AN IMPLEMENTATION OF THE QMR METHOD BASED ON COUPLED TWO-TERM RECURRENCES*

ROLAND W. FREUND† AND NOËL M. NACHTIGAL‡

Abstract. Recently, the authors proposed a new Krylov subspace iteration, the quasi-minimal residual algorithm (QMR), for solving non-Hermitian linear systems. In the original implementation of the QMR method, the Lanczos process with look-ahead is used to generate basis vectors for the underlying Krylov subspaces. In the Lanczos algorithm, these basis vectors are computed by means of three-term recurrences. It has been observed that, in finite precision arithmetic, vector iterations based on three-term recursions are usually less robust than mathematically equivalent coupled two-term vector recurrences.

This paper presents a look-ahead algorithm that constructs the Lanczos basis vectors by means of coupled two-term recursions. Some implementation details are given, and the look-ahead strategy is described. A new implementation of the QMR method, based on this coupled two-term algorithm, is proposed. A simplified version of the QMR algorithm without look-ahead is also presented, and the special case of QMR for complex symmetric linear systems is considered. Results of numerical experiments comparing the original and the new implementations of the QMR method are reported.

Key words. Krylov subspace iteration, quasi-minimal residual method, non-Hermitian matrices, coupled two-term recurrences, look-ahead techniques, complex symmetric matrices

AMS(MOS) subject classifications. 65F10, 65N22

1. Introduction.

Recently, we have proposed a new Krylov subspace iteration, the quasi-minimal residual algorithm (QMR) [9], for solving general nonsingular non-Hermitian systems of linear equations

$$Ax = b. \tag{1.1}$$

The QMR method is closely related to the classical biconjugate gradient algorithm (BCG) due to Lanczos [14]. The BCG method aims at generating approximate solutions for (1.1) that satisfy a Galerkin condition. Unfortunately, for non-Hermitian matrices A, such iterates need not always exist, and this is the source of one of the two possible breakdowns—triggered by division by 0—that can occur during each iteration step of BCG. The second breakdown is equivalent to the possible breakdown—also triggered by division by 0—of the nonsymmetric Lanczos process [13]. In finite precision arithmetic, it is unlikely that one encounters exact breakdowns in the BCG algorithm. However, near-breakdowns can occur, which can cause a build-up of round-off in successive iterations. Another problem with BCG is the lack of a residual minimization

* This work was supported by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

† AT&T Bell Laboratories, Room 2C–420, 600 Mountain Avenue, P.O. Box 636, Murray Hill, New Jersey 07974–0636. This research was performed while this author was in residence at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, California 94035.

‡ RIACS, Mail Stop T041–5, NASA Ames Research Center, Moffett Field, California 94035.

property for its iterates, which leads to a typically erratic convergence behavior, with wild oscillations in the residual norm.

The QMR method offers remedies for these problems. It generates iterates that are defined by a quasi-minimization of the residual norm, rather than a Galerkin condition. This eliminates the oscillations and leads to a smooth and nearly monotone convergence behavior. In contrast to BCG, a QMR iterate always exists at each iteration step, and this excludes breakdowns caused by non-existent iterates. Moreover, possible breakdowns in the underlying Lanczos process are prevented by using look-ahead techniques. Therefore, except for the rare event of an incurable breakdown, breakdowns cannot occur in the QMR method.

In the original QMR algorithm [9], an implementation of the Lanczos method with look-ahead is used to generate basis vectors for the underlying Krylov subspaces. In the Lanczos process, these basis vectors are generated by means of three-term recurrences. It has been observed that, in finite precision arithmetic, vector iterations based on three-term recursions are usually less robust than mathematically equivalent coupled two-term vector recurrences. In this paper, we present a general look-ahead algorithm based on coupled two-term recursions for constructing basis vectors of Krylov subspaces. Based on this algorithm we then propose a new implementation of the QMR method.

The remainder of the paper is organized as follows. In § 2, we briefly review the Lanczos process and the original QMR algorithm. In § 3, we present a sketch of the proposed look-ahead procedure for constructing Lanczos vectors by means of coupled two-term recurrences. In § 4, we discuss the look-ahead strategy for this algorithm, and in § 5, we give some implementation details. Next we combine the coupled two-term procedure with the QMR approach. In § 6, we outline the resulting implementation for the general case of QMR with look-ahead. In § 7, we present a simplified version of the QMR algorithm without look-ahead. In § 8, we consider a variant of QMR for the special case of complex symmetric linear systems. In § 9, we report results of numerical experiments comparing the original and the new implementations of the QMR method. Finally, in § 10, we make some concluding remarks.

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, $M^T = \left[m_{kj}\right]$ and $M^H = \left[\overline{m}_{kj}\right]$ denote the transpose and the conjugate transpose, respectively, of the matrix $M = \left[m_{jk}\right]$. We use $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$ for the largest and smallest singular value of M, respectively. The vector norm $\|x\| := \sqrt{x^H x}$ is always the Euclidean norm and $\|M\| := \sigma_{\max}(M)$ is the corresponding matrix norm. We denote by

$$\mathcal{P}_n := \left\{ \varphi(\lambda) \equiv \gamma_0 + \gamma_1 \lambda + \cdots + \gamma_n \lambda^n \;\middle|\; \gamma_0, \gamma_1, \cdots, \gamma_n \in \mathbb{C} \right\}$$

the set of all complex polynomials of degree at most n. The nth Krylov subspace of $\mathbb{C}^N$ generated by $c \in \mathbb{C}^N$ and the $N \times N$ matrix B is defined by

$$K_n(c, B) := \operatorname{span}\{c, Bc, \cdots, B^{n-1}c\},$$

and we will make use of the fact that

$$K_n(c, B) = \{\varphi(B)c \mid \varphi \in \mathcal{P}_{n-1}\}.$$

Furthermore, it is always assumed that A is an $N \times N$ matrix, singular or nonsingular.

Finally, we remark that, for complex matrices, there are two equivalent formulations of the Lanczos process, using either A^T or A^H. In this paper, we have chosen the formulation with A^T, for two reasons. First, it avoids complex conjugation of the scalars in some of the recurrence relations, and second, the recursions reduce immediately for the special case of complex symmetric matrices.

2. The QMR algorithm. In this section, we briefly describe the QMR method and its original implementation [9]. We remark that, in [9], QMR was proposed for nonsingular linear systems (1.1). Freund and Hochbruck [8] showed that the QMR method can also be applied to singular square systems, and that it always generates well-defined iterates. However, as discussed in [8], these iterates converge to a meaningful solution of (1.1) only for consistent systems with coefficient matrices of index 1. An important special case for which these conditions are satisfied is consistent singular systems with a one-dimensional null space, i.e.,

$$b \in \{Ax \mid x \in \mathbb{C}^N\} \quad \text{and} \quad \dim\{x \in \mathbb{C}^N \mid Ax = 0\} = 1. \tag{2.1}$$

In this paper, we always consider the QMR method for the general case of $N \times N$ linear systems, with singular or nonsingular coefficient matrices A.

2.1. Krylov subspace methods. Let $x_0 \in \mathbb{C}^N$ be an arbitrary initial guess for the linear system (1.1), and denote by $r_0 := b - Ax_0$ the corresponding residual vector. An iterative scheme for solving (1.1) is called a *Krylov subspace method* if, for any choice of x_0, it produces approximate solutions of the form

$$x_n \in x_0 + K_n(r_0, A), \quad n = 1, 2, \cdots. \tag{2.2}$$

Clearly, the design of a Krylov subspace algorithm consists of two main parts: the construction of suitable basis vectors for the Krylov subspaces $K_n(r_0, A)$ in (2.2), and the choice of the actual iterates x_n. The QMR method is an example of a Krylov subspace iteration, where the basis vectors are generated by means of the nonsymmetric Lanczos process, and the iterates are characterized by a quasi-minimal residual property. Next, we describe these two main ingredients of QMR.

2.2. The Lanczos process. The Lanczos method is started with two vectors,

$$v_1 = r_0/\rho_1, \quad \text{where} \quad \rho_1 = \|r_0\|, \tag{2.3}$$

and an arbitrary second starting vector

$$w_1 \in \mathbb{C}^N \quad \text{with} \quad \|w_1\| = 1. \tag{2.4}$$

It then produces two sequences of vectors

$$\{v_j\}_{j=1}^n \quad \text{and} \quad \{w_j\}_{j=1}^n \tag{2.5}$$

such that, for $n = 1, 2, \cdots,$

$$
\begin{aligned}
\operatorname{span}\{v_1, v_2, \cdots, v_n\} &= K_n(v_1, A), \\
\operatorname{span}\{w_1, w_2, \cdots, w_n\} &= K_n(w_1, A^T),
\end{aligned}
\tag{2.6}
$$

and the two sets are biorthogonal or block biorthogonal, i.e.,

$$
W_n^T V_n = D_n,
\tag{2.7}
$$

where D_n is a diagonal or block diagonal matrix. Here and in the sequel, we denote by

$$
V_n := [\, v_1 \quad v_2 \quad \cdots \quad v_n \,] \quad \text{and} \quad W_n := [\, w_1 \quad w_2 \quad \cdots \quad w_n \,]
\tag{2.8}
$$

the matrices containing the Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ as columns. We remark that the conditions (2.6)–(2.7) determine the vectors (2.5) only up to scaling. Throughout this paper, we always scale the Lanczos vectors to have unit length:

$$
\|v_n\| = \|w_n\| = 1, \quad n = 1, 2, \cdots .
\tag{2.9}
$$

The crucial point of the Lanczos process is that vectors satisfying (2.6)–(2.7) can be constructed by means of short vector recursions. In the classical Lanczos algorithm [13], the vectors are generated using simple three-term recurrences:

$$
\begin{aligned}
\tilde{v}_{n+1} &= A v_n - v_n \mu_n - v_{n-1} \nu_n, \\
\rho_{n+1} &= \|\tilde{v}_{n+1}\|, \quad v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1},
\end{aligned}
\tag{2.10}
$$

$$
\begin{aligned}
\tilde{w}_{n+1} &= A^T w_n - w_n \mu_n - w_{n-1} (\nu_n \rho_n / \xi_n), \\
\xi_{n+1} &= \|\tilde{w}_{n+1}\|, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1},
\end{aligned}
\tag{2.11}
$$

$$
\text{where} \quad \mu_n = w_n^T A v_n / w_n^T v_n, \quad \nu_n = \xi_n w_n^T v_n / w_{n-1}^T v_{n-1}.
\tag{2.12}
$$

In this case, the matrix D_n in (2.7) is diagonal and nonsingular:

$$
D_n = \operatorname{diag}(\delta_1, \delta_2, \cdots, \delta_n), \quad \text{where} \quad \delta_j := w_j^T v_j \neq 0.
\tag{2.13}
$$

Furthermore, we note that, using the notation introduced in (2.8), the recurrence relations (2.10)–(2.11) for the first $n+1$ Lanczos vectors $\{v_j\}_{j=1}^{n+1}$ and $\{w_j\}_{j=1}^{n+1}$ can be written compactly in matrix form:

$$
A V_n = V_{n+1} H_n,
\tag{2.14}
$$

$$
A^T W_n = W_{n+1} \Gamma_{n+1}^{-1} H_n \Gamma_n .
\tag{2.15}
$$

Here, H_n is an $(n+1) \times n$ tridiagonal matrix, and

$$
\Gamma_n := \operatorname{diag}(\gamma_1, \gamma_2, \cdots, \gamma_n), \quad \text{where} \quad \gamma_j := \begin{cases} 1 & \text{if} \quad j = 1, \\ \gamma_{j-1} \rho_j / \xi_j & \text{if} \quad j > 1, \end{cases}
\tag{2.16}
$$

is a diagonal scaling matrix with positive diagonal entries. Finally, for later use, we note that all subdiagonal elements of H_n are nonzero, and therefore

$$\operatorname{rank} H_n = n. \tag{2.17}$$

Unfortunately, in the classical Lanczos algorithm, breakdowns cannot be excluded. Indeed, by (2.12), division by 0 will occur during the construction of v_{n+1} and w_{n+1} if $w_n^T v_n = 0$, but $w_n \neq 0$ and $v_n \neq 0$. Parlett, Taylor, and Liu [15] were the first to devise a practical modification of the Lanczos procedure that uses look-ahead to skip over possible *exact breakdowns* ($w_n^T v_n = 0$) or *near-breakdowns* ($w_n^T v_n$ is nonzero, but small in some sense). The QMR algorithm is based on a different implementation of the look-ahead Lanczos method, recently developed by Freund, Gutknecht, and Nachtigal [7]. Next, we briefly sketch this look-ahead Lanczos procedure.

As in the classical Lanczos algorithm, two sequences of Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ are generated, starting with (2.3) and (2.4). Again, we will use the matrix notation V_n and W_n defined in (2.8). As before, these vectors satisfy (2.6)–(2.7), but now D_n is, in general, only a block diagonal matrix, with $l := l(n)$ square blocks of dimension h_j, $j = 1, 2, \cdots, l$, on the diagonal. More precisely, we have

$$D_n = \operatorname{diag}(D^{(1)}, D^{(2)}, \cdots, D^{(l)}), \quad \text{where} \quad D^{(j)} := (W^{(j)})^T V^{(j)}. \tag{2.18}$$

Here, the matrices $V^{(j)}$ and $W^{(j)}$, $j = 1, 2, \cdots, l$, are defined by partitioning V_n and W_n into blocks, according to the look-ahead steps taken:

$$V_n = [\, V^{(1)} \quad V^{(2)} \quad \cdots \quad V^{(l)} \,] \quad \text{and} \quad W_n = [\, W^{(1)} \quad W^{(2)} \quad \cdots \quad W^{(l)} \,]. \tag{2.19}$$

We remark that the matrices $V^{(j)}$ and $W^{(j)}$ are of size $N \times h_j$, and they contain as their columns the Lanczos vectors constructed in the jth look-ahead step. The integer h_j is called the length of the jth look-ahead step, and l in (2.18)–(2.19) is the number of look-ahead steps that have been performed during the first n steps of the Lanczos process. For later use, we introduce some further notation. For $j = 1, 2, \cdots$, we denote by n_j the index of the first vectors of the blocks $V^{(j)}$ and $W^{(j)}$ in (2.19); hence, we have

$$V^{(j)} = [\, v_{n_j} \quad v_{n_j+1} \quad \cdots \quad] \quad \text{and} \quad W^{(j)} = [\, w_{n_j} \quad w_{n_j+1} \quad \cdots \quad]. \tag{2.20}$$

Note that the indices n_j satisfy

$$1 =: n_1 < n_2 < \cdots < n_l \leq n < n_{l+1}. \tag{2.21}$$

The vectors v_{n_j} and w_{n_j} are called *regular*, while the remaining vectors in the blocks $V^{(j)}$ and $W^{(j)}$ are called *inner*. We remark that, in view of (2.7) and (2.18), the regular vectors are biorthogonal to all previous Lanczos vectors, i.e.,

$$w_i^T v_{n_j} = w_{n_j}^T v_i = 0 \quad \text{for all} \quad i = 1, 2, \cdots, n_j - 1, \tag{2.22}$$

while the inner vectors in the jth blocks $V^{(j)}$ and $W^{(j)}$ are biorthogonal to all Lanczos vectors from the previous blocks, but not necessarily to the Lanczos vectors in the jth blocks. Finally, we note that, in (2.18), the blocks $D^{(j)}$, $j = 1, 2, \cdots, l-1$, are all nonsingular, while the last block $D^{(l)}$ is nonsingular if $n_{l+1} = n+1$, i.e., if v_{n+1} and w_{n+1} are constructed as regular vectors.

In the look-ahead algorithm, the Lanczos vectors (2.5) are again generated using only short vector recurrences, which now involve vectors from the last two blocks $V^{(l)}$, $V^{(l-1)}$ and $W^{(l)}$, $W^{(l-1)}$, instead of just v_n, v_{n-1} and w_n, w_{n-1}, as in the classical algorithm. For example, v_{n+1} is computed by means of the relations

$$\begin{aligned} \tilde{v}_{n+1} &= Av_n - V^{(l)}\mu_n - V^{(l-1)}\nu_n, \\ \rho_{n+1} &= \|\tilde{v}_{n+1}\|, \quad v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \end{aligned} \tag{2.23}$$

where $\mu_n \in \mathbb{C}^{h_l}$ and $\nu_n \in \mathbb{C}^{h_{l-1}}$ are suitable coefficient vectors. The second Lanczos vector w_{n+1} is obtained similarly. As before, the recurrence relations for the Lanczos vectors can be summarized in the form (2.14)–(2.15). Here, H_n is now a block tridiagonal matrix with l square blocks on the diagonal, where the jth block has dimension $h_j \times h_j$, $j = 1, 2, \cdots, l$. In addition, H_n is also an upper Hessenberg matrix. Furthermore, (2.16) and (2.17) remain valid, and ρ_j and ξ_j in (2.16) are scaling factors used to ensure the normalization (2.9) of the Lanczos vectors, see (2.23).

We would like to stress that the look-ahead strategy (see [7] and also § 4 below) is such that the algorithm performs mostly standard Lanczos steps, i.e., look-ahead steps of length $h_j = 1$ with blocks $V^{(j)}$ and $W^{(j)}$ that consist of only single Lanczos vectors. True look-ahead steps, i.e., steps of size $h_j > 1$, are only used to avoid exact and near-breakdowns. Typically, except for contrived examples, only few true look-ahead steps occur, and their size is usually small, mostly $h_j = 2$. We note that, if only steps of length $h_j = 1$ are performed, then the look-ahead Lanczos algorithm reduces to the classical algorithm. Finally, we remark that so-called incurable breakdowns [18, 12] can occur in the Lanczos process. Such breakdowns cannot be remedied by look-ahead, and indeed, in such a case, the look-ahead Lanczos algorithm would build a block of size $h_l = \infty$. Fortunately, incurable breakdowns are extremely rare, and they do not present a problem in practice.

The look-ahead Lanczos algorithm is intimately connected with formally orthogonal polynomials, see, e.g., [12, 7]. In particular, we will use the fact that each pair of Lanczos vectors v_j and w_j can be expressed in the form

$$v_j = \phi_{j-1}(A)v_1 \quad \text{and} \quad w_j = \gamma_j \phi_{j-1}(A^T)w_1, \tag{2.24}$$

where $\phi_{j-1} \in \mathcal{P}_{j-1}$ is of exact degree $j-1$ and $\gamma_j > 0$ is defined in (2.16).

For further details and properties of the look-ahead Lanczos algorithm, we refer the reader to [7].

2.3. The quasi-minimal residual property. In the QMR method, the vectors $\{v_j\}_{j=1}^n$ generated by the look-ahead Lanczos algorithm are used as a basis for the Krylov subspace $K_n(r_0, A)$ in (2.2). The nth QMR iterate x_n is then defined by

$$x_n = x_0 + V_n z_n, \tag{2.25}$$

where $z_n \in \mathbb{C}^n$ is the unique solution of the least squares problem

$$\|f_{n+1} - \Omega_{n+1} H_n z_n\| = \min_{z \in \mathbb{C}^n} \|f_{n+1} - \Omega_{n+1} H_n z\|. \tag{2.26}$$

Here

$$f_{n+1} := \omega_1 \rho_1 \cdot [\,1 \quad 0 \quad \cdots \quad 0\,]^T \in \mathbb{R}^{n+1}, \tag{2.27}$$

with ρ_1 given in (2.3), and

$$\Omega_{n+1} := \operatorname{diag}(\omega_1, \omega_2, \cdots, \omega_{n+1}), \quad \omega_j > 0, \quad j = 1, 2, \cdots, n+1, \tag{2.28}$$

is an arbitrary diagonal weighting matrix. Note that, in view of (2.17) and (2.28), the $(n+1) \times n$ matrix $\Omega_{n+1} H_n$ has full rank n. This guarantees that there always exists a unique solution of (2.26). Furthermore, we remark that the standard choice for the weights in (2.28) is

$$\omega_j = 1 \quad \text{for all} \quad j. \tag{2.29}$$

However, there are instances (see [10]) where the use of different weights is crucial, and therefore, we formulate the QMR method in the general setting (2.28).

From (2.25), (2.14), (2.27), and (2.3), it follows that the residual vector $r_n := b - Ax_n$ corresponding to x_n satisfies

$$r_n = V_{n+1} \Omega_{n+1}^{-1} \left(f_{n+1} - \Omega_{n+1} H_n z_n \right). \tag{2.30}$$

Hence, in view of (2.26), the nth QMR iterate x_n is characterized by a minimization of the second factor in (2.30); this is just the *quasi-minimal residual* property. The relation (2.30) shows that the scaling (2.29) is very natural, in the sense that all columns of $V_{n+1}\Omega_{n+1}^{-1}$ in the representation (2.30) of r_n are treated equally. We remark that the QMR iterates x_n can be easily updated from step to step. Due to the block tridiagonal structure of H_n, this update can be implemented with only short recurrences; see [9] for details. Finally, we note that the quasi-minimal residual property can be used to derive convergence results for the QMR method; we refer the reader to [9, 6].

3. A coupled two-term procedure with look-ahead. In this section, we consider a different approach to constructing the Lanczos vectors. The basic idea is to break up the three-term recurrences in the Lanczos process into coupled two-term recurrences, by using—in addition to the Lanczos vectors—a suitable second set of basis vectors for the underlying Krylov subspaces. In § 9, we will illustrate that QMR based on this coupled two-term procedure has better numerical properties than the original implementation of QMR based on three-term recurrences.

3.1. The general setting. In the following, let $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ always denote the sequence of vectors generated by the look-ahead Lanczos algorithm, as described in § 2.2. We assume that we are also given a second set of basis vectors

$$\{p_j\}_{j=1}^n \quad \text{and} \quad \{q_j\}_{j=1}^n \tag{3.1}$$

for the Krylov subspaces $K_n(v_1, A)$ and $K_n(w_1, A^T)$. More precisely, we consider vectors (3.1) that—in analogy to (2.24)—are of the form

$$p_j = \psi_{j-1}(A)v_1 \quad \text{and} \quad q_j = \gamma_j \psi_{j-1}(A^T)w_1, \tag{3.2}$$

where $\gamma_j > 0$ is given by (2.16), and $\psi_{j-1} \in \mathcal{P}_{j-1}$ is of exact degree $j-1$ with the same leading coefficient as the polynomial ϕ_{j-1} in (2.24). To distinguish between the two bases, we will often refer to the Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ as the *V–W sequence* and to the vectors (3.1) as the *P–Q sequence*.

From (2.24) and (3.2), we conclude that, for each $n = 1, 2, \cdots$,

$$p_n = v_n - \sum_{i=1}^{n-1} p_i u_{in}, \tag{3.3}$$

$$q_n = w_n - \sum_{i=1}^{n-1} q_i u_{in}(\gamma_n/\gamma_i), \tag{3.4}$$

with suitable coefficients $u_{in} \in \mathbb{C}$, $i = 1, 2, \cdots, n-1$. Similarly, in view of (2.23), (2.24), (3.2), and (2.16), we have

$$\tilde{v}_{n+1} = Ap_n - \sum_{i=1}^{n} v_i l_{in}, \quad \rho_{n+1} = \|\tilde{v}_{n+1}\|, \quad v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \tag{3.5}$$

$$\tilde{w}_{n+1} = A^T q_n - \sum_{i=1}^{n} w_i l_{in}(\gamma_n/\gamma_i), \quad \xi_{n+1} = \|\tilde{w}_{n+1}\|, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}, \tag{3.6}$$

with suitable coefficients $l_{in} \in \mathbb{C}$, $i = 1, 2, \cdots, n$. Note that (3.3)–(3.6) are coupled recurrences for generating the P–Q and V–W sequences: first, p_n and q_n are computed by means of (3.3)–(3.4), and then, the next Lanczos pair v_{n+1} and w_{n+1} is obtained from (3.5)–(3.6). Of course, it remains to specify the actual choice of the P–Q sequence. In order to minimize work and storage of previous vectors, the goal here is to select these vectors such that the recurrences (3.3)–(3.6) are as short as possible.

It will be convenient to use—in addition to (2.8)—the notation

$$P_n := [\, p_1 \quad p_2 \quad \cdots \quad p_n \,] \quad \text{and} \quad Q_n := [\, q_1 \quad q_2 \quad \cdots \quad q_n \,].$$

The recurrences (3.3)–(3.6) for the vectors $\{p_j\}_{j=1}^n$, $\{q_j\}_{j=1}^n$, $\{v_j\}_{j=1}^{n+1}$, and $\{w_j\}_{j=1}^{n+1}$ can then be written compactly in matrix form:

$$V_n = P_n U_n, \qquad AP_n = V_{n+1} L_n, \tag{3.7}$$

$$W_n = Q_n \Gamma_n^{-1} U_n \Gamma_n, \quad A^T Q_n = W_{n+1} \Gamma_{n+1}^{-1} L_n \Gamma_n. \tag{3.8}$$

Here, U_n is an upper triangular matrix and L_n is an upper Hessenberg matrix given by

$$U_n := \begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad \text{and} \quad L_n := \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1n} \\ \rho_2 & l_{22} & & \vdots \\ 0 & \rho_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & l_{nn} \\ 0 & \cdots & 0 & \rho_{n+1} \end{bmatrix}, \tag{3.9}$$

respectively, and Γ_n is the diagonal matrix defined in (2.16). Note that, by eliminating P_n in (3.7), we obtain

$$AV_n = V_{n+1} L_n U_n. \tag{3.10}$$

By comparing (3.10) with (2.14), it follows that

$$H_n = L_n U_n, \tag{3.11}$$

i.e., the matrices (3.9) define a factorization of the block tridiagonal Hessenberg matrix H_n generated by the look-ahead Lanczos algorithm.

Recall from (2.7) and (2.18) that the Lanczos vectors are block biorthogonal. These biorthogonality relations determine the coefficients l_{in} in (3.5)–(3.6). For example, consider the case that v_{n+1} and w_{n+1} are constructed as regular vectors. Then, in view of (2.22), we have the condition $W_n^T v_{n+1} = 0$, which, by (3.5), is equivalent to:

$$0 = W_n^T A p_n - \sum_{i=1}^{n} W_n^T v_i l_{in}. \tag{3.12}$$

Using (2.7), (2.8), and the first equation in (3.8), we deduce from (3.12) that

$$\begin{bmatrix} l_{1n} \\ \vdots \\ l_{nn} \end{bmatrix} = D_n^{-1} \Gamma_n U_n^T \Gamma_n^{-1} Q_n^T A p_n. \tag{3.13}$$

Recall from (2.18), (2.16), and (3.9) that the matrices D_n^{-1}, Γ_n, and U_n^T are block diagonal, diagonal, and lower triangular, respectively. Hence the relation (3.13) implies that the vector $Q_n^T A p_n$ determines the length of the recurrences (3.5)–(3.6). In particular, in order to obtain recursions that are as short as possible, the P–Q sequence should be chosen such that the vector $Q_n^T A p_n$ has as many leading zeros as possible. The same conclusion also holds for the case that v_{n+1} and w_{n+1} are constructed as inner vectors.

Motivated by this discussion, we require that the vectors in the P–Q sequence be A-biorthogonal or block A-biorthogonal, in the sense that the matrix

$$E_n := Q_n^T A P_n \tag{3.14}$$

should be diagonal or block diagonal. Note that the vector $Q_n^T A p_n$ is just the nth column of E_n. Furthermore, we remark that $q_n^T A P_n$, the nth row of E_n, has the same zero structure as $Q_n^T A p_n$. This is a consequence of relation (3.16) in the following lemma, which we will also need later on.

LEMMA 3.1. *Let* $\{v_i\}_{i=1}^n$, $\{w_i\}_{i=1}^n$ *and* $\{p_i\}_{i=1}^n$, $\{q_i\}_{i=1}^n$ *be vectors satisfying* (2.24) *and* (3.2), *respectively, and let* $\Gamma_n = \operatorname{diag}(\gamma_1, \gamma_2, \cdots, \gamma_n)$. *Let* D_n *and* E_n *be the matrices given by* (2.7) *and* (3.14), *respectively. Then*:

$$D_n \Gamma_n = (D_n \Gamma_n)^T, \tag{3.15}$$

$$E_n \Gamma_n = (E_n \Gamma_n)^T. \tag{3.16}$$

Proof. Using the polynomial representation (2.24) for the V–W vectors and the fact that polynomials in a matrix commute, we obtain

$$w_i^T v_j \gamma_j = \gamma_i w_1^T \phi_{i-1}(A) \phi_{j-1}(A) v_1 \gamma_j = \gamma_j w_1^T \phi_{j-1}(A) \phi_{i-1}(A) v_1 \gamma_i = w_j^T v_i \gamma_i,$$

and thus (3.15). The relation (3.16) follows similarly. □

3.2. The coupled algorithm without look-ahead. Next, we briefly consider the case that the V–W sequence consists of the vectors generated by the classical Lanczos algorithm without look-ahead. Recall from (2.13) that here the matrix D_n in (2.7) is diagonal, and that the Lanczos vectors are biorthogonal:

$$w_j^T v_n = \begin{cases} 0 & \text{if} \quad j \neq n, \\ \delta_n \neq 0 & \text{if} \quad j = n. \end{cases} \tag{3.17}$$

Suppose that it is possible to construct the vectors in the P–Q sequence such that the matrix E_n in (3.14) is also diagonal:

$$E_n = \operatorname{diag}(\epsilon_1, \epsilon_2, \cdots, \epsilon_n), \quad \text{where} \quad \epsilon_j := q_j^T A p_j \neq 0. \tag{3.18}$$

By (3.14) and (3.18), the P–Q vectors are then A-biorthogonal:

$$q_j^T A p_n = \begin{cases} 0 & \text{if} \quad j \neq n, \\ \epsilon_n \neq 0 & \text{if} \quad j = n. \end{cases} \tag{3.19}$$

With (2.13), (3.9), and (3.19), we deduce from (3.13) that

$$l_{in} = 0, \quad i = 1, 2, \cdots, n-1, \quad \text{and} \quad l_{nn} = \beta_n := \epsilon_n / \delta_n. \tag{3.20}$$

Furthermore, by multiplying (3.3) from the left by $q_j^T A$ and by using (3.19), (3.6), and (3.17), we obtain that, for $j = 1, 2, \cdots, n-1$,

$$\begin{aligned} 0 = q_j^T A p_n &= q_j^T A v_n - \epsilon_j u_{jn} \\ &= (A^T q_j)^T v_n - \epsilon_j u_{jn} \\ &= \xi_{j+1} w_{j+1}^T v_n - \epsilon_j u_{jn}. \end{aligned} \tag{3.21}$$

With (3.17), it follows from (3.21) that

$$u_{in} = 0, \quad i = 1, 2, \cdots, n-2, \quad \text{and} \quad u_{n-1,n} = \xi_n \delta_n / \epsilon_{n-1}. \tag{3.22}$$

In view of (3.20) and (3.22), all but the last terms vanish in each of the sums in (3.3)–(3.6), and hence (3.3)–(3.6) reduces to a coupled two-term procedure for generating the P–Q and V–W sequences.

The nth iteration of the resulting algorithm can be summarized as follows.

ALGORITHM 3.2. (nth iteration of the coupled algorithm without look-ahead)

1) *If* $\epsilon_{n-1} = 0$, *then stop.*
   Otherwise, compute $\delta_n = w_n^T v_n$.
   If $\delta_n = 0$, *then stop.*
2) *Compute*

$$p_n = v_n - p_{n-1}(\xi_n \delta_n / \epsilon_{n-1}),$$
$$q_n = w_n - q_{n-1}(\rho_n \delta_n / \epsilon_{n-1}).$$

3) *Compute* $\epsilon_n = q_n^T A p_n$, $\beta_n = \epsilon_n / \delta_n$, *and set*

$$\tilde{v}_{n+1} = A p_n - v_n \beta_n, \quad \rho_{n+1} = \|\tilde{v}_{n+1}\|,$$
$$\tilde{w}_{n+1} = A^T q_n - w_n \beta_n, \quad \xi_{n+1} = \|\tilde{w}_{n+1}\|.$$

4) *If* $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$, *then stop.*
   Otherwise, set

$$v_{n+1} = \tilde{v}_{n+1} / \rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1} / \xi_{n+1}.$$

We remark that the vectors in the P–Q and V–W sequences are, up to scaling, just the search directions and the residual vectors generated by the BCG method [14, 4]. In particular, Algorithm 3.2 can be viewed as the nth iteration of a rescaled version of BCG, where the computation of the BCG iterates is omitted.

In exact arithmetic, one of the termination checks in steps 1) or 4) of the coupled two-term procedure will be satisfied after at most N iterations. Normally, the algorithm stops due to $\rho_{n+1} = 0$ or $\xi_{n+1} = 0$, and then the procedure has constructed a basis for the invariant subspaces $K_n(v_1, A)$ or $K_n(w_1, A^T)$, respectively. This is called regular termination. If $\epsilon_{n-1} = 0$ or $\delta_n = 0$ occurs, then the algorithm has to be stopped to avoid division by 0. This is referred to as an exact breakdown. Recall from § 2.2 that $\delta_n = 0$ signals a breakdown in the classical Lanczos algorithm. Note that, like BCG, the coupled two-term procedure now has a second source of breakdown, namely the case $\epsilon_{n-1} = 0$. It can be shown that the condition $\epsilon_{n-1} = 0$ corresponds to a breakdown in the BCG algorithm due to an nth iterate not being defined by the Galerkin condition.

In finite precision arithmetic, exact breakdowns are rather unlikely. However, near-breakdowns, where δ_n or ϵ_{n-1} is nonzero, but small in some sense, may occur, leading to numerical instabilities in subsequent iterations. Next, we sketch a coupled two-term procedure that uses look-ahead in the construction of both the V–W and P–Q sequences to avoid exact and near-breakdowns.

3.3. The general algorithm with look-ahead. For describing the look-ahead in the V–W sequence, we will use the notations (2.18)–(2.21) introduced in § 2.2. In particular, the integer $l := l(n)$ denotes the number of look-ahead steps that have been performed during the construction of the first n vectors $\{v_i\}_{i=1}^n$ and $\{w_i\}_{i=1}^n$ in the

V–W sequence, and the n_j's in (2.21) are the indices of the regular vectors. Recall that, by (2.7) and (2.18), the V–W vectors satisfy the block biorthogonality conditions

$$(W^{(i)})^T V^{(j)} = \begin{cases} 0 & \text{if} \quad i \neq j, \\ D^{(j)} & \text{if} \quad i = j, \end{cases} \qquad i, j = 1, 2, \cdots, l. \tag{3.23}$$

Here, the blocks $D^{(j)}$ are all nonsingular, except for possibly $D^{(l)}$. However, we have that necessarily

$$D^{(l)} \quad \text{is nonsingular, if } n_{l+1} = n + 1. \tag{3.24}$$

Next, we introduce similar notations for describing the look-ahead in the P–Q sequence. We denote by $k := k(n)$ the number of look-ahead steps that have been performed during the construction of the first $n-1$ vectors $\{p_i\}_{i=1}^{n-1}$ and $\{q_i\}_{i=1}^{n-1}$ in the P–Q sequence. In analogy to (2.19), we partition these vectors into blocks, according to the look-ahead steps taken:

$$P_{n-1} = [\, P^{(1)} \quad P^{(2)} \quad \cdots \quad P^{(k)} \,] \quad \text{and} \quad Q_{n-1} = [\, Q^{(1)} \quad Q^{(2)} \quad \cdots \quad Q^{(k)} \,]. \tag{3.25}$$

Recall from (3.14) that the P–Q vectors are constructed to be block A-biorthogonal. More precisely, we have

$$E_{n-1} = \operatorname{diag}(E^{(1)}, E^{(2)}, \cdots, E^{(k)}),$$

or, equivalently,

$$(Q^{(i)})^T A P^{(j)} = \begin{cases} 0 & \text{if} \quad i \neq j, \\ E^{(j)} & \text{if} \quad i = j, \end{cases} \qquad i, j = 1, 2, \cdots, k. \tag{3.26}$$

Here, the blocks $E^{(j)}$ are nonsingular, except for possibly the last block $E^{(k)}$. In analogy to (2.21), we denote by m_j the indices of the first vectors of the blocks $P^{(j)}$ and $Q^{(j)}$ in (3.25). Hence, for $j = 1, 2, \cdots, k$, we have

$$P^{(j)} = [\, p_{m_j} \quad p_{m_j+1} \quad \cdots \quad] \quad \text{and} \quad Q^{(j)} = [\, q_{m_j} \quad q_{m_j+1} \quad \cdots \quad]. \tag{3.27}$$

Furthermore, the indices m_j satisfy

$$1 =: m_1 < m_2 < \cdots < m_k < n \leq m_{k+1}. \tag{3.28}$$

We remark that, by (3.26) and (3.27), the vectors p_{m_j} and q_{m_j} are A-biorthogonal to all previous P–Q vectors, i.e.,

$$q_i^T A p_{m_j} = q_{m_j}^T A p_i = 0 \quad \text{for all} \quad i = 1, 2, \cdots, m_j - 1.$$

Therefore, using the same notation as for the V–W sequence, we refer to the vectors p_{m_j} and q_{m_j} as *regular* vectors, while the remaining vectors in (3.27) are called *inner*. Finally, it turns out that, in (3.26), the last block $E^{(k)}$ has to be nonsingular, if p_n and q_n are constructed as regular vectors. This means that, in analogy to (3.24),

$$E^{(k)} \quad \text{is nonsingular, if } m_{k+1} = n. \tag{3.29}$$

After these preliminaries, we can now sketch the actual algorithm. Let $n \geq 1$, and assume that we have already generated the first $n-1$ vectors $\{p_i\}_{i=1}^{n-1}$ and $\{q_i\}_{i=1}^{n-1}$ of the P–Q sequence, and the first n vectors $\{v_i\}_{i=1}^{n}$ and $\{w_i\}_{i=1}^{n}$ of the V–W sequence.

First, the next pair of P–Q vectors, p_n and q_n, is constructed, using the recurrences (3.3)–(3.4). Here, the coefficients u_{in} need to be chosen such that p_n satisfies the corresponding A-biorthogonality conditions (3.26). We note that, in view of (3.16), the A-biorthogonality relations for the vector q_n are then also fulfilled. With (3.6) and (3.23), one readily verifies that

$$q_i^T A v_n = (A^T q_i)^T v_n = 0 \quad \text{for all} \quad i = 1, 2, \cdots, n_l - 2.$$

By rewriting this in terms of the blocks $Q^{(j)}$, we obtain that

$$(Q^{(j)})^T A v_n = 0 \quad \text{for all} \quad j = 1, 2, \cdots, k^\star - 1, \tag{3.30}$$

where $k^\star := k^\star(n)$ is given by

$$k^\star = \max\Big\{ j \;\Big|\; 1 \leq j \leq k \;\text{ and }\; m_j \leq \max\{1, n_l - 1\} \Big\}. \tag{3.31}$$

The orthogonality conditions (3.30), together with (3.26), imply that, in (3.3), we have $u_{in} = 0$ for all $i < m_{k^\star}$. Thus the recurrences (3.3)–(3.4) reduce to

$$p_n = v_n - \sum_{i=m_{k^\star}}^{n-1} p_i u_{in}, \tag{3.32}$$

$$q_n = w_n - \sum_{i=m_{k^\star}}^{n-1} q_i u_{in} (\gamma_n / \gamma_i). \tag{3.33}$$

Now we need to determine the coefficients u_{in} for $m_{k^\star} \leq i \leq n-1$. This is done by enforcing the remaining A-biorthogonality conditions (3.26) for p_n and the vectors q_i, namely

$$q_i^T A p_n = 0 \quad \text{for all} \quad i = m_{k^\star}, m_{k^\star} + 1, \cdots, m^\star. \tag{3.34}$$

Here, $m^\star := n-1$ if p_n and q_n are constructed as regular vectors, and $m^\star := m_k - 1$ otherwise. Note that, in view of (3.28), we always have $m_k - 1 \leq m^\star$. First, we consider (3.34) for the indices i in the range $m_{k^\star} \leq i \leq m_k - 1$. Using (3.32) and (3.26), we deduce from (3.34) that

$$U_{m_{k^\star}:m_k-1,n} = (\operatorname{diag}(\, E^{(k^\star)} \quad \cdots \quad E^{(k-1)}\,))^{-1} \,[\, Q^{(k^\star)} \quad \cdots \quad Q^{(k-1)}\,]^T A v_n. \tag{3.35}$$

Here and in the sequel, we use the notation

$$M_{i:j,n} := [\, m_{in} \quad m_{i+1,n} \quad \cdots \quad m_{jn}\,]^T$$

for vectors consisting of successive elements of the nth column of the matrix $M = [\, m_{ij}\,]$. If p_n and q_n are constructed as regular vectors, then we also have to ensure that (3.34) holds for i with $m_k \leq i \leq n-1$, and this gives

$$U_{m_k:n-1,n} = (E^{(k)})^{-1} (Q^{(k)})^T A v_n. \tag{3.36}$$

We remark that, by (3.29), the matrix $E^{(k)}$ in (3.36) is necessarily nonsingular since the vectors p_n and q_n are regular. If p_n and q_n are constructed as inner vectors, then $m^\star = m_k - 1$, and (3.34) yields no conditions for the choice of the coefficients u_{in} with $m_k \le i < n-1$. In this case, we set

$$u_{in} = \zeta_{in} \quad \text{for} \quad i = m_k, m_k+1, \cdots, n-1, \tag{3.37}$$

where $\zeta_{in} \in \mathbb{C}$ can be chosen arbitrarily. Finally, if p_n and q_n are regular, we update the "regular" indices (3.28) by setting $m_{k+1} := n$ and $k := k+1$. This completes the construction of the p_n and q_n vectors.

In a second step, we now compute the next pair of V–W vectors, v_{n+1} and w_{n+1}, using the recurrences (3.5)–(3.6) Here, we have to determine the coefficients l_{in} such that v_{n+1} satisfies the corresponding biorthogonality conditions (3.23). Note that, in view of (3.15), the relations (3.23) for w_{n+1} are then fulfilled automatically. We proceed similar to the construction of p_n and q_n. By using (3.26) and the fact that the columns of the matrices Q_i and W_i span the same space, it is easily verified that

$$(W^{(j)})^T A p_n = 0 \quad \text{for all} \quad j = 1, 2, \cdots, l^\star - 1, \tag{3.38}$$

where $l^\star := l^\star(n)$ is given by

$$l^\star = \max\left\{ j \;\middle|\; 1 \le j \le l \text{ and } n_j \le m_k \right\}. \tag{3.39}$$

From (3.38), we conclude that the recurrences (3.5)–(3.6) reduce as follows:

$$\tilde{v}_{n+1} = A p_n - \sum_{i=n_{l^\star}}^{n} v_i l_{in}, \tag{3.40}$$

$$\tilde{w}_{n+1} = A^T q_n - \sum_{i=n_{l^\star}}^{n} w_i l_{in} (\gamma_n / \gamma_i). \tag{3.41}$$

The recurrence coefficients l_{in} in (3.40)–(3.41) are determined by enforcing the remaining biorthogonality conditions (3.23) for the vectors v_{n+1} and w_i, $n_{l^\star} \le i \le n$. This gives

$$L_{n_{l^\star}:n_l-1,n} = (\operatorname{diag}(\, D^{(l^\star)} \quad \cdots \quad D^{(l-1)}\,))^{-1} \, [\, W^{(l^\star)} \quad \cdots \quad W^{(l-1)}\,]^T A p_n. \tag{3.42}$$

Moreover, if v_{n+1} and w_{n+1} are constructed as regular vectors, then

$$L_{n_l:n,n} = (D^{(l)})^{-1} (W^{(l)})^T A p_n. \tag{3.43}$$

Note that, by (3.24), the matrix $D^{(l)}$ in (3.43) is necessarily nonsingular since v_{n+1} and w_{n+1} are regular. If v_{n+1} and w_{n+1} are built as inner vectors, then we set

$$l_{in} = \eta_{in} \quad \text{for} \quad i = n_l, n_l+1, \cdots, n, \tag{3.44}$$

where $\eta_{in} \in \mathbb{C}$ can be chosen arbitrarily. Finally, if v_{n+1} and w_{n+1} are regular, then we update the indices (2.21) by setting $n_{l+1} := n+1$ and $l := l+1$.

The resulting coupled procedure for generating the P–Q and V–W sequences can be sketched as follows.

ALGORITHM 3.3. (Coupled algorithm with look-ahead)

0) *Choose* $v_1,\ w_1 \in \mathbb{C}^N$ *with* $\|v_1\| = \|w_1\| = 1$.
   Set $V^{(1)} = v_1,\ W^{(1)} = w_1,\ D^{(1)} = w_1^T v_1$.
   Set $k = 1,\ m_k = 1,\ l = 1,\ n_l = 1$.

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

1) *Determine* $k^\star$ *from* (3.31).
2) *Decide whether to construct* p_n *and* q_n *as regular or inner vectors and go to* 3) *or* 4), *respectively.*
3) *Compute* p_n *and* q_n *by means of* (3.35)–(3.36) *and* (3.32)–(3.33).
   Set $m_{k+1} = n,\ k = k+1,\ P^{(k)} = Q^{(k)} = \emptyset$ *and go to* 5).
4) *Compute* p_n *and* q_n *by means of* (3.35), (3.37), *and* (3.32)–(3.33).
5) *Set*

$$P^{(k)} = [\, P^{(k)} \quad p_n \,], \quad Q^{(k)} = [\, Q^{(k)} \quad q_n \,], \quad E^{(k)} = (Q^{(k)})^T A P^{(k)}.$$

6) *Determine* $l^\star$ *from* (3.39).
7) *Decide whether to construct* v_{n+1} *and* w_{n+1} *as regular or inner vectors and go to* 8) *or* 9), *respectively.*
8) *Compute* $\tilde{v}_{n+1}$ *and* $\tilde{w}_{n+1}$ *by means of* (3.42)–(3.43) *and* (3.40)–(3.41).
   Set $n_{l+1} = n+1,\ l = l+1,\ V^{(l)} = W^{(l)} = \emptyset$ *and go to* 10).
9) *Compute* $\tilde{v}_{n+1}$ *and* $\tilde{w}_{n+1}$ *by means of* (3.42), (3.44), *and* (3.40)–(3.41).
10) *Compute* $\rho_{n+1} = \|\tilde{v}_{n+1}\|$ *and* $\xi_{n+1} = \|\tilde{w}_{n+1}\|$.
   If $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$, *then stop.*
   Otherwise, set

$$\begin{aligned} &v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}, \\ &V^{(l)} = [\, V^{(l)} \quad v_{n+1} \,], \quad W^{(l)} = [\, W^{(l)} \quad w_{n+1} \,], \quad D^{(l)} = (W^{(l)})^T V^{(l)}. \end{aligned}$$

We remark that Algorithm 3.3 reduces to the coupled two-term procedure described in § 3.2 if all vectors in the P–Q and V–W sequences are built as regular vectors. Note that in this case, we have $k(n) = n-1$, $n_{k^\star} = n-1$, $l(n) = n$, and $n_{l^\star} = n$ for all n.

4. The look-ahead strategy. As described in § 3.2, there are two possible breakdowns in the coupled two-term procedure without look-ahead: one associated with the V–W sequence, and another associated with the P–Q sequence. In particular, Algorithm 3.2 will encounter an exact breakdown in the V–W sequence if $w_n^T v_n = 0$, or in the P–Q sequence if $q_{n-1}^T A p_{n-1} = 0$. The exact breakdowns of the two sequences are not independent of each other, as was pointed out by Gutknecht in [11]. For a full description of the structure and coupling of the exact breakdowns, we refer the reader to [11, 2] and the references given there. However, in practice one is also concerned with avoiding near-breakdowns, that is, situations when $w_n^T v_n$ or $q_{n-1}^T A p_{n-1}$ are not exactly zero, but are small in some sense.

In the coupled procedure with look-ahead, which we sketched in § 3.3, exact and near-breakdowns in the P–Q respectively V–W sequence are prevented by building the next pair of vectors p_n and q_n respectively v_{n+1} and w_{n+1} as inner vectors. In this section, we describe the look-ahead strategy that is used to decide in steps 2) and 7) of Algorithm 3.3 whether vectors are constructed as regular or inner vectors.

Recall from § 2.2 that the vectors $\{v_i\}_{i=1}^{n+1}$ and $\{w_i\}_{i=1}^{n+1}$ in the look-ahead Lanczos algorithm satisfy a block three-term recurrence that can be written compactly as (2.14)–(2.15). By eliminating V_n and W_n in (3.7) and (3.8), one obtains a similar recurrence relation for the vectors $\{p_i\}_{i=1}^{n}$ and $\{q_i\}_{i=1}^{n}$ of the P–Q sequence:

$$AP_{n-1} = P_n U_n L_{n-1} \quad \text{and} \quad A^T Q_{n-1} = Q_n \Gamma_n^{-1} U_n L_{n-1} \Gamma_{n-1}. \tag{4.1}$$

By using the A-biorthogonality of p_n and q_n, it is easy to show that the recurrences for the P–Q sequence are also three-term recurrences, or, in the general case, block three-term recurrences. We note that, in (4.1), the matrix $U_n L_{n-1}$ of recurrence coefficients is obtained by multiplying the factors L_{n-1} and U_n from the decompositions (3.11) of H_{n-1} and H_n, respectively, in reverse order. This was first remarked by Rutishauser [16] for the special case of no look-ahead, and recently by Gutknecht [11], for the general case.

The look-ahead strategy consists of monitoring breakdowns in the two sequences independently. For the V–W sequence, the criteria used are the same as those proposed in [7]:

$$\sigma_{\min}(D^{(l)}) \geq eps, \tag{4.2}$$

$$n(A) \geq \sum_i \left| \left(L_n U_n \right)_{in} \right|, \tag{4.3}$$

$$n(A) \geq \sum_i \frac{\gamma_n}{\gamma_i} \left| \left(L_n U_n \right)_{in} \right|, \tag{4.4}$$

where eps is machine epsilon, and $n(A)$ is an estimate for the norm of A. The Lanczos vectors v_{n+1} and w_{n+1} are built as regular vectors only if all three of the above checks are satisfied. The check (4.2) ensures that the diagonal blocks $D^{(j)}$ are nonsingular, while the checks (4.3)–(4.4) ensure that the size of the coefficients μ_n and ν_n in (2.23) and in the corresponding relation for $\tilde{w}_{n+1}$ do not exceed an estimate $n(A)$ for the norm of A. The first condition is needed since the inverse of $D^{(l)}$ appears in μ_n, while the second and third conditions attempt to ensure that the components from $K_n(r_0, A)$ and from $K_n(w_1, A^T)$ do not dominate the Av_n and $A^T w_n$ terms, respectively. Another motivation for these checks is as follows. The symmetric Lanczos process for Hermitian matrices A generates tridiagonal matrices H_n that satisfy

$$\|H_n\| \le \|A\| \quad \text{for all} \quad n. \tag{4.5}$$

For the classical nonsymmetric Lanczos algorithm, the relation (4.5) does not hold in general. Indeed, formally, we have $\|H_n\| = \infty$ if a breakdown occurs. As David Day pointed out to us[1], an "ideal" look-ahead Lanczos procedure would ensure that (4.5) also holds for non-Hermitian matrices. The criteria (4.3)–(4.4) can be viewed as a cheap way of modeling the conditions (4.5). We remark that the checks (4.3)–(4.4) take advantage of the normalization (2.9) of the Lanczos vectors.

For the P–Q sequence, the criteria are similar: the diagonal blocks $E^{(j)}$ must be nonsingular, and the size of the last columns of $U_n L_{n-1}$ and of $\Gamma_n^{-1} U_n L_{n-1} \Gamma_{n-1}$ must not exceed the estimate $n(A)$ for the norm of A. Singularity of $E^{(k)}$ is once again checked from its smallest singular value:

$$\sigma_{\min}(E^{(k)}) \ge eps.$$

However, for the second and third checks, it is no longer sufficient to compute just the norm of the last column of the matrices of recurrence coefficients, as the vectors p_n and q_n are not normalized to unit length. Instead, one must check

$$n(A)\|p_n\| \ge \sum_i \left| \left(U_n L_{n-1}\right)_{i,n-1} \right| \|p_i\|, \tag{4.6}$$

and

$$n(A)\|q_n\| \ge \sum_i \frac{\gamma_{n-1}}{\gamma_i} \left| \left(U_n L_{n-1}\right)_{i,n-1} \right| \|q_i\|. \tag{4.7}$$

This means that the look-ahead strategy for the P–Q sequence requires the computation of the two norms $\|p_n\|$ and $\|q_n\|$ at each step, work that would otherwise not be needed by the algorithm. Once again, the vectors p_n and q_n are built as regular vectors only if all three of the above checks are satisfied. We remark that the look-ahead strategy presented here builds regular vectors in preference to inner vectors, and will therefore build as few inner vectors as possible.

Finally, we note that other look-ahead strategies are also possible. For example, Gutknecht [11] proposed a look-ahead strategy that assumes that the near-breakdowns encountered in the two sequences have the same structure as the exact breakdowns. We have chosen to monitor the two sequences independently; nevertheless, our strategy will recover the exact-breakdown structure if only exact breakdowns are considered.

5. Implementation details. In this section, we present a detailed description of our implementation of the coupled two-term recurrence Lanczos algorithm with look-ahead. We are interested in obtaining an implementation that requires only two inner

[1] Private communication, Berkeley, March 1992.

products per iteration to compute all the coefficients of the recurrence formulas. Recall that the look-ahead strategy (4.6)–(4.7) for the P–Q sequence and the normalization (2.9) require a total of four norm computations, so that the implementation will require two inner products and four norms per iteration.

Let us denote by

$$G_{n-1} := U_n L_{n-1} \tag{5.1}$$

the matrix of recurrence coefficients for the three-term recurrences for P_n and Q_n in (4.1). Let us also introduce the auxiliary matrices

$$F_n := W_n^T A P_n \quad \text{and} \quad \tilde{F}_n := Q_n^T A V_n,$$

whose columns are needed in (3.35)–(3.36) and (3.42)–(3.43). It turns out that these two matrices are essentially the transpose of each other.

LEMMA 5.1. *The matrices F_n and $\tilde{F}_n$ satisfy:*

$$F_n \Gamma_n = (\tilde{F}_n \Gamma_n)^T.$$

Proof. The proof is similar to the proof of Lemma 3.1. □

The nth iteration of the implementation will update the matrices D_{n-1}, E_{n-1}, F_{n-1}, L_{n-1}, U_{n-1}, P_{n-1}, Q_{n-1}, V_n, and W_n, to D_n, E_n, F_n, L_n, U_n, P_n, Q_n, V_{n+1}, and W_{n+1}, respectively.

We first list an outline of the algorithm as we have implemented it. This is essentially the same as Algorithm 3.3, up to the order of the steps and more details.

ALGORITHM 5.2. (Coupled algorithm with look-ahead)

0) *Choose v_1, $w_1 \in \mathbb{C}^N$ with $\|v_1\| = \|w_1\| = 1$, and compute $w_1^T v_1$.*
Set $k = 1$, $m_k = 1$, $l = 1$, $n_l = 1$.

For $n = 1, 2, \cdots$, do:

1) *Update D_{n-1} to D_n.*
2) *Determine $k^\star$ from* (3.31):
$$k^\star = \max\left\{ j \;\middle|\; 1 \le j \le k \;\text{ and }\; m_j \le \max\{1, n_l - 1\} \right\}.$$
3) *Compute $F_{n,1:n-1}$ from* (5.2) *below, using L_{n-1} and $D_{n,1:n}$.*
Then compute $\tilde{F}_{1:n-1,n}$ by Lemma 5.1.
4) *Check whether $E^{(k)}$ is nonsingular:*
$$\texttt{innerp} = \sigma_{\min}(E^{(k)}) < eps.$$
5) *Compute the part of $U_{1:n,n}$ that is determined by biorthogonality:*
$$U_{m_i:m_{i+1}-1,n} = (E^{(i)})^{-1} (Q^{(i)})^T A v_n = (E^{(i)})^{-1} \tilde{F}_{m_i:m_{i+1}-1,n}, \quad i = k^\star, \cdots, k-1.$$

If `innerp`*, go to* 6)*. Otherwise, set*

$$U_{m_k:n-1,n} = (E^{(k)})^{-1}(Q^{(k)})^T A v_n = (E^{(k)})^{-1}\tilde{F}_{m_k:n-1,n}.$$

6) *Build the part of p_n and q_n that is common to both regular and inner vectors:*

$$\hat{p}_n = v_n - \sum_{i=m_{k^\star}}^{m_k-1} p_i u_{in},$$

$$\hat{q}_n = w_n - \sum_{i=m_{k^\star}}^{m_k-1} q_i u_{in}(\gamma_n/\gamma_i).$$

If `innerp`*, go to* 11)*.*

7) *Build and check the coefficient $G_{m_k:n-1,n-1}$. If* `innerp`*, go to* 11)*.*
8) *Build p_n and q_n as regular vectors:*

$$p_n = \hat{p}_n - \sum_{i=m_k}^{n-1} p_i u_{in},$$

$$q_n = \hat{q}_n - \sum_{i=m_k}^{n-1} q_i u_{in}(\gamma_n/\gamma_i).$$

Compute Ap_n, $q_n^T A p_n$, $\|p_n\|$, and $\|q_n\|$.

9) *Build and check the coefficient $G_{m_k:n-1,n}$. If* `innerp`*, go to* 11)*.*
10) *Set $m_{k+1} = n$, $k = k+1$, and go to* 12)*.*
11) *Choose the inner recurrence coefficients u_{in}, $i = m_k, \cdots, n-1$, and build p_n and q_n as inner vectors:*

$$p_n = \hat{p}_n - \sum_{i=m_k}^{n-1} p_i u_{in},$$

$$q_n = \hat{q}_n - \sum_{i=m_k}^{n-1} q_i u_{in}(\gamma_n/\gamma_i).$$

Compute Ap_n, $q_n^T A p_n$, $\|p_n\|$, and $\|q_n\|$.

12) *If $\|p_n\| = 0$, or $\|q_n\| = 0$, then stop.*
13) *Compute $A^T q_n$.*
14) *Update E_{n-1} to E_n.*
15) *Determine $l^\star$ from* (3.39)*:*

$$l^\star = \max\left\{ j \;\middle|\; 1 \le j \le l \;\; \text{and} \;\; n_j \le m_k \right\}.$$

16) *Compute* $F_{1:n,n}$ *from* (5.3) *below, using* E_n *and* U_n.

17) *Check whether* $D^{(l)}$ *is nonsingular*:

$$\texttt{innerv} = \sigma_{\min}(D^{(l)}) < eps.$$

18) *Compute the part of* $L_{1:n,n}$ *that is determined by biorthogonality*:

$$L_{n_i:n_{i+1}-1,n} = (D^{(i)})^{-1}(W^{(i)})^T Ap_n = (D^{(i)})^{-1}F_{n_i:n_{i+1}-1,n}, \quad i = l^\star, \cdots, l-1,$$

If `innerv`, *go to* 19). *Otherwise, set*

$$L_{n_l:n,n} = (D^{(l)})^{-1}(W^{(l)})^T Ap_n = (D^{(l)})^{-1}F_{n_l:n,n}.$$

19) *Build the part of* v_{n+1} *and* w_{n+1} *that is common to both regular and inner vectors*:

$$\hat{v}_{n+1} = Ap_n - \sum_{i=n_{l^\star}}^{n_l-1} v_i l_{in},$$

$$\hat{w}_{n+1} = A^T q_n - \sum_{i=n_{l^\star}}^{n_l-1} w_i l_{in}(\gamma_n/\gamma_i).$$

If `innerv`, *go to* 24).

20) *Build and check the coefficient* $H_{n_l:n,n}$. *If* `innerv`, *go to* 24).

21) *Build* v_{n+1} *and* w_{n+1} *as regular vectors*:

$$\tilde{v}_{n+1} = \hat{v}_{n+1} - \sum_{i=n_l}^{n} v_i l_{in},$$

$$\tilde{w}_{n+1} = \hat{w}_{n+1} - \sum_{i=n_l}^{n} w_i l_{in}(\gamma_n/\gamma_i).$$

Compute $\rho_{n+1} = l_{n+1,n} = \|\tilde{v}_{n+1}\|$, $\xi_{n+1} = \|\tilde{w}_{n+1}\|$.
If $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$, *then stop.*
Otherwise, set $\gamma_{n+1} = \gamma_n \rho_{n+1}/\xi_{n+1}$, *and compute* $\tilde{w}_{n+1}^T \tilde{v}_{n+1}$.

22) *Build and check the coefficient* $H_{n_l:n,n+1}$. *If* `innerv`, *go to* 24).

23) *Set* $n_{l+1} = n+1$, $l = l+1$, *and go to* 25).

24) *Choose the inner recurrence coefficients* l_{in}, $i = n_l, \cdots, n$, *and build* v_{n+1} *and* w_{n+1} *as inner vectors*:

$$\tilde{v}_{n+1} = \hat{v}_{n+1} - \sum_{i=n_l}^{n} v_i l_{in},$$

$$\tilde{w}_{n+1} = \hat{w}_{n+1} - \sum_{i=n_l}^{n} w_i l_{in}(\gamma_n/\gamma_i).$$

Compute $\rho_{n+1} = l_{n+1,n} = \|\tilde{v}_{n+1}\|$, $\xi_{n+1} = \|\tilde{w}_{n+1}\|$.

If $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$, *then stop.*

Otherwise, set $\gamma_{n+1} = \gamma_n \rho_{n+1}/\xi_{n+1}$, *and compute* $\tilde{w}_{n+1}^T \tilde{v}_{n+1}$.

25) *Set*

$$v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1},$$
$$w_{n+1}^T v_{n+1} = \tilde{w}_{n+1}^T \tilde{v}_{n+1}/(\rho_{n+1}\xi_{n+1}).$$

Step 1. The diagonal term $w_n^T v_n$ has already been computed directly, at the end of the previous step. Next, using

$$\begin{aligned} F_{n-1} &= W_{n-1}^T A P_{n-1} = W_{n-1}^T V_n L_{n-1} \\ &= D_{n-1} L_{1:n-1,1:n-1} + l_{n,n-1} D_{1:n-1,n} \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}, \end{aligned} \tag{5.2}$$

the remainder of the last column of D_n is computed from D_{n-1}, F_{n-1}, and L_{n-1}. The last row of D_n is obtained by symmetry, using (3.15) from Lemma 3.1.

Step 7. We build $G_{m_k:n-1,n-1}$, which would be the coefficient of the $P^{(k)}$ and $Q^{(k)}$ blocks in the three-term recurrences for p_n and q_n. Using (5.1), one has:

$$G_{i,n-1} = \sum_{j=i}^{n} u_{ij} l_{j,n-1}, \quad i = m_k, \cdots, n-1.$$

We then check (4.6)–(4.7), and set `innerp` to `TRUE` if at least one of the two checks fails.

Step 9. We build $G_{m_k:n-1,n}$, which would be the coefficient of the $P^{(k)}$ and $Q^{(k)}$ blocks in the three-term recurrences for p_{n+1} and q_{n+1}. It is straightforward to show that

$$G_{m_k:n-1,n} = (E^{(k)})^{-1} (Q^{(k)})^T A A p_n.$$

Moreover, we have

$$\begin{aligned} Q_{n-1}^T A A p_n &= \left(A^T Q_{n-1}\right)^T A p_n = \left(Q_n \Gamma_n^{-1} U_n L_{n-1} \Gamma_{n-1}\right)^T A p_n \\ &= \Gamma_{n-1} L_{n-1}^T U_n^T \Gamma_n^{-1} Q_n^T A p_n \\ &= \Gamma_{n-1} L_{n-1}^T U_n^T \Gamma_n^{-1} \begin{bmatrix} 0 & \cdots & 0 & q_n^T A p_n \end{bmatrix}^T \\ &= \frac{\gamma_{n-1}}{\gamma_n} l_{n,n-1} \begin{bmatrix} 0 & \cdots & 0 & q_n^T A p_n \end{bmatrix}^T. \end{aligned}$$

We then check (4.6)–(4.7), and set `innerp` to `TRUE` if at least one of the two checks fails.

Step 14. The diagonal term $q_n^T A p_n$ has already been computed directly, as part of Step 8) or Step 11). Next, using

$$F_n = W_n^T A P_n = \Gamma_n U_n^T \Gamma_n^{-1} Q_n^T A P_n, \tag{5.3}$$

the remainder of the last row of E_n is computed from E_{n-1}, $F_{1:n,1:n-1}$, and U_n. The last column of E_n is obtained by symmetry, using (3.16) from Lemma 3.1.

Step 20. We build $H_{n_l:n,n}$, which would be the coefficient of the $V^{(l)}$ and $W^{(l)}$ blocks in the three-term recurrences for v_{n+1} and w_{n+1}. Using (3.11), one has:

$$H_{in} = \sum_{j=i}^{n} l_{ij} u_{jn}, \quad i = n_l, \cdots, n.$$

We then check (4.3)–(4.4), and set `innerv` to `TRUE` if at least one of the two checks fails.

Step 22. We build $H_{n_l:n,n+1}$, which would be the coefficient of the $V^{(l)}$ and $W^{(l)}$ blocks in the three-term recurrences for v_{n+2} and w_{n+2}. It is straightforward to show that

$$H_{n_l:n,n+1} = (D^{(l)})^{-1} (W^{(l)})^T A v_{n+1}.$$

Moreover, we have

$$\begin{aligned} W_n^T A v_{n+1} &= \left(A^T W_n\right)^T A v_{n+1} = \left(W_{n+1} \Gamma_{n+1}^{-1} L_n U_n \Gamma_n\right)^T v_{n+1} \\ &= \Gamma_n U_n^T L_n^T \Gamma_{n+1}^{-1} W_{n+1}^T v_{n+1} \\ &= \Gamma_n U_n^T L_n^T \Gamma_{n+1}^{-1} \left[0 \quad \cdots \quad 0 \quad w_{n+1}^T v_{n+1}\right]^T \\ &= \frac{\gamma_n}{\gamma_{n+1}} l_{n+1,n} \left[0 \quad \cdots \quad 0 \quad w_{n+1}^T v_{n+1}\right]^T. \end{aligned}$$

We then check (4.3)–(4.4), and set `innerv` to `TRUE` if at least one of the two checks fails.

We remark that the checks in steps 7) and 9), and 20) and 22), are actually slightly relaxed versions of (4.3)–(4.4), and (4.6)–(4.7), respectively, since the indices checked are only a subset of the full range appearing in (4.3)–(4.4) and (4.6)–(4.7). We also note that the algorithm above requires minimal inputs from the user. Recall that *eps* in steps 4) and 17) is machine epsilon. Furthermore, the estimate $n(A)$ for the norm of the matrix can be updated dynamically, as was done in [7].

The coupled Lanczos Algorithm 5.2 requires per iteration the computation of two inner products and four vector norms. We conclude this section by noting that, in Algorithm 5.2, the choice of the inner recurrence coefficients (3.37) and (3.44) is arbitrary. In our implementation of the algorithm, we have used

$$\begin{aligned} u_{n-1,n} &= 1, \\ u_{n-2,n} &= 1, \quad \text{when} \quad m_k \le n-2, \\ u_{in} &= 0, \quad \text{for} \quad i = m_k, \cdots, n-3, \\ l_{nn} &= 1, \\ l_{n-1,n} &= 1, \quad \text{when} \quad n_l \le n-1, \\ l_{in} &= 0, \quad \text{for} \quad i = n_l, \cdots, n-2, \end{aligned}$$

for the inner vector recurrence coefficients.

6. An implementation of QMR with look-ahead. We now return to linear systems (1.1) and the QMR method. In this section, we propose an implementation of QMR method based on the coupled two-term look-ahead Algorithm 3.3.

Recall that the nth QMR iterate x_n is defined by (2.25)–(2.26) in terms of the matrices V_n and H_n generated by the look-ahead Lanczos algorithm. In the original implementation of QMR, the solution z_n of the least squares problem (2.26) is computed by means of a QR decomposition of the matrix $\Omega_{n+1}H_n$.

Here we consider the case that the Lanczos vectors are constructed using the coupled two-term Algorithm 3.3. Recall that Algorithm 3.3 yields as a by-product the factors L_n and U_n in the decomposition (3.11) of H_n. Using the factorization (3.11) and setting $y_n := U_n z_n$, we can rewrite the definition (2.25)–(2.26) of x_n as follows:

$$x_n = x_0 + V_n U_n^{-1} y_n, \tag{6.1}$$

where y_n is the unique solution of the least squares problem

$$\|f_{n+1} - \Omega_{n+1} L_n y_n\| = \min_{y \,\in\, \mathbb{C}^n} \|f_{n+1} - \Omega_{n+1} L_n y\|. \tag{6.2}$$

Here, as before, f_{n+1} is given by (2.27) and Ω_{n+1} is defined in (2.28). We remark that the least squares problem (6.2) is actually cheaper to solve than the original one (2.26). The reason is that the matrix L_n in (6.2) has fewer nonzero elements than H_n in (2.26). For example, if no look-ahead steps are taken, then H_n is tridiagonal, while L_n is a lower bidiagonal matrix. This special case will be considered in more detail in § 7.

As discussed in [9], solutions of least squares problems of the type (6.2) can be easily updated from step to step, using the QR decomposition of $\Omega_{n+1}L_n$,

$$\Omega_{n+1} L_n = Q_n^H \begin{bmatrix} R_n \\ 0 \end{bmatrix}, \tag{6.3}$$

where Q_n is a unitary $(n+1)\times(n+1)$ matrix, and R_n is a nonsingular upper triangular $n \times n$ matrix. With this, the least squares problem (6.2) becomes

$$\begin{aligned} \min_{y \,\in\, \mathbb{C}^n} \|f_{n+1} - \Omega_{n+1} L_n y\| &= \min_{y \,\in\, \mathbb{C}^n} \left\| Q_n^H \left(Q_n f_{n+1} - \begin{bmatrix} R_n \\ 0 \end{bmatrix} y \right) \right\| \\ &= \min_{y \,\in\, \mathbb{C}^n} \left\| Q_n f_{n+1} - \begin{bmatrix} R_n \\ 0 \end{bmatrix} y \right\|. \end{aligned}$$

This gives for y_n:

$$y_n = R_n^{-1} t_n, \quad \text{where} \quad t_n = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix}, \quad \begin{bmatrix} t_n \\ \tilde{\tau}_{n+1} \end{bmatrix} := Q_n f_{n+1}. \tag{6.4}$$

Finally, we note that it is possible to update the QMR iterate at each step, as was done in the original QMR algorithm. Full implementation details were given in [9, Sect. 4], and we will not repeat them here. The point is that the QMR iterates have an update formula of the form

$$x_n = x_{n-1} + d_n \tau_n. \tag{6.5}$$

Here, τ_n is given by (6.4), and d_n is an auxiliary search direction defined as the last column of the matrix $V_n U_n^{-1} R_n^{-1} = P_n R_n^{-1}$, which appears in (6.1) after inserting y_n from (6.4). The vectors d_n are also updated with short recurrences: the recurrence for d_n involves only as many vectors as the recurrence for v_{n+1}. For full details of the update procedure for d_n, we refer the reader to [9].

The basic outline of the resulting implementation of QMR based on the coupled two-term Algorithm 3.3 is then as follows.

ALGORITHM 6.1. (QMR based on coupled recurrences)

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $r_0 = b - Ax_0$, $\rho_1 = \|r_0\|$, $v_1 = r_0/\rho_1$.
Choose $w_1 \in \mathbb{C}^N$ *with* $\|w_1\| = 1$.

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

1) *Perform the nth iteration of the coupled two-term Algorithm* 3.3.
This yields matrices L_n, P_n, U_n, *and* V_{n+1}, *which satisfy* (3.7).
2) *Update the QR factorization* (6.3) *of* $\Omega_{n+1} L_n$ *and the vector* t_n *in* (6.4).
3) *Update the QMR iterate* x_n *by means of* (6.5).
4) *If* x_n *has converged, then stop.*

In [9], various properties of the QMR method are given. For example, it is shown how existing BCG iterates can be easily recovered from the QMR process, and how estimates for the QMR residual norms can be obtained at no extra costs. We would like to stress that these properties also hold true for the particular implementation of QMR sketched in Algorithm 6.1.

7. An implementation of QMR without look-ahead. In this section, we present the simplification of Algorithm 6.1 to the case where no look-ahead is used. We also briefly address the issue of preconditioning.

Let M be a given nonsingular $N \times N$ matrix that approximates in some sense the coefficient matrix A of (1.1). Suppose further that M is decomposed as

$$M = M_1 M_2. \tag{7.1}$$

Then, one applies the QMR algorithm to the system

$$A'y' = b', \tag{7.2}$$

where $A' = M_1^{-1} A M_2^{-1}$, $b' = M_1^{-1} b$, and $x' = M_2 x$. It is easy to see that the linear systems (1.1) and (7.2) are equivalent, and that one can transform back from the iterates x_n' and the residuals r_n' of the system (7.1) to the iterates x_n and the residuals r_n of the system (1.1). For example, while applying QMR to the preconditioned system (7.2), it is possible to write the resulting algorithm in terms of the quantities corresponding to the original system (1.1); this is what is done below.

We now present a version of the QMR algorithm based on the coupled Algorithm 3.2, which does not have look-ahead. We remark that in this case, by (3.22) and (3.20), the matrix U_n in (3.9) is upper bidiagonal, and L_n is a lower bidiagonal matrix. We also implement preconditioning, as discussed above. The resulting QMR algorithm is as follows.

ALGORITHM 7.1. (QMR based on coupled recurrences without look-ahead)

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $r_0 = b - Ax_0$.

Compute $\rho_1 = \|M_1^{-1} r_0\|$ *and set* $v_1 = r_0/\rho_1$.

Choose $w_1 \in \mathbb{C}^N$ *with* $\|M_2^{-T} w_1\| = 1$.

Set $p_0 = q_0 = d_0 = 0$, $c_0 = \epsilon_0 = \xi_1 = 1$, $\vartheta_0 = 0$, $\eta_0 = -1$.

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

1) *If* $\epsilon_{n-1} = 0$, *then stop.*

Compute $\delta_n = w_n^T M^{-1} v_n$.

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

2) *Compute*

$$\begin{aligned} p_n &= M^{-1} v_n - p_{n-1}(\xi_n \delta_n/\epsilon_{n-1}), \\ q_n &= M^{-T} w_n - q_{n-1}(\rho_n \delta_n/\epsilon_{n-1}). \end{aligned}$$

3) *Compute* $\epsilon_n = q_n^T A p_n$, $\beta_n = \epsilon_n/\delta_n$, *and set*

$$\begin{aligned} \tilde{v}_{n+1} &= A p_n - v_n \beta_n, \quad \rho_{n+1} = \|M_1^{-1} \tilde{v}_{n+1}\|, \\ \tilde{w}_{n+1} &= A^T q_n - w_n \beta_n, \quad \xi_{n+1} = \|M_2^{-T} \tilde{w}_{n+1}\|. \end{aligned}$$

4) *Compute*

$$\begin{aligned} \vartheta_n &= \frac{\omega_{n+1}\rho_{n+1}}{\omega_n c_{n-1}|\beta_n|}, \quad c_n = \frac{1}{\sqrt{1+\vartheta_n^2}}, \quad \eta_n = -\eta_{n-1}\frac{\rho_n c_n^2}{\beta_n c_{n-1}^2}, \\ d_n &= p_n \eta_n + d_{n-1}(\vartheta_{n-1} c_n)^2, \quad x_n = x_{n-1} + d_n. \end{aligned}$$

5) *If* $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$, *then stop.*

Otherwise, set

$$v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}.$$

As pointed out in § 3.2, the vectors computed in steps 2) and 3) of Algorithm 7.1 are rescaled versions of vectors used in BCG. Freund and Szeto [10] have used this connection to derive an implementation of QMR without look-ahead that is directly based on BCG. Their Algorithm 3.1 in [10] and Algorithm 7.1 are mathematically equivalent.

8. QMR for complex symmetric matrices. In this section, we briefly discuss the application of the QMR Algorithm 7.1 to the solution of complex symmetric linear systems, i.e., systems with

$$A = A^T \in \mathbb{C}^{N \times N}.$$

We note that the QMR approach was originally proposed by Freund in [5] for exactly this class of linear systems.

The benefit of applying a Lanczos method to complex symmetric systems is that the underlying Lanczos algorithm simplifies naturally. Recall that in the coupled Algorithm 3.3, the second starting vector w_1 is arbitrary. In the case of a symmetric matrix, if w_1 is chosen equal to v_1, then it is easy to show that $w_n = v_n$ and $q_n = p_n$, for all n. Thus, the recurrences for w_n and q_n can be eliminated, saving roughly half the amount of work and storage.

However, we remark that—in contrast to the Lanczos algorithm for Hermitian matrices, where breakdowns are excluded—the Lanczos process for complex symmetric also requires look-ahead in order to avoid exact and near-breakdowns. This is discussed in detail in [5].

The only other issue in the case of complex symmetric systems is that the preconditioner M in (7.1) must also be symmetric, i.e.,

$$M = M_1 M_2 = (M_1 M_2)^T = M^T. \tag{8.1}$$

For example, this is always guaranteed if the decomposition (7.1) is "symmetric" in the sense that

$$M_2 = M_1^T. \tag{8.2}$$

However, we stress that the condition (8.2) is not necessary, and M_1 and M_2 can be arbitrary matrices satisfying (8.1). We remark that standard preconditioning techniques, such as incomplete factorization, yield symmetric preconditioners (8.1) when applied to symmetric matrices A.

In this case, one can apply the QMR Algorithm 7.1 to the resulting preconditioned system. Once again writing everything in terms of the quantities corresponding to the original system (1.1), one obtains the following iteration.

ALGORITHM 8.1. (QMR without look-ahead for complex symmetric systems)

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $r_0 = b - Ax_0$.

Compute $\rho_1 = \|M_1^{-1} r_0\|$ *and set* $v_1 = r_0/\rho_1$.

Set $p_0 = d_0 = 0$, $c_0 = \epsilon_0 = 1$, $\vartheta_0 = 0$, $\eta_0 = -1$.

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

1) *If* $\epsilon_{n-1} = 0$, *then stop.*

Compute $\delta_n = v_n^T M^{-1} v_n$.

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

2) *Compute*

$$p_n = M^{-1}v_n - p_{n-1}(\rho_n\delta_n/\epsilon_{n-1}).$$

3) *Compute* $\epsilon_n = p_n^T A p_n$, $\beta_n = \epsilon_n/\delta_n$, *and*

$$\tilde{v}_{n+1} = Ap_n - v_n\beta_n, \quad \rho_{n+1} = \|M_1^{-1}\tilde{v}_{n+1}\|.$$

4) *Compute*

$$\begin{aligned} \vartheta_n &= \frac{\omega_{n+1}\rho_{n+1}}{\omega_n c_{n-1}|\beta_n|}, \quad c_n = \frac{1}{\sqrt{1+\vartheta_n^2}}, \quad \eta_n = -\eta_{n-1}\frac{\rho_n c_n^2}{\beta_n c_{n-1}^2}, \\ d_n &= p_n\eta_n + d_{n-1}(\vartheta_{n-1}c_n)^2, \quad x_n = x_{n-1} + d_n. \end{aligned}$$

5) *If* $\rho_{n+1} = 0$, *then stop.*
Otherwise, set

$$v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}.$$

9. Numerical experiments. In this section, we present a few numerical examples. We compare the original and the new implementation of the QMR algorithm, as well as illustrate an application of the coupled QMR algorithm to the solution of complex symmetric linear systems.

In Figures 9.1–9.3 below, we always show the true relative residual norm $\|r_n\|/\|r_0\|$ plotted versus the iteration index n.

Example 9.1. This example is taken from [1], and is meant to illustrate the typical behavior that can be expected from the new implementation of QMR when compared to the original implementation. We consider the differential equation

$$Lu = f \quad \text{on} \quad (0,1)\times(0,1), \tag{9.1}$$

where

$$\begin{aligned} Lu := &-\frac{\partial}{\partial x}\left(e^{-xy}\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(e^{xy}\frac{\partial u}{\partial y}\right) \\ &+20(x+y)\frac{\partial u}{\partial x} + 20\frac{\partial}{\partial x}\left((x+y)u\right) + \frac{1}{1+x+y}u, \end{aligned}$$

with Dirichlet boundary conditions $u = 0$. We discretized (9.1) using centered differences on a 29×29 grid with mesh size $h = 1/30$. This leads to a linear system $Ax = b$, where A is a nonsymmetric matrix of order $N = 900$ with 4380 nonzero entries. We ran the original and the new implementations of QMR, both with look-ahead and with the same starting conditions, until the true residual norm $\|r_n\|$ was not reduced any further. The vectors b and w_1 were random vectors, the initial guess x_0 was zero, and the example was run without preconditioning. The original QMR algorithm is plotted with a dotted line; it stagnated at 6.7E−14, and it built 6 look-ahead blocks of size 2.

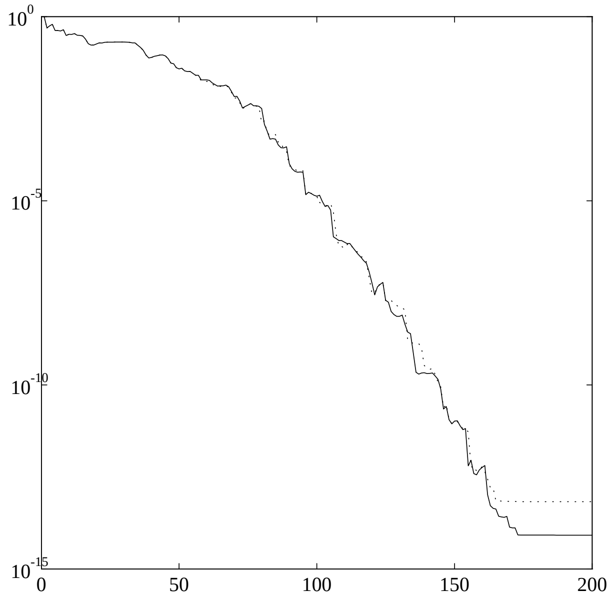

FIG. 9.1. *Convergence curves for Example* 9.1.

The coupled QMR algorithm is plotted with a solid line; it stagnated at 8.3E−15, and it built 4 blocks of size 2 in the V–W sequence and 7 blocks of size 2 in the P–Q sequence. This behavior seems to be fairly typical, in that usually the new implementation is better than the original implementation, but the difference is not very large.

However, there are cases where the coupled implementation is significantly better than the original QMR implementation. The next example is of this type.

Example 9.2. This is a linear system that arises in performance modeling of multiprocessor systems, using Petri net analysis. In such applications, one obtains large sparse singular matrices A with null spaces of dimension 1, and one needs to compute a nontrivial basis vector for this null space. This leads to a linear system of the form

$$Ax = 0,$$

and thus condition (2.1) is satisfied. We used a matrix A of size $N = 3663$ with 23397 non-zero elements. The vector b was zero, while the initial guess x_0 and the starting vector w_1 were both random. The linear system is a difficult one and does not converge easily without preconditioning. We used the variant described in [9] of Saad's ILUT preconditioner [17], with no additional fill-in allowed and a drop tolerance of 0.001, which generated a preconditioning matrix M with 23397 elements. The original QMR algorithm is plotted with a dotted line; it stagnated at 2.9E−5, and it built 2 blocks of size 2. On the other hand, the new implementation, plotted with a solid line, stagnated at 1.2E−12, and it built 1 block of size 2 in the V–W sequence and 3 blocks of size 2 in the P–Q sequence.

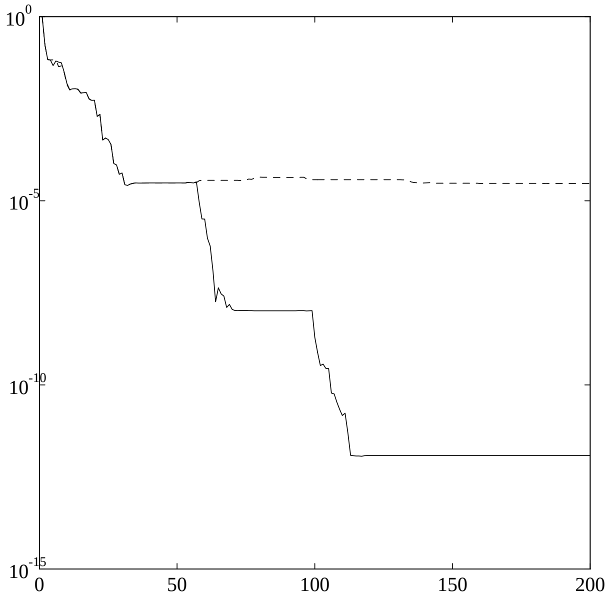

FIG. 9.2. *Convergence curves for Example* 9.2.

Example 9.3. Here A is the complex symmetric YOUNG1C matrix from the Harwell-Boeing test collection of sparse matrices [3]. The matrix arises in a scattering problem in aerodynamics research; it is of dimension $N = 841$ with 4089 non-zero elements. We ran Algorithm 8.1, with various complex symmetric ILUT preconditioners. In all cases, the iteration was started with the same random vector for b and zero initial guess x_0. This system is also a difficult one, and if not preconditioned, the QMR algorithm requires around 700 iterations to reach the stagnation level of 2.5E−14; the corresponding convergence curve is plotted in a dashed line. However, the ILUT preconditioner is quite effective in this example, especially at higher levels of allowed fill-in and/or drop tolerance. The graph shows, in order of solid lines from right to left, ILUT with no additional fill-in and 0.001 drop tolerance (2375 non-zero elements), ILUT with 5 additional fill-in and 0.001 drop tolerance (5171 non-zero elements), ILUT with 10 additional fill-in and 0.001 drop tolerance (9320 non-zero elements), and finally ILUT with 16 additional fill-in and 0.0 drop tolerance (13329 non-zero elements). As can be seen, all variants reach roughly the same stagnation level, around 1.0E−14. However, they do so in fewer and fewer iterations, and in fact, for this example, the additional time spent computing the denser preconditioners was always made up by converging in fewer iterations.

10. Concluding remarks. We have presented a new look-ahead algorithm for constructing Lanczos vectors based on coupled two-term recurrences instead of the usual three-term recurrences. We then discussed a new implementation of the quasi-minimal residual algorithm, using the coupled process to build the basis for the Krylov

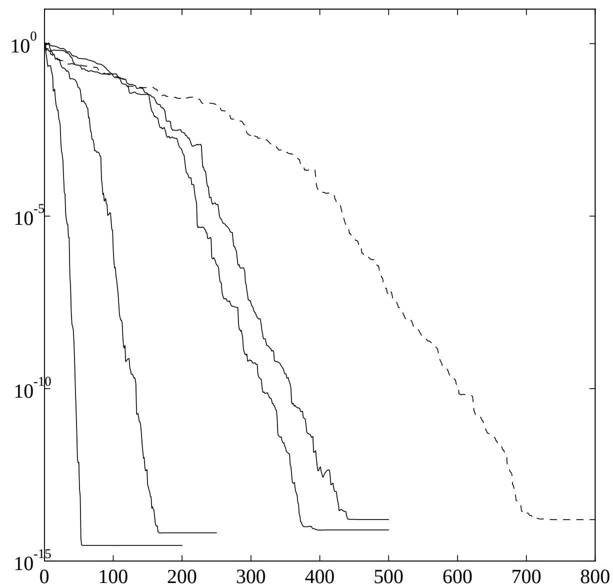

FIG. 9.3. *Convergence curves for Example* 9.3.

space. While the theoretical results derived for the original algorithm carry over to the new one, the latter was shown in examples to have better numerical properties. We also briefly covered an implementation of the new QMR method without look-ahead, as well as the application of the QMR algorithm to the solution of complex symmetric linear systems, where the underlying Lanczos process naturally simplifies. Finally, an extended version of this paper, with a more detailed implementation section, is available as a RIACS Technical Report.

FORTRAN 77 codes for the proposed coupled-two term look-ahead procedure and the resulting new implementation of the QMR algorithm can be obtained electronically from the authors (na.freund@na-net.ornl.gov or na.nachtigal@na-net.ornl.gov). We note that FORTRAN 77 codes for the original implementation of QMR and the underlying look-ahead Lanczos algorithm are available from netlib by sending an email message consisting of the single line "send lalqmr from misc" to netlib@ornl.gov or netlib@research.att.com.

Acknowledgement. The authors wish to acknowledge the fruitful discussions held with Martin Gutknecht and Tedd Szeto. Marlis Hochbruck and Uwe Seidel have provided the matrix for Example 9.2. Finally, we would like to thank Susanne Freund for her careful reading of the manuscript.

REFERENCES

[1] J. K. CULLUM AND R. A. WILLOUGHBY, *A practical procedure for computing eigenvalues of*

large sparse nonsymmetric matrices, in Large Scale Eigenvalue Problems, J. K. Cullum and R. A. Willoughby, eds., North-Holland, 1986, pp. 193–240.

[2] A. DRAUX, *Polynômes Orthogonaux Formels — Applications*, Lecture Notes in Mathematics 974, Springer-Verlag, Berlin, Germany, 1983.

[3] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[4] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis Dundee 1975, G. A. Watson, ed., Lecture Notes in Mathematics 506, Berlin, Germany, 1976, Springer-Verlag, pp. 73–89.

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

[6] ———, *Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations*, Technical Report 92.07, RIACS, NASA Ames Research Center, Moffett Field, CA, March 1992. To appear in Numerical Methods of Approximation Theory.

[7] R. W. FREUND, M. H. GUTKNECHT, AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, Technical Report 91.09, RIACS, NASA Ames Research Center, Moffett Field, CA, April 1991. To appear in the SIAM Journal on Scientific and Statistical Computing.

[8] R. W. FREUND AND M. HOCHBRUCK, *On the use of two QMR algorithms for solving singular systems and applications in Markov chain modeling*, Technical Report 91.25, RIACS, NASA Ames Research Center, Moffett Field, CA, December 1991.

[9] R. W. FREUND AND N. M. NACHTIGAL, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[10] R. W. FREUND AND T. SZETO, *A quasi-minimal residual squared algorithm for non-Hermitian linear systems*. Proceedings of the 1992 Copper Mountain Conference on Iterative Methods, April 1992.

[11] M. H. GUTKNECHT, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part II*, IPS Research Report 90–16, IPS, ETH, Zürich, Switzerland, September 1990.

[12] ———, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 594–639.

[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 (1950), pp. 255–282.

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

[15] B. N. PARLETT, D. R. TAYLOR, AND Z. A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.

[16] H. RUTISHAUSER, *Der Quotienten-Differenzen-Algorithmus*, vol. 7 of Mitteilungen aus dem Institut für angewandte Mathematik an der ETH Zürich, Birkhaüser, Basel, Switzerland, 1957.

[17] Y. SAAD, *ILUT: a dual threshold incomplete LU factorization*, Research Report UMSI 92/38, University of Minnesota Supercomputer Institute, Minneapolis, MN, March 1992.

[18] D. R. TAYLOR, *Analysis of the look ahead Lanczos algorithm*, PhD Thesis, Department of Mathematics, University of California, Berkeley, CA, November 1982.