A Block-QMR Algorithm for Non-Hermitian Linear Systems With Multiple Right-Hand Sides

Roland W. Freund

AT&T Bell Laboratories
600 Mountain Avenue, Room 2C–420
Murray Hill, New Jersey 07974–0636, U.S.A.
email: *freund@research.att.com*

and

Manish Malhotra*

Department of Civil Engineering
Stanford University
Stanford, California 94305–4020, U.S.A.
email: *manish@am-sun2.stanford.edu*

ABSTRACT

Many applications require the solution of multiple linear systems that have the same coefficient matrix, but differ in their right-hand sides. Instead of applying an iterative method to each of these systems individually, it is more efficient to employ a block version of the method that generates iterates for all the systems simultaneously. In this paper, we propose a block version of Freund and Nachtigal's quasi-minimal residual (QMR) method for the iterative solution of non-Hermitian linear systems. The block-QMR method uses a novel Lanczos-type process for multiple starting vectors, which was recently developed by Aliaga, Boley, Freund, and Hernández, to compute suitable basis vectors for the underlying block Krylov subspaces. We describe the basic block-QMR method, and also give important implementation details. In particular, we show how to incorporate deflation to drop converged linear systems, and to delete linearly and almost linearly dependent vectors in the underlying block Krylov sequences. Numerical results are reported that illustrate typical features of the block-QMR method.

---

*Part of this work was performed while this author was visiting AT&T Bell Laboratories. The research of this author was supported in part by ONR via contract N00014-92-J-1774 with Stanford University.

1. INTRODUCTION

There are important applications that require the solution of multiple systems of linear equations,

$$Ax^{(j)} = b^{(j)}, \quad j = 1, 2, \ldots, m, \tag{1.1}$$

with the same coefficient matrix A, but m different right-hand sides $b^{(j)}$, $j = 1, 2, \ldots, m$. If all m vectors $b^{(j)}$ are available simultaneously, then (1.1) is equivalent to the block system of linear equations,

$$AX = B, \tag{1.2}$$

where

$$B = [\, b^{(1)} \quad b^{(2)} \quad \cdots \quad b^{(m)} \,] \quad \text{and} \quad X = [\, x^{(1)} \quad x^{(2)} \quad \cdots \quad x^{(m)} \,].$$

For example, block systems (1.2) arise in the numerical simulation of wave scattering and wave propagation, where the multiple right-hand sides correspond to incident waves coming in at different angles, or to excitation sources placed at different locations. Block systems (1.2) also arise in the solution of bordered linear systems of the form

$$\begin{bmatrix} A & C \\ D & F \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}, \tag{1.3}$$

where A is assumed to be nonsingular, and the border matrices D and C only have few rows and columns, respectively. Provided that systems with coefficient matrix A are easy to solve, an attractive approach to (1.3) is to first perform one step of block elimination with A. The solution of (1.3) is then computed as follows:

$$\begin{aligned} &\text{Solve} \quad \left(F - DA^{-1}C\right) z = c - DA^{-1}b \quad \text{for} \quad z; \\ &\text{Set} \quad y = A^{-1}b - A^{-1}Cz. \end{aligned}$$

This procedure requires the vector $x := A^{-1}b$ and the matrix $Z := A^{-1}C$, which can be obtained by solving the block system

$$A\,[\, x \quad Z \,] = [\, b \quad C \,].$$

Direct methods, which are based on an LU factorization of A, can trivially be adapted to multiple linear systems (1.1). The main work in direct methods is in the factorization phase, which involves only A and thus needs to be performed only once. The solutions of multiple systems are then obtained at the cost of two backsolves per right-hand side. It is far more intricate to extend iterative methods for linear systems with a single right-hand side, to procedures for multiple linear systems. Iterative methods, such as Krylov-subspace methods, depend on both the coefficient matrix and the right-hand side, and therefore, quantities from a previous run cannot be simply reused for a different right-hand side.

One approach to extend an iterative method for single to multiple systems, is to devise a block variant that is applied to the block formulation (1.2) of multiple linear systems (1.1). A key issue in the design of block iterative methods is the need for *deflation*. The iterates for the m different systems (1.2) that are produced by a block method will, in general, converge at different stages of the block iteration. An efficient and robust block method needs to be able to detect and then deflate converged systems. Each such deflation reduces the block size, and thus the block method needs to be able to handle varying block sizes. For block Krylov-subspace methods, deflation is also crucial in order to delete linearly and almost linearly dependent vectors in the underlying block Krylov sequences. An added difficulty arises for Lanczos-type block methods for non-Hermitian systems, since they involve two different block Krylov sequences. In these methods, deflation can now occur independently in both sequences, and consequently, the block sizes in the two sequences may become different in the course of the iteration, even though they were identical at the beginning.

In this paper, we propose a block version of Freund and Nachtigal's quasi-minimal residual (QMR) method [8] for the solution of non-Hermitian linear systems with single right-hand sides. The QMR algorithm is a Krylov-subspace iteration that uses the look-ahead variant [7] of the classical nonsymmetric Lanczos process [13] to generate basis vectors for the underlying Krylov subspaces. The QMR iterates are defined by a quasi-minimization of the residual norm, which leads to smooth convergence behavior. The block-QMR method (referred to as BL-QMR hereafter) is an extension of QMR to multiple linear systems. The BL-QMR method uses a novel Lanczos-type process for multiple starting vectors, which was recently developed by Aliaga, Boley, Freund, and Hernández [1], to compute suitable basis vectors for the two underlying block Krylov subspaces. The BL-QMR iterates are characterized by a block version of the quasi-minimization property, which can be formulated as a matrix least-squares problem. The underlying Lanczos-type process can handle the most general case of block Krylov sequences with arbitrary block sizes, and in particular, can also handle deflation in both sequences. The BL-QMR method employs the deflation procedure of the Lanczos-type process to detect and delete linearly and almost linearly dependent vectors in the underlying block Krylov sequences. In addition, BL-QMR also includes a deflation procedure to identify and drop linear systems whose solution can be recovered from the solutions of the remaining multiple linear systems. As in the case of the classical Lanczos process [13] for single starting vectors, it cannot be excluded that exact breakdowns or near-breakdowns—triggered by division by zero or by numbers close to zero—occur in the Lanczos-type process used in BL-QMR. As in the case of single starting vectors, the problem of breakdowns can be remedied by incorporating look-ahead into the algorithm. In fact, a detailed description of a look-ahead version of the Lanczos-type process for multiple starting vector is given in [1]. However, in order to keep the exposition relatively short, we will discuss only the BL-QMR method without look-ahead in this paper.

A number of authors have proposed other Lanczos-based block Krylov-subspace methods for the solution of multiple non-Hermitian linear systems. O'Leary [18] was the first to devise a block version of the biconjugate gradient (BCG) method [14], which is closely related to the classical Lanczos process [13]. A different version of block BCG was recently proposed by Simoncini [20]. Boyse and Seidl [3] developed a block variant of the special QMR algorithm [5] for complex symmetric linear systems. However, none of the algorithms in [3, 18, 20] can handle deflations or variable block sizes. The proposed BL-QMR method appears to be the first Lanczos-based block Krylov-subspace iteration for non-Hermitian systems that includes deflation and allows varying block sizes. For the case of Hermitian positive definite systems, Simon and Yeremin [19] and Nikishin and Yeremin [17] devised a block version of the conjugate gradient (CG) algorithm [11] that allows varying block sizes. The situation there is different from the non-Hermitian case, as block CG only involves one block Krylov sequence.

The remainder of this paper is organized as follows. In Section 2, we introduce the notion of block Krylov-subspace methods, and discuss their potential speed-up over solving multiple systems individually. In Section 3, we briefly review the Lanczos-type algorithm for multiple starting vectors proposed in [1], and list a few of its properties. In Section 4, we describe the BL-QMR method, and show how it is affected by deflation. In Section 5, we give implementation details for BL-QMR. In Section 6, we show how BL-QMR simplifies for J-symmetric and J-Hermitian matrices. In Section 7, results of numerical experiments are reported. Finally, we make some concluding remarks in Section 8.

Throughout this paper, all vectors and matrices are allowed to have real or complex entries. As usual, $\overline{M} = [\,\overline{m_{jk}}\,]$, $M^T = [\,m_{kj}\,]$, and $M^H = \overline{M}^T = [\,\overline{m_{kj}}\,]$ denote the complex conjugate, transpose, and the conjugate transpose respectively, of the matrix $M = [\,m_{jk}\,]$. The vector norm $\|x\| := \sqrt{x^H x}$ is always the Euclidean norm, and $\|M\| := \max_{\|x\|=1} \|Mx\|$ is the corresponding matrix norm. In Section 7, we will also use the Frobenius norm $\|M\|_F = \sqrt{\sum_{j,k} |m_{jk}|^2}$ of a matrix $M = [\, m_{jk} \,]$.

2. BLOCK KRYLOV-SUBSPACE METHODS

In this section, we introduce the notions of block Krylov subspaces and block Krylov-subspace methods.

From now on, $A \in \mathbb{C}^{N \times N}$ denotes a given $N \times N$ matrix. Whenever we consider linear systems with A, we also assume that A is nonsingular. Furthermore, $m \geq 1$ and $p \geq 1$ are assumed to be given integers, with m being the number of right-hand sides of the multiple linear systems (1.1). Finally,

$$R = [\, r_1 \quad r_2 \quad \cdots \quad r_m \,] \in \mathbb{C}^{N \times m} \tag{2.1}$$

and

$$L = [\, l_1 \quad l_2 \quad \cdots \quad l_p \,] \in \mathbb{C}^{N \times p} \tag{2.2}$$

are assumed to be arbitrary given $N \times m$ and $N \times p$ matrices, respectively. The columns $r_1, r_2, \ldots, r_m$ of R and the columns $l_1, l_2, \ldots, l_p$ of L will be used as right and left starting vectors for the Lanczos-type Algorithm 3.1, which will be described in Section 3.

2.1. Block Krylov Sequences and Subspaces

The matrix A and the block of right starting vectors R induce the *block Krylov sequence*

$$R, AR, A^2R, \ldots, A^{j-1}R, \ldots . \tag{2.3}$$

Note that, by (2.1), each block $A^{j-1}R$ in (2.3) consists of m vectors, $A^{j-1}r_1, A^{j-1}r_2, \ldots, A^{j-1}r_m$. Moreover each of these vectors is of length N, and thus at most N of the vectors in (2.3) can be linearly independent. By scanning the vectors in (2.3) from left to right and deleting each vector that is either linearly dependent or in some sense "almost" linearly dependent on previous vectors, we obtain a *deflated* block Krylov sequence whose vectors are all linearly independent. This process of deleting linearly and almost linearly dependent vectors is referred to as *deflation* in the sequel. Moreover, we say that the deflation is *exact* if only the linearly dependent vectors are deleted, and we call it *inexact* otherwise. Clearly, in finite-precision arithmetic, it is crucial to also incorporate inexact deflation.

By the Krylov structure of (2.3), a vector $A^{j-1}r_i$ being linearly or almost linearly dependent on previous vectors implies that all vectors $A^k r_i$, $k \geq j$, are also linearly dependent or almost linearly dependent on previous vectors. Consequently, the deflated block Krylov sequence is of the form

$$R^{(1)}, AR^{(2)}, A^2R^{(3)}, \ldots, A^{j_{\max}-1}R^{(j_{\max})}. \tag{2.4}$$

Here, for each $j = 1, 2, \ldots, j_{\max}$, $R^{(j)}$ is a submatrix of $R^{(j-1)}$, with $R^{(j)} \neq R^{(j-1)}$ if, and only if, at least one vector was deleted while scanning the vectors of the j-th block $A^{j-1}R$ in (2.3). (For $j = 1$, we set $R^{(0)} = R$.) We denote by m_j the number of columns of $R^{(j)}$. Note that

$$m \geq m_1 \geq m_2 \geq \cdots \geq m_{j_{\max}} \geq 1,$$

and that $n_{\max} := m_1 + m_2 + \cdots + m_{j_{\max}}$ is just the total number of vectors in the deflated block Krylov sequence (2.4).

Finally, for each $n = 1, 2, \ldots, n_{\max}$, we denote by $\mathcal{K}_n^{\mathrm{dl}}(A, R)$ the subspace of $\mathbb{C}^N$ spanned by the first n vectors of the deflated block Krylov sequence (2.4). We call $\mathcal{K}_n^{\mathrm{dl}}(A, R)$ the n-th *block Krylov subspace* (generated by A and R). Since, by construction, the vectors in (2.4) are all linearly independent, $\mathcal{K}_n^{\mathrm{dl}}(A, R)$ is a subspace of dimension n.

The BL-QMR method also involves the block Krylov sequence

$$L, A^T L, (A^T)^2 L, \ldots, (A^T)^{k-1} L, \ldots, \tag{2.5}$$

that is induced by A^T and the block of left starting vectors L given in (2.2). By performing the above deflation procedure on (2.5), we obtain the deflated block Krylov sequence

$$L^{(1)}, A^T L^{(2)}, (A^T)^2 L^{(3)}, \ldots, (A^T)^{k_{\max}-1} L^{(k_{\max})}. \tag{2.6}$$

Here, for each $k = 1, 2, \ldots, k_{\max}$, $L^{(k)}$ is a submatrix of $L^{(k-1)}$, with $L^{(k)} \neq L^{(k-1)}$ if, and only if, at least one vector was deleted while scanning the vectors of the k-th block $A^{k-1}L$ in (2.5). (For $k = 1$, we set $L^{(0)} = L$.) In analogy to the definition of $\mathcal{K}_n^{\mathrm{dl}}(A, R)$, we denote by $\mathcal{K}_n^{\mathrm{dl}}(A^T, L)$ the subspace of $\mathbb{C}^N$ spanned by the first n vectors of the deflated block Krylov sequence (2.6), and we call $\mathcal{K}_n^{\mathrm{dl}}(A^T, L)$ the *n-th block Krylov subspace* (generated by A^T and L). By construction, the vectors in (2.6) are linearly independent, and thus $\mathcal{K}_n^{\mathrm{dl}}(A^T, L)$ is a subspace of dimension n.

2.2. Block Krylov-Subspace Iterations

We now turn to multiple linear systems, which we assume to be formulated as a block system (1.2).

We call an iterative procedure for the solution of (1.2) a *block Krylov-subspace method* if it generates a sequence of block iterates

$$X_\mu = \begin{bmatrix} x_\mu^{(1)} & x_\mu^{(2)} & \cdots & x_\mu^{(m)} \end{bmatrix} \in \mathbb{C}^{N \times m}, \quad \mu = 0, 1, \ldots, \tag{2.7}$$

such that, for each $j = 1, 2, \ldots, m$,

$$x_\mu^{(j)} \in x_0^{(j)} + \mathcal{K}_\mu^{\mathrm{dl}}(A, R). \tag{2.8}$$

Here, the block

$$X_0 = \left[\, x_0^{(1)} \quad x_0^{(2)} \quad \cdots \quad x_0^{(m)} \,\right], \tag{2.9}$$

is any initial guess for the exact solution $A^{-1}B$ of (1.2), $R = R_0 = B - AX_0$ is the corresponding initial block residual, and $\mathcal{K}_\mu^{\mathrm{dl}}(A, R)$, $\mu = 0, 1, \ldots$, are the block Krylov subspaces defined in Section 2.1.

Throughout this paper, we use the admittedly unconventional notation, $\mu = 0, 1, \ldots$, for the index of the iterates of a block Krylov-subspace method, rather than the standard notation n. Retaining the notation introduced in [1], we use n as the iteration counter for the Lanczos-type Algorithm 3.1 on which BL-QMR is based.

2.3. Potential Speed-Up

An alternative approach to using a block Krylov-subspace method for the solution of (1.2), is to solve each of the m systems (1.1) individually with a typical Krylov-subspace method for a single right-hand side. Using the same initial vectors as in (2.9), these m individual runs of the Krylov-subspace method yield m sequences of iterates

$$\tilde{x}_\mu^{(j)} \in x_0^{(j)} + \mathcal{K}_\mu(A, r_0^{(j)}), \quad \mu = 0, 1, \ldots, \tag{2.10}$$

where $j = 1, 2, \ldots, m$. Here, $r_0^{(j)} = b^{(j)} - Ax_0^{(j)}$ is the initial residual of the j-th system in (1.1), and

$$\mathcal{K}_\mu(A, r_0^{(j)}) = \operatorname{span}\{\, r_0^{(j)}, Ar_0^{(j)}, A^2 r_0^{(j)}, \ldots, A^{\mu-1} r_0^{(j)} \,\}$$

is the μ-th Krylov subspace of $\mathbb{C}^N$ generated by A and $r_0^{(j)}$. Note that $\mathcal{K}_\mu(A, r_0^{(j)})$ has dimension μ, as long as $1 \le \mu \le \dim \mathcal{K}_N(A, r_0^{(j)})$.

Recall that the block Krylov subspace $\mathcal{K}_\mu^{\mathrm{dl}}(A, R)$ in (2.8) also has dimension μ. Therefore, at iteration step μ, the block Krylov-subspace method and the m individual runs of the Krylov-subspace method for single systems generate iterates, given by (2.8) and (2.10) respectively, from subspaces of the same dimension μ. However, the work to build up the spaces in (2.8) and (2.10) is different. The work for generating Krylov and block Krylov subspaces is dominated by the matrix-vector products with A, and so we only count these to compare the two approaches. Furthermore, for simplicity, we assume that no deflation has occurred while building $\mathcal{K}_\mu^{\mathrm{dl}}(A, R)$. It is easy to verify that exactly μ matrix-vector products are required to generate $\mathcal{K}_\mu^{\mathrm{dl}}(A, R)$ from A, B, and X_0. Building each of the Krylov subspaces $\mathcal{K}_\mu(A, r_0^{(j)})$ in (2.10) also involves μ matrix-vector products. However, since there are m of these subspaces, we need a total of $\mu \cdot m$ matrix-vector products with A. Therefore, m individual runs of the Krylov-subspace method require m times more matrix-vector products with A than the block Krylov-subspace method, to generate iterates from subspaces of the same dimension. For Lanczos-type methods that also involve matrix-vector products with A^T, a similar argument shows that the m individual runs also require m times more such products than the block Krylov-subspace method. Of course, the subspaces in the individual runs are different from the subspaces in the block iteration. Nevertheless, the above argument shows that block iterations have the potential for significant speed-up over solving multiple systems individually. The numerical results reported in Section 7 clearly confirm that block methods are faster than the individual solves.

3. A LANCZOS-TYPE ALGORITHM

In this section, we briefly review the Lanczos-type algorithm for multiple starting vectors proposed in [1], and list a few of its properties.

3.1. A Sketch of the Algorithm

We continue to use the notation introduced in Section 2. In particular, R and L are assumed to be given matrices of the form (2.1) and (2.2), respectively, whose columns are the multiple starting vectors.

Given A, R, and L, the Lanczos-type algorithm described in [1] generates two sequences of *right and left Lanczos vectors*,

$$v_1, v_2, \ldots, v_n \quad \text{and} \quad w_1, w_2, \ldots, w_n, \quad n = 1, 2, \ldots,$$

respectively, that span the block Krylov subspaces $\mathcal{K}_n^{\mathrm{dl}}(A, R)$ and $\mathcal{K}_n^{\mathrm{dl}}(A^T, L)$, i.e.,

$$\begin{aligned} \operatorname{span}\{v_1, v_2, \ldots, v_n\} &= \mathcal{K}_n^{\mathrm{dl}}(A, R), \\ \operatorname{span}\{w_1, w_2, \ldots, w_n\} &= \mathcal{K}_n^{\mathrm{dl}}(A^T, L), \end{aligned} \tag{3.1}$$

and that are biorthogonal:

$$w_j^T v_k = \begin{cases} 0, & \text{if } j \neq k, \\ \delta_j \neq 0, & \text{if } j = k. \end{cases} \tag{3.2}$$

Next, we give a complete statement of the algorithm. For a derivation and an explanation of the "history indices" μ_n and ϕ_n, we refer the reader to [1].

ALGORITHM 3.1. (Sketch of the Lanczos-type method with deflation, but without look-ahead.)

INPUT: *Matrix* $A \in \mathbb{C}^{N \times N}$;

m right starting vectors $r_1, r_2, \ldots, r_m \in \mathbb{C}^N$;

p left starting vectors $l_1, l_2, \ldots, l_p \in \mathbb{C}^N$.

0) *Set* $\mu = -m$ *and* $\phi = -p$.
   Set $\mathcal{I}_v = \emptyset$ *and* $\mathcal{I}_w = \emptyset$.

For $n = 1, 2, \ldots,$ *do* (Build n-th pair of Lanczos vectors.) :

1) (Build the right Lanczos vector v_n first.)

   1a) *Set* $\mu = \mu + 1$. *If* $\mu = n$, *then stop.*

   1b) *Set*

$$v = \begin{cases} r_{\mu+m}, & \text{if } \mu \leq 0, \\ Av_\mu, & \text{otherwise.} \end{cases}$$

   1c) *Set*

$$i_v = \begin{cases} 1, & \text{if } \mu \leq 0, \\ \max\{1, \phi_\mu\}, & \text{otherwise.} \end{cases}$$

   Compute

$$t_{i,\mu} = \frac{w_i^T v}{\delta_i} \quad \text{for all } i \text{ with } i_v \leq i \leq n-1 \text{ or } i \in \mathcal{I}_v. \tag{3.3}$$

   Set

$$v = v - \sum_{i=i_v}^{n-1} v_i t_{i,\mu} - \sum_{\substack{i \in \mathcal{I}_v, \\ i < i_v.}} v_i t_{i,\mu}. \tag{3.4}$$

   1d) *If* $\|v\| \leq$ `dtol`, *then deflate the vector* v *and do the following:*

   i. *If* $\mu > 0$ *and the deflated vector* v *is nonzero, then set* $\mathcal{I}_w = \mathcal{I}_w \cup \{\mu\}$.

   ii. *Repeat Step* 1.

2) (Build the left Lanczos vector w_n next.)

   2a) *Set* $\phi = \phi + 1$. *If* $\phi = n$, *then stop.*

   2b) *Set*

$$w = \begin{cases} l_{\phi+p}, & \text{if } \phi \leq 0, \\ A^T w_\phi, & \text{otherwise.} \end{cases}$$

   2c) *Set*

$$i_w = \begin{cases} 1, & \text{if } \phi \leq 0, \\ \max\{1, \mu_\phi\}, & \text{otherwise.} \end{cases}$$

   Compute

$$\tilde{t}_{i,\phi} = \frac{v_i^T w}{\delta_i} \quad \text{for all } i \text{ with } i_w \leq i \leq n-1 \text{ or } i \in \mathcal{I}_w. \tag{3.5}$$

Set

$$w = w - \sum_{i=i_w}^{n-1} w_i \tilde{t}_{i,\phi} - \sum_{\substack{i \in \mathcal{I}_w, \\ i < i_w}} w_i \tilde{t}_{i,\phi}. \tag{3.6}$$

2d) *If* $\|w\| \leq$ `dtol`*, then deflate the vector* w *and do the following:*

i. *If* $\phi > 0$ *and the deflated vector* w *is nonzero, then set* $\mathcal{I}_v = \mathcal{I}_v \cup \{\phi\}$.

ii. *Repeat Step 2.*

3) (Normalize v and w to obtain the n-th pair of Lanczos vectors.)
Set

$$v_n = \frac{v}{t_{n,\mu}} \quad \textit{and} \quad w_n = \frac{w}{\tilde{t}_{n,\phi}}, \tag{3.7}$$

where $t_{n,\mu}$ *and* $\tilde{t}_{n,\phi}$ *are scaling factors given by*

$$t_{n,\mu} = \|v\| \quad \textit{and} \quad \tilde{t}_{n,\phi} = \|w\|. \tag{3.8}$$

Set $\mu_n = \mu$ *and* $\phi_n = \phi$.

4) (Compute δ_n and check for breakdown.)
Set

$$\delta_n = w_n^T v_n. \tag{3.9}$$

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

Remark 3.1. In exact arithmetic, linearly dependent vectors in the block Krylov sequences (2.3) and (2.5) can be detected and deflated in Steps 1d) and 2d), respectively, of Algorithm 3.1 by setting `dtol` $= 0$; hereafter, we refer to this type of deflation as *exact deflation*. In finite precision arithmetic, it becomes necessary to use a small, but finite tolerance, `dtol` > 0, in order to monitor and deflate vectors that are almost linearly dependent; hereafter, we refer to this type of deflation as *inexact deflation*.

Remark 3.2. In practice, the computations in Steps 1c) and 2c), involving the biorthogonalization of vectors v and w against previous vectors, should be implemented by rearranging the loops in (3.4) and (3.6) so that updates on v and w are performed at each i loop. Such a rearrangement corresponds to performing computations in the manner of the modified Gram-Schmidt method and, in general, yields improved numerical properties.

REMARK 3.3. In exact arithmetic, the termination check $\delta_n = 0$ in Step 4) of Algorithm 3.1 is satisfied normally when either $v_n = 0$ or $w_n = 0$. However, it can also happen that Algorithm 3.1 stops because $\delta_n = 0$, but $v_n \neq 0$ and $w_n \neq 0$. In finite precision arithmetic, near breakdowns may occur, i.e., $\delta_n \approx 0$, but $v_n \not\approx 0$ and $w_n \not\approx 0$, which can cause numerical instabilities in subsequent iterations. These problems associated with potential breakdowns and potential instabilities of the Lanczos-type algorithm can be remedied by using a look-ahead variant of Algorithm 3.1 described in [1]. However, for the sake of brevity, in this paper we develop the BL-QMR algorithm based on the Lanczos-type method without look-ahead.

REMARK 3.4. The main expense in obtaining the recurrence coefficients $t_{i,\mu}$ and $\tilde{t}_{i,\phi}$ in (3.3) and (3.5) is the computation of inner products. We remark that these coefficients can be determined cheaply using only about half of the number of inner products that seem necessary in (3.3) and (3.5). This approach is based on exploiting the biorthogonality (3.2) of the vectors in the v and w sequences, to relate the recurrence coefficients in the two sequences, and is similar to the approach employed in the classical Lanczos algorithm.

3.2. Properties of the Algorithm

Next, we list some properties of Algorithm 3.1, which will be used in later sections. In view of (3.7) and (3.8), we have

$$\|v_n\| = \|w_n\| = 1, \quad n = 1, 2, \dots .$$

It is convenient to introduce the notation

$$V^{(n)} := [\, v_1 \quad v_2 \quad \dots \quad v_n \,] \quad \text{and} \quad W^{(n)} := [\, w_1 \quad w_2 \quad \dots \quad w_n \,].$$

Hence, by (3.1), we have

$$\begin{aligned} \mathcal{K}_n^{\mathrm{dl}}(A, R) &= \{\, V^{(n)} z \mid z \in \mathbb{C}^n \,\}, \\ \mathcal{K}_n^{\mathrm{dl}}(A^T, L) &= \{\, W^{(n)} z \mid z \in \mathbb{C}^n \,\}, \end{aligned} \tag{3.10}$$

and, by (3.2) and (3.9),

$$(W^{(n)})^T V^{(n)} = \operatorname{diag}(\delta_1, \delta_2, \dots, \delta_n) \quad n = 1, 2, \dots . \tag{3.11}$$

The recursions for constructing the v's in (3.4) can also be rewritten in matrix formulation as follows:

$$A V^{(\mu)} = V^{(n)} T_e^{(\mu)} + V_{\mathrm{dl}}^{(\mu)}, \quad \mu > 0. \tag{3.12}$$

Here, $\mu = n - m_{\rm cr}$, and $m_{\rm cr}$ denotes the reduced size of the current block in the block Krylov sequence (2.4) due to deflations. Clearly, $m - m_{\rm cr}$ is equal to the total number of deflations performed in the v sequence up to iteration n of Algorithm 3.1. Further, the matrix $T_e^{(\mu)} \in \mathbb{C}^{n\times\mu}$ in (3.12) contains recurrence coefficients appearing in (3.4) such that

$$T_e^{(\mu)} := [\, t_{i,j} \,]_{1\leq i\leq n,\; 1\leq j\leq \mu} \;, \tag{3.13}$$

where the elements $t_{i,j}$ which are not defined in (3.3) are set to be zero. Also, $V_{\rm dl}^{(\mu)} \in \mathbb{C}^{N\times\mu}$ in (3.12) is a matrix containing at most $m - m_{\rm cr}$ nonzero columns corresponding to the deflated v vectors. If `dtol` is the tolerance used for deflation, then we have

$$\|V_{\rm dl}^{(\mu)}\| \;\leq\; \mathtt{dtol}\cdot\sqrt{m - m_{\rm cr}}.$$

We note that, in the absence of any deflations, $T_e^{(\mu)}$ is a banded matrix with a lower bandwidth $m+1$ and upper bandwidth $p+1$. Each deflation step reduces the lower or upper bandwidth by one. Moreover, if deflations are performed in the w sequence then $\mathcal{I}_v \neq \emptyset$ and, consequently, the corresponding rows of $T_e^{(\mu)}$ have additional nonzero entries in columns extending beyond the $p+1$ upper diagonals. We denote the banded part of $T_e^{(\mu)}$, obtained by ignoring nonzero entries with row indices in $\mathcal{I}_v$ which extend beyond its upper bandwidth, by $T_b^{(\mu)}$. Then, (3.12) can be rewritten as follows:

$$AV^{(\mu)} = V^{(n)}T_b^{(\mu)} + \widehat{V}_{\rm dl}^{(\mu)}, \quad \mu > 0. \tag{3.14}$$

Here, $T_b^{(\mu)}$ has lower bandwidth $m+1$ and upper bandwidth $p+1$, and $\widehat{V}_{\rm dl}^{(\mu)}$ includes the columns of $V_{\rm dl}^{(\mu)}$ together with additional nonzero columns that arise from the product in the first term of (3.12) corresponding to entries of $T_e^{(\mu)}$ that are not included in $T_b^{(\mu)}$.

Notice that, the matrix $T_e^{(\mu)}$ defined in (3.13) only contains recurrence coefficients $t_{i,\mu}$ for $\mu > 0$. We define the matrix containing recurrence coefficients in (3.3) up to the iteration when $\mu = 0$ by

$$\rho := [\, t_{i,-m+j} \,]_{1\leq i\leq m_1,\; 1\leq j\leq m} \;, \tag{3.15}$$

where the elements $t_{i,j}$ which are not defined in (3.3) are set to be zero. Here, m_1 is the size of the first block $R^{(1)}$ in the deflated block Krylov sequence (2.4). Note that $m_1 = m$, unless columns of the starting block R are deflated. Using (3.3), (3.4), and (3.15), the recurrences for the initial block of Lanczos vectors, $v_1, v_2, \ldots, v_{m_1}$, can be stated as follows:

$$V^{(m_1)}\rho + V_{\rm dl}^{(0)} = R. \tag{3.16}$$

Here, the j-th column of $V_{\rm dl}^{(0)} \in \mathbb{C}^{N\times m}$ is either the zero vector, or is the deflated v vector at iteration n, where n corresponds to $\mu = -m + j$.

4. THE BLOCK-QMR ALGORITHM

In this section, we describe the BL-QMR algorithm for the simultaneous solution of linear systems (1.1), or equivalently (1.2), with a nonsingular matrix A.

4.1. The Block-QMR Approach

The BL-QMR algorithm is a block Krylov-subspace method that constructs iterates X_μ of the form (2.7) and (2.8), starting with the given block of initial guess vectors $X_0 \in \mathbb{C}^{N \times m}$.

Let the block of residual vectors corresponding to the block iterate X_μ be denoted by

$$R_\mu = B - AX_\mu, \quad \mu = 0, 1, 2, \dots .$$

The BL-QMR method employs $R = R_0$ as the block of right starting vectors in Algorithm 3.1, and generates the right Lanczos vectors $v_1, v_2, \dots, v_n$, $n = 1, 2, \dots$, that satisfy (3.1). From (3.10), it is clear that the v's span the first n linearly independent columns of the block Krylov sequence

$$R_0, AR_0, A^2R_0, \dots,$$

while any dependent (or nearly dependent) columns are deflated. Each time such a deflation is encountered while generating the v sequence, we can also drop one of the columns of X_μ from subsequent BL-QMR iterations. In effect, this corresponds to deflating, or dropping, one of the m linear systems in (1.1) from subsequent iterations. We describe the procedure to do this later in Section 4.2; however, at this point, we introduce some convenient notation.

Let $X_\mu^{\mathrm{cr}} \in \mathbb{C}^{N \times m_{\mathrm{cr}}}$, $\mu = 1, 2, \dots$, denote the current block iterate containing those columns of X_μ that are retained, after each deflation, for subsequent BL-QMR iterations. Let $\mathcal{I} = \{ i_1, i_2, \dots, i_{m_{\mathrm{cr}}} \}$ be the set containing column indices of those linear systems in (1.1) which are present in X_μ^{cr} at iteration $n = \mu + m_{\mathrm{cr}}$. Then, clearly, X_μ^{cr} consists of columns of X_μ given by

$$X_\mu^{\mathrm{cr}} := \begin{bmatrix} x_\mu^{(i_1)} & x_\mu^{(i_2)} & \cdots & x_\mu^{(i_{m_{\mathrm{cr}}})} \end{bmatrix}.$$

Further, we denote the subset of linear systems in (1.1) that correspond to columns of the currently active block iterate X_μ^{cr} by

$$AX^{\mathrm{cr}} = B^{\mathrm{cr}}. \tag{4.1}$$

Here, the matrices $X^{\mathrm{cr}}, B^{\mathrm{cr}} \in \mathbb{C}^{N \times m_{\mathrm{cr}}}$ are given by

$$B^{\mathrm{cr}} := \begin{bmatrix} b^{(i_1)} & b^{(i_2)} & \cdots & b^{(i_{m_{\mathrm{cr}}})} \end{bmatrix},$$
$$X^{\mathrm{cr}} := \begin{bmatrix} x^{(i_1)} & x^{(i_2)} & \cdots & x^{(i_{m_{\mathrm{cr}}})} \end{bmatrix}.$$

In the rest of the paper, following the same notation as above, we use the superscript $^{\mathrm{cr}}$ to denote matrices with m_{cr} columns that correspond to the current set of linear systems in (4.1).

Note that the block of left starting vectors, $L \in \mathbb{C}^{N \times p}$, is still unspecified. Due to lack of a criterion for the choice of L, one can start by setting $p = m$ and choosing the columns of L as random vectors. Recall that, $m_{\mathrm{cr}} = n - \mu$ denotes the size of the current block in the right block Krylov sequence, at iteration n of Algorithm 3.1. Similarly, at iteration n, the size of the current block in the left block Krylov sequence is given by $n - \phi$. It is important to note that, since deflations in the v and w sequences can occur independent of each other, the sizes of current blocks in the two sequences may not always be equal, even though $p = m$ initially.

For the block iterate X_μ^{cr}, by (2.8) and (3.10), we have the parametrization

$$X_\mu^{\mathrm{cr}} = X_0^{\mathrm{cr}} + V^{(\mu)} Z, \quad Z \in \mathbb{C}^{\mu \times m_{\mathrm{cr}}}. \tag{4.2}$$

From (2.8), (3.14) and (3.16), the residual block R_μ^{cr} corresponding to X_μ^{cr} satisfies

$$\begin{aligned} R_\mu^{\mathrm{cr}} &= B^{\mathrm{cr}} - A X_\mu^{\mathrm{cr}} \\ &= R_0^{\mathrm{cr}} - A V^{(\mu)} Z \\ &= R_0^{\mathrm{cr}} - V^{(n)} T_b^{(\mu)} Z - \widehat{V}_{\mathrm{dl}}^{(\mu)} Z \\ &= V^{(n)} \left(\begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - T_b^{(\mu)} Z \right) - \widehat{V}_{\mathrm{dl}}^{(\mu)} Z. \end{aligned} \tag{4.3}$$

Here, the matrix $\rho^{\mathrm{cr}} \in \mathbb{C}^{m_1 \times m_{\mathrm{cr}}}$ contains the columns of ρ in (3.15) that correspond to (4.1). Ideally, we would like to choose $Z \in \mathbb{C}^{\mu \times m_{\mathrm{cr}}}$ in (4.2) such that $\|R_\mu^{\mathrm{cr}}\|$ is minimal. However, in general, since $V^{(n)}$ is not unitary and $\widehat{V}_{\mathrm{dl}}^{(\mu)}$ has some nonzero columns, this would be too expensive.

We will instead minimize just the Euclidean norm of the bracketed terms in (4.3), i.e., we will choose $Z = Z^{(\mu)} \in \mathbb{C}^{\mu \times m_{\mathrm{cr}}}$ as the solution of the matrix least-squares problem

$$\left\| \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - T_b^{(\mu)} Z^{(\mu)} \right\| = \min_{Z \in \mathbb{C}^{\mu \times m_{\mathrm{cr}}}} \left\| \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - T_b^{(\mu)} Z \right\|. \tag{4.4}$$

For the sake of clarity in describing the basic steps of the block-QMR approach, we first consider the solution of the matrix least-squares problem (4.4) for the case when no deflation occurs at iteration n, while generating v_n. In this case, we have

$$V^{(n)} = [\, V^{(n-1)} \quad v_n \,],$$

so that $V^{(n)}$ has full column rank n, and additionally, by (3.13), $T_b^{(\mu)}$ is an $n \times \mu$ matrix with full column rank μ. This guarantees that the solution $Z^{(\mu)}$ of (4.4) is unique, and hence via (4.2) defines a unique block iterate X_μ^{cr}. We now proceed to outline the basic steps involved in solving the matrix least-squares problem (4.4) for this case. The discussion of the special case when deflation occurs during iteration n is deferred to Section 4.2.

For solving (4.4), we use the standard approach based on a QR decomposition of $T_b^{(\mu)}$:

$$T_b^{(\mu)} = (Q^{(\mu)})^H \begin{bmatrix} R^{(\mu)} \\ 0 \end{bmatrix}. \tag{4.5}$$

Here, $Q^{(\mu)}$ is a unitary $n \times n$ matrix, and $R^{(\mu)}$ is a nonsingular upper triangular $\mu \times \mu$ matrix. Inserting (4.5) in (4.4) yields

$$\min_Z \left\| \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - T_b^{(\mu)} Z \right\| = \min_{Z \in \mathbb{C}^{\mu \times m_{\mathrm{cr}}}} \left\| Q^{(\mu)} \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - \begin{bmatrix} R^{(\mu)} \\ 0 \end{bmatrix} Z \right\|.$$

Let

$$\tilde{t}_\mu^{\mathrm{cr}} := \begin{bmatrix} t_\mu^{\mathrm{cr}} \\ \tau_\mu^{\mathrm{cr}} \end{bmatrix} = Q^{(\mu)} \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix}, \tag{4.6}$$

where $t_\mu^{\mathrm{cr}} \in \mathbb{C}^{\mu \times m_{\mathrm{cr}}}$ and $\tau_\mu^{\mathrm{cr}} \in \mathbb{C}^{m_{\mathrm{cr}} \times m_{\mathrm{cr}}}$. Hence, $Z^{(\mu)}$ is given by

$$Z^{(\mu)} = (R^{(\mu)})^{-1} t_\mu^{\mathrm{cr}}, \tag{4.7}$$

and the BL-QMR iterate in (4.2) becomes

$$X_\mu^{\mathrm{cr}} = X_0^{\mathrm{cr}} + V^{(\mu)} (R^{(\mu)})^{-1} t_\mu^{\mathrm{cr}}. \tag{4.8}$$

Furthermore, we note that

$$\left\| \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - T_b^{(\mu)} Z^{(\mu)} \right\| = \| \tau_\mu^{\mathrm{cr}} \|.$$

Using this relation, we deduce from (4.3) that

$$\begin{aligned} \| R_\mu^{\mathrm{cr}} \| &= \left\| V^{(n)} \left(\begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - T_b^{(\mu)} Z \right) - \widehat{V}_{\mathrm{dl}}^{(\mu)} Z \right\| \\ &\le \left\| V^{(n)} \right\| \cdot \left\| \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - T_b^{(\mu)} Z \right\| + \left\| \widehat{V}_{\mathrm{dl}}^{(\mu)} Z \right\| \\ &\le \sqrt{n}\, \| \tau_\mu^{\mathrm{cr}} \| + \left\| \widehat{V}_{\mathrm{dl}}^{(\mu)} Z \right\|. \end{aligned} \tag{4.9}$$

The term $\|\widehat{V}_{\rm dl}^{(\mu)} Z\|$ in (4.9) is usually of the order of the deflation tolerance `dtol`. Omitting that term, we obtain from (4.9) the approximate upper bound

$$\|R_\mu^{\rm cr}\| \;\leq\; \sqrt{n}\,\|\tau_\mu^{\rm cr}\|$$

for the residual. In our implementation of BL-QMR, we use this approximate upper bound to monitor residual norms, and true residuals are computed only in the last few iteration steps, when the upper bound becomes comparable to the desired convergence tolerance.

4.2. How Deflation Affects BL-QMR

Next, we describe the effect of deflations in the Lanczos-type Algorithm 3.1 on the procedure for updating the block iterates $X_\mu^{\rm cr}$. In view of the basic steps for solving the matrix least-squares problem (4.4) described above, we remark that deflations in the w sequence do not directly affect the procedure for performing QMR updates. However, the situation is quite different when deflations are encountered in the v sequence. For simplicity, we will discuss only the case of exact deflations. For that case, we now address the following three issues:

- First, we show that there exists a unique solution $Z^{(\mu)}$ to the minimization problem (4.4) in the presence of deflations.
- Next, the procedure for solving the matrix least-squares problem in the deflation case is outlined.
- Finally, the procedure for dropping a linear system in (1.2) is given.

We begin by considering the solution of (4.4) when a deflation is encountered at Step 1c) of Algorithm 3.1 while generating v_n. In this case, we set $v_n = 0$, which in turn implies that

$$V^{(n)} = [\, V^{(n-1)} \quad 0\,], \tag{4.10}$$

so that $V^{(n)}$ now only has column rank $n-1$. Correspondingly, since $t_{n,\mu} = 0$ in (3.3), we also have

$$T_b^{(\mu)} = \left[\begin{array}{ccc|c} & & & \vdots \\ & T_b^{(\mu-1)} & & t_{n-2,\mu} \\ & & & t_{n-1,\mu} \\ \hline 0 & \cdots & 0 & 0 \end{array}\right]. \tag{4.11}$$

Therefore, the QR factorization of $T_b^{(\mu)}$ in (4.11) can be described as

$$T_b^{(\mu)} = \begin{bmatrix} Q^H & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R^{(\mu)} \\ 0 \end{bmatrix}, \tag{4.12}$$

where Q is an $(n-1)\times(n-1)$ unitary matrix, and $R^{(\mu)}$ is an $\mu\times\mu$ upper triangular matrix.

PROPOSITION 4.1. *There exists a unique solution of the matrix least-squares problem* (4.4) *in the presence of only exact deflations.*

Proof. Clearly, in the presence of only exact deflations, we have $T_b^{(\mu)} = T_e^{(\mu)}$ and $\widehat{V}_{\mathrm{dl}}^{(\mu)} = 0$ for all $\mu > 0$. Using (3.14), (4.10), and (4.12), we then have

$$AV^{(\mu)} = \begin{bmatrix} V^{(n-1)} & 0 \end{bmatrix} T_b^{(\mu)} = V^{(n-1)} Q^H \begin{bmatrix} R^{(\mu)} \\ 0 \end{bmatrix}. \tag{4.13}$$

Since A is nonsingular, (4.13) implies that the QR decomposition in (4.12) results in an upper triangular factor $R^{(\mu)}$ with full column rank μ, and consequently, the matrix least-squares problem (4.4) has a unique solution. ■

Next, we consider the solution of (4.4) at a deflation step. Introducing (4.12) in (4.4), we get

$$\left\| \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - T_e^{(\mu)} Z^{(\mu)} \right\| = \min_{Z \in \mathbb{C}^{\mu\times m_{\mathrm{cr}}}} \left\| \begin{bmatrix} Q & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix} - \begin{bmatrix} R^{(\mu)} \\ 0 \end{bmatrix} Z \right\|. \tag{4.14}$$

In this case, the first matrix product within the norm in (4.14) can be written as

$$\tilde{t}_\mu^{\mathrm{cr}} := \begin{bmatrix} t_\mu^{\mathrm{cr}} \\ \tilde{\tau}_\mu^{\mathrm{cr}} \\ 0 \end{bmatrix} = \begin{bmatrix} Q & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \rho^{\mathrm{cr}} \\ 0 \end{bmatrix}, \tag{4.15}$$

where $t_\mu^{\mathrm{cr}} \in \mathbb{C}^{\mu\times m_{\mathrm{cr}}}$ and $\tilde{\tau}_\mu^{\mathrm{cr}} \in \mathbb{C}^{(m_{\mathrm{cr}}-1)\times m_{\mathrm{cr}}}$. Furthermore, in view of Proposition 4.1, we have

$$Z^{(\mu)} = (R^{(\mu)})^{-1} t_\mu^{\mathrm{cr}}. \tag{4.16}$$

Therefore, the update of the BL-QMR iterates at a deflation step is of the same form as (4.8), i.e.,

$$X_\mu^{\mathrm{cr}} = X_0^{\mathrm{cr}} + V^{(\mu)} \left(R^{(\mu)}\right)^{-1} t_\mu^{\mathrm{cr}}. \tag{4.17}$$

However, we note that, with each deflation the upper bandwidth of $R^{(\mu)}$ also reduces by one. This has the important consequence of shorter recurrences for the update of the BL-QMR iterates (4.17), as we describe in greater detail in Section 5.

Finally, we show how to deflate a linear system when deflation is encountered in generating the v sequence. At a deflation step, the residual block (4.3) corresponding to the iterate $X_\mu^{\rm cr}$ in (4.17) is given by

$$R_\mu^{\rm cr} = V^{(n)} \begin{bmatrix} Q^H & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \tilde{\tau}_\mu^{\rm cr} \\ 0 \end{bmatrix} = V^{(n-1)} Q^H \begin{bmatrix} 0 \\ \tilde{\tau}_\mu^{\rm cr} \end{bmatrix}. \tag{4.18}$$

Since $\tilde{\tau}_\mu^{\rm cr}$ is a rectangular $(m_{\rm cr} - 1) \times m_{\rm cr}$ matrix,

$$\text{there exists a} \quad \gamma \neq 0, \quad \text{such that} \quad \tilde{\tau}_\mu^{\rm cr} \gamma = 0. \tag{4.19}$$

By employing (4.19) in (4.18), we obtain

$$R_\mu^{\rm cr} \gamma = V^{(n-1)} Q^H \begin{bmatrix} 0 \\ \tilde{\tau}_\mu^{\rm cr} \end{bmatrix} \gamma = 0. \tag{4.20}$$

Therefore, using (4.20), we can express the approximate solution of the single linear system

$$A x_\gamma = B^{\rm cr} \gamma,$$

in terms of a linear combination of columns of $X_\mu^{\rm cr}$ by setting

$$x_\gamma = X_\mu^{\rm cr} \gamma. \tag{4.21}$$

Consequently, it is possible to delete one of the linear systems from subsequent BL-QMR iterations; e.g., one can delete the j-th system in (4.1) where j is such that

$$|\gamma_j| = \max_{1 \leq i \leq m_{\rm cr}} |\gamma_i|. \tag{4.22}$$

Accordingly, we also delete the j-th column in $X_\mu^{\rm cr}$, $R_\mu^{\rm cr}$, $\rho^{\rm cr}$, and $\tilde{\tau}^{\rm cr}$ to obtain updated matrices with $m_{\rm cr} - 1$ columns each. Finally, we update $m_{\rm cr} \leftarrow m_{\rm cr} - 1$, and then set $\tau^{\rm cr} = \tilde{\tau}^{\rm cr} \in \mathbb{C}^{m_{\rm cr} \times m_{\rm cr}}$.

The solution vector corresponding to the j-th system which was deflated from $m_{\rm cr}$ systems in (4.22) can be constructed at iteration $n_{\rm max} = \mu_{\rm max} + m_{j_{\rm max}}$, when all solution vectors in the updated block iterate $X_{n_{\rm max}}^{\rm cr}$ have converged. By using (4.19) and (4.21), we then set

$$x_{\mu_{\rm max}}^{(i_j)} = \Big(x_\gamma - \sum_{\substack{k=1 \\ k \neq j}}^{m_{\rm cr}} x_{\mu_{\rm max}}^{(i_k)} \gamma_k\Big) \,/\, \gamma_j.$$

4.3. Sketch of the BL-QMR Algorithm

In this section, we summarize the basic structure of the BL-QMR algorithm.

ALGORITHM 4.2. (Sketch of the BL-QMR method with deflation, but without look-ahead.)
INPUT: *Matrices* $A \in \mathbb{C}^{N\times N}$ *and* $B \in \mathbb{C}^{N\times m}$.

0) *Choose* $X_0 \in \mathbb{C}^{N\times m}$ *and set* $R = R_0 = B - AX_0$.
   Set $p = m$, *and choose* $L \in \mathbb{C}^{N\times m}$.
   Set $\mu = -m$ *and* $\phi = -p$.
   Set $\mathcal{I}_v = \emptyset$ *and* $\mathcal{I}_w = \emptyset$.
   Set $m_{\text{cr}} = m$.

For $n = 1, 2, \ldots,$ *do*:

1) (Build the right Lanczos vector v_n.)
   1*a*) *Perform Steps* 1*a*)–1*c*) *of the Lanczos-type Algorithm* 3.1.
   1*b*) *If* v *is not deflated, i.e.,* $\|v\| >$ `dtol`, *then do the following*:
   Set
   $$v_n = \frac{v}{t_{n,\mu}}, \quad \textit{where} \quad t_{n,\mu} = \|v\|.$$
   Set $\mu_n = \mu$.
   1*c*) *If* v *is deflated, i.e.,* $\|v\| \le$ `dtol`, *then do the following*:
      i) *If* $\mu \le 0$, *then do the following:*
      Delete one column vector from $X_0^{\text{cr}}, R_0^{\text{cr}}, \rho^{\text{cr}}$.
      Set $m_{\text{cr}} = m_{\text{cr}} - 1$.
      Repeat Step 1).
      ii) *If* $\mu > 0$ *and the deflated vector* v *is nonzero, then set* $\mathcal{I}_w = \mathcal{I}_w \cup \{\mu\}$.

2) *If* $\mu > 0$, *then do the following* (Perform BL-QMR update of X_μ^{cr}.) :
   2*a*) *Update the QR factorization* (4.5) *or* (4.12) *of* $T_b^{(\mu)}$, *and* t_μ^{cr} *in* (4.6) *or* (4.15).
   2*b*) *Compute*
   $$\begin{aligned} X_\mu^{\text{cr}} &= X_0^{\text{cr}} + V^{(\mu)}\big(R^{(\mu)}\big)^{-1} t_\mu^{\text{cr}} \\ &= X_{\mu-1}^{\text{cr}} + p_\mu y_\mu^T, \end{aligned} \tag{4.23}$$
   where p_μ *and* y_μ^T *are given by* (5.10) *and* (5.13).
   2*c*) *Check if all solution vectors in* X_μ^{cr} *have converged.*
   If yes, then recover solution vectors corresponding to deflated linear systems and stop.
   2*d*) *Check if we need to deflate in* $AX^{\text{cr}} = B^{\text{cr}}$.
   If yes, then delete one column vector from X_μ^{cr}, R_μ^{cr}, *and* $\tilde{\tau}^{\text{cr}}$.
   Set $\tau^{\text{cr}} = \tilde{\tau}^{\text{cr}}$.

Set $m_{\rm cr} = m_{\rm cr} - 1$.
Repeat Steps 1) *and* 2).

3) (Build the left Lanczos vector w_n.)

3*a*) *Perform Step* 2) *of the Lanczos-type Algorithm* 3.1.
3*b*) *Set*

$$w_n = \frac{w}{\tilde{t}_{n,\phi}}, \quad \textit{where} \quad \tilde{t}_{n,\phi} = \|w\|.$$

Set $\phi_n = \phi$.

4) (Compute δ_n and check for breakdown.)
Set

$$\delta_n = w_n^T v_n.$$

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

5. IMPLEMENTATION DETAILS

In this section, we give some of the details for the actual implementation of Steps 2*a*), 2*b*), and 2*c*) in the BL-QMR Algorithm 4.2. Note that, because of deflation, these steps may be performed several times for the same value of the loop index n. Therefore, throughout this section, we use the value of the parameter μ to uniquely describe the instance when computations within Steps 2*a*), 2*b*), and 2*c*) are performed, and accordingly refer to that as iteration μ.

First, we note that the QR decomposition (4.5) of $T_b^{(\mu)}$ can be computed by means of Givens rotations, taking advantage of the fact that $T_b^{(\mu)}$ is a banded matrix with lower bandwidth less than or equal to $m+1$. In view of this, the unitary factor in (4.5) can be described recursively as:

$$Q^{(\mu)} = \begin{cases} \begin{bmatrix} I_{\mu-1} & 0 \\ 0 & Q_\mu \end{bmatrix} \begin{bmatrix} Q^{(\mu-1)} & 0 \\ 0 & 1 \end{bmatrix}, & \text{if } \mu > 1, \\ \qquad\qquad Q_1, & \text{if } \mu = 1. \end{cases} \tag{5.1}$$

Since, typically $T_b^{(\mu)}$ has only $m_{\rm cr}$ nonzero elements below the diagonal in the μ-th column (except in the case of a deflation step), each of the matrices $Q_\mu \in \mathbb{C}^{(m_{\rm cr}+1)\times(m_{\rm cr}+1)}$, $\mu = 1, 2, \ldots$, is a product of $m_{\rm cr}$ Givens rotations:

$$Q_\mu = G^\mu_{m_{\rm cr}} \begin{bmatrix} 1 & 0 \\ 0 & G^\mu_{m_{\rm cr}-1} \end{bmatrix} \cdots \begin{bmatrix} I_{m_{\rm cr}-1} & 0 \\ 0 & G^\mu_1 \end{bmatrix}. \tag{5.2}$$

Here, for each $j = 1, 2, \ldots, m_{\mathrm{cr}}$,

$$G_j^\mu = \begin{bmatrix} c_j^\mu & s_j^\mu & 0 \\ -\bar{s}_j^\mu & c_j^\mu & 0 \\ 0 & 0 & I_{j-1} \end{bmatrix}, \quad \text{with } c_j^\mu \in \mathbb{R},\ s_j^\mu \in \mathbb{C},\ (c_j^\mu)^2 + |s_j^\mu|^2 = 1. \tag{5.3}$$

However, if a deflation occurs at iteration μ, then $Q_\mu \in \mathbb{C}^{(m_{\mathrm{cr}}+1)\times(m_{\mathrm{cr}}+1)}$ is a product of $m_{\mathrm{cr}} - 1$ Givens rotations that can be described as:

$$Q_\mu = \begin{bmatrix} G_{m_{\mathrm{cr}}-1}^\mu & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & G_{m_{\mathrm{cr}}-2}^\mu & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} I_{m-2} & 0 & 0 \\ 0 & G_1^\mu & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.4}$$

where the matrices $G_j^\mu, j = 1, 2, \ldots, m_{\mathrm{cr}} - 1$, are again given by (5.3). Furthermore, the unitary factor in (4.5) at iteration $\mu + 1$, immediately following a deflation at iteration μ, is given by

$$Q^{(\mu+1)} = \begin{bmatrix} I_\mu & 0 \\ 0 & Q_{\mu+1} \end{bmatrix} Q^{(\mu)},$$

instead of (5.1). Note that, since $T_b^{(\mu)}$ is a banded matrix with both upper and lower bandwidths less than or equal to $m + 1$, the upper triangular factor $R^{(\mu)}$ in (4.5) is also banded with its upper bandwidth less than or equal to $2m + 1$. Moreover, the upper bandwidth of $R^{(\mu)}$ decreases further due to deflations.

Next, we establish that the decomposition in (4.5) can be updated from the factorization of $T_b^{(\mu-1)}$ at the previous iteration $\mu - 1$. In order to obtain $R^{(\mu)}$, one only needs to compute its last column,

$$\begin{aligned} [\,\theta_1 \quad \theta_2 \quad \ldots \quad \theta_\mu\,]^T &= R^{(\mu)} e_\mu^{(\mu)}, \\ \text{where} \quad e_\mu^{(\mu)} &= [\,0 \quad \ldots \quad 1\,]^T \in \mathbb{R}^\mu, \end{aligned} \tag{5.5}$$

and append it to $R^{(\mu-1)}$. This can be done in the following two stages.

In the first stage, the elements $\theta_1, \theta_2, \ldots, \theta_{\mu-1}$ in (5.5) are obtained by applying Givens rotations, introduced up to iteration $\mu - 1$ to the last column of $T_b^{(\mu)}$. From (3.3), we know that the last column of $T_b^{(\mu)}$ has zero elements in row positions $1, 2, \ldots, i_\star - 1$, where $i_\star = \max\{1, \phi_\mu\}$. Furthermore, amongst all previous Givens rotations, the rotation which first applies to row $i_\star$ was introduced at iteration $j_\star = \max\{1, \mu_{i_\star}\}$. As a result, we only need to store and apply Givens rotations $Q_{j_\star}, Q_{j_\star+1}, \ldots, Q_{\mu-1}$, given by (5.2) or (5.4), to the last column of $T_b^{(\mu)}$. By setting

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ \theta_{j_\star} \\ \vdots \\ \theta_{\mu-1} \\ \zeta_0 \\ \vdots \\ \zeta_{m_{\mathrm{cr}}} \end{bmatrix} = \begin{bmatrix} I_{\mu-2} & 0 & 0 \\ 0 & Q_{\mu-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{\mu-3} & 0 & 0 \\ 0 & Q_{\mu-2} & 0 \\ 0 & 0 & I_2 \end{bmatrix} \cdots \qquad (5.6)$$

$$\cdots \begin{bmatrix} I_{j_\star-1} & 0 & 0 \\ 0 & Q_{j_\star} & 0 \\ 0 & 0 & I_{\mu-j_\star} \end{bmatrix} T_b^{(\mu)} e_\mu^{(\mu)},$$

we obtain the desired vector (5.5) except for its last component θ_μ. The following algorithm sketches the underlying procedure in (5.6).

Algorithm 5.1. (Sketch of the procedure to apply previous Givens rotations to the last column of $T_b^{(\mu)}$ at iteration μ of Algorithm 4.2.)

0) *Set* $i_\star = \max\{1, \phi_\mu\}$.
   Set $[\,\xi_1, \xi_2, \dots, \xi_n\,]^T = T_b^{(\mu)} e_\mu^{(\mu)}$.

For $i = i_\star, \dots, n-1$ *do* (Loop over nonzero entries in $\xi_1, \xi_2, \dots, \xi_{n-1}$.) :

1) *Set* $j_b = \mu_i$.

2) *If* $j_b > 0$, *then do the following*:

   2a) *Set*
   $$j_e = \begin{cases} \mu_{i+1} - 1, & \textit{if } i < n-1, \\ \mu - 1, & \textit{if } i = n-1. \end{cases}$$
   Set $m_j = i - j_b$.

   2b) (Apply m_j Givens rotations introduced at iteration $\mu = j_b$ of Algorithm 4.2.)
   Set
   $$[\,\xi_{j_b}, \dots, \xi_i\,]^T = Q_{j_b}\,[\,\xi_{j_b}, \dots, \xi_i\,]^T .$$
   Set $\theta_{j_b} = \xi_{j_b}$.

   2c) (Apply Givens rotations introduced at deflation steps $\mu = j_b + 1, \dots, j_e$ of Algorithm 4.2.)
   For $j = j_b + 1, \dots, j_e$ *do*:

   i) *Set*
   $$[\,\xi_j, \dots, \xi_{j+m_j}\,]^T = Q_j\,[\,\xi_j, \dots, \xi_{j+m_j}\,]^T .$$
   Set $\theta_j = \xi_j$.
   Set $m_j = m_j - 1$.

3) *If* $i = n-1$, *then set* $[\,\zeta_0, \ldots, \zeta_{m_{\rm cr}}\,] = [\,\xi_\mu, \ldots, \xi_n\,]$ *and stop.*

In the second stage, we introduce a new set of suitably chosen Givens rotations that comprise the Q_μ in (5.2), in order to zero out the last $m_{\rm cr}$ elements $\zeta_1, \zeta_2, \ldots, \zeta_{m_{\rm cr}}$ of the vector obtained in (5.6). Note that in the case of deflation at iteration μ, by (4.11), (5.4), and (5.6), we already have $\zeta_{m_{\rm cr}} = t_{n,\mu} = 0$; therefore, here only $m_{\rm cr} - 1$ new Givens rotations of the form (5.4) are necessary. In either case, the coefficients c_j^μ and s_j^μ in (5.3) can be easily determined in analogy to the following 2×2 example. In order to find coefficients c and s in $Q(c,s)$ such that

$$Q(c,s)\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \begin{bmatrix} c & s\\ -\bar{s} & c\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \begin{bmatrix}\widehat{\alpha}\\ 0\end{bmatrix},$$

where $c \in \mathbb{R}, s \in \mathbb{C}, c^2 + |s|^2 = 1$, we set

$$\begin{cases} c = \frac{|\alpha|}{\sqrt{|\alpha|^2+|\beta|^2}}, & \bar{s} = c\frac{\beta}{\alpha}, \quad \text{if } \alpha \neq 0,\\ c = 0, & \bar{s} = 1, \quad \text{if } \alpha = 0, \end{cases} \tag{5.7}$$

and, thereafter, obtain

$$\widehat{\alpha} = c\alpha + s\beta. \tag{5.8}$$

Following the above steps, coefficients c_j^μ, s_j^μ, $j = 1, 2, \ldots, m_{\rm cr}$, in (5.2) and (5.3) can be determined as follows:

1. Initially, set $\widehat{\alpha} = \zeta_{m_{\rm cr}}$,

2. Repeat (5.7)–(5.8) with $\alpha = \zeta_{m_{\rm cr}-j}$, $\beta = \widehat{\alpha}$ for $j = 1, 2, \ldots, m_{\rm cr}$ to obtain $c_j^\mu = c$, $s_j^\mu = s$ at each j.

Similarly, for the case when deflation occurs, c_j^μ, s_j^μ, $j = 1, 2, \ldots, m_{\rm cr} - 1$ in (5.4) and (5.3) can be determined by initially setting $\widehat{\alpha} = \zeta_{m_{\rm cr}-1}$ and repeating (5.7)–(5.8) with $\alpha = \zeta_{m_{\rm cr}-1-j}$, $\beta = \widehat{\alpha}$ for $j = 1, 2, \ldots, m_{\rm cr} - 1$ to obtain $c_j^\mu = c$, $s_j^\mu = s$ at each j. In either case, at the end we set $\theta_\mu = \widehat{\alpha}$.

Using (4.6) and (5.1), we can update the matrix $t_\mu^{\rm cr}$ in (4.8) as follows:

$$\begin{bmatrix} t_\mu^{\rm cr}\\ \tau_\mu^{\rm cr}\end{bmatrix} = \begin{bmatrix} I_{\mu-1} & 0\\ 0 & Q_\mu\end{bmatrix}\begin{bmatrix} t_{\mu-1}^{\rm cr}\\ \tau_{\mu-1}^{\rm cr}\\ 0\end{bmatrix} = \begin{bmatrix} t_{\mu-1}^{\rm cr}\\ Q_\mu \begin{bmatrix}\tau_{\mu-1}^{\rm cr}\\ 0\end{bmatrix}\end{bmatrix}. \tag{5.9}$$

It follows from (5.9) that $t_\mu^{\rm cr}$ differs from $t_{\mu-1}^{\rm cr}$ in only its last row, and hence we can write

$$t_\mu^{\rm cr} = \begin{bmatrix} t_{\mu-1}^{\rm cr}\\ y_\mu^T\end{bmatrix}, \quad \text{where} \quad y_\mu^T \in \mathbb{C}^{1\times m_{\rm cr}}. \tag{5.10}$$

Note that in order to obtain y_μ^T one needs to compute the product in (5.9), which involves applying $m_{\rm cr}$ Givens rotations in the following way:

$$\begin{bmatrix} y_\mu^T \\ \tau_\mu^{\rm cr} \end{bmatrix} = G_{m_{\rm cr}}^\mu \begin{bmatrix} 1 & 0 \\ 0 & G_{m_{\rm cr}-1}^\mu \end{bmatrix} \cdots \begin{bmatrix} I_{m_{\rm cr}-1} & 0 \\ 0 & G_1^\mu \end{bmatrix} \begin{bmatrix} \tau_{\mu-1}^{\rm cr} \\ 0 \end{bmatrix}.$$

We recall that computations of this nature are also encountered in Steps 2*b*) and 2*c*) of Algorithm 5.1. Therefore, before proceeding further, we sketch the procedure for applying Q_μ, which contains $m_{\rm cr}$ Givens rotations of the form (5.2), to an arbitrary column vector $[\,\alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_{m_{\rm cr}+1}\,]^T$. The elements of the resulting vector,

$$\begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{m_{\rm cr}+1} \end{bmatrix} = Q_\mu \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{m_{\rm cr}+1} \end{bmatrix},$$

can be computed using the following formulae:

$$\begin{aligned} \beta_1 &= c_{m_{\rm cr}}\alpha_1 + \sum_{k=2}^{m_{\rm cr}} \left(c_{m_{\rm cr}+1-k} \left(\prod_{i=1}^{k-1} s_{m_{\rm cr}+1-i} \right) \alpha_k \right) \\ &\quad + \left(\prod_{i=1}^{m_{\rm cr}} s_{m_{\rm cr}+1-i} \right) \alpha_{m_{\rm cr}+1}, \\ \beta_j &= -\bar{s}_{m_{\rm cr}+2-j}\ \alpha_{j-1} + c_{m_{\rm cr}+2-j}\ c_{m_{\rm cr}+1-j}\ \alpha_j \\ &\quad + \sum_{k=j+1}^{m_{\rm cr}} \left(c_{m_{\rm cr}+2-j} \left(\prod_{i=1}^{k-j} s_{m_{\rm cr}+2-j-i} \right) c_{m_{\rm cr}+1-k}\ \alpha_k \right) \\ &\quad + \ c_{m_{\rm cr}+2-j} \left(\prod_{i=1}^{m_{\rm cr}+1-j} s_{m_{\rm cr}+2-j-i} \right) \alpha_k, \quad j = 2, \ldots, m_{\rm cr}, \\ \beta_{m_{\rm cr}+1} &= -\bar{s}_1\ \alpha_{m_{\rm cr}} + c_1\ \alpha_{m_{\rm cr}+1}. \end{aligned}$$

Here, for the sake of clarity, we have omitted the superscript $^\mu$ from coefficients c_j^μ and s_j^μ.

Next, we describe the actual computation of the BL-QMR iterates $X_\mu^{\rm cr}$ in (4.23). We define vectors p_j via

$$P^{(\mu)} = [\,p_1 \quad p_2 \quad \ldots \quad p_\mu\,] := V^{(\mu)}(R^{(\mu)})^{-1}. \tag{5.11}$$

Then, from (4.23), (4.7), and (5.10), it follows that

$$X_\mu^{\rm cr} = X_{\mu-1}^{\rm cr} + p_\mu y_\mu^T, \tag{5.12}$$

and therefore, the BL-QMR iterates can be updated via just a rank-one update. Next, following the approach in [8], we establish how to compute p_μ. From (5.5), (5.6), and (5.11), one obtains the relation

$$p_\mu = (v_\mu - \sum_{i=j_\star}^{\mu-1} p_i \theta_i) \,/\, \theta_\mu, \tag{5.13}$$

and, since $(\mu - j_\star) \leq 2m$, we have that p_μ can be updated via short recurrences.

6. BL-QMR FOR J-SYMMETRIC AND J-HERMITIAN MATRICES

It is well known that the classical Lanczos process for general non-symmetric matrices A simplifies when applied to complex symmetric A or Hermitian matrices A. The resulting complex symmetric and Hermitian Lanczos processes only involve one sequence of Lanczos vectors, rather than two sequences as in the case of general A. The classical Lanczos process can also be simplified when A is a J-symmetric or J-Hermitian matrix, by choosing an appropriate left starting vector; see [9] and the references there.

The concept of simplification immediately carries over to the Lanczos-type Algorithm 3.1, provided that $m = p$, and to the BL-QMR method based on Algorithm 3.1. We now briefly describe this simplification, and give one example.

Let $J \in \mathbb{C}^{N \times N}$ be a nonsingular matrix. A matrix $A \in \mathbb{C}^{N \times N}$ is called *J-symmetric* if

$$A^T J = JA.$$

A matrix $A \in \mathbb{C}^{N \times N}$ is called *J-Hermitian* if

$$A^H J = JA.$$

First, assume that A is a J-symmetric matrix for some given matrix J. Recall that the left starting block L in the BL-QMR method can be chosen arbitrarily, while the right starting block $R = R_0$ is given as the initial block residual. We now couple L to R by choosing $L = JR$. Using this relation together with the J-symmetry of A, one easily verifies that all the vectors v_n and w_n generated by Algorithm 3.1 are coupled as follows:

$$w_n = \eta_n \, J \cdot v_n \quad \text{for all} \quad n = 1, 2, \ldots, \tag{6.1}$$

where $\eta_n \in \mathbb{C}$, $\eta_n \neq 0$, are suitable normalization factors.

In view of (6.1), Algorithm 3.1 and thus BL-QMR simplifies for J-symmetric matrices A in that we only need to generate the right Lanczos vectors $v_1, v_2, \ldots$, while the left Lanczos vectors $w_1, w_2, \ldots$, are obtained via (6.1). In particular, the resulting simplified BL-QMR method does not require matrix-vector multiplications with A^T. Clearly, the simplification is only viable if the matrix-vector products with J, which are required in (6.1), are cheap. Fortunately, this is the case in many important situations. Next, we give such an example.

EXAMPLE 6.1. We consider the solution of complex symmetric block systems:

$$AX = B, \quad \text{where} \quad A = A^T \in \mathbb{C}^{N \times N} \quad \text{and} \quad B \in \mathbb{C}^{N \times m}. \tag{6.2}$$

Let $M = M^T \in \mathbb{C}^{N \times N}$ be a nonsingular, complex symmetric matrix that we want to use as a preconditioner for the system (6.2). To allow for left, right, or two-sided preconditioning, we write the preconditioner M in the form $M = M_1 \cdot M_2$. We stress that M_1 and M_2 are arbitrary matrices such that

$$M = M_1 M_2 = M_2^T M_1^T = M^T, \tag{6.3}$$

and that M_1 and M_2 need not be the transpose of each other. Then, instead of (6.2), we solve the equivalent preconditioned block system

$$\begin{aligned} &A'X' = B', \\ &\text{where} \quad A' = M_1^{-1} A M_2^{-1}, \quad X' = M_2 X, \quad \text{and} \quad B' = M_1^{-1} b. \end{aligned} \tag{6.4}$$

Note that $M_1 = I$ for right preconditioning, and $M_2 = I$ for left preconditioning. We now set

$$J = M_1^T M_2^{-1}. \tag{6.5}$$

Using $A = A^T$ and (6.3), one readily verifies that

$$(A')^T J = M_2^{-T} A M_2^{-1} = M_1^T M_2^{-1} M_1^{-1} A M_2^{-1} = J A'.$$

Hence, the matrix A' is J-symmetric, and the BL-QMR method simplifies when applied to the preconditioned block system (6.4).

Now assume that A is a J-Hermitian matrix for some given matrix J. Here, we choose L as the complex conjugate of JR, i.e., $L = \overline{JR}$. Again, it is easy to verify that then all the vectors v_n and w_n generated by Algorithm 3.1 are coupled as follows:

$$w_n = \eta_n \, \overline{J \cdot v_n} \quad \text{for all} \quad n = 1, 2, \ldots, \tag{6.6}$$

where $\eta_n \in \mathbb{C}$, $\eta_n \neq 0$, are suitable normalization factors. Therefore, for J-Hermitian matrices A, the Lanczos-type Algorithm 3.1 and thus the BL-QMR method simplify, and the recurrences for w_n can be replaced by (6.6).

7. NUMERICAL EXAMPLES

In this section, we present some numerical examples to illustrate the typical convergence behavior of the BL-QMR method, and also to demonstrate the importance of deflation in block Krylov-subspace methods.

The examples considered here arise in acoustic wave propagation, and have coefficient matrices A that are complex symmetric. The resulting block systems are thus of the form (6.2), and we apply the simplified BL-QMR method discussed in Section 6. When solving realistic problems, as is typical for other Krylov-subspace methods, it is important to employ an efficient preconditioning technique in conjunction with the BL-QMR method. Here, we use the SSOR preconditioner [2] given by

$$M_1 = (D + \omega L)D^{-1/2}, \quad M_2 = D^{-1/2}(D + \omega L^T), \tag{7.1}$$

where D, L, and L^T, respectively, are the diagonal, strictly lower triangular, and strictly upper triangular part of the complex symmetric matrix $A = L + D + L^T$. Although the choice of ω effects the performance of the preconditioner for the class of problems considered here (see [6]), in the numerical tests reported here, we always used $\omega = 1.0$. Instead of solving (6.2), we now apply the BL-QMR algorithm to the preconditioned system (6.4). We note that, when M_1 and M_2 are given by (7.1), the Eisenstat trick [4] can be used to compute a matrix-vector product with the preconditioned matrix A' of (6.4) at a cost only slightly more than the cost of a matrix-vector product with A. Furthermore, we remark that, by (7.1), the matrix J in (6.5) reduces to $J = I$, the identity matrix. Therefore, the matrix-vector products with J in the simplified BL-QMR method are completely trivial for the preconditioner (7.1).

Finally, we note that, in all examples, we stop the BL-QMR iterations when the residual vectors for the undeflated linear systems of the preconditioned system (6.4) satisfy

$$\max_{j=1,2,\ldots,m_{\mathrm{cr}}} \|r_\mu^{(j)}\| < 10^{-6} \cdot \|r_0^{(j)}\|.$$

EXAMPLE 7.1. We consider linear systems with multiple right-hand sides that arise in time-harmonic acoustic scattering. In these problems, it is required to determine the pressure field due to scattering of an incoming monochromatic plane wave by a rigid obstacle submerged in an infinite fluid medium, for different angles of the incoming wave. The corresponding boundary-value problem involves the Helmholtz equation in an infinite domain, and in two dimensions can be stated as follows:

$$-\nabla^2 p - k^2 p \;=\; 0 \quad \text{in} \quad \Omega_\infty\,, \tag{7.2}$$

$$\nabla p \cdot n \;=\; g(\theta) \quad \text{on} \quad \Gamma_s\,, \tag{7.3}$$

$$\lim_{r\to\infty} \sqrt{r}\left(\frac{\partial p}{\partial r} - ikp\right) \;=\; 0\,. \tag{7.4}$$

Here, p denotes the scattered pressure in the infinite fluid domain Ω_∞, k is the acoustic wavenumber, r and θ denote the polar coordinates, $i = \sqrt{-1}$, $g(\theta)$ is the prescribed Neumann data due to a plane wave incident at θ, and n is the unit outward normal to the scatterer boundary Γ_s. Equation (7.4) is the Sommerfeld radiation condition, which asserts that at infinity all waves are outgoing. The infinite domain in these problems is treated by introducing an artificial boundary close to the scatterer. The finite fluid domain enclosed between the scatterer and the artificial boundary is then discretized using finite elements. Additionally, in order to eliminate spurious reflection of waves, an absorbing boundary condition also needs to be imposed on this artificial boundary. In our numerical tests, we employ the Dirichlet-to-Neumann (DtN) boundary condition [12] for this purpose. The finite-element discretization of the resulting problem is described in [15, 16]. Here, we note that discretization of the DtN boundary condition leads to a dense, complex symmetric matrix whose size is equal to the number of grid points on the artificial boundary.

In our numerical tests, we consider scattering from an end-capped rigid cylinder with an aspect ratio $l/d = 8.0$. The DtN condition is applied on a circular artificial boundary of radius l, and the non-dimensional wavenumber is taken as $kd = \pi/3$. The finite-element discretization is chosen to adequately resolve the pressure field, and resulted in a complex symmetric coefficient matrix A of order 3200 and 44034 nonzeros [15]. The m columns in the block of right-hand sides B arise from (7.3) for various plane waves incident at $\theta_j = j\Delta\theta$, $j = 0, 1, \ldots, m-1$, with $\Delta\theta = 1.5°$.

Table 7.1 shows the number of iterations needed for convergence of the BL-QMR algorithm for various block sizes m, with $X_0 = 0$. Notice that as m increases, fewer number of iterations per each right-hand side are needed, and also that several deflations are performed when $m \geq 10$. We remark that these deflations were necessary to enforce linear independence (up to specified $\texttt{dtol} = 10^{-6}$) amongst the initial block of Lanczos vectors. In order to illustrate the effect of these deflations on convergence of block Krylov iterations, we set $\texttt{dtol} = 10^{-14}$ so that no deflations are performed in Algorithm 4.2. The resulting iteration counts, which are shown in Table 7.2,

TABLE 7.1
ITERATION COUNTS FOR CONVERGENCE OF BL-QMR WITH INCREASING BLOCK SIZE m AND $X_0 = 0$.

Number of RHS (m)	1	5	10	15	20	25
Number of deflations	0	0	2	5	9	13
Number of iterations	133	278	463	565	775	928

clearly demonstrate that the convergence of block iterations deteriorates considerably if linear independence of the starting block is not enforced strictly. In Figure 7.1, we illustrate the typical convergence behavior of linear systems that remain after initial deflations; these convergence curves indicate a smooth and almost monotonic reduction of BL-QMR residuals for each of the 8 undeflated systems corresponding to the case $m = 10$ in Table 7.1.

TABLE 7.2
ITERATION COUNTS FOR CONVERGENCE OF BL-QMR WITHOUT DEFLATION AND $X_0 = 0$.

Number of RHS (m)	Number of iterations(μ_t)	$\|\|R_{\mu_t}\|\|_F / \|\|R_0\|\|_F$
1	133	6.75×10^{-7}
5	278	8.27×10^{-7}
10	626	7.15×10^{-7}
15	2000	5.55×10^{-1}
20	2000	4.79×10^{-1}
25	2000	7.03×10^{-1}

In order to avoid excessive number of deflations in the initial block of Lanczos vectors when $X_0 = 0$, one can alternatively choose X_0 as a matrix with random entries from a uniform distribution on the interval (-1,1); we denote this choice as $X_0 = \text{rand}(-1, 1)$. Table 7.3 gives the iteration counts required with $X_0 = \text{rand}(-1, 1)$ for each of the block sizes m considered in

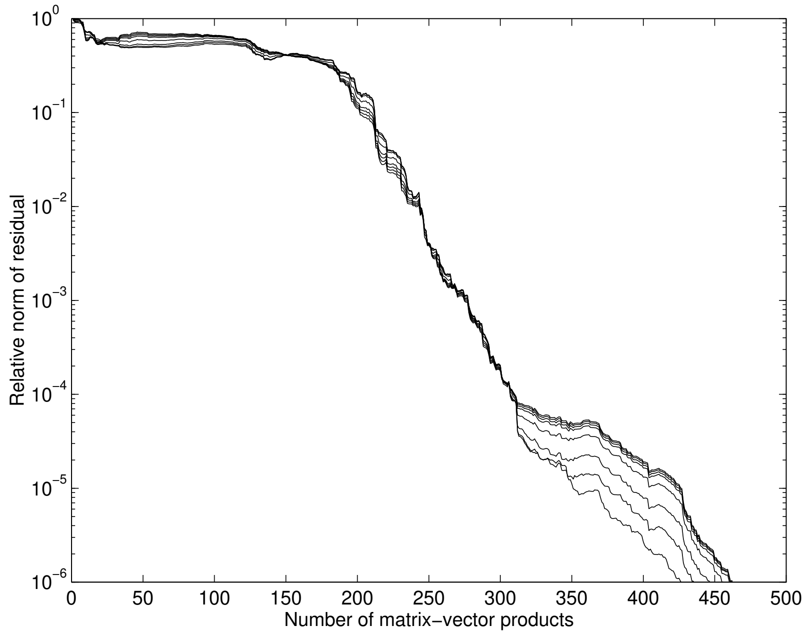

FIG. 7.1. Convergence curves for each of the 8 undeflated linear systems for the case $m = 10$ in Table 7.1 (see Example 7.1).

Table 7.1. Notice that $X_0 = \mathrm{rand}(-1, 1)$, in fact, yields faster convergence of the block algorithm for the current example; for instance, when $m = 25$ only 21.9 iterations per each RHS were required and no deflations were performed. To illustrate the faster convergence of BL-QMR iterations with increasing block size m, we plot in Figure 7.2 the Frobenius norm of the residual block R_μ, for each of the runs in Table 7.3, against its iteration number normalized with the block size m.

We remark that choosing $X_0 = \mathrm{rand}(-1, 1)$ may not always be advantageous and, as the next example illustrates, in some cases the natural choice of setting $X_0 = 0$ can potentially be more useful, provided deflations are enforced to maintain linear independence of the Lanczos vectors.

EXAMPLE 7.2. We consider a fluid-structure interaction problem that involves determining: i) the vibrations of an elastic body due to external forces acting on it, and ii) the acoustic pressure radiated into the surrounding fluid medium due to vibrations of the elastic body. The corresponding boundary-value problem is described by the time-harmonic equations of lin-

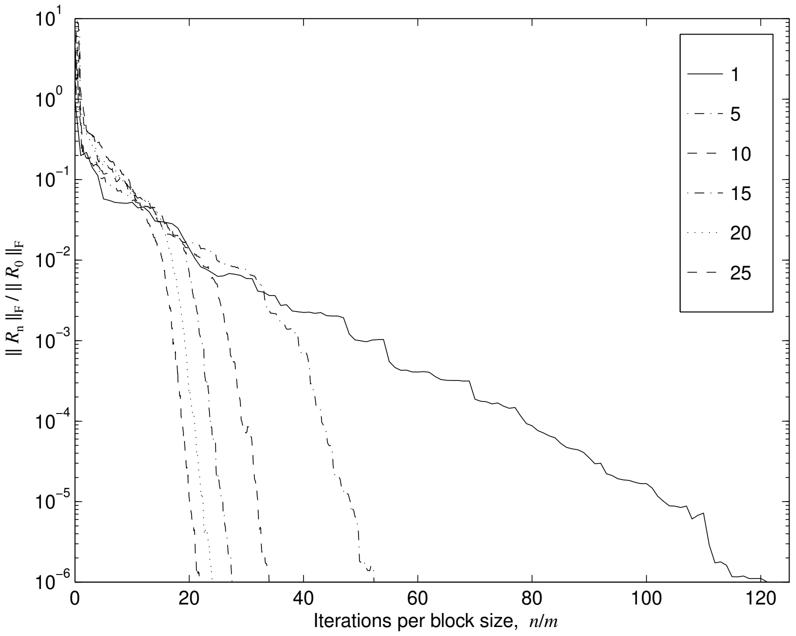

FIG. 7.2. An illustration of the per RHS convergence obtained with increasing block size m and $X_0 = \mathrm{rand}(-1, 1)$ (see Example 7.1).

ear elasticity that govern the deformation of the elastic body, together with equations (7.2) and (7.4) that govern the propagation of acoustic waves in the infinite fluid, and two boundary conditions that couple the fluid and structural unknowns by enforcing compatibility of tractions and the normal displacement across the fluid-structure interface. The multiple right-hand sides in this example arise due to multiple-load cases applied on the elastic body.

In our numerical tests, we consider a canonical two-dimensional interaction problem studied in [10], where four interconnected plates are considered to be immersed underwater and the external load is taken as a unit point force on one of the plates. The linear system of equations is obtained via a Galerkin finite-element discretization of the problem; for a description of the governing equations, and the geometrical and material properties employed here, we refer the reader to [10, Chapter 5]. Here, we note that the resulting coefficient matrix A is complex symmetric, of order 7040 and has 86346 nonzero entries. In addition to the single right-hand side case, we consider the case $m = 5$ where the influence of shifting the given point load to four adjacent grid points in the finite-element mesh is examined.

TABLE 7.3
ITERATION COUNTS FOR CONVERGENCE OF BL-QMR WITH INCREASING BLOCK SIZE m AND $X_0 = \text{rand}(-1, 1)$.

Number of RHS (m)	1	5	10	15	20	25
Number of iterations	122	262	342	413	481	548
Number of iter./RHS	122	52.4	34.2	27.5	24.1	21.9

The iteration counts for convergence of BL-QMR are shown in Table 7.4. Notice that, unlike Example 7.1, the choice $X_0 = 0$ requires fewer total iterations than $X_0 = \text{rand}(-1, 1)$, in both cases $m = 1, 5$. Also, we note that three deflations were performed at iterations $\mu = 3, 4, 5$ of Algorithm 4.2 when $X_0 = 0$. For this case, we show in Figure 7.3 the norms of the candidate Lanczos vectors v that are first generated in Step 1a) of Algorithm 4.2 and then checked in Step 1b) against the deflation tolerance `dtol` $= 10^{-6}$. The figure illustrates that linearly dependent vectors in the v sequence could be detected easily in this example. Finally, we note that if one chooses $X_0 = 0$ but does not enforce deflations, then the convergence of block iterations for $m = 5$ case deteriorates significantly and the block residuals only satisfy $\|R_\mu\|_F < (1.64 \times 10^{-2}) \cdot \|R_0\|_F$, even after $\mu = 2500$ iterations.

TABLE 7.4
ITERATION COUNTS FOR CONVERGENCE OF BL-QMR FOR $m = 1, 5$ WITH DIFFERENT CHOICES OF X_0.

Number of RHS (m)	$X_0 = 0$		$X_0 = \text{rand}(-1, 1)$	
	Number of deflations	Number of iterations	Number of deflations	Number of iterations
1	0	570	0	578
5	3	538	0	889

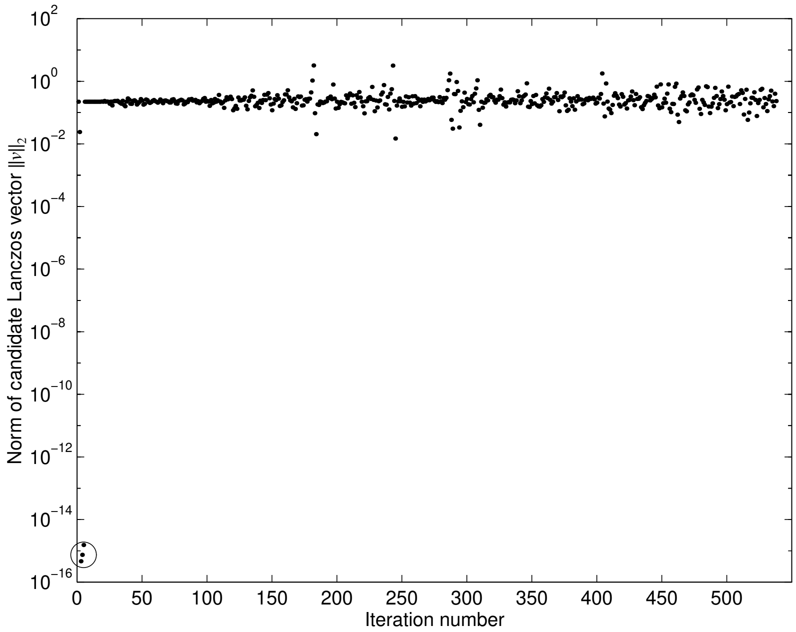

FIG. 7.3. Norms of candidate Lanczos vectors checked for deflation at each iteration for the case $m = 5$ with $X_0 = 0$ (see Example 7.2).

8. CONCLUDING REMARKS

We have proposed a block-QMR method (BL-QMR) for the iterative solution of non-Hermitian linear systems with multiple right-hand sides. The BL-QMR method is a block Krylov-subspace iteration that employs a recently developed novel Lanczos-type process for multiple starting vectors to generate suitable basis vectors for the underlying block Krylov subspaces. The use of this Lanczos-type process allowed us to incorporate into BL-QMR general deflation procedures that can handle deflation due to linearly or almost linearly dependent block Krylov vectors, as well as deflation due to convergence of one of the multiple systems.

The BL-QMR method described in this paper is based on a no-look-ahead version of the underlying Lanczos-type algorithm, and thus breakdowns cannot be excluded. The problem of potential breakdowns can be remedied by employing a look-ahead variant of the Lanczos-type algorithm described in [1]. While the derivation of the resulting BL-QMR method with look-ahead is fairly routine, it would have made this paper considerably longer. Therefore, we opted to describe only the BL-QMR method without look-ahead. However, we stress that we intend to make available code with implementations of BL-QMR, both with and without look-ahead.

Many interesting aspects of the BL-QMR method remain to be investigated. Our numerical experience shows that, typically, the BL-QMR method is less prone to loss of biorthogonality than the QMR method for single systems. Intuitively, this is to be expected since the underlying Lanczos-type Algorithm 3.1 for multiple starting vectors explicitly enforces biorthogonality among a greater number of vectors than the Lanczos process for single starting vectors on which QMR is based. However, a more quantitative understanding of this reduced loss of biorthogonality is needed.

Another interesting problem is the choice of appropriate preconditioners for BL-QMR. In [6], we compare various preconditioners for multiple linear systems arising in wave propagation problems. It turns out that the efficiency of preconditioners depends crucially on the number m of right-hand sides. For the problems and preconditioners considered in [6], SSOR becomes the most efficient preconditioner as m increases. More sophisticated preconditioners, such as hierarchical bases, are superior to or competitive with SSOR only for very small values of m. Much more work is needed to understand preconditioning for the BL-QMR method for specific problem classes.

In the context of parallel and distributed computing, block iterations in general, and hence the BL-QMR method, become particularly attractive. Recall that the main work involved in Krylov-subspace methods is the computation of matrix-vector products with A and, in the case of Lanczos-based iterations, possibly A^T. For block Krylov-subspace methods, the matrix-vector products corresponding to a whole block can be computed in parallel. In particular, this is the case for the BL-QMR method. Note that, in the BL-QMR method, all matrix-vector products with A and A^T are performed in the underlying Lanczos-type Algorithm 3.1. Consider Algorithm 3.1 after iteration n. Then, the matrix-vector products with A that will be needed in the next m_{cr} $(= n - \mu)$ iterations of Algorithm 3.1 are given by

$$A \cdot v_{\mu+1}, A \cdot v_{\mu+2}, \ldots, A \cdot v_n. \tag{8.1}$$

We note that all the vectors $v_{\mu+1}, v_{\mu+2}, \ldots, v_n$ in (8.1) are already available after iteration n, and therefore, we can precompute in parallel all the matrix-vector products (8.1) required in the next $n - \mu$ iterations of Algorithm 3.1. Similarly, after iteration n, we can precompute

$$A^T \cdot w_{\phi+1}, A^T \cdot w_{\phi+2}, \ldots, A^T \cdot w_n,$$

which are all the matrix-vector products with A^T needed within the next $n - \phi$ iterations of Algorithm 3.1. Recall that, if no deflation occurs, then $m = n - \mu$ and $p = n - \phi$ are just the initial block sizes. Due to these computational advantages, block iterations appear to be attractive even for the solution of a single linear system; see, e.g., [17]. In order to employ block iterations for a single linear system, the Lanczos-type Algorithm 3.1 is used with blocks of left and right starting vectors while the solution iterates are determined by solving a least-squares problem similar to the QMR method for single linear systems. In view of the above discussion, it is clear that computations inherent in the BL-QMR method are particularly attractive for developing efficient parallel implementations. We are currently investigating the parallel implementation of the BL-QMR method on a cluster of workstations, and this will be described in a forthcoming report.

REFERENCES

1 J. I. Aliaga, D. L. Boley, R. W. Freund, and V. Hernández, A Lanczos-Type Algorithm for Multiple Starting Vectors, Numerical Analysis Manuscript No. 95–11, AT&T Bell Laboratories, Murray Hill, Nov. 1995.

2 O. Axelsson, A survey of preconditioned iterative methods for linear systems of algebraic equations, *BIT* 25:166–187 (1985).

3 W. E. Boyse and A. A. Seidl, A block QMR method for computing multiple simultaneous solutions to complex symmetric systems, *SIAM J. Sci. Comput.* 17 (1996), to appear.

4 S. C. Eisenstat, Efficient implementation of a class of preconditioned conjugate gradient methods, *SIAM J. Sci. Statist. Comput.* 2:1–4 (1981).

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

6 R. W. Freund and M. Malhotra, Iterative Solution of Multiple Radiation and Scattering Problems in Structural Acoustics Using a Block Quasi-Minimal Residual Algorithm, Numerical Analysis Manuscript, AT&T Bell Laboratories, Murray Hill, in preparation.

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

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

9 R. W. Freund and N. M. Nachtigal, Software for simplified Lanczos and QMR algorithms, *Appl. Numer. Math.* 19:319–341 (1995).

10 K. Grosh, Design and Analysis of Computational Methods for Structural Acoustics, Ph.D. Thesis, Stanford University, Stanford, 1994.

11 M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards*, 49:409–436 (1952).

12 J. B. Keller and D. Givoli, Exact non-reflecting boundary conditions, *J. Comput. Phys.*, 82:172–192, (1989).

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

14 C. Lanczos, Solution of systems of linear equations by minimized iterations, *J. Res. Nat. Bur. Standards* 49:33–53 (1952).

15 M. Malhotra and P. M. Pinsky, Parallel Preconditioning Based on h-Hierarchical Finite Elements with Application to Acoustics, Manuscript SCCM-95-05, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, Apr. 1995.

16 M. Malhotra and P. M. Pinsky, A matrix-free interpretation of the nonlocal Dirichlet-to-Neumann radiation boundary condition, Preprint, 1995.

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

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

19 H. D. Simon and A. Yu. Yeremin, A new approach to construction of efficient iterative schemes for massively parallel applications: variable block CG and BiCG methods and variable block Arnoldi procedure, in *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing* (R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold, and D. A. Reed, Eds.), SIAM, Philadelphia, 1993, pp. 57–60.

20 V. Simoncini, A New Version of Block BICG for Multiple Systems, Manuscript, 1994.