\eta}_0^{\mathrm{cr}} \\ &= \begin{bmatrix} \mathbf{Q}_n^\star(1{:}\nu(n-1),1{:}s_0)\boldsymbol{\eta}_0^{\mathrm{cr}} \\ \mathbf{U}_n'^\star(:,1{:}s_{n-1})\mathbf{Q}_n^\star(\nu(n-1)+1{:}\nu(n),1{:}s_0)\boldsymbol{\eta}_0^{\mathrm{cr}} \end{bmatrix}. \end{aligned}$$

The second equality follows from the fact that the first update happens when $n = 2$ so that $\nu(n) > s_0$. Comparing this with (3.15), it follows that

$$\begin{bmatrix} \mathbf{t}_1^{\mathrm{cr}} \\ \mathbf{t}_2^{\mathrm{cr}} \end{bmatrix} = \begin{bmatrix} \mathbf{t}_1'^{\mathrm{cr}} \\ \mathbf{U}_n'^\star(:,1{:}s_{n-1})\mathbf{t}_2'^{\mathrm{cr}} \end{bmatrix},$$

which may also be written as

$$\begin{aligned} \mathbf{t}_1^{\mathrm{cr}} &= \begin{bmatrix} \mathbf{t}_1'^{\mathrm{cr}} \\ \mathbf{U}_n'^\star(1{:}s_{n-1},1{:}s_{n-1})\mathbf{t}_2'^{\mathrm{cr}} \end{bmatrix} \\ \mathbf{t}_2^{\mathrm{cr}} &= \mathbf{U}_n'^\star(s_{n-1}+1{:}s_{n-1}+s_n,1{:}s_{n-1})\mathbf{t}_2'^{\mathrm{cr}} \end{aligned} \tag{3.16}$$

So the first $\nu(n-1)$ components of $\mathbf{t}_1^{\mathrm{cr}}$ are not changed. For $n \geq 1$, we now define $\mathbf{P}_n :\equiv$

$[p_0, \dots, p_{\nu(n)-1}] :\equiv \mathbf{Y}_n (\mathbf{R}_n^{\mathrm{QR}})^{-1}$ so that (3.11) can be rewritten as

$$\begin{aligned}\mathbf{X}_n^{\mathrm{cr}} &= \mathbf{X}_0^{\mathrm{cr}} + \mathbf{P}_n \mathbf{t}_1^{\mathrm{cr}} = \mathbf{X}_0^{\mathrm{cr}} + [\,\mathbf{P}_{n-1}\ \mathbf{p}_{n-1}\,] \begin{bmatrix} \mathbf{t}_1'^{\mathrm{cr}} \\ \mathbf{U}_n'^{\star}(1{:}s_{n-1}, 1{:}s_{n-1}) \mathbf{t}_2'^{\mathrm{cr}} \end{bmatrix} \\ &= \mathbf{X}_{n-1}^{\mathrm{cr}} + \mathbf{p}_{n-1} \mathbf{U}_n'^{\star}(1{:}s_{n-1}, 1{:}s_{n-1}) \mathbf{t}_2'^{\mathrm{cr}}. \end{aligned} \tag{3.17}$$

Here by $\mathbf{p}_{n-1}$ we denote the last s_{n-1} columns of $\mathbf{P}_n$. In order to perform this update, we need first need to compute $\mathbf{p}_{n-1}$. Considering the last block column of the equation $\mathbf{P}_n \mathbf{R}_n^{\mathrm{QR}} = \mathbf{Y}_n$ we get

$$\mathbf{P}_n \begin{bmatrix} \mathbf{0}_{\nu(\mathsf{m}(n-1)-1)\times s_{n-1}} \\ \boldsymbol{\theta}_{\mathsf{m}(n-1)-1,n-1} \\ \vdots \\ \boldsymbol{\theta}_{n-2,n-1} \\ \boldsymbol{\theta}_{n-1,n-1} \end{bmatrix} = \mathbf{y}_{n-1}$$

and therefore

$$\mathbf{p}_{n-1} = \left(\mathbf{y}_{n-1} - \sum_{j=\max\{\mathsf{m}(n-1)-1,0\}}^{n-2} \mathbf{p}_j \boldsymbol{\theta}_{j,n-1} \right) \boldsymbol{\theta}_{n-1,n-1}^{-1}. \tag{3.18}$$

This recurrence is short in the sense that the blocks $\mathbf{p}_0, \dots, \mathbf{p}_{\mathsf{m}(n-1)-2}$ are not needed.

Now we insert some remarks about complex Householder transformations [10, 19]. Let $y = (\xi_1, \dots, \xi_n)^\top$ be a complex vector. We look for a unitary matrix $\mathbf{U} \in \mathbb{C}^{n\times n}$ such that $\mathbf{U}y = \alpha e_1$ where $\alpha \in \mathbb{C}$. If $\xi_2 = \cdots = \xi_n = 0$ we set $\mathbf{U} = \mathbb{1}_n$ and are done. Otherwise, as $\mathbf{U}$ is unitary, it follows that $|\alpha| = \|y\|_2$. For $v \in \mathbb{C}^n$ the matrix

$$\mathbf{H}_v = \mathbb{1}_n - 2\frac{vv^\star}{\langle v, v\rangle} = \mathbb{1}_n + \beta vv^\star$$

where $\langle v, w\rangle :\equiv v^\star w$ and $\beta = -2/\langle v, v\rangle \in \mathbb{R}$ is called a *Householder reflection*. The matrix $\mathbf{H}_v$ describes a reflection on the hyperplane of $\mathbb{C}^n$ orthogonal to v. We note that $\mathbf{H}_v$ is hermitian and unitary, i. e. $\mathbf{H}_v^\star = \mathbf{H}_v$ and $\mathbf{H}_v\mathbf{H}_v^\star = \mathbb{1}_n$. Application of $\mathbf{H}_v$ to a vector y gives $\mathbf{H}_v y = y + \beta\langle v, y\rangle v$. According to our goal, given y we want to find a $v \in \mathbb{C}^n$ such that $\mathbf{H}_v y = \alpha e_1$. This yields

$$y - \alpha e_1 = -\beta\langle v, y\rangle v \tag{3.19}$$

so in particular $v \in \operatorname{span}\{y - \alpha e_1\}$. Since $\mathbf{H}_v = \mathbf{H}_{\lambda v}$ for every $\lambda \in \mathbb{C}\setminus\{0\}$, we may choose $v = y - \alpha e_1$. Now

$$\langle v, y\rangle = \langle y - \alpha e_1, y\rangle = \|y\|_2^2 - \overline{\alpha}\xi_1.$$

On the other hand, (3.19) becomes $v = -\beta\langle v, y\rangle v$, which gives

$$\langle v, y\rangle = -\beta^{-1} \in \mathbb{R}.$$

Let $\xi_1 = |\xi_1| e^{i\theta}$ and $\alpha = \|y\|_2 e^{i\theta_\alpha}$. Then

$$\langle v, y\rangle = \|y\|_2^2 - \|y\|_2 |\xi_1| e^{i(\theta - \theta_\alpha)}$$

Since $\langle v, y\rangle$ must be real, we are left with the two following choices:

- $\alpha = -\|y\|_2 e^{i\theta}$, then the first component of v and β are given by

$$\begin{aligned} \xi_1 - \alpha &= \big(|\xi_1| + \|y\|_2\big)\, e^{i\theta} \\ \beta = -\langle v, y\rangle^{-1} &= \frac{-1}{\|y\|_2\big(\|y\|_2 + |\xi_1|\big)}. \end{aligned}$$

- $\alpha = +\|y\|_2 e^{i\theta}$, now the first component of v is

$$\xi_1 - \alpha = \big(|\xi_1| - \|y\|_2\big)\, e^{i\theta} = \frac{|\xi_1|^2 - \|y\|_2^2}{|\xi_1| + \|y\|_2}\, e^{i\theta} = \frac{-\big(|\xi_2|^2 + \cdots + |\xi_n|^2\big)}{|\xi_1| + \|y\|_2}\, e^{i\theta} \tag{3.20}$$

  where the last expression prevents cancellation. The formula for β is now

$$\beta = -\langle v, y\rangle^{-1} = \frac{-1}{\|y\|_2\big(\|y\|_2 - |\xi_1|\big)} = \frac{-\big(|\xi_1| + \|y\|_2\big)}{\|y\|_2\big(|\xi_2|^2 + \cdots + |\xi_n|^2\big)}. \tag{3.21}$$

The second choice has the property that if α is real, then it is also positive, and so we adopt it in Algorithm 18, which is similar to [10, Algorithm 5.1.1]. We note that in line 18.2, a scaling should first be applied in order to prevent a possible overflow. In line 18.18 we ensure that the first

Algorithm 18 $[v, \beta] = \mathbf{house}(y)$

Input: Vector $y \in \mathbb{C}^n$

Output: Vector $v \in \mathbb{C}^n$ with $v(1) = 1$ and $\beta \in \mathbb{R}$ such that $\mathbf{H}_v y = \alpha e_1$.

```
 1: n = length(y)
 2: σ = y(2:n)^⋆ y(2:n)
 3: v = [1; y(2:n)]
 4: if σ = 0 then
 5:     /* Nothing to do, so set H_v to the identity: */
 6:     β = 0
 7: else
 8:     μ = sqrt(|y(1)|^2 + σ).     /* μ = ||y||_2 */
 9:     if y(1) ≠ 0 then
10:         θ = arg(y(1)).
11:     else
12:         θ = 0.
13:     end if
14:     /* Now compute v(1) according to (3.20). */
15:     v(1) = (−σ)/(|y(1)| + μ) e^{iθ}
16:     /* Compute β via (3.21). */
17:     β = −(|y(1)| + μ)|v(1)|^2/(μσ)
18:     v = v/v(1)
19: end if
```

component of v is one, which allows us to replace the zeroed elements of a matrix by the essential part of v.

The next step is to outline the computation of $\mathbf{U}'_n$ on the basis of Algorithm 18. As an example, assume that $s_{n-1} = 4$ and $s_n = 3$. According to (3.14), we consider the matrix

$$\begin{bmatrix} \boldsymbol{\tau}'_{n-1,n-1} \\ \boldsymbol{\tau}_{n,n-1} \end{bmatrix} = \left(\begin{array}{cccc} \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \\ \hline \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \end{array}\right).$$

Now we have to find Householder matrices $\mathbf{H}_{1,n}, \ldots, \mathbf{H}_{s_{n-1},n}$ with the property that

$$\begin{bmatrix} \boldsymbol{\theta}_{n-1,n-1} \\ \mathbf{0}_{s_n \times s_{n-1}} \end{bmatrix} = \mathbf{H}_{s_{n-1},n} \ldots \mathbf{H}_{1,n} \begin{bmatrix} \boldsymbol{\tau}'_{n-1,n-1} \\ \boldsymbol{\tau}_{n,n-1} \end{bmatrix} = \mathbf{U}'^{\star}_n \begin{bmatrix} \boldsymbol{\tau}'_{n-1,n-1} \\ \boldsymbol{\tau}_{n,n-1} \end{bmatrix},$$

in particular $\mathbf{U}'_n = \mathbf{H}_{1,n} \dots \mathbf{H}_{s_{n-1},n}$. In the above example, assume that the matrices $\mathbf{H}_{1,n}$ und $\mathbf{H}_{2,n}$ have been determined so that

$$\mathbf{H}_{2,n}\mathbf{H}_{1,n} \begin{bmatrix} \boldsymbol{\tau}'_{n-1,n-1} \\ \boldsymbol{\tau}_{n,n-1} \end{bmatrix} = \left(\begin{array}{cccc} \circ & \circ & \circ & \circ \\ & \circ & \circ & \circ \\ & & \bullet & \circ \\ & & \bullet & \circ \\ \hline & & \bullet & \circ \\ & & \bullet & \circ \\ & & \bullet & \circ \end{array} \right).$$

The highlighted vector now determines the next Householder matrix $\mathbf{H}_{3,n} = \text{block diag}(\mathbb{1}_2, \mathbf{H}'_{3,n})$. To get $\mathbf{H}'_{3,n}$ we run Algorithm 18 with the highlighted vector as input. The resulting algorithm is formulated as Algorithm 19. After termination, in the preceding example, the matrix M has the structure shown in the following Figure, where the bullets mark the matrix $\boldsymbol{\theta}_{n-1,n-1}$. Additionally, the stars represent the essential part of the Householder vector of $\mathbf{H}'_{3,n}$.

$$\begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \circ & \bullet & \bullet & \bullet \\ \circ & \circ & \bullet & \bullet \\ \circ & \circ & \star & \bullet \\ \circ & \circ & \star & \circ \\ \circ & \circ & \star & \circ \\ \circ & \circ & \star & \circ \end{pmatrix}$$

Algorithm 19 Computation of $\mathbf{U}'_n$

Input: Matrices $\boldsymbol{\tau}'_{n-1,n-1}$ and $\boldsymbol{\tau}_{n,n-1}$

Output: Matrix containing $\boldsymbol{\theta}_{n-1,n-1}$ in the upper triangle and the essential parts of the Householder vectors of $\mathbf{H}_{1,n}, \dots, \mathbf{H}_{s_{n-1},n}$ in the (strictly) lower triangle.

```
/* Initialization: */
M = [ τ'_{n-1,n-1} ]
    [ τ_{n,n-1}    ]
for i = 1, ..., s_{n-1} do
    [β, v] = house(M(i:s_{n-1} + s_n, i))
    /* Now H'_{i,n} = 1_{s_{n-1}+s_n-i+1} + β v v^*. */
    M(i:s_{n-1} + s_n, i:s_{n-1}) = M(i:s_{n-1} + s_n, i:s_{n-1}) + β v v^* M(i:s_{n-1} + s_n, i:s_{n-1})
    /* Save essential part of v in the lower triangle of M: */
    M(i + 1:s_{n-1} + s_n, i) = v(2:s_{n-1} + s_n + i + 1)
end for
```

3.2.2 Statement of BLQMR

In the preceding subsections, all the low level operations involved in the BLQMR algorithm have been described so that we now turn to a high level description of BLQMR, see Algorithm 20, thereby concentrating on points not already covered in the previous subsections. First, we need a variable to store the index of the matrix $\underline{\mathbf{T}}_n$ which has been QR decomposed; we call it n_{qr}. Since $\underline{\mathbf{T}}_1$ is the first matrix which is decomposed, n_{qr} is initialized to zero. If $n_{\text{qr}} > 0$, then the QR factorization of $\underline{\mathbf{T}}_{n_{\text{qr}}}$ has been computed.

When new clusters have been built, we check whether this has lead to new right blocks being available. At line 20.22, the number of the last available right block is computed (recall that at this point, k_l is the last final regular index, so if $n_{\max} > 0$, then $\mathbf{y}_{n_{\max}-1}$ is the last right block which will not change any more). If new blocks turn out to be available, the **for** loop on line 20.23 is executed and performs one or several updates of the solution block $\mathbf{X}_i^{\text{cr}}$. Then the convergence is checked,

which means that we either estimate or compute the maximal relative unpreconditioned residual norm. If the convergence criterion is satisfied, it is necessary to restore the eliminated solutions according to (3.5). Finally, at line 20.29, we check whether some columns of $\mathbf{X}_i^{\mathrm{cr}}$ can be eliminated.

Algorithm 20 BLQMR

Input: Matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$, block $\mathbf{B}' \in \mathbb{C}^{N\times s'}$ of right-hand sides

1: Check $\mathbf{B}'$ for deflation $\rightarrow \mathbf{B}$, s.
2: Choose block $\mathbf{X}_0 \in \mathbb{C}^{N\times s}$ of initial guesses and compute initial residuals $\mathbf{R}_0 = \mathbf{B} - \mathbf{A}\mathbf{X}_0$.
3: Check $\mathbf{R}_0$ for deflation $\rightarrow \mathbf{y}_0$, $\nu(1)$, s_0.
4: Choose block $\widetilde{\mathbf{y}} \in \mathbb{C}^{N\times s}$ of left Lanczos vectors and check it for deflation $\rightarrow \widetilde{\mathbf{y}}_0$, $\widetilde{\nu}(1)$, $\widetilde{s}_0$.
5: Execute Algorithm 15 with $n = 0$ and $\mathbf{X} = \mathbf{X}_0$.
6: */* Initializations, including: */*
7: $l = 0$, $n = 1$, $\widetilde{n} = 1$, $k_0 = 0$, $n_{\mathrm{qr}} = 0$.
8: LA1 Fix initial choice for k_1 under the condition $1 \le k_1 \le \nu(1)$.
9: **loop**
10: */* 1) Compute enough $\widetilde{y}$ vectors: */*
11: Extend left block Krylov space according to Algorithm 7.
12: */* 2) Test whether current value of k_{l+1} leads to a nonsingular $\boldsymbol{\delta}_l$: */*
13: Set $\mathbf{z}_l = [\, y_{k_l}, \ldots, y_{k_{l+1}-1} \,]$, $\widetilde{\mathbf{z}}_l = [\, \widetilde{y}_{k_l}, \ldots, \widetilde{y}_{k_{l+1}-1} \,]$ and $r_l = k_{l+1} - k_l$.
14: Compute $\boldsymbol{\delta}_l = \widetilde{\mathbf{z}}_l^{\star}\mathbf{z}_l$ and $\Delta_l = r_l - \mathrm{rank}(\boldsymbol{\delta}_l)$.
15: Execute lines 14 to 25 of Algorithm 6.
16: **if** $\Delta_l > 0$ **then**
17: LA2 Increase k_{l+1}.
18: **else**
19: Orthogonalize $[\, y_{k_{l+1}}, \ldots, y_{\nu(n)-1} \,]$ against $\widetilde{\mathbf{z}}_l$ and $[\, \widetilde{y}_{k_{l+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1} \,]$ against $\mathbf{z}_l$. Renormalize the orthogonalized vectors and adapt $\underline{\mathbf{T}}_n$ and $\underline{\widetilde{\mathbf{T}}}_{\widetilde{n}}$.
20: LA3 Fix new initial choice for k_{l+2}.
21: Set $l = l + 1$.
22: Compute $n_{\max} = \max\{n \mid \nu(n) \le k_l\}$.
23: **for** $i = n_{\mathrm{qr}} + 1, \ldots, n_{\max}$ **do**
24: Obtain QR decomposition of $\underline{\mathbf{T}}_i$ according to Algorithm 17.
25: Compute $\mathbf{p}_{i-1}$ via (3.18).
26: $\mathbf{X}_i^{\mathrm{cr}} = \mathbf{X}_{i-1}^{\mathrm{cr}} + \mathbf{p}_{i-1}\mathbf{U}_i'^{\star}(1{:}s_{i-1}, 1{:}s_{i-1})\mathbf{t}_2^{\mathrm{cr}}$
27: $\mathbf{t}_2^{\mathrm{cr}} = \mathbf{U}_i'^{\star}(s_{i-1}+1{:}s_{i-1}+s_i, 1{:}s_{i-1})\mathbf{t}_2^{\mathrm{cr}}$
28: Check convergence
29: Execute Algorithm 15 with $n = i$ and $\mathbf{X} = \mathbf{X}_i^{\mathrm{cr}}$.
30: **end for**
31: $n_{\mathrm{qr}} = n_{\max}$
32: **end if**
33: */* 3) Compute enough y vectors: */*
34: LA4 Fix minimum number of right vectors $\rightarrow$ minnum.
35: Extend right block Krylov space according to Algorithm 8.
36: **end loop**

3.3 BLBICG

In this part, we construct a version of BLBICG based on the variants of the block Lanczos process developed in the first section of this work. In contrast to the algorithms presented in [17, 20], deflation of right or left Lanczos vectors does not lead to a restart of the process, and we also have a procedure for overcoming (near) breakdowns at our disposal. As outlined in [13] for the nonblock case, there are also several variants of the BLBICG method of which we consider BLBIORES and

BlBiOMin. We first give some remarks which apply to both of them.

As in the nonblock case, BlBiCG is based on a Galerkin condition. Let $\widetilde{\mathcal{Z}}_l :\equiv \operatorname{span}\{\widetilde{y}_0, \ldots, \widetilde{y}_{k_l-1}\}$ and assume that there is an index n such that $k_l = \nu(n)$. If

$$\mathbb{C}^N = \mathbf{A}\mathcal{B}_n \oplus \widetilde{\mathcal{Z}}_l^{\perp} \tag{3.22}$$

we may conclude by [4, Lemma 2.2] that the oblique projection onto $\mathbf{A}\mathcal{B}_n$ along $\widetilde{\mathcal{Z}}_l^{\perp}$ exists. So in this case there is a unique block $\mathbf{k}_n^{\mathrm{cr}} \in \mathbb{C}^{\nu(n)\times s_n}$ such that $\mathbf{R}_n^{\mathrm{cr}} = \mathbf{R}_0^{\mathrm{cr}} - \mathbf{A}\mathbf{Y}_n\mathbf{k}_n^{\mathrm{cr}}$ satisfies the Galerkin condition $\mathbf{R}_n^{\mathrm{cr}} \perp \widetilde{\mathcal{Z}}_l$. It follows that $r_n^{(i)} \in \mathcal{B}_{n+1} \ominus \mathcal{B}_n$ for $i \in J_n$ and by (3.8)

$$\begin{aligned}\mathbf{0} \overset{!}{=} \widetilde{\mathbf{Z}}_l^{\star}\mathbf{R}_n^{\mathrm{cr}} &= \widetilde{\mathbf{Z}}_l^{\star}\mathbf{Y}_{n+1}(\underline{\mathbf{e}}_0\boldsymbol{\eta}_0^{\mathrm{cr}} - \underline{\mathbf{T}}_n\mathbf{k}_n^{\mathrm{cr}}) \\ &= \left[\begin{array}{ccc|c} \boldsymbol{\delta}_0 & & & \\ & \ddots & & \mathbf{0}_{k_l\times s_n} \\ & & \boldsymbol{\delta}_{l-1} & \end{array}\right] (\underline{\mathbf{e}}_0\boldsymbol{\eta}_0^{\mathrm{cr}} - \underline{\mathbf{T}}_n\mathbf{k}_n^{\mathrm{cr}}) = \mathbf{e}_0\boldsymbol{\eta}_0^{\mathrm{cr}} - \mathbf{T}_n\mathbf{k}_n^{\mathrm{cr}},\end{aligned}$$

where $\mathbf{T}_n$ denotes the first $\nu(n)$ rows of $\underline{\mathbf{T}}_n$. As $\mathbf{k}_n^{\mathrm{cr}}$ is uniquely determined by the above Galerkin condition, the system

$$\mathbf{T}_n\mathbf{k}_n^{\mathrm{cr}} = \mathbf{e}_0\boldsymbol{\eta}_0^{\mathrm{cr}} \tag{3.23}$$

must have a unique solution and hence $\mathbf{T}_n$ is nonsingular. If, on the other hand, condition (3.22) is not satisfied, then $\mathbf{T}_n$ may be singular and it may happen that no approximate solution $\mathbf{X}_n$ exists at this step.

The solution of (3.23), if it exists, is done by combining the computation of the LU decomposition of $\mathbf{T}_n$ with updating the blocks of the solution vectors and the residuals. First, this process is described for BlBiORes.

3.3.1 BlBiORes

Since BlBiORes is based on BlBiO, we need a procedure for updating the LU decomposition of $\mathbf{T}_n$. By making use of the structure of $\mathbf{T}_n$, we are lead to a slight modification of the standard block LU decomposition [10, Section 3.2.10] which incorporates pivoting within the block rows of $\mathbf{T}_n$. This makes sure that the length of the recursions for updating the blocks of iterates and residuals is not increased by pivoting. We first describe the details of this process and then its limitations.

The first LU decomposition is done when $\mathbf{T}_1 = \boldsymbol{\tau}_{0,0}$ is available, and we use pivoting here; therefore we have $\boldsymbol{\pi}_0\boldsymbol{\tau}_{0,0} = \mathbf{L}_1\mathbf{U}_1$ with a permutation matrix $\boldsymbol{\pi}_0$ of order s_0. Next we assume that we already have computed an LU decomposition of $\mathbf{T}_n$, namely $\mathbf{P}_n\mathbf{T}_n = \mathbf{L}_n\mathbf{U}_n$. We split $\mathbf{T}_{n+1}$ as follows

$$\mathbf{T}_{n+1} = \begin{bmatrix}\mathbf{T}_n & \mathbf{t}_1 \\ \mathbf{t}_2 & \boldsymbol{\tau}_{n,n}\end{bmatrix}$$

where

$$\mathbf{t}_1 = \begin{bmatrix}\mathbf{0}_{\nu(\mathsf{m}(n))\times s_n} \\ \boldsymbol{\tau}_{\mathsf{m}(n),n} \\ \vdots \\ \boldsymbol{\tau}_{n-1,n}\end{bmatrix}, \qquad \mathbf{t}_2 = \begin{bmatrix}\mathbf{0}_{s_n\times\nu(n-1)} & \boldsymbol{\tau}_{n,n-1}\end{bmatrix}.$$

According to our requirements about pivoting, we have the following ansatz for the permutation matrix $\mathbf{P}_{n+1}$

$$\mathbf{P}_{n+1} = \begin{bmatrix}\mathbf{P}_n & \mathbf{0}_{\nu(n)\times s_n} \\ \mathbf{0}_{s_n\times\nu(n)} & \boldsymbol{\pi}_n\end{bmatrix} = \text{block diag}(\boldsymbol{\pi}_0, \ldots, \boldsymbol{\pi}_n)$$

with permutation matrices $\boldsymbol{\pi}_i \in \mathbb{R}^{s_i\times s_i}$. By partitioning $\mathbf{L}_{n+1}$ and $\mathbf{U}_{n+1}$ in the same way, we get

$$\begin{aligned}\mathbf{T}_{n+1} = \mathbf{P}_{n+1}^{\star}\mathbf{L}_{n+1}\mathbf{U}_{n+1} &= \begin{bmatrix}\mathbf{P}_n^{\star} & \mathbf{0}_{\nu(n)\times s_n} \\ \mathbf{0}_{s_n\times\nu(n)} & \boldsymbol{\pi}_n^{\star}\end{bmatrix}\begin{bmatrix}\mathbf{L}_n & \mathbf{0}_{\nu(n)\times s_n} \\ \mathbf{L}_{21} & \mathbf{L}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{U}_n & \mathbf{U}_{12} \\ \mathbf{0}_{s_n\times\nu(n)} & \mathbf{U}_{22}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{P}_n^{\star}\mathbf{L}_n\mathbf{U}_n & \mathbf{P}_n^{\star}\mathbf{L}_n\mathbf{U}_{12} \\ \boldsymbol{\pi}_n^{\star}\mathbf{L}_{21}\mathbf{U}_n & \boldsymbol{\pi}_n^{\star}\mathbf{L}_{21}\mathbf{U}_{12} + \boldsymbol{\pi}_n^{\star}\mathbf{L}_{22}\mathbf{U}_{22}\end{bmatrix}.\end{aligned}$$

So we obtain $\mathbf{P}_n^\star \mathbf{L}_n \mathbf{U}_{12} = \mathbf{t}_1$ and need to solve the triangular system

$$\mathbf{L}_n \mathbf{U}_{12} = \begin{bmatrix} \mathbf{0}_{\nu(\mathsf{m}(n)) \times s_n} \\ \boldsymbol{\pi}_{\mathsf{m}(n)} \boldsymbol{\tau}_{\mathsf{m}(n),n} \\ \vdots \\ \boldsymbol{\pi}_{n-1} \boldsymbol{\tau}_{n-1,n} \end{bmatrix} \tag{3.24}$$

for $\mathbf{U}_{12}$. The next step is to solve $\boldsymbol{\pi}_n^\star \mathbf{L}_{21} \mathbf{U}_n = \mathbf{t}_2$ for $\boldsymbol{\pi}_n^\star \mathbf{L}_{21}$ (the permutation matrix $\boldsymbol{\pi}_n$ is not known at this point). The lower right part yields $\boldsymbol{\pi}_n^\star \mathbf{L}_{22} \mathbf{U}_{22} = \boldsymbol{\tau}_{n,n} - \boldsymbol{\pi}_n^\star \mathbf{L}_{21} \mathbf{U}_{12}$, i. e. we have to find an LU decomposition (with pivoting) of the matrix on the right-hand side. Finally, $\mathbf{L}_{21}$ is computed.

The LU decomposition with this sort of pivoting works as long as $\mathbf{T}_n$ is nonsingular. If $\mathbf{T}_n$ is singular, we get a singular matrix $\mathbf{U}_n$, and then every $\mathbf{U}_{n'}$ with $n' > n$ computed by the above scheme is also singular. So if m is the smallest index greater than n such that $\mathbf{T}_m$ is nonsingular, then the above scheme fails to produce an LU decomposition of $\mathbf{T}_m$. Consequently, BlBiORes breaks down as soon as a singular $\mathbf{T}_n$ occurs.

Now we turn to the derivation of the recursions for the iterate and residual blocks. If we define $\mathbf{k}'_n :\equiv \mathbf{U}_n \mathbf{k}_n^{\mathrm{cr}}$, then (3.23) becomes $\mathbf{L}_n \mathbf{k}'_n = \mathbf{P}_n \mathbf{e}_0 \boldsymbol{\eta}_0^{\mathrm{cr}}$. Since $\mathbf{k}'_n$ may be partitioned as $\mathbf{k}'_n = \begin{bmatrix} \mathbf{k}'_{n-1} \\ \mathbf{k}'_{n-1,n} \end{bmatrix}$, where $\mathbf{k}'_{n-1,n}$ is of size $s_{n-1} \times s_n$, we get from $\mathbf{L}_n \mathbf{k}'_n = \mathbf{P}_n \mathbf{e}_0 \boldsymbol{\eta}_0^{\mathrm{cr}}$

$$\begin{aligned} \mathbf{k}'_1 &= \mathbf{L}_1^{-1} \mathbf{P}_1 \mathbf{e}_0 \boldsymbol{\eta}_0^{\mathrm{cr}}, \\ \mathbf{k}'_{n-1,n} &= -\mathbf{l}_{n-1,n-1}^{-1} \mathbf{l}_{n-1,n-2} \mathbf{k}'_{n-2,n-1} \quad \text{for } n \geq 2. \end{aligned} \tag{3.25}$$

Here, $\mathbf{k}'_{n-2,n-1}$ denotes the last s_{n-2} rows of $\mathbf{k}'_{n-1}$. Next, we have to derive the recursion for the block $\mathbf{X}_n^{\mathrm{cr}}$ of iterates:

$$\begin{aligned} \mathbf{X}_n^{\mathrm{cr}} &= \mathbf{X}_0^{\mathrm{cr}} + \mathbf{Y}_n \mathbf{k}_n^{\mathrm{cr}} = \mathbf{X}_0^{\mathrm{cr}} + \mathbf{Y}_n \mathbf{T}_n^{-1} \mathbf{e}_0 \boldsymbol{\eta}_0^{\mathrm{cr}} = \mathbf{X}_0^{\mathrm{cr}} + \mathbf{Y}_n \mathbf{U}_n^{-1} \mathbf{k}'_n \\ &= \mathbf{X}_0^{\mathrm{cr}} + \mathbf{V}_n \mathbf{k}'_n \end{aligned} \tag{3.26}$$

with $\mathbf{V}_n :\equiv \mathbf{Y}_n \mathbf{U}_n^{-1}$. If we partition $\mathbf{V}_n$ in the usual way as $\mathbf{V}_n = [\, \mathbf{v}_0, \ldots, \mathbf{v}_{n-1} \,]$, then we obtain

$$\begin{aligned} \mathbf{v}_0 &= \mathbf{y}_0 \mathbf{u}_{0,0}^{-1} \\ \mathbf{v}_n &= \Big(\mathbf{y}_n - \sum_{j=\mathsf{m}(n)}^{n-1} \mathbf{v}_j \mathbf{u}_{j,n} \Big) \mathbf{u}_{n,n}^{-1} \qquad \text{for } n \geq 1, \end{aligned} \tag{3.27}$$

where all the linear systems which have to be solved involve upper triangular matrices. Now, using (3.25), we may write (3.26) as

$$\mathbf{X}_n^{\mathrm{cr}} = \mathbf{X}_{n-1}^{\mathrm{cr}} - \mathbf{v}_{n-1} \mathbf{l}_{n-1,n-1}^{-1} \mathbf{l}_{n-1,n-2} \mathbf{k}'_{n-2,n-1}.$$

By multiplying (3.27) with $\mathbf{A}$, we obtain

$$\begin{aligned} \mathbf{A}\mathbf{v}_0 &= \mathbf{A}\mathbf{y}_0 \mathbf{u}_{0,0}^{-1} \\ \mathbf{A}\mathbf{v}_n &= \Big(\mathbf{A}\mathbf{y}_n - \sum_{j=\mathsf{m}(n)}^{n-1} \mathbf{A}\mathbf{v}_j \mathbf{u}_{j,n} \Big) \mathbf{u}_{n,n}^{-1} \qquad \text{for } n \geq 1, \end{aligned} \tag{3.28}$$

and since the matrix vector products $\mathbf{A}\mathbf{y}_i$ have to be computed anyway, we may use (3.28) to compute $\mathbf{A}\mathbf{v}_i$, which allows us to update the residual block according to

$$\mathbf{R}_n^{\mathrm{cr}} = \mathbf{R}_{n-1}^{\mathrm{cr}} - \mathbf{A}\mathbf{v}_{n-1} \mathbf{l}_{n-1,n-1}^{-1} \mathbf{l}_{n-1,n-2} \mathbf{k}'_{n-2,n-1}.$$

Once the residuals are computed, we need to check whether some columns of $\mathbf{X}_n^{\mathrm{cr}}$ (which at this point has s_{n-1} columns) can be deflated. If $s_n < s_{n-1}$, we have to find $s_{n-1} - s_n$ elements of $\ker(\mathbf{R}_n^{\mathrm{cr}})$, which can be done by computing $s_{n-1} - s_n$ left singular vectors of $\mathbf{R}_n^{\mathrm{cr}}$ corresponding to zero singular values. See Algorithm 21 for a statement of BlBiORes.

Algorithm 21 BLBIORES

Input: Matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$, block $\mathbf{B}' \in \mathbb{C}^{N\times s'}$ of right-hand sides

Check $\mathbf{B}'$ for deflation $\rightarrow \mathbf{B}$, s.
Choose block $\mathbf{X}_0 \in \mathbb{C}^{N\times s}$ of initial guesses and compute initial residuals $\mathbf{R}_0 = \mathbf{B} - \mathbf{A}\mathbf{X}_0$.
Check $\mathbf{R}_0$ for deflation $\rightarrow \mathbf{y}_0$, $\nu(1)$, s_0.
Choose block $\widetilde{\mathbf{y}} \in \mathbb{C}^{N\times s}$ of left Lanczos vectors and check it for deflation $\rightarrow \widetilde{\mathbf{y}}_0$, $\widetilde{\nu}(1)$, $\widetilde{s}_0$.
Execute Algorithm 15 with $n = 0$ and $\mathbf{X} = \mathbf{X}_0$.
/ Initializations, including: */*
$l = 0$, $n = 1$, $\widetilde{n} = 1$, $k_0 = 0$, $n_{\mathrm{lu}} = 0$.
LA1 Fix initial choice for k_1 under the condition $1 \le k_1 \le \nu(1)$.
loop
  / 1) Compute enough $\widetilde{y}$ vectors: */*
  Extend left block Krylov space according to Algorithm 7.
  / 2) Test whether current value of k_{l+1} leads to a nonsingular $\boldsymbol{\delta}_l$: */*
  Set $\mathbf{z}_l = [\, y_{k_l}, \ldots, y_{k_{l+1}-1} \,]$, $\widetilde{\mathbf{z}}_l = [\, \widetilde{y}_{k_l}, \ldots, \widetilde{y}_{k_{l+1}-1} \,]$ and $r_l = k_{l+1} - k_l$.
  Compute $\boldsymbol{\delta}_l = \widetilde{\mathbf{z}}_l^{\star}\mathbf{z}_l$ and $\Delta_l = r_l - \mathrm{rank}(\boldsymbol{\delta}_l)$.
  Execute lines 14 to 25 of Algorithm 6 (BLBIO).
  if $\Delta_l > 0$ **then**
    LA2 Increase k_{l+1}.
  else
    Orthogonalize $[\, y_{k_{l+1}}, \ldots, y_{\nu(n)-1} \,]$ against $\widetilde{\mathbf{z}}_l$ and $[\, \widetilde{y}_{k_{l+1}}, \ldots, \widetilde{y}_{\widetilde{\nu}(\widetilde{n})-1} \,]$ against $\mathbf{z}_l$. Renormalize the orthogonalized vectors and adapt $\underline{\mathbf{T}}_n$ and $\underline{\widetilde{\mathbf{T}}}_{\widetilde{n}}$.
    LA3 Fix new initial choice for k_{l+2}.
    Set $l = l + 1$.
    Execute Algorithm 22.
  end if
  / 3) Compute enough y vectors: */*
  LA4 Fix minimum number of right vectors $\rightarrow$ minnum.
  Extend right block Krylov space according to Algorithm 8.
end loop

Algorithm 22 Updating solutions and residuals in BLBIORES

Compute $n_{\max} = \max\{n \mid \nu(n) \leq k_l\}$.
for $i = n_{\text{lu}} + 1, \ldots, n_{\max}$ **do**
  if $i = 1$ **then**
    Compute LU decomposition of $\boldsymbol{\tau}_{0,0}$ with pivoting: $\boldsymbol{\pi}_0\boldsymbol{\tau}_{0,0} = \mathbf{L}_1\mathbf{U}_1$.
  else
    Solve (3.24) (with n replaced by $i-1$) for $\mathbf{U}_{12}$.
    Solve $\mathbf{t}\mathbf{U}_{i-1} = \mathbf{t}_2$ for $\mathbf{t}$.
    Compute LU decomposition of $\boldsymbol{\tau}_{i-1,i-1} - \mathbf{t}\mathbf{U}_{12}$ with pivoting: $\boldsymbol{\pi}^{\star}_{i-1}\mathbf{L}_{22}\mathbf{U}_{22} = \boldsymbol{\tau}_{i-1,i-1} - \mathbf{t}\mathbf{U}_{12}$.
    Compute $\mathbf{L}_{21} = \boldsymbol{\pi}_{i-1}\mathbf{t}$.
  end if
  Compute $\mathbf{v}_{i-1}$ via (3.27).
  if $i = 1$ **then**
    Solve $\mathbf{L}_1\mathbf{k}' = \boldsymbol{\pi}_1\mathbf{e}_0\boldsymbol{\eta}_0^{\text{cr}}$ for $\mathbf{k}'$.
  else
    Solve $\mathbf{l}_{i-1,i-1}\mathbf{k}' = -\mathbf{l}_{i-1,i-2}\mathbf{k}'$ for $\mathbf{k}'$.
  end if
  $\mathbf{X}_i^{\text{cr}} = \mathbf{X}_{i-1}^{\text{cr}} + \mathbf{v}_{i-1}\mathbf{k}'$
  Compute $\mathbf{A}\mathbf{v}_{i-1}$ via (3.28).
  $\mathbf{R}_i^{\text{cr}} = \mathbf{R}_{i-1}^{\text{cr}} - \mathbf{A}\mathbf{v}_{i-1}\mathbf{k}'$
  Check convergence
  if $s_i - s_{i-1} > 0$ **then**
    Compute $s_i - s_{i-1}$ elements of $\ker(\mathbf{R}_i^{\text{cr}})$.
    Eliminate $s_i - s_{i-1}$ columns from $\mathbf{X}_i^{\text{cr}}$.
  end if
end for
$n_{\text{lu}} = n_{\max}$

3.3.2 BlBiOMin

Another option for implementing the BlBiCG method is to start from BlBiOC instead of BlBiO. This means that we already have the LU decomposition of $\mathbf{T}_n$. Consequently, we only need to outline how to update the block $\mathbf{k}'_n$. Algorithm 23 gives the statements which are executed in addition to BlBiOC.

Algorithm 23 BlBiOMin

1: During the initializations:
2: Set $n_{\text{sol}} = 0$, nreg $= [\,0\,]$, lreg $= [\,0\,]$.
3: After line 12.39:
4: Check if there is an index i such that $k_l = \nu(i)$.
5: **if** there is such in index i **then**
6: nreg $= [\,\text{nreg}\ i\,]$
7: lreg $= [\,\text{lreg}\ l\,]$
8: **end if**
9: tmp = find(nreg $>$ nsol & nreg $< n$)
10: **if** tmp is nonempty **then**
11: ind = tmp $-$ 1 /* *nreg*(*ind*) = *nsol* */
12: **if** nsol $= 0$ **then**
13: Compute $\boldsymbol{\eta}_0$ such that $\mathbf{R}_0 = \mathbf{y}_0\boldsymbol{\eta}_0$.
14: rhs $= \begin{bmatrix} \mathbb{1}_{s_0} \\ \mathbf{0}_{\nu(\text{nreg}(2))-s_0\times s_0} \end{bmatrix} \boldsymbol{\eta}_0$
15: kpr $= \mathbf{0}_{\nu(\text{nreg}(2))\times m_{\text{cr}}}$
16: start $= 0$
17: shift $= 0$
18: **else**
19: rhs $= \mathbf{0}_{\nu(\text{nreg(ind+1)})-\nu(\text{nsol})\times m_{\text{cr}}}$
20: Set kpr to the last $s_{\text{nsol}-1}$ rows of kpr.
21: start $= \nu(\text{nsol} - 1)$
22: shift $= \nu(\text{nsol})$
23: **end if**
24: **for** $j = \text{lreg(ind)}, \ldots, \text{lreg(ind} + 1) - 1$ **do**
25: tmp1 $= L(k_j + 1{:}k_{j+1}, \text{start} + 1{:}k_j)$
26: tmp2 $= L(k_j + 1{:}k_{j+1}, k_j + 1{:}k_{j+1})$
27: Solve tmp2 $\mathbf{t} = \text{rhs}(k_j + 1 - \text{shift}{:}k_{j+1} - \text{shift}, :) - \text{tmp1} * \text{kpr}(1{:}k_j - \text{start}, :)$ for $\mathbf{t} \rightarrow$ kpr$(k_j + 1 - \text{start}{:}k_{j+1} - \text{start}, :)$.
28: **end for**
29: **for** $j = \text{nsol}, \ldots, \text{nreg(ind} + 1) - 1$ **do**
30: $\mathbf{X}^{\text{cr}}_{j+1} = \mathbf{X}^{\text{cr}}_j + \mathbf{v}_j\text{kpr}(\nu(j) + 1 - \text{start}, \nu(j + 1) - \text{start}, :)$
31: $\mathbf{R}^{\text{cr}}_{j+1} = \mathbf{R}^{\text{cr}}_j - \mathbf{A}\mathbf{v}_j\text{kpr}(\nu(j) + 1 - \text{start}, \nu(j + 1) - \text{start}, :)$
32: Check convergence.
33: Check $\mathbf{X}^{\text{cr}}_{j+1}$ for deflation.
34: nsol $= \text{nreg(ind} + 1)$
35: lsol $= \text{lreg(ind} + 1)$
36: **end for**
37: **end if**

During the initializations, we execute line 23.2. The variable `nsol` gives the largest n such that $\mathbf{X}^{\text{cr}}_n$ has been computed. Then there are two new arrays `nreg` and `lreg` which save all the indices i and l such that $\nu(i) = k_l$. Since $\nu(0) = k_0 = 0$, they initially both contain zero. Lines 23.4 to 23.37 are executed whenever new clusters have been completed. Our first task then is to check whether there is an index i with $k_l = \nu(i)$, and if yes, arrays `nreg` and `lreg` are enlarged. The statement in line 23.9 checks whether the solution $\mathbf{X}^{\text{cr}}_{\text{nsol}}$ can be updated, which is the case if and only if some element

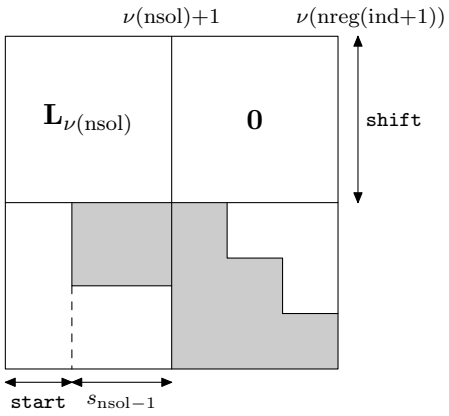


Figure 13: Structure of $\mathbf{L}_{\nu(\mathrm{nreg(ind+1)})}$

of `nreg` is larger than `nsol` and smaller than n. The latter of these two conditions is necessary because at this point, $\mathbf{T}_{n-1} \in \mathbb{C}^{\nu(n-1)\times\nu(n-1)}$ has been computed so we can at most compute $\mathbf{X}^{\mathrm{cr}}_{n-1}$. But k_l might be equal to $\nu(n)$, and in this case, we have to prevent n from being added to `tmp`. When line 23.9 has been executed, `tmp` is an array containing all those indices i such that `nreg`(i) satisfies the two conditions.

It is not hard to see that `tmp` actually contains at most one element. For if there were two different elements i_1 and i_2 of `tmp`, then index i_1 would already have been treated at the latest when the regular index $k_{\mathrm{lreg}(i_2)}$ was fixed and then `nsol` would have been set at least to i_1.

If `nsol` is still zero, we compute $\mathbf{k}'_{\mathrm{nreg}(2)}$ by solving

$$\mathbf{L}_{\mathrm{nreg}(2)}\mathbf{k}'_{\mathrm{nreg}(2)} = \begin{bmatrix} \mathbb{1}_{s_0} \\ \mathbf{0}_{\nu(\mathrm{nreg}(2))-s_0\times s_0} \end{bmatrix} \boldsymbol{\eta}_0$$

in the **for** loop starting in line 23.24. The zero matrix in the right-hand block may be empty. The block $\mathbf{k}'_{\mathrm{nreg}(2)}$ allows us to compute $\mathbf{X}^{\mathrm{cr}}_1, \ldots, \mathbf{X}^{\mathrm{cr}}_{\mathrm{nreg}(2)}$, see the loop in line 23.29.

In case `nsol` is positive, we have computed $\mathbf{k}'_{\mathrm{nsol}}$ and want to obtain $\mathbf{k}'_{\mathrm{nreg(ind+1)}}$ where `ind` is chosen such that `nreg(ind)` = `nsol`. The two variables `start` and `shift` are chosen as shown in Figure 13. We see that only the last $s_{\mathrm{nsol}-1}$ rows of $\mathbf{k}'_{\mathrm{nsol}}$ are still needed whence `start` (which will be used with `kpr`) must be set to $\nu(\mathrm{nsol}-1)$. Similarly, only rows $\nu(\mathrm{nsol})+1$ to $\nu(\mathrm{nreg(ind}+1))$ of `rhs` are needed for the present update (and these rows are all zero) so that `shift` obtains the value $\nu(\mathrm{nsol})$. Finally, when $\mathbf{k}'_{\mathrm{nreg(ind+1)}}$ is computed, we may compute $\mathbf{X}^{\mathrm{cr}}_{\mathrm{nsol}}, \ldots, \mathbf{X}^{\mathrm{cr}}_{\mathrm{nreg(ind+1)}}$.

3.4 Numerical experiments

All the experiments described in this subsection have been carried out with Matlab 7 on a PC with a 2.8GHz Pentium IV processor running Linux. The following preconditioners have been used:

- The SSOR preconditioner where the preconditioned system $\widehat{\mathbf{A}}\widehat{\mathbf{X}} = \widehat{\mathbf{B}}$ was given by

  $$\widehat{\mathbf{A}} :\equiv \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}, \qquad \widehat{\mathbf{X}} :\equiv \mathbf{M}_2\mathbf{X}, \qquad \widehat{\mathbf{B}} :\equiv \mathbf{M}_1^{-1}\mathbf{B},$$

  where

  $$\mathbf{M}_1 :\equiv (\mathbf{D}+\mathbf{L})\mathbf{D}^{-1/2}, \qquad \mathbf{M}_2 :\equiv \mathbf{D}^{-1/2}(\mathbf{D}+\mathbf{U})$$

  if we temporarily denote the diagonal, strictly lower and strictly upper triangular parts of $\mathbf{A}$ by $\mathbf{D}$, $\mathbf{L}$ and $\mathbf{U}$, respectively. This preconditioner was used for all experiments shown in Table 4.
- An incomplete LU factorization as computed by the Matlab command `[L,U] = luinc(A,'0')`. The Matlab Help says that the first output argument is a matrix $\mathbf{L}$ of the form $\mathbf{L} = \mathbf{P}\widetilde{\mathbf{L}}$ with a permuation matrix $\mathbf{P}$ and a lower unit triangular matrix $\widetilde{\mathbf{L}}$. The second output is an upper triangular matrix $\mathbf{U}$. The sparsity patterns of $\mathbf{L}$, $\mathbf{U}$ and $\mathbf{A}$ are not comparable, but $\mathbf{LU}$ agrees

with $\mathbf{A}$ over the sparsity pattern of $\mathbf{A}$. If $\mathbf{U}$ is well-conditioned the preconditioned system $\widehat{\mathbf{A}}\widehat{\mathbf{X}} = \widehat{\mathbf{B}}$ is now given by

$$\widehat{\mathbf{A}} :\equiv \mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}, \qquad \widehat{\mathbf{X}} :\equiv \mathbf{U}\mathbf{X}, \qquad \widehat{\mathbf{B}} :\equiv \mathbf{L}^{-1}\mathbf{B}.$$

In Table 5, this preconditioner will be denoted by "ILU('0')".

- Incomplete LU factorization with drop tolerance (ILUT) $d = \sqrt{\epsilon}$ where $\epsilon \approx 2.2 \cdot 10^{-16}$ is the machine epsilon. According to Matlab's help, this factorization is computed in the same (column-oriented) manner as the LU factorization except after each column of $\mathbf{L}$ and $\mathbf{U}$ has been calculated, all entries in that column which are smaller in magnitude than the local drop tolerance, which is $d \cdot \|\mathbf{A}(:,i)\|$, are dropped from $\mathbf{L}$ or $\mathbf{U}$. The diagonal elements of $\mathbf{U}$ are not affected by this rule so that we also replace zero diagonal entries of $\mathbf{U}$ by the local drop tolerance. The matrices $\mathbf{L}$ and $\mathbf{U}$ can be computed in Matlab by first defining a structure `opt` as follows

```
opt.udiag = 1;
opt.droptol = sqrt(eps);
```

Then `[L,U] = luinc(A,opt)` yields the two factors.

For the convergence tests, we always computed the maximum *unpreconditioned* residual norm relative to the initial residual, i.e.

$$\max_{1 \le i \le s_0} \frac{\|r_n^{(i)}\|_2}{\|r_0^{(i)}\|_2}, \tag{3.29}$$

where $r_n^{(i)}$ is the unpreconditioned residual of system i. Convergence was achieved if this maximum was at most 10^{-6}. In all the plots below, the above maximum is plotted on the y axis.

In order to avoid extremely large clusters, we limited the cluster size to $3s_0$ in all experiments described in this subsection. This means that if a cluster contained $3s_0$ or more vectors, it was completed irrespective of the condition of the corresponding $\boldsymbol{\delta}_l$ (and $\boldsymbol{\delta}'_l$ in the case of BLBIOMIN). Moreover, the maximum number of right and left Lanczos vectors was limited to $100s_0$ ($200s_0$ for matrix Memplus) so that the iteration stopped whenever $\nu(n)$ or $\widetilde{\nu}(\widetilde{n})$ got larger than this limit. Finally, for all methods, the deflation tolerance used in the HRRQR decomposition was set to 10^{-6}.

A Matlab implementation of the block QMR algorithm developed in [8] (denoted FM–BLQMR from now on) can be found on the website [21]. It turns out, however, that this program has the following features:

- It does not contain look-ahead and is therefore susceptible to the Lanczos breakdown.
- Although the FM–BLQMR algorithm allows the blockwise computation of the matrix vector products (MVs), this is not implemented. Instead, all MVs are computed vectorwise.
- For the convergence checks, true residuals are always computed. As the block Krylov spaces used in [8] can be augmented vector by vector, the convergence can be checked whenever the dimension of the block Krylov spaces increases by one. In contrast, with the algorithms developed in this work, a convergence check can only be done when a whole right block has been obtained in its final form (and if look-ahead occurs, even then not always). So although we also used true residuals for checking the convergence, this leads to a lot more additional MVs in the FM–BLQMR program than in the other programs.

For these reasons, it is not entirely straightforward to compare the performance of the FM–BLQMR algorithm with our algorithms. The second and third point lead to more time being spent for the computation of MVs for FM–BLQMR. Therefore, we think that the following correction of the total CPU time for FM–BLQMR may be adequate

$$t_{\mathrm{c}}^{\mathrm{FM}} = t^{\mathrm{FM}} - t_{\mathrm{MV}}^{\mathrm{FM}} + \frac{N_{\mathrm{MV}}^{\mathrm{FM}}}{N_{\mathrm{MV}}^{\mathrm{Bl}}} t_{\mathrm{MV}}^{\mathrm{Bl}}, \tag{3.30}$$

where Table 3 gives the definitions of the quantities in this equation. So if the ratio $N_{\mathrm{MV}}^{\mathrm{FM}}/N_{\mathrm{MV}}^{\mathrm{Bl}}$ is about one, we replace the time needed by FM–BLQMR for the MVs by the time needed by

$N_{\mathrm{MV}}^{\mathrm{FM}}$	number of MVs for building the block Krylov spaces for FM–BLQMR
$N_{\mathrm{MV}}^{\mathrm{Bl}}$	number of MVs for building the block Krylov spaces for BLQMR
$t_{\mathrm{c}}^{\mathrm{FM}}$	corrected CPU time for FM–BLQMR
t^{FM}	measured CPU time for FM–BLQMR
$t_{\mathrm{MV}}^{\mathrm{FM}}$	total time for MVs (also for checking the convergence) in FM–BLQMR
$t_{\mathrm{MV}}^{\mathrm{Bl}}$	total time for MVs (also for checking the convergence) in BLQMR

Table 3: Quantities appearing in (3.30)

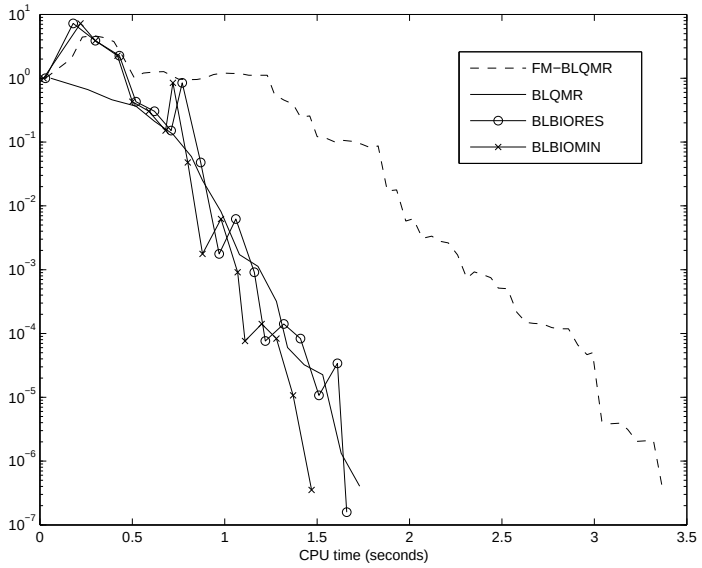


Figure 14: First example with 5 right-hand sides

BLQMR for the MVs. This reflects the fact that with both methods, the MVs can be computed equally efficiently.

Our first experiment is taken from [8, Example 7.1]. We consider the differential equation

$$-\nabla \cdot (e^{xy}\nabla u) + 25(x+y+z)\frac{\partial u}{\partial x} + \left(1 + \frac{1}{1+x+y+z}\right) u = f \tag{3.31}$$

on the unit cube in $\mathbb{R}^3$ with homogeneous Dirichlet boundary conditions. Centered differences on a uniform $15 \times 15 \times 15$ grid are used to discretize the left hand side of (3.31). The resulting matrix $\mathbf{A}$ is real nonsymmetric and of order 3375. The function f appearing on the right-hand side of (3.31) is not specified; instead we use a block $\widehat{\mathbf{B}}$ of five right-hand sides as follows:

$$\widehat{\mathbf{B}} = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 & \widehat{\mathbf{A}}^3 b_1 \end{bmatrix},$$

where $b_1, \ldots, b_4$ are random vectors. The last vector $\widehat{\mathbf{A}}^3 b_1$ ensures that at least one right Lanczos vector (and consequently also one linear system) has to be deflated during the solution. The initial guess $\mathbf{X}_0$ was the zero matrix, and five random vectors were used as left starting block. The convergence curves are given in Figure 14. On the x axis, the CPU time as returned by Matlab's `cputime` function is plotted. Table 4 gives some more information: the first column gives the CPU time for the whole iteration, the second column contains for FM–BLQMR the dimension of the left block Krylov space at the end and for the methods developed in this work the number $\widetilde{\nu}(\widetilde{n}_t)$, the final number of left vectors. The third column shows the analogue information for the right spaces,

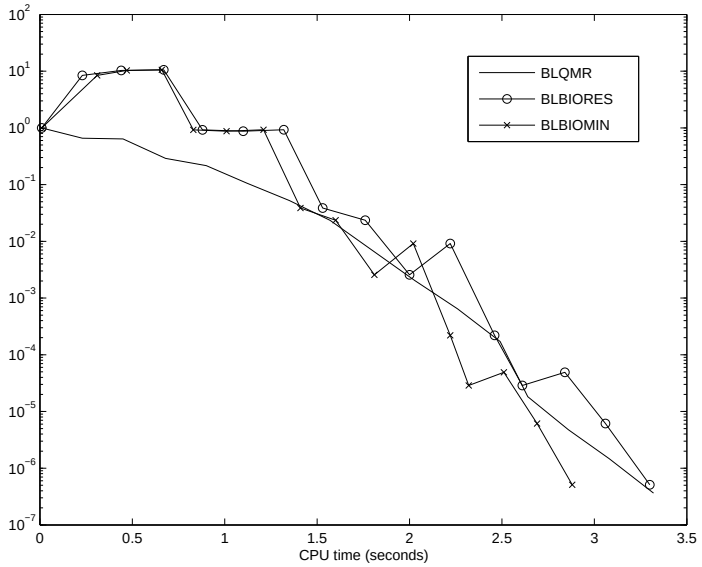


Figure 15: First example with 10 right-hand sides

the fourth the total time for MVs with $\widehat{\mathbf{A}}$, the fifth the same for $\widehat{\mathbf{A}}^\star$ (in the case of FM–BLQMR: with $\widehat{\mathbf{A}}^\top$) and the next the total time for all MVs. Finally, the corrected time for FM–BLQMR according to (3.30) is also given.

It can be seen that FM–BLQMR needs only ldim + rdim $-\ \widetilde{s}_0 = 137$ MVs to build the two final block Krylov spaces while with larger clusters, BLQMR needs 141 and BLBIORES 145. Since $t_c^{\mathrm{FM}} = 1.81$, we conclude that in this example, the FM–BLQMR algorithm is about as fast as BLQMR.

Next we take the same example but 10 right-hand sides instead of five. So

$$\widehat{\mathbf{B}} = \left[b_1 \quad b_2 \quad \ldots \quad b_9 \quad \widehat{\mathbf{A}}^3 b_1 \right],$$

where $b_1, \ldots, b_9$ are random vectors. This time FM–BLQMR stops at iteration 106 with a Lanczos breakdown. Table 4 therefore gives the results for the other methods and Figure 15 shows the convergence curves. We see that in this example, BLBIORES and BLBIOMIN generate the same residual norms.

The next examples are from set QCD found at [16]. In all these examples, the matrix $\mathbf{A}$ has order 3072 and is of the form $\mathbf{A} = \mathbb{1} - k\mathbf{D}$ with $0 \leq k < k_c$ where k_c represents a critical parameter depending on the matrix $\mathbf{D}$. The first 12 standard unit vectors of $\mathbb{C}^{3072}$ form the right-hand sides $\mathbf{B}$ and 12 random vectors are taken as left starting block. We take the matrices conf5.0-00l4x4-1000, conf5.0-00l4x4-1800 and conf6.0-00l4x4-3000 which are all complex nonsymmetric. This time, the corrected times for FM–BLQMR are clearly higher than the corresponding runtimes for BLQMR.

	large clusters							small clusters					
	CPU	ldim	rdim	MV $\widehat{\mathbf{A}}$	MV $\widehat{\mathbf{A}}^\star$	MV tot	t_c^{FM}	CPU	ldim	rdim	MV $\widehat{\mathbf{A}}$	MV $\widehat{\mathbf{A}}^\star$	MV tot
	Example 1 with 5 rhs												
FM–BlQMR	3.48	71	71	2.12	0.78	2.90	1.81	3.26	71	71	2.17	0.72	2.89
BlQMR	1.85	75	71	0.79	0.48	1.27		1.73	75	71	0.77	0.43	1.20
BlBiORes	1.78	75	75	0.41	0.46	0.87		1.73	75	75	0.40	0.44	0.84
BlBiOMin	1.62	75	71	0.44	0.49	0.93		1.71	75	75	0.44	0.47	0.91
	Example 1 with 10 rhs												
BlQMR	3.37	150	147	1.57	0.83	2.40		4.10	150	147	1.62	0.85	2.47
BlBiORes	3.36	150	147	0.80	0.83	1.63		4.10	150	147	0.80	0.83	1.63
BlBiOMin	2.94	150	147	0.95	0.97	1.92		4.43	140	147	0.92	0.86	1.78
	conf5.0-00l4x4-1000												
FM–BlQMR	117.61	335	335	64.06	25.19	89.25	41.83	120.20	335	335	65.57	25.92	91.49
BlQMR	18.66	324	324	7.97	5.05	13.02		28.05	336	336	8.25	5.25	13.50
BlBiORes	17.71	336	336	4.18	5.25	9.43		26.27	336	336	4.18	5.27	9.45
BlBiOMin	17.24	336	336	4.59	5.64	10.23		35.14	336	336	4.48	5.55	10.03
	conf5.0-00l4x4-1800												
FM–BlQMR	58.96	170	170	32.09	12.69	44.78	20.91	58.96	170	170	32.51	12.69	45.20
BlQMR	9.38	168	168	4.11	2.54	6.65		14.85	180	180	4.37	2.72	7.09
BlBiORes	9.22	180	180	2.27	2.71	4.98		13.95	180	180	2.23	2.71	4.94
BlBiOMin	8.95	180	180	2.52	2.98	5.50		18.32	180	180	2.49	2.97	5.46
	conf6.0-00l4x4-3000												
FM–BlQMR	76.64	218	218	41.61	16.57	58.18	27.14	76.75	218	218	41.88	16.54	58.42
BlQMR	12.18	216	216	5.29	3.31	8.60		18.94	228	228	5.57	3.48	9.05
BlBiORes	11.19	216	216	2.71	3.30	6.01		16.93	216	216	2.74	3.34	6.08
BlBiOMin	10.79	216	216	2.93	3.61	6.54		22.13	216	216	2.95	3.56	6.51

Table 4: Results, part 1 (SSOR preconditioning)

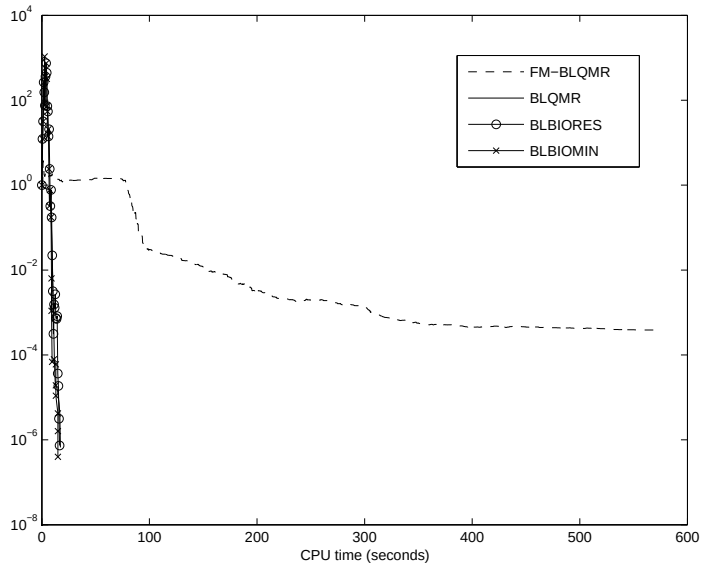


Figure 16: Sherman5 with 20 starting vectors

Next we turn to some Harwell-Boeing matrices [16]. Here we take $s_0 = 5$, 10, 15 and 20 random vectors as (preconditioned) right-hand sides. These experiments were only run with the larger clusters. See Table 5. In this table, an asterisk * means that the method did not converge but managed to reduce the quantity (3.29) at least to 10^{-3}. A bar — indicates failure to reach 10^{-3} and LB means that FM–BLQMR stopped with a Lanczos breakdown.

For Add20, BLBIOMIN always failed because it could not complete a cluster so that the maximum cluster size $3s_0$ was reached and the cluster was completed despite a possible rank deficiency. The rounding errors introduced in this way seem to have spoiled the convergence. A similar problem occurred in the case of Memplus. For matrix Sherman5, FM–BLQMR failed to converge for 15 and 20 starting vectors. Figure 16 shows that in the latter case, FM–BLQMR did not reach 10^{-4}. Concerning Watt1 with 15 rhs, FM–BLQMR almost converged, see Figure 17.

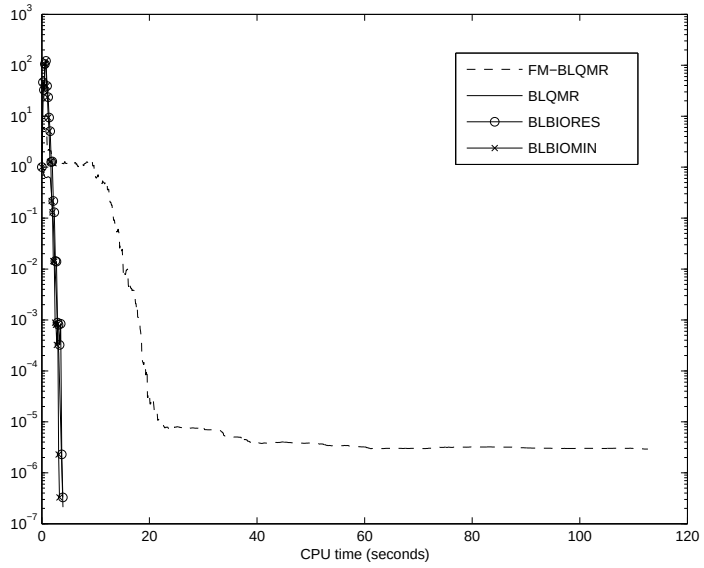


Figure 17: Watt1 with 15 starting vectors

# rhs	5		10		15		20	
	CPU	t_c^{FM}	CPU	t_c^{FM}	CPU	t_c^{FM}	CPU	t_c^{FM}
	add20 (SSOR)							
FM–BlQMR	12.62	5.86	37.41	13.77	69.58	24.03	LB	—
BlQMR	5.46		8.23		11.09		13.98	
BlBiORes	5.43		8.63		11.72		16.36	
BlBiOMin	—		—		—		—	
	memplus (SSOR)							
FM–BlQMR	199.22	80.32	1051.38*	292.77	LB	—	LB	—
BlQMR	137.05*		128.45		175.36		200.52	
BlBiORes	—		142.67		231.58		288.34	
BlBiOMin	—		—		—		—	
	orsreg1 (ILU('0'))							
FM–BlQMR	14.89	5.72	37.07	10.60	81.65	21.12	112.64	30.01
BlQMR	5.03		7.75		9.13		10.14	
BlBiORes	4.70		7.11		8.39		9.93	
BlBiOMin	4.38		6.47		7.55		8.42	
	pores2 (ILUT)							
FM–BlQMR	1.91	0.43	6.32	0.89	13.23	1.76	22.13	2.84
BlQMR	0.36		0.61		0.84		1.43	
BlBiORes	0.30		0.51		0.72		1.22	
BlBiOMin	0.37		0.55		0.79		1.26	
	psmigr3							
FM–BlQMR	22.82	4.76	58.41	7.95	110.68	13.13	176.39	21.56
BlQMR	4.22		6.24		8.36		10.39	
BlBiORes	4.06		6.35		8.26		12.36	
BlBiOMin	3.86		6.36		7.49		10.49	
	sherman5 (ILU('0'))							
FM–BlQMR	19.79	7.10	56.51	14.08	323.58*	70.83	572.71*	123.01
BlQMR	6.57		8.91		15.85		17.52	
BlBiORes	6.49		8.13		14.95		16.71	
BlBiOMin	8.06		10.01		11.56		15.12	
	watt1 (ILU('0'))							
FM–BlQMR	5.35	2.72	11.59	4.44	112.66*	41.59	44.76	17.64
BlQMR	1.73		2.86		3.98		4.91	
BlBiORes	1.73		3.06		3.95		5.35	
BlBiOMin	1.50		2.72		3.34		4.48	
	west1505 (ILUT)							
FM–BlQMR	1.36	0.39	4.67	0.87	9.39	1.64	15.44	2.75
BlQMR	0.32		0.75		0.84		1.14	
BlBiORes	0.28		0.65		0.75		1.03	
BlBiOMin	0.31		0.67		0.79		1.03	

Table 5: Results, part 2

3.5 Conclusions

We are aware that much more experience is needed to evaluate the performance of the algorithms developed in this thesis. Still, the numerical experiments presented in the last subsection show the following tendencies:

- If more than five right-hand sides were used, then the vectorwise approach followed in FM–BlQMR leads to slow convergence.
- As was to be expected, large clusters lead to better performance for BlQMR, BlBiORes and BlBiOMin.
- Concerning the performance of BlQMR, BlBiORes and BlBiOMin relative to each other, there seems to be no clear winner.

References

[1] J. I. Aliaga, D. L. Boley, R. W. Freund, and V. Hernández. A Lanczos-type method for multiple starting vectors. *Math. Comp.*, 69(232):1577–1601, 2000.

[2] Zhaojun Bai, David Day, and Qiang Ye. ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 20(4):1060–1082 (electronic), 1999.

[3] T. F. Chan. Rank revealing QR factorizations. *Linear Algebra Appl.*, 88/89:67–82, 1987.

[4] M. Eiermann and O. Ernst. Geometric aspects of the theory of Krylov space methods. *Acta Numerica*, 10:251–312, 2001.

[5] R. D. Fierro, P. C. Hansen, and P. S. K. Hansen. UTV Tools: Matlab templates for rank-revealing UTV decompositions. *Numerical Algorithms*, 20:165–194, 1999.

[6] R. Fletcher. Conjugate gradient methods for indefinite systems. In G. A. Watson, editor, *Numerical Analysis, Dundee, 1975*, volume 506 of *Lecture Notes in Mathematics*, pages 73–89. Springer, Berlin, 1976.

[7] L. V. Foster. Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra Appl.*, 74:47–71, 1986.

[8] R. W. Freund and M. Malhotra. A block QMR algorithm for non-hermitian linear systems with multiple right-hand sides. *Linear Algebra Appl.*, 254:119–157, 1997.

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

[10] G. Golub and C. F. van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[11] M. H. Gutknecht. A general framework for recursions for Krylov space solvers. Preprint.

[12] M. H. Gutknecht. The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the qd algorithm. in Preliminary Proceedings of the Copper Mountain Conference on Iterative Methods, April 1990. `http://www.sam.math.ethz.ch/~mhg/pub/CopperMtn90.ps.gz` and `CopperMtn90-7.ps.gz`.

[13] M. H. Gutknecht. Lanczos-type solvers for nonsymmetric linear systems of equations. *Acta Numerica*, 6:271–397, 1997.

[14] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bureau Standards*, 45:255–281, 1950.

[15] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bureau Standards*, 49:33–53, 1952.

[16] MatrixMarket Website. `http://math.nist.gov/MatrixMarket`.

[17] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra Appl.*, 29:293–322, 1980. Received Sep. 25, 1978.

[18] A. Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Math. Comp.*, 33(146):680–687, 1979. Received Mar. 9, 1978.

[19] T. Schmelzer. Block Krylov methodes for symmetric linear systems. Master's thesis, University of Kaiserslautern, 2004.

[20] V. Simoncini. A stabilized QMR version of block BICG. *SIAM J. Matrix Anal. Appl.*, 18(2):419–434, 1997. Received Mar. 16, 1994.

[21] The BL-QMR Algorithm. `http://www.bell-labs.com/project/BLQMR`.

[22] D. M. Young and K. C. Jea. Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods. *Linear Algebra Appl.*, 34:159–194, 1980.

Curriculum vitae

25.07.1968	born in Zürich
1975 – 1981	Primary School in Vättis and Eggersriet
1981 – 1987	Kantonsschule St. Gallen
1987 – 1993	Studies of Physics at ETH Zürich
1993	Diploma in theoretical Physics
1995 – 1998	Swiss Center for Scientific Computing, ETH Zürich
1999 – 2005	Seminar for Applied Mathematics, ETH Zürich