TRANSPOSE-FREE QUASI-MINIMAL RESIDUAL METHODS FOR NON-HERMITIAN LINEAR SYSTEMS

ROLAND W. FREUND*

Abstract. Recently, Freund and Nachtigal proposed a novel conjugate gradient-type method, the quasi-minimal residual algorithm (QMR), for the iterative solution of general non-Hermitian systems of linear equations. The QMR method is based on the nonsymmetric Lanczos process, and thus, like the latter, QMR requires matrix-vector multiplications with both the coefficient matrix of the linear system and its transpose. However, in certain applications, the transpose is not readily available, and generally, it is desirable to trade in multiplications with the transpose for matrix-vector products with the original matrix.

This paper gives a survey of transpose-free algorithms that are based on the quasi-minimal residual approach. First, it is shown that, in principle, the transpose in the standard QMR method can always be eliminated by choosing special starting vectors. Examples are given for which this approach is practical. Then, two transpose-free QMR methods, the TFQMR algorithm and the QMR squared algorithm, for general non-Hermitian systems are described. Some theory for ideal transpose-free QMR and TFQMR is presented. Results of numerical experiments are reported. Finally, some open problems are mentioned.

Key words. non-Hermitian linear systems, nonsymmetric Lanczos process, quasi-minimal residual property, transpose-free, conjugate gradients squared, symmetric indefinite linear systems

AMS(MOS) subject classifications. 65F10, 65N22, 65K05

1. Introduction.

The classical conjugate gradient method (CG) [27]—used in conjunction with preconditioning [5], [30]—is one of the most powerful iterative techniques for solving Hermitian positive definite systems of linear equations. Among the many properties of CG, the following two are especially important:

(i) The method generates iterates that are characterized by a minimization property over Krylov subspaces induced by the coefficient matrix of the linear system.

(ii) The iterates are computable by means of short vector updates, and thus storage requirements and work per iteration are low and constant throughout the iteration.

For general non-Hermitian linear systems, the situation is less satisfactory. It was shown [8], [29], [39] that ideal non-Hermitian extensions of CG with both properties (i) and (ii) do not exist. Instead, most CG-type methods for non-Hermitian linear systems fulfill either (i) or (ii), but not both. In the first category, the most successful scheme is the generalized minimal residual algorithm (GMRES) [36]. It generates iterates defined by a minimal residual property, and thus it fulfills (i). However, the condition (ii) is violated, since storage requirements and work per iteration grow linearly with the iteration number. In the

*AT&T Bell Laboratories, Room 2C–420, 600 Mountain Avenue, Murray Hill, New Jersey 07974–0636. This research was performed while the author was in residence at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, California 94035, and it was supported by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association.

second category, the archetype is the biconjugate gradient algorithm (BCG) [32], [9]. BCG is based on simple coupled two-term recurrences, and thus the condition (ii) is satisfied. However, the BCG iterates are defined by a Galerkin condition rather than a minimization property (i), which means that the algorithm can exhibit—and typically does—a rather irregular convergence behavior with wild oscillations in the residual norm.

Although one cannot fulfill (i) and (ii) simultaneously, it is possible to devise CG-type methods for general non-Hermitian linear systems that satisfy (ii) and a relaxed version of (i), namely a quasi-minimization property. An iterative scheme of this type, the quasi-minimal residual algorithm (QMR), was recently proposed by Freund [11] for the special case of complex symmetric matrices, and by Freund and Nachtigal [18] for the case of general nonsingular non-Hermitian linear systems. Freund and Hochbruck [17] extended the QMR method to linear systems with singular square coefficient matrices.

The QMR method is based on the nonsymmetric Lanczos process, and thus, like the latter, QMR requires matrix-vector multiplications with both the coefficient matrix A of the linear system and its transpose A^T. This is a disadvantage in certain applications where products $A \cdot v$ can be obtained efficiently without explicitly generating A, while products $A^T \cdot w$ are not readily available. Furthermore, in QMR, the transpose A^T only appears in the construction of vectors that are used to compute the recurrence coefficients for the Lanczos process. However, these vectors are never used directly in the construction of the actual QMR iterates. Therefore, it is desirable to have QMR methods that do not involve the transpose. Various such transpose-free QMR algorithms have been proposed, see [12], [4], [20], [22]. In this paper, we give an overview of these different transpose-free QMR approaches, and we present some extensions of the results given in [12].

The remainder of this paper is organized as follows. In § 2, we briefly review the standard QMR method. In § 3, we consider the ideal transpose-free QMR algorithm, and we discuss some cases where this approach is viable. In § 4, we describe the TFQMR method. In § 5, we give convergence results for QMR and TFQMR. In § 6, we present an algorithm for computing the zeros of the QMR and TFQMR residual polynomials. In § 7, we describe the QMR squared algorithm. In § 8, results of numerical experiments are reported Finally, in § 9, we make some concluding remarks, and we mention some open problems.

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, $M^T = [m_{kj}]$ and $M^H = [\overline{m_{kj}}]$ denote the 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. The set of eigenvalues of a square matrix M is denoted by $\lambda(M)$. 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 $M \in \mathbb{C}^{N\times N}$ is defined by

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

and we will make use of the fact that

$$K_n(c, M) = \{\varphi(M)c \bigm| \varphi \in \mathcal{P}_{n-1}\}.$$

2. Review of the standard QMR method. We are concerned with the iterative solution of systems of linear equations

$$Ax = b, \tag{2.1}$$

where A is a general, singular or nonsingular, $N \times N$ matrix with real or complex entries, and $b \in \mathbb{C}^N$. In this section, we briefly review the standard QMR method [18].

2.1. Krylov subspace iterations. All algorithms described in this paper fall into the category of *Krylov subspace iterations*: they produce approximate solutions x_n of (2.1) of the form

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

Here $x_0 \in \mathbb{C}^N$ is an arbitrary initial guess, and $r_0 := b - Ax_0$ is the corresponding residual vector. In the sequel, we always use the notation

$$r_n := b - Ax_n$$

for the residual vector associated with the iterate x_n.

The standard QMR method is a Krylov subspace iteration, where a look-ahead Lanczos algorithm is used to generate basis vectors for the subspaces $K_n(r_0, A)$, and the iterates x_n are characterized by a quasi-minimal residual property.

2.2. The look-ahead Lanczos algorithm. In the classical nonsymmetric Lanczos algorithm [31], breakdowns—more precisely, division by 0—can occur. Parlett, Taylor, and Liu [34] were the first to demonstrate that breakdowns and near-breakdowns (division by a nonzero, but small number) in the standard algorithm can be avoided by using so-called *look-ahead* techniques. Recently, Freund, Gutknecht, and Nachtigal [16] developed a look-ahead Lanczos algorithm that is different from the procedure described in [34]. The QMR method is based on the algorithm proposed in [16]. Next we sketch some key properties of this look-ahead Lanczos procedure; a detailed description of the algorithm can be found in [16]. For further properties of the Lanczos process, we refer the reader to [26], [15], [21] and the references given there.

Starting with arbitrary initial vectors

$$v_1,\, w_1 \in \mathbb{C}^N, \quad \text{with} \quad \|v_1\| = \|w_1\| = 1, \tag{2.3}$$

the look-ahead Lanczos algorithm generates two sequences of vectors $v_1, v_2, \cdots, v_n$ and $w_1, w_2, \cdots, w_n$, $n = 1, 2, \cdots$, such that

$$\begin{aligned} \operatorname{span}\{v_1, v_2, \cdots, v_n\} &= K_n(v_1, A), \\ \operatorname{span}\{w_1, w_2, \cdots, w_n\} &= K_n(w_1, A^T), \end{aligned} \tag{2.4}$$

and the two sequences are block biorthogonal:

$$(W^{(j)})^T V^{(k)} = 0 \quad \text{for all} \quad j \neq k, \quad j, k = 1, 2, \cdots, l(n). \tag{2.5}$$

The matrices $W^{(j)}$ and $V^{(k)}$ in (2.5) are defined by partitioning the first n Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ into blocks, according to the look-ahead steps taken:

$$\begin{aligned} V_n &:= [\, v_1 \quad v_2 \quad \cdots \quad v_n \,] = [\, V^{(1)} \quad V^{(2)} \quad \cdots \quad V^{(l(n))} \,], \\ W_n &:= [\, w_1 \quad w_2 \quad \cdots \quad w_n \,] = [\, W^{(1)} \quad W^{(2)} \quad \cdots \quad W^{(l(n))} \,]. \end{aligned} \tag{2.6}$$

More precisely, the matrices $V^{(k)}$ and $W^{(k)}$ are both of order $N \times \mu_k$, and their columns are just the Lanczos vectors corresponding to the kth look-ahead step. Here, μ_k denotes the length of the kth look-ahead step. Moreover, in (2.6), $l(n)$ denotes the number of look-ahead steps that have been performed during the first n iterations of the Lanczos process. We remark that the conditions (2.4)–(2.5) determine the Lanczos vectors only up to scaling. In the sequel, we always assume that the Lanczos vectors are scaled to have unit length:

$$\|v_n\| = \|w_n\| = 1, \quad n = 1, 2, \cdots . \tag{2.7}$$

We stress that the algorithm performs mostly standard Lanczos steps, i.e., look-ahead steps of size $\mu_k = 1$ with blocks $V^{(k)}$ and $W^{(k)}$ that consist of only single Lanczos vectors. True look-ahead steps, i.e., steps of size $\mu_k > 1$, 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 $\mu_k = 2$. We note that, if only steps of length $\mu_k = 1$ are performed, then the look-ahead Lanczos algorithm reduces to the classical algorithm.

The crucial point of the look-ahead Lanczos algorithm is that the vectors satisfying (2.4) and (2.5) can be generated by means of short recurrences. For the first $n+1$ vectors $\{v_j\}_{j=1}^{n+1}$ and $\{w_j\}_{j=1}^{n+1}$, these recurrence relations can be written compactly in matrix form as follows:

$$AV_n = V_{n+1} T_n \quad \text{and} \quad A^T W_n = W_{n+1} \Gamma_{n+1}^{-1} T_n \Gamma_n. \tag{2.8}$$

Here T_n is an $(n+1) \times n$ upper Hessenberg matrix, which is also block tridiagonal with square blocks of order μ_k on the diagonal, and

$$\Gamma_n := \operatorname{diag}(\gamma_1, \gamma_2, \cdots, \gamma_n), \quad \text{with} \quad \gamma_j > 0 \quad \text{for all} \quad j, \tag{2.9}$$

is a scaling matrix. Furthermore, we note that T_n always has full column rank, and thus

$$\operatorname{rank} T_n = n. \tag{2.10}$$

Finally, we will make use of the fact that each pair of Lanczos vectors v_n and w_n can be expressed in the form

$$v_n = \varphi_{n-1}(A)v_1 \quad \text{and} \quad w_n = \gamma_n \varphi_{n-1}(A^T)w_1, \tag{2.11}$$

where $\varphi_{n-1} \in \mathcal{P}_{n-1}$ is of exact degree $n-1$ and $\gamma_n > 0$ is as in (2.9). We refer to the φ_j's in (2.11) as the *look-ahead Lanczos polynomials*.

2.3. The QMR iterates. In (2.3), we now choose the starting vector

$$v_1 := r_0/\rho_1, \quad \text{where} \quad \rho_1 := \|r_0\|, \tag{2.12}$$

while w_1 is still an arbitrary unit vector. In view of (2.4) and (2.6), the columns of V_n span the subspace $K_n(r_0, A)$ in (2.2). Hence the nth iterate x_n of any Krylov subspace iteration can be written in the form

$$x_n = x_0 + V_n z_n, \quad \text{where} \quad z_n \in \mathbb{C}^n. \tag{2.13}$$

The nth iterate of the standard QMR method is given by (2.13) where $z_n \in \mathbb{C}^n$ is chosen as the unique solution of the least-squares problem

$$\|f_n - H_n z_n\| = \min_{z \in \mathbb{C}^n} \|f_n - H_n z\|. \tag{2.14}$$

Here H_n is a rescaled version of the Lanczos matrix T_n from (2.8), namely

$$H_n := \Omega_{n+1} T_n, \tag{2.15}$$

where

$$\Omega_{n+1} := \operatorname{diag}(\omega_1, \omega_2, \cdots, \omega_{n+1}), \quad \text{with} \quad \omega_j > 0 \quad \text{for all} \quad j, \tag{2.16}$$

is a still arbitrary diagonal scaling matrix. Moreover, f_n in (2.14) is given by

$$f_n := \omega_1 \rho_1 \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \in \mathbb{R}^{n+1}. \tag{2.17}$$

We remark that, by (2.10) and (2.15)–(2.16), the matrix H_n has full column rank, and this guarantees that z_n is uniquely defined by (2.14).

Using (2.13), (2.12), the first relation in (2.8), (2.15), and (2.17), one readily verifies that the residual vector r_n corresponding to x_n satisfies

$$r_n = V_{n+1}\Omega_{n+1}^{-1}(f_n - H_n z_n). \tag{2.18}$$

Hence, in view of (2.14), the nth QMR iterate is characterized by a minimization of the second factor in (2.18); this is just the *quasi-minimal residual property*. We note that the standard choice for the scaling matrix (2.16) is

$$\omega_j = 1 \quad \text{for all} \quad j. \tag{2.19}$$

In view of (2.18) and (2.7), the scaling (2.19) is very natural, in the sense that all columns of the matrix $V_{n+1}\Omega_{n+1}^{-1}$ in the representation (2.18) of r_n are treated equally. We stress that there are situations (see § 7.2) where it is crucial to use a scaling different from (2.19); this is the reason for formulating the QMR method for general scaling matrices (2.16).

The original implementation [18] of the QMR method is directly based on the definition (2.13)–(2.14) of the QMR iterates. We remark that, due to the block tridiagonal structure of H_n, the solution z_n of (2.14), and hence the QMR iterate x_n can be easily updated from step to step. The resulting update for x_n only involves short vector recurrences; see [18] for details.

Of course, there are also other possible implementations of the QMR method. Suppose that we have a factorization

$$H_n = L_n U_n \tag{2.20}$$

of H_n into an upper Hessenberg matrix L_n and an upper triangular matrix U_n. Then, the definition (2.13)–(2.14) of the nth QMR iterate can be rewritten as follows:

$$x_n = x_0 + P_n y_n, \tag{2.21}$$

where $P_n := V_n U_n^{-1}$ and y_n ($:= U_n z_n$) is the unique solution of

$$\|f_n - L_n y_n\| = \min_{y \in \mathbb{C}^n} \|f_n - L_n y\|. \tag{2.22}$$

An implementation of QMR based on (2.21)–(2.22) was recently proposed by Freund and Nachtigal [19]. Here, instead of (2.8), the Lanczos vectors $\{v_j\}_{j=1}^{n+1}$ are generated by coupled recurrences of the form

$$V_n = P_n U_n, \quad AP_n = V_{n+1}\Omega_{n+1}^{-1} L_n,$$

and analogous recursions are used to compute $\{w_j\}_{j=1}^{n+1}$. If no look-ahead steps are taken in this coupled procedure, then the matrices L_n and U_n are lower and upper bidiagonal matrices, respectively. For a detailed description of this implementation of QMR based on coupled recurrences, we refer the reader to [19].

3. The ideal transpose-free QMR method. We now turn to transpose-free QMR iterations for solving linear systems (2.1). Note that, in the standard QMR method, the transpose A^T only appears in the recurrences for the vectors $\{w_j\}_{j=1}^{n+1}$ in (2.8). These vectors are needed for generating the entries of the Lanczos recurrence matrix T_n; however, they are not used in the actual computation of the QMR iterates based on (2.13)–(2.14). Freund and Zha [22] have shown that, in principle, it is always possible to eliminate the transpose in the standard QMR method by choosing a special starting vector w_1. In this section, we describe the resulting *ideal* transpose-free QMR method.

3.1. Simplification of the Lanczos process. Suppose that we are given a nonsingular matrix $S \in \mathbb{C}^{N\times N}$ such that

$$A^T S = SA. \tag{3.1}$$

We remark that a matrix S satisfying (3.1) always exists, since any square matrix A is similar to its transpose A^T (see, e.g., [28, Chap. 3.2]). Recall that, for the standard QMR method, the second starting vector w_1 in (2.3) can be chosen arbitrarily. We now set

$$w_1 := \frac{1}{\|Sv_1\|} S v_1. \tag{3.2}$$

Then, using (3.1)–(3.2), the polynomial representation (2.11) of v_n and w_n, and the normalization condition (2.7), one readily verifies that

$$w_n = \frac{\gamma_n}{\|Sv_1\|} S v_n = \frac{1}{\|Sv_n\|} S v_n \quad \text{for all} \quad n. \tag{3.3}$$

Hence, for the special starting vector (3.2), the look-ahead Lanczos algorithm simplifies: instead of running both recurrences in (2.8) for both Lanczos sequences $v_1, v_2, \cdots$, and $w_1, w_2, \cdots$, only the first one is used to generate the v_n's, while the w_n's are computed by means of (3.3). Clearly, this eliminates the need of A^T, at the expense of one matrix-vector product $S \cdot v_n$ at each iteration step. Therefore, this ideal transpose-free QMR method is only viable if the matrix S is such that products with S can be computed cheaply. Examples of this type are given in § 3.3.

3.2. An implementation without look-ahead. In this section, we present an actual implementation of the ideal transpose-free QMR method. For simplicity, we only state the algorithm without look-ahead; the general case with look-ahead is treated in [22]. Moreover, in practical applications, the matrix S in (3.1) usually corresponds to the inverse of a preconditioner M, and we formulate the algorithm in this setting.

Let $M \in \mathbb{C}^{N\times N}$ be an arbitrary given nonsingular matrix. We then define

$$J := SM \quad \text{and} \quad T := AM, \tag{3.4}$$

and we state the ideal transpose-free QMR algorithm in terms of the matrices T, M, and J, instead of A and S. We remark that, with (3.4), the relation (3.1) for S translates into the equivalent condition

$$T^T J = M^T J M^{-1} T \tag{3.5}$$

for J. The motivation for the transition to T, instead of A, is as follows. Suppose we want to solve the linear system

$$Ty = b, \tag{3.6}$$

using the matrix M as a right preconditioner. We apply the ideal transpose-free QMR method to the preconditioned system

$$Ax = b, \quad \text{where} \quad A := TM^{-1} \quad \text{and} \quad x := My.$$

By rewriting the resulting algorithm in terms of the original linear system (3.6), one obtains the following ideal transpose-free QMR algorithm for solving (3.6), using a right preconditioning matrix M.

ALGORITHM 3.1 ("IDEAL" TRANSPOSE-FREE QMR WITHOUT LOOK-AHEAD).

(1) *Start:*

(a) *Choose* $y_0 \in \mathbb{C}^N$.

(b) *Set* $r_0 = b - Ty_0$, $\rho_1 = \|r_0\|$, $v_1 = r_0/\rho_1$, $p_0 = d_0 = 0$;
$\chi_0 = \varepsilon_0 = 1$, $\vartheta_0 = 0$, $\eta_0 = -1$.

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

(a) *Compute* $\hat{w}_n$ *by solving* $M\hat{w}_n = v_n$.
Set $\delta_n = (J\hat{w}_n)^T v_n$, $\beta_{n-1} = \rho_n \delta_n / \varepsilon_{n-1}$;
$p_n = \hat{w}_n - p_{n-1}\beta_{n-1}$.

(b) *Set* $\varepsilon_n = (Jp_n)^T T p_n$, $\alpha_n = \varepsilon_n / \delta_n$;
$\tilde{v}_{n+1} = Tp_n - v_n \alpha_n$, $\rho_{n+1} = \|\tilde{v}_{n+1}\|$, $v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}$.

(c) *Set* $\vartheta_n = (\omega_{n+1}\rho_{n+1})^2 / (\omega_n^2 \chi_{n-1} |\alpha_n|^2)$, $\chi_n = 1/(1 + \vartheta_n)$;
$\eta_n = -\eta_{n-1}\rho_n \chi_n / (\alpha_n \chi_{n-1})$;
$d_n = p_n \eta_n + d_{n-1}(\vartheta_{n-1}\chi_n)$, $y_n = y_{n-1} + d_n$.

If y_n *has converged, then stop.*

Note that, in Algorithm 3.1, we only generate the vectors $\hat{w}_n := M^{-1}v_n$ $(= J^{-1}Sv_n)$, instead of (3.3). The computation of $\hat{w}_n$ involves the usual solution of a linear system with the preconditioner M as coefficient matrix. In addition, Algorithm 3.1 involves two matrix-vector products $J \cdot \hat{w}_n$ and $J \cdot p_n$ at each step. In typical applications (see § 3.3), the matrix J is either the identity or the antidiagonal identity matrix, and then these multiplications with J do not represent extra work.

Finally, we remark that Algorithm 3.1 uses coupled recurrences to generate the Lanczos vectors $\{v_j\}_{j=1}^{n+1}$, based on a factorization (2.20) of H_n into a lower and upper bidiagonal matrix L_n and U_n, respectively. For details of the derivation of Algorithm 3.1, we refer the reader to [22], [19].

3.3. Examples. In this section, we give examples for linear systems (3.6) with preconditioning matrices M that satisfy (3.5).

3.3.1. Complex symmetric systems. Many important applications lead to linear systems (3.6) with complex non-Hermitian, but symmetric coefficient matrices $T = T^T$; we refer the reader to [11] and the references given there. Standard preconditioning techniques, as for instance incomplete factorization [30], yield complex symmetric preconditioners $M = M^T$ when applied to complex symmetric systems. For complex symmetric T and M, the condition (3.5) is trivially satisfied, where $J := I_N$ is the $N \times N$ identity matrix.

3.3.2. Real symmetric indefinite systems. Another important special case is real linear systems (3.6) where $T = T^T \in \mathbb{R}^{N\times N}$ is symmetric, but indefinite and $M = M^T \in \mathbb{R}^{N\times N}$ is a nonsingular preconditioner for (3.6). Again, condition (3.5) is trivially satisfied with $J := I_N$. The resulting QMR Algorithm 3.1 for solving real symmetric indefinite systems has the same work and storage requirements as SYMMLQ and MINRES [33], which are the standard CG-like iterations for such systems. However, both SYMMLQ and MINRES can only be used with symmetric positive definite preconditioners M. This is a rather restrictive condition for highly indefinite systems for which standard preconditioning techniques yield indefinite preconditioners. In contrast to SYMMLQ and MINRES, the QMR Algorithm 3.1 can be combined with any nonsingular symmetric preconditioner M.

We remark that a particular class of highly indefinite symmetric systems are real linear systems (3.6) with coefficient matrices of the form

$$T = \begin{bmatrix} H & G^T \\ G & 0 \end{bmatrix}, \tag{3.7}$$

where H is symmetric positive semidefinite. Such systems arise, for example, in incompressible flow computations [25] and as subproblems in optimization algorithms, such as interior-point methods for linear and nonlinear programs or sequential programming methods for constrained nonlinear programs [23], [40].

3.3.3. Centrosymmetric systems. A matrix $T \in \mathbb{C}^{N\times N}$ is called *centrosymmetric* if it is symmetric with respect to the antidiagonal, i.e.,

$$T = J_N T^T J_N,$$

where J_N is the $N \times N$ antidiagonal identity matrix defined by

$$J_N := \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \iddots & 1 & 0 \\ 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{N\times N}.$$

One easily verifies that, for centrosymmetric linear systems (3.6) with centrosymmetric preconditioners M, the condition (3.5) is satisfied with $J := J_N$. Hence the QMR Algorithm 3.1 can be used for the iterative solution of centrosymmetric linear systems. Note that matrix-vector products $J \cdot v$ just correspond to reversing the order of the entries of v.

We note that Toeplitz matrices, i.e., matrices whose elements are constant along each diagonal, are in particular centrosymmetric. Chan and Strang [3] suggested to use circulant matrices as preconditioners for Toeplitz systems. In this case, the linear systems with the preconditioner M can be solved efficiently be means of the fast Fourier transformation. Since circulant matrices are centrosymmetric, the QMR Algorithm 3.1 can be used for the iterative solution of Toeplitz systems (3.6) with circulant preconditioners.

Finally, we remark that further examples, for which the ideal transpose-free QMR method is practical, are given in [22].

4. The TFQMR method. In this section, we sketch the *transpose-free quasi-minimal residual* method (TFQMR) for solving general non-Hermitian linear systems (2.1).

4.1. Sonneveld's observation. Sonneveld [37] with his *conjugate gradients squared* algorithm (CGS) was the first to propose a transpose-free BCG-type Krylov subspace iteration for non-Hermitian linear systems (2.1). All scalar quantities that appear in the nth iteration of BCG are of the form

$$w^T v, \quad \text{where} \quad v = \varphi(A)v_1, \; w = \psi(A^T)w_1 \quad \text{and} \quad \varphi, \psi \in \mathcal{P}_n. \tag{4.1}$$

Sonneveld observed that, by rewriting (4.1) in the form

$$w^T v = w_1^T(\psi(A)\varphi(A)v_1),$$

the BCG scalars can be obtained without using A^T, by working with vectors of the form $\psi(A)\varphi(A)v_1$ instead of $\varphi(A)v_1$ and $\psi(A^T)w_1$. Although the CGS algorithm produces the

BCG scalars, it generates a different set of iterates than BCG. More precisely, the CGS iterates x_{2n}^{CGS}, $n = 0, 1, \cdots$, are defined by

$$r_{2n}^{\mathrm{CGS}} = b - Ax_{2n}^{\mathrm{CGS}} = (\psi_n^{\mathrm{BCG}}(A))^2 r_0. \tag{4.2}$$

Here ψ_n^{BCG} is given by the polynomial representation

$$\begin{gathered} r_n^{\mathrm{BCG}} = b - Ax_n^{\mathrm{BCG}} = \psi_n^{\mathrm{BCG}}(A) r_0, \\ \text{where} \quad \psi_n^{\mathrm{BCG}} \in \mathcal{P}_n \quad \text{and} \quad \psi_n^{\mathrm{BCG}}(0) = 1, \end{gathered} \tag{4.3}$$

of the nth BCG residual vector. We remark that, in CGS and BCG, roughly the same work and storage is required to generate the iterates

$$x_{2n}^{\mathrm{CGS}} \in x_0 + K_{2n}(r_0, A) \quad \text{and} \quad x_n^{\mathrm{BCG}} \in x_0 + K_n(r_0, A),$$

respectively.

Due to the squaring of the polynomial in (4.2), the CGS algorithm usually converges faster than BCG. However, the squaring also amplifies the erratic convergence behavior of BCG, and there are cases where BCG converges, while CGS diverges. Finally, breakdowns—triggered by division by 0—can occur in the CGS algorithm. Actually, in exact arithmetic, CGS breaks down in the nth step if, and only if, BCG breaks down during the nth iteration.

4.2. The general TFQMR method. The nth CGS iterate x_{2n}^{CGS} is defined in (4.2) by squaring the BCG polynomial, and in general, it does not satisfy a minimization or quasi-minimization property. In [12], we proposed a transpose-free Krylov subspace iteration, the TFQMR algorithm, that uses the CGS search directions to generate iterates characterized by a quasi-minimal residual property. The TFQMR algorithm in [12] is derived directly from the CGS process, and like the latter, it can break down. In this section, we present an extension of the TFQMR method that is based on the look-ahead Lanczos polynomials, and thus avoids breakdowns.

Instead of one CGS iterate x_{2n}^{CGS}, the TFQMR method produces two iterates

$$x_m \in x_0 + K_m(r_0, A), \quad m = 2n-1,\, 2n, \tag{4.4}$$

in step n. Therefore, in the sequel, we will use $m = 1, 2, \cdots$, as the iteration index in our description of TFQMR. First, based on the look-ahead Lanczos polynomials given in (2.11), we define vectors

$$u_m := \begin{cases} \dfrac{\varphi_{n-1}(A)\varphi_n(A) r_0}{\|\varphi_{n-1}(A)\varphi_n(A) r_0\|} & \text{if m=2n}, \\ \dfrac{(\varphi_n(A))^2 r_0}{\|(\varphi_n(A))^2 r_0\|} & \text{if m=2n+1}. \end{cases} \tag{4.5}$$

Since each polynomial φ_n is of exact degree n, we have

$$K_m(r_0, A) = \{U_m z \mid z \in \mathbb{C}^m\}, \tag{4.6}$$

where

$$U_m := [\, u_1 \quad u_2 \quad \cdots \quad u_m \,]. \tag{4.7}$$

Furthermore, the vectors $\{u_j\}_{j=1}^{m+1}$ satisfy recurrences of the form

$$AU_m = U_{m+1} F_m, \tag{4.8}$$

where F_m is an $(m+1) \times m$ upper Hessenberg matrix with nonzero subdiagonal elements. In particular, we have

$$\operatorname{rank} F_m = m. \tag{4.9}$$

Finally, note that, in view of (2.11), $\varphi_0(\lambda) \equiv 1$, and thus, by (4.5), $u_1 = r_0/\rho_1$, where $\rho_1 := \|r_0\|$.

By (4.6), the iterates (4.4) can be parametrized in the form

$$x_m = x_0 + U_m z_m, \quad \text{where} \quad z_m \in \mathbb{C}^m, \tag{4.10}$$

and, with (4.8), the corresponding residual vector r_m satisfies

$$r_m = U_{m+1} \Omega_{m+1}^{-1} (f_m - H_m z_m). \tag{4.11}$$

Here, Ω_{m+1} is again an arbitrary scaling matrix of the form (2.16), f_m is defined as in (2.17), and

$$H_m := \Omega_{m+1} F_m. \tag{4.12}$$

In analogy to the definition (2.13)–(2.14) of the standard QMR iterates, the mth TFQMR iterate is given by (4.10) where z_m is chosen so that the Euclidean norm of the second factor in (4.11) is as small as possible, i.e., $z_m \in \mathbb{C}^m$ is the solution of the least-squares problem

$$\|f_m - H_m z_m\| = \min_{z \in \mathbb{C}^m} \|f_m - H_m z\|. \tag{4.13}$$

Note that, in view of (4.9), z_m is uniquely defined by (4.13). We remark that—analogous to the scaling (2.19) in the case of the QMR method—the standard choice for the weights in Ω_{m+1} is

$$\omega_j = 1 \quad \text{for all} \quad j, \tag{4.14}$$

so that all columns of the matrix $U_{m+1}\Omega_{m+1}^{-1}$ in the representation (4.11) of r_m are scaled to have unit length. Finally, we would like to stress that the TFQMR method and the standard QMR method are not mathematically equivalent. In particular, the sets of iterates generated by TFQMR and QMR are different.

4.3. An implementation of TFQMR without look-ahead. Recall that the basis vectors in (4.5) involve the look-ahead Lanczos polynomials. As a result, any robust implementation of TFQMR requires look-ahead in order to avoid breakdowns or near-breakdowns. It is an open question, whether TFQMR with look-ahead can be implemented with only short vector recurrences and with only two matrix-vector multiplications by A per nth iteration. An actual implementation of TFQMR without look-ahead was derived in [12]. In this section, we present this algorithm.

We assume that no breakdowns occur in the CGS algorithm, and we denote by $\{y_j\}_{j=1}^{2n}$ the search directions generated during the first n CGS iterations. In analogy to (4.7), we set

$$Y_m := [\, y_1 \quad y_2 \quad \cdots \quad y_m \,].$$

The columns of Y_m and U_m both span $K_m(r_0, A)$, and thus we have $U_m = Y_m C_m$, where C_m is a nonsingular upper triangular matrix. By rewriting (4.10) and (4.13) in terms of Y_m, it follows that the mth TFQMR iterate is given by

$$x_m = x_0 + Y_m q_m,$$

where $q_m := C_m z_m$ is the unique solution of

$$\|f_m - H_m C_m^{-1} z_m\| = \min_{z \in \mathbb{C}^m} \|f_m - H_m C_m^{-1} z\|. \tag{4.15}$$

It turns out (see [12]) that the matrix $H_m C_m^{-1}$ is lower bidiagonal, and the solution of (4.15) and thus x_m can be easily updated from step to step. Together with the fact that the CGS search directions can be generated by means of short recurrences, it follows that TFQMR without look-ahead can be implemented using only short vector updates.

Of course, in practice, the TFQMR method is combined with preconditioning. Let $M \in \mathbb{C}^{N \times N}$ be a nonsingular preconditioner for the linear system (2.1), and assume that M is given in the form $M = M_1 M_2$. The algorithm is then applied to the preconditioned system

$$A' x' = M_1^{-1} b, \quad \text{where} \quad A' := M_1^{-1} A M_2^{-1} \quad \text{and} \quad x' := M_2 x, \tag{4.16}$$

and it generates approximate solutions x'_m of (4.16). We denote by $x_m := M_2^{-1} x'_m$ the corresponding iterates for the original system (2.1). We remark that usually one chooses $M_1 = I_N$ and $M_2 = M$ (right preconditioning), or $M_1 = M$ and $M_2 = I_N$ (left preconditioning).

Next, we state the resulting preconditioned TFQMR algorithm without look-ahead, for the standard weighting strategy (4.14).

ALGORITHM 4.1 (TFQMR WITHOUT LOOK-AHEAD; WITH WEIGHTS (4.14)).

(1) *Start:*

(a) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $x'_0 = M_2 x_0$.

(b) *Set* $\hat{u}_1 = y_1 = M_1^{-1} b - A' x'_0$, $v_0 = A' y_1$, $d_0 = 0$;

$\tau_0 = \|\hat{u}_1\|$, $\vartheta_0 = 0$, $\eta_0 = 0$.

(c) *Choose* $\tilde{r}_0$ *such that* $\rho_0 = \tilde{r}_0^H \hat{u}_1 \neq 0$.

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

(a) *Set* $\sigma_{n-1} = \tilde{r}_0^H v_{n-1}$, $\alpha_{n-1} = \rho_{n-1}/\sigma_{n-1}$;

$y_{2n} = y_{2n-1} - \alpha_{n-1} v_{n-1}$.

(b) *For* $m = 2n-1, 2n$ *do:*

- *Set* $\hat{u}_{m+1} = \hat{u}_m - \alpha_{n-1} A' y_m$;
- $\vartheta_m = \|\hat{u}_{m+1}\|/\tau_{m-1}$, $c_m = 1/\sqrt{1+\vartheta_m^2}$;
- $\tau_m = \tau_{m-1}\vartheta_m c_m$, $\eta_m = c_m^2 \alpha_{n-1}$;
- $d_m = y_m + (\vartheta_{m-1}^2 \eta_{m-1}/\alpha_{n-1}) d_{m-1}$;
- $x'_m = x'_{m-1} + \eta_m d_m$.
- *If* x'_m *has converged, then set* $x_m = M_2^{-1} x'_m$ *and stop.*

(c) *Set* $\rho_n = \tilde{r}_0^H \hat{u}_{2n+1}$, $\beta_n = \rho_n/\rho_{n-1}$;

$y_{2n+1} = \hat{u}_{2n+1} + \beta_n y_{2n}$;

$v_n = A' y_{2n+1} + \beta_n (A' y_{2n} + \beta_n v_{n-1})$.

We note that the vectors $\hat{u}_m$ in Algorithm 4.1 are rescaled versions of the vectors u_m defined in (4.5). Moreover, the polynomials φ_{n-1} and φ_n in (4.5) correspond to the classical nonsymmetric Lanczos process without look-ahead.

Finally, we remark that it is straightforward to rewrite Algorithm 4.1 in terms of the quantities A, M_1^{-1}, M_2^{-1}, and x_m, instead of A' and x'_m. However, in some cases it is more advantageous to work directly with the preconditioned matrix A'. For example, this is true for SSOR preconditioning, where products with the preconditioned matrix A' can be computed efficiently using Eisenstat's trick [7].

5. Convergence results for QMR and TFQMR. The quasi-minimal residual property of the QMR and TFQMR iterates is strong enough to deduce from it convergence results for both methods, see [18], [14]. In this section, we state these results under some additional weak assumptions. Formulations of the convergence results for the general case can be found in [14] (for nonsingular A) and [17] (for singular A).

Note that, in our description of QMR and TFQMR, we have included the case of singular coefficient matrices A. Indeed, as was shown by Freund and Hochbruck [17], both QMR and TFQMR generate well-defined iterates in the case of singular A. However, like other Krylov subspace methods, the iterates converge to a meaningful solution of (2.1) only for special cases, for instance consistent singular systems with one-dimensional null spaces. This is discussed in detail in [17]. In this section, the coefficient matrix A of (2.1) is always assumed to be nonsingular.

We need some further notation. For any $(n+1) \times n$ matrix H_n, we denote by

$$\tilde{H}_n = [\, I_n \quad 0 \,]\, H_n \tag{5.1}$$

the matrix obtained from H_n by deleting its last row. In exact arithmetic, the look-ahead Lanczos algorithm sketched in § 2.2 terminates after at most N steps, either because the process has found an invariant subspace or due to an incurable breakdown, see, e.g., [26], [16]. We denote by L $(\leq N)$ the termination index of the look-ahead Lanczos algorithm. Let $\lambda_1, \lambda_2, \cdots, \lambda_L$ be the zeros of the last look-ahead Lanczos polynomial φ_L, and let $\tilde{T}_L$ be the last $L \times L$ Lanczos matrix defined by (2.8) and (5.1). It is well known (see, e.g., [26]) that

$$\{\lambda_1, \lambda_2, \cdots, \lambda_L\} = \lambda(\tilde{T}_L) \subseteq \lambda(A). \tag{5.2}$$

For the remainder of this section, we assume that the zeros of φ_L are all distinct. In view of (5.2), this guarantees that the matrix $\tilde{T}_L$ is similar to the diagonal matrix

$$D_T := \operatorname{diag}(\lambda_1, \lambda_2, \cdots, \lambda_L). \tag{5.3}$$

Moreover, we consider the QMR and TFQMR methods only for the standard weighting strategies (2.19) and (4.14), respectively.

After these preliminaries, we can now state the convergence results. First, we consider the standard QMR method.

THEOREM 5.1. *Let L be the termination index of the look-ahead Lanczos algorithm, and assume that the zeros $\lambda_1, \lambda_2, \cdots, \lambda_L$ of φ_L are all distinct. Let D_T be the diagonal matrix defined in* (5.3). *Then, for $n = 1, 2, \cdots, L-1$, the residual vectors r_n generated by the standard QMR method with weights* (2.19) *satisfy:*

$$\begin{aligned} \frac{\|r_n\|}{\|r_0\|} \;\leq\; c_T\, \sqrt{n+1} \min_{\varphi \in \mathcal{P}_n:\, \varphi(0)=1} \; \max_{\lambda \in \lambda(A)} |\varphi(\lambda)|, \\ \text{where} \quad c_T := \min_{X:\; X^{-1}\tilde{T}_L X = D_T} \|X\| \cdot \|X^{-1}\|. \end{aligned} \tag{5.4}$$

We remark that—except for the factor $\sqrt{n+1}$, which reflects the quasi-minimality, and a slightly different constant c_T—the upper bound (5.4) is the same as the standard estimate [36] for algorithms such as GMRES that are based on a true minimal residual property.

Next, we turn to the TFQMR method. As we pointed out in [13], TFQMR terminates at iteration step $m = 2L$. Let $\tilde{F}_{2L}$ be the corresponding last TFQMR matrix as defined by (4.8) and (5.1). From the results in [14], it follows that

$$\{\lambda_1, \lambda_2, \cdots, \lambda_L\} = \lambda(\tilde{F}_{2L}),$$

where each λ_j is an eigenvalue of algebraic multiplicity 2 and of geometric multiplicity 1. Hence $\tilde{F}_{2L}$ has the Jordan canonical form

$$\begin{aligned} J_F &:= \operatorname{diag}(J(\lambda_1), J(\lambda_2), \cdots, J(\lambda_L)), \\ &\quad \text{where} \quad J(\lambda_j) := \begin{bmatrix} \lambda_j & 1 \\ 0 & \lambda_j \end{bmatrix} \quad \text{for all} \quad j. \end{aligned} \tag{5.5}$$

The convergence result for TFQMR can now be formulated as follows.

THEOREM 5.2. *Let L be the termination index of the look-ahead Lanczos algorithm, and assume that the zeros $\lambda_1, \lambda_2, \cdots, \lambda_L$ of φ_L are all distinct. Let J_F be the Jordan matrix defined in* (5.5). *Then, for $m = 1, 2, \cdots, 2L-1$, the residual vectors r_m generated by the TFQMR method with weights* (4.14) *satisfy*:

$$\begin{aligned} \frac{\|r_m\|}{\|r_0\|} &\leq c_F \sqrt{m+1} \min_{\varphi \in \mathcal{P}_m:\, \varphi(0)=1} \max_{\lambda \in \lambda(A)} \left(|\varphi(\lambda)| + |\varphi'(\lambda)|\right), \\ &\quad \textit{where} \quad c_F := \min_{X:\, X^{-1}\tilde{F}_{2L}X = J_F} \|X\| \cdot \|X^{-1}\|. \end{aligned} \tag{5.6}$$

Recall that, in the general case, n steps of the standard QMR method require n matrix-vector multiplications with A and n multiplications with A^T, while m TFQMR iterations involve m matrix-vector products with A only. In other words, with $2n$ matrix-vector multiplications, the QMR and TFQMR methods generate iterates

$$x_n^{\mathrm{QMR}} \in x_0 + K_n(r_0, A) \quad \text{and} \quad x_{2n}^{\mathrm{TFQMR}} \in x_0 + K_{2n}(r_0, A),$$

respectively. However, as the presence of the term $|\varphi'(\lambda)|$ in the upper bound (5.6) indicates, x_{2n}^{TFQMR} is not an optimal quasi-minimal residual iterate in the shifted Krylov subspace $x_0 + K_{2n}(r_0, A)$. Note that, in view of Theorem 5.1, such an optimal quasi-minimal residual iterate would satisfy an estimate of the type (5.4) (with n replaced by $2n$).

6. Computing zeros of residual polynomials. The residual vector r_n corresponding to the nth iterate of any Krylov subspace method can be expressed in terms of polynomials:

$$r_n = \psi_n(A) r_0, \quad \text{where} \quad \psi_n \in \mathcal{P}_n \quad \text{and} \quad \psi_n(0) = 1.$$

In general, any $\psi_n \in \mathcal{P}_n$ with $\psi_n(0) = 1$ is called an nth *residual polynomial*. It is well known (see, e.g., [13]) that minimal residual algorithms, such as GMRES, are based on special residual polynomials, namely so-called *kernel polynomials* [38]. In [13], we introduced the more general concept of *quasi-kernel polynomials*, and we showed that both QMR and TFQMR are based on particular instances of quasi-kernel polynomials. The zeros of the GMRES, QMR, and TFQMR residual polynomials yield approximate eigenvalues of A that—in the context of solving linear systems—are often more appropriate than the usual *Ritz values* obtained from the Arnoldi and Lanczos processes. In order to distinguish the two sets of approximate eigenvalues, we refer to the zeros of residual polynomials as *pseudo-Ritz values*. We remark that the notation pseudo-Ritz values was first introduced in [10] for the zeros of kernel polynomial associated with Hermitian indefinite linear systems.

In this section, we describe an algorithm for computing the GMRES, QMR, and TFQMR pseudo-Ritz values. First, we recall the following result.

THEOREM 6.1 ([13, COR. 5.3]). *The nth QMR (with weights (2.19)), TFQMR (with weights (4.14)), and GMRES residual polynomials are given by*

$$\psi_n(\lambda) \equiv \frac{\det\left(H_n^H H_n - \lambda \tilde{H}_n^H\right)}{\det\left(H_n^H H_n\right)}.$$

Here, in the case of QMR and TFQMR, H_n is defined in (2.15) and (4.12) (both with $\Omega_{n+1} := I_{n+1}$), respectively; and in the case of GMRES, H_n is the $(n+1) \times n$ upper Hessenberg matrix generated by n steps of the Arnoldi process (see, e.g., [36]). In all three cases, $\tilde{H}_n$ is the $n \times n$ matrix defined by (5.1).

In view of Theorem 6.1, the QMR, TFQMR, and GMRES pseudo-Ritz values can be computed via solution of a generalized eigenvalue problem of the form

$$H_n^H H_n z = \lambda \tilde{H}_n^H z, \quad z \in \mathbb{C}^n, \ z \neq 0. \tag{6.1}$$

However, for numerical reasons, it is advisable to avoid forming the matrix $H_n^H H_n$ in (6.1). Instead, as described in [13], one rewrites (6.1) using a QR decomposition of H_n of the form

$$Q_n H_n = \begin{bmatrix} R_n \\ 0 \end{bmatrix}. \tag{6.2}$$

Here, R_n is an upper triangular matrix, and Q_n is a unitary matrix of the form

$$Q_n = G_n \begin{bmatrix} G_{n-1} & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} G_{n-2} & 0 \\ 0 & I_2 \end{bmatrix} \cdots \begin{bmatrix} G_1 & 0 \\ 0 & I_{n-1} \end{bmatrix}, \tag{6.3}$$

where, for $j = 1, 2, \cdots, n$,

$$(6.4) \qquad G_j = \begin{bmatrix} I_{j-1} & 0 & 0 \\ 0 & c_j & -\overline{s_j} \\ 0 & s_j & c_j \end{bmatrix}, \quad \text{with} \quad c_j \in \mathbb{R},\ s_j \in \mathbb{C},\ c_j^2 + |s_j|^2 = 1.$$

With (6.2), the generalized eigenvalue problem (6.1) can be reformulated as follows:

$$(6.5) \qquad R_n z = \lambda \tilde{Q}_n z, \quad z \in \mathbb{C}^n,\ z \neq 0,$$

where

$$(6.6) \qquad \tilde{Q}_n = \begin{bmatrix} I_{n-1} & 0 \\ 0 & c_n \end{bmatrix} G_{n-1} \begin{bmatrix} G_{n-2} & 0 \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} G_1 & 0 \\ 0 & I_{n-2} \end{bmatrix}.$$

Hence, we have the following algorithm for computing pseudo-Ritz values.

ALGORITHM 6.2 (FOR COMPUTING QMR, TFQMR, OR GMRES PSEUDO-RITZ VALUES).

Input: *Let H_n be the $(n+1) \times n$ Hessenberg matrix given by* (2.15) *with $\Omega_{n+1} = I_{n+1}$ (for QMR), by* (4.12) *with $\Omega_{n+1} = I_{n+1}$ (for TFQMR), or by the Arnoldi process (for GMRES).*

(a) *Compute the QR decomposition* (6.2) *of H_n with Q_n of the form* (6.3)–(6.4).

(b) *Compute $\tilde{Q}_n$ using* (6.6).

(c) *Apply the QZ algorithm* (*see, e.g.,* [24, Chap. 7.7]) *to the generalized eigenvalue problem* (6.5). *The resulting eigenvalues are the pseudo-Ritz values corresponding to QMR, TFQMR, or GMRES, respectively.*

7. The QMR squared algorithm.

In this section, we discuss yet another transpose-free QMR approach based on "squaring" the standard QMR method, in analogy to the derivation of the CGS residuals (4.2) by squaring the BCG residuals (4.3).

7.1. The basic approach.

For the moment, we assume that the weights (2.16) are arbitrary, but fixed. Let x_n^{QMR} be the nth QMR iterate given by (2.13)–(2.14), corresponding to these weights. Let $\psi_n \in \mathcal{P}_n$ with $\psi_n(0) = 1$ be the associated residual polynomial:

$$(7.1) \qquad r_n^{\mathrm{QMR}} = b - Ax_n^{\mathrm{QMR}} = \psi_n(A) r_0.$$

We define the nth iterate x_{2n} of the *quasi-minimal residual squared* algorithm (QMRS) by squaring the relation (7.1). Thus we have,

$$r_{2n} = b - Ax_{2n} = \left(\psi_n(A)\right)^2 r_0,$$

and the iterate itself is given by

$$x_{2n} = x_0 + \phi_{2n}(A) r_0, \quad \text{where} \quad \phi_{2n}(\lambda) \equiv \frac{1 - \psi_n^2(\lambda)}{\lambda}. \tag{7.2}$$

Of course, it remains to give an actual algorithm for computing the QMRS iterates defined in (7.2). It is an open question whether there exists an implementation of QMRS with look-ahead that uses only short vector updates and only two matrix-vector products per nth step.

7.2. An implementation of QMRS without look-ahead. In [20], Freund and Szeto derived an implementation of QMRS without look-ahead, and we now briefly describe this algorithm. The key observation here is that the QMR and BCG residuals are connected by a relation of the form

$$\begin{aligned} r_n^{\mathrm{QMR}} &= \mu_n r_{n-1}^{\mathrm{QMR}} + \nu_n r_n^{\mathrm{BCG}}, \\ &\quad \text{where} \quad 0 \leq \mu_n \leq 1 \quad \text{and} \quad \nu_n := 1 - \mu_n, \end{aligned} \tag{7.3}$$

which is due to Freund and Nachtigal [18, Prop. 4.1]. By rewriting (7.3), by means of (4.3) and (7.1), in terms of residual polynomials, we obtain

$$\psi_n(\lambda) \equiv \mu_n \psi_{n-1}(\lambda) + \nu_n \psi_n^{\mathrm{BCG}}(\lambda). \tag{7.4}$$

By squaring the relation (7.4) and by using (7.2) and (4.2), one easily verifies that the QMRS iterates satisfy

$$\begin{aligned} x_{2n} &= \mu_n^2 x_{2n-2} + \nu_n^2 x_{2n}^{\mathrm{CGS}} + 2\mu_n \nu_n f_n, \\ &\quad \text{where} \quad f_n := x_0 + A^{-1}(I_N - \psi_{n-1}(A)\psi_n^{\mathrm{BCG}}(A)) r_0. \end{aligned} \tag{7.5}$$

It turns out (see [20] for details) that the auxiliary vector f_n can be updated with short vector recursions, at the expense of one extra matrix-vector multiplication with A at each iteration. Recall from [37] that the CGS iterates x_{2n}^{CGS} can be generated by means of short vector updates and two matrix-vector products with A per step. Thus, in view of (7.5), the computation of the QMRS iterates x_{2n} involves only short vector updates, but three matrix-vector products with A per nth iteration. Note that both CGS and TFQMR involve only two multiplications by A at the nth step.

So far, we have not specified how to choose the weights ω_j in (2.16) for the QMRS algorithm. Ideally, as in the standard QMR method, we would like to use unit weights (2.19). It turns out that this is not possible. Indeed, in [20], we showed that the computation of the recurrence coefficients μ_n and ν_n involves the previous recurrence coefficients, as well as a factor of the form

$$\omega_{n+1}^2 \| r_n^{\mathrm{BCG}} \|^2. \tag{7.6}$$

Hence, the standard choice $\omega_{n+1} = 1$ would require the knowledge of $\|r_n^{\mathrm{BCG}}\|$. However, in the QMRS algorithm, only the squared residual vector r_{2n}^{CGS} is available, instead of r_n^{BCG}. Hence the choice $\omega_{n+1} = 1$ is not feasible, and instead we set

$$\omega_{n+1} := \frac{\sqrt{\|r_n^{\mathrm{CGS}}\| \cdot \|r_0\|}}{\|r_n^{\mathrm{BCG}}\|}. \tag{7.7}$$

Note that, with (7.7), the factor (7.6) reduces to

$$\|r_n^{\mathrm{CGS}}\| \cdot \|r_0\|,$$

and consequently, the coefficients μ_n and ν_n corresponding to the weights (7.7) can be updated in the course of the QMRS algorithm. Note that, with (4.2)–(4.3), we can express (7.7) in the form

$$\omega_{n+1} := \frac{\sqrt{\|(\varphi_n^{\mathrm{BCG}}(A))^2 r_0\| \cdot \|r_0\|}}{\|\varphi_n^{\mathrm{BCG}}(A) r_0\|},$$

which suggests that $\omega_{n+1} \approx 1$.

Finally, we state the resulting implementation of QMRS without look-ahead and with weights (7.7). As in § 4.3, we formulate the algorithm in terms of the preconditioned linear system (4.16).

Algorithm 7.1 (QMRS without look-ahead; with weights (7.7)).

(1) *Start:*

(a) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $z_0 = x_0' = M_2 x_0$.

(b) *Set* $s_0 = t_0 = p_0 = u_0 = w_0 = M_1^{-1} b - A' x_0'$, $v_0 = A' p_0$, $f_0 = 0$;
$\xi_0 = \|w_0\|^2$, $\eta_0 = 0$, $\mu_0 = 0$, $\nu_0 = 1$.

(c) *Choose* $\tilde{r}_0$ *such that* $\rho_0 = \tilde{r}_0^H w_0 \neq 0$.

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

(a) *Set* $\sigma_{n-1} = \tilde{r}_0^H v_{n-1}$, $\alpha_{n-1} = \rho_{n-1}/\sigma_{n-1}$;
$g_n = u_{n-1} - \alpha_{n-1} v_{n-1}$, $f_n = \mu_{n-1} f_{n-1} + \nu_{n-1} z_{n-1} + \alpha_{n-1} s_{n-1}$;
$z_n = z_{n-1} + \alpha_{n-1}(u_{n-1} + g_n)$, $w_n = w_{n-1} - \alpha_{n-1} A'(u_{n-1} + g_n)$.

(b) *Set* $\eta_n = \|w_n\| \cdot \|r_0\| / \xi_{n-1}$, $\nu_n = 1/(1+\eta_n)$, $\mu_n = \eta_n \nu_n$, $\xi_n = \xi_{n-1}\mu_n$;
$x_{2n}' = \mu_n^2 x_{2n-2}' + 2\mu_n \nu_n f_n + \nu_n^2 z_n$.
If x_{2n}' *has converged, then set* $x_{2n} = M_2^{-1} x_{2n}'$ *and stop.*

(c) *Set* $\rho_n = \tilde{r}_0^H w_n$, $\beta_n = \rho_n / \rho_{n-1}$;
$u_n = w_n + \beta_n g_n$, $p_n = u_n + \beta_n(g_n + \beta_n p_{n-1})$;
$v_n = A' p_n$, $y_n = t_{n-1} - \alpha_{n-1} A' s_{n-1}$;
$t_n = \nu_n w_n + \mu_n y_n$, $s_n = t_n + \beta_n(\nu_n g_n + \mu_n s_{n-1})$.

We remark that, for simplicity, we have set $w_n := M_1^{-1} r_{2n}^{\mathrm{CGS}}$ and $z_n := M_2 x_{2n}^{\mathrm{CGS}}$ in Algorithm 7.1.

8. Numerical examples. In this section, we present numerical results for a few examples. The figures always show the relative residual norms $\|r_n\|/\|r_0\|$ (for QMR Algorithm 3.1 and standard QMR), respectively $\|r_{2n}\|/\|r_0\|$ (for TFQMR, QMRS, and CGS), plotted versus the iteration index n.

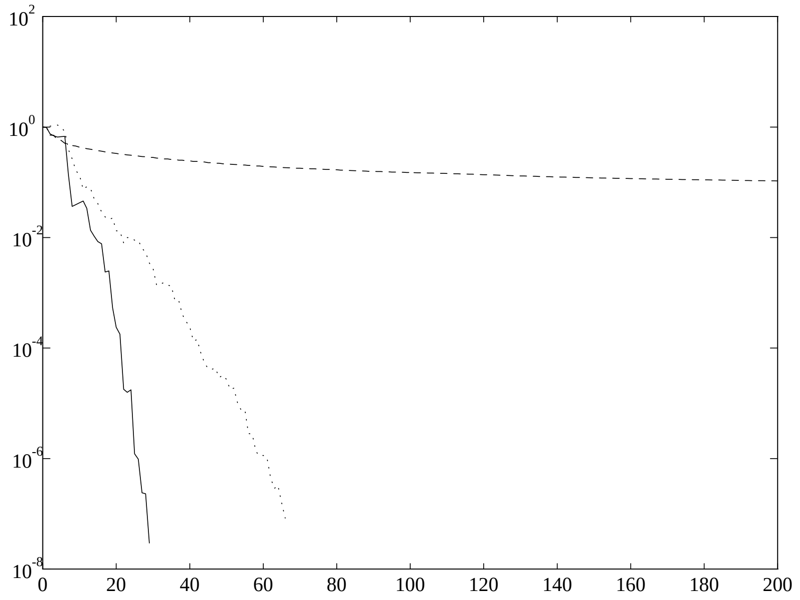

FIG. 8.1. *Convergence curves for Example 1.*

8.1. Example 1. Our first example is a real linear system (3.6) with a nonsingular symmetric indefinite coefficient matrix T of the form (3.7). The order of T is $N = 1000$, and the upper left block H in (3.7) is a positive definite 600×600 matrix. The matrix T is highly indefinite with 600 positive and 400 negative eigenvalues. For this system we obtained a symmetric indefinite preconditioner by computing a Bunch-Parlett factorization [2] of the form

$$M = LDL^T \tag{8.1}$$

of an approximation M to T. Here L is a unit lower triangular matrix and D is a nonsingular block diagonal matrix with 1×1 and 2×2 blocks. We ran the ideal transpose-free QMR Algorithm 3.1 with $J = I_N$ and preconditioner M, and we compared it to MINRES [33]. Recall that MINRES can only be used with a symmetric positive definite preconditioner $\hat{M}$. Following a proposal made in [23], we used the matrix $\hat{M} = L\hat{D}L^T$ that is obtained from (8.1) by constructing a block diagonal matrix $\hat{D}$ whose eigenvalues are just the absolute values of those of the indefinite matrix D. The resulting convergence curves for QMR (solid line) and MINRES (dotted line) are shown in Fig. 8.1. In this example, QMR converges more than twice as fast as MINRES. Note that both algorithms have the same work and storage requirements per iteration.

If no preconditioning is used, then the QMR Algorithm 3.1 and MINRES are mathematically equivalent. In Fig. 8.1, we also plotted the residual norm for a run of Algorithm 3.1 without look-ahead (dashed line). Note that the method does not converge in a reasonable number of iterations. We remark that all Krylov subspace iterations typically converge very slowly for highly indefinite coefficient matrices.

The next two examples are linear systems (2.1) with nonsingular nonsymmetric coefficient matrices. In both cases, we ran the TFQMR Algorithm 4.1, the QMRS Algorithm 7.1, and CGS.

8.2. Example 2. This example is the SHERMAN5 matrix from the Harwell-Boeing set of sparse test matrices [6]. This is a matrix of order $N = 3312$, and it has 20793 nonzero elements. This example was run without preconditioning. The convergence curves for QMRS (solid line), TFQMR (dashed line), and CGS (dotted line) are shown in Fig. 8.2. Note that CGS does diverge, while QMRS and TFQMR converge.

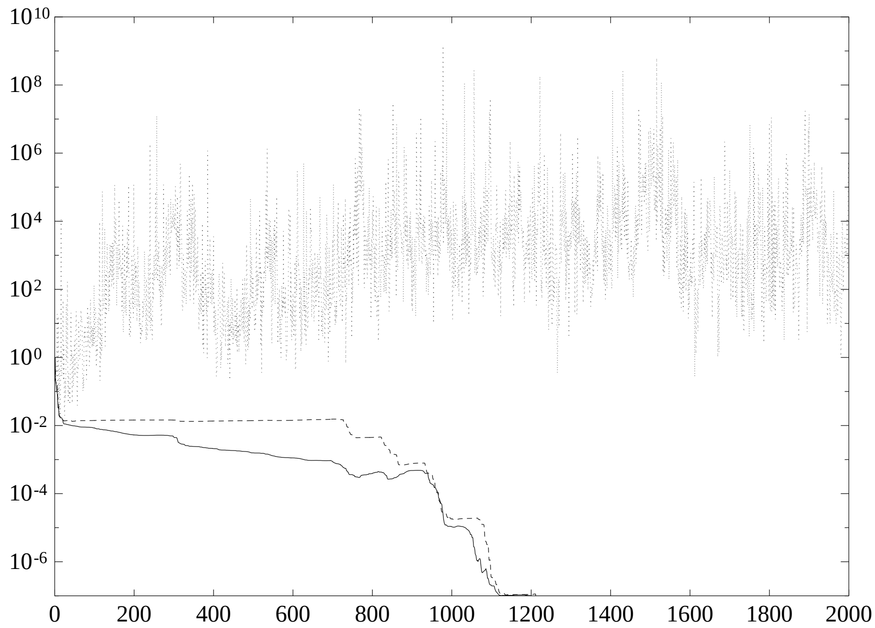

FIG. 8.2. *Convergence curves for Example 2.*

8.3. Example 3. This example is a linear system arising from the partial differential equation

$$-\Delta u + 40\left(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y} + z\frac{\partial u}{\partial z}\right) - 240u = f \quad \text{on} \quad (0,1)\times(0,1)\times(0,1).$$

We discretize this equation using centered differences on a uniform $25 \times 25 \times 25$ grid with mesh size $h = 1/26$. The resulting linear system has a sparse coefficient matrix A of order $N = 15625$ with 105625 nonzero elements. For this example, we also ran the standard QMR algorithm. All algorithms were run with a variant [18] of an ILU preconditioner due to Saad [35]. The convergence curves for QMRS (solid line), TFQMR (dashed line), CGS (dotted line), and standard QMR (dash-dotted line) are shown in Fig. 8.3.

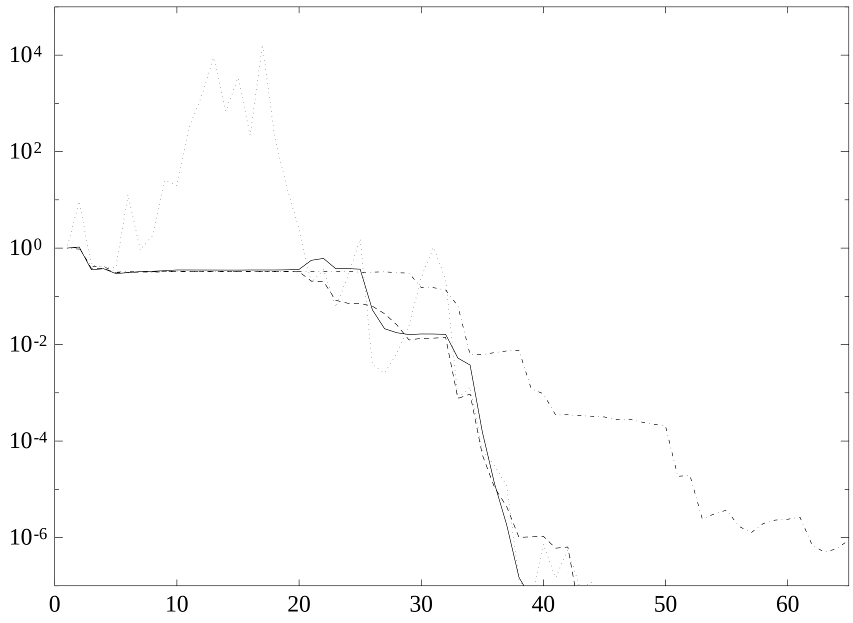

FIG. 8.3. *Convergence curves for Example 3.*

9. Concluding remarks and open problems. The standard QMR method for solving general non-Hermitian linear systems requires matrix-vector products with both the coefficient matrix and its transpose. We have discussed various QMR-type approaches that do not involve the transpose.

We have shown that, in principle, it is always possible to eliminate the need for the transpose in the standard QMR method by choosing special starting vectors. However, the resulting ideal transpose-free QMR method is practical only for special cases, such as complex symmetric, symmetric indefinite, or Toeplitz systems. We have then described two transpose-free QMR schemes for general non-Hermitian matrices, the TFQMR and QMRS methods, and we presented implementations of both approaches for the case that no look-ahead steps are taken. We stress that standard QMR, TFQMR, and QMRS are not mathematically equivalent. In particular, the sets of iterates generated by the three algorithms are different in general. Our numerical experiments with TFQMR and QMRS show that both algorithms usually display a similar convergence behavior. However, since QMRS requires three matrix-vector multiplications with the coefficient matrix per iteration, compared to only two multiplications per TFQMR iteration, the TFQMR algorithm appears to be preferable to QMRS.

We remark that Chan, de Pillis, and van der Vorst [4] have proposed yet another transpose-free QMR algorithm. Their scheme is mathematically equivalent to the standard QMR method without look-ahead. The use of the transpose is avoided by generating the Lanczos tridiagonal matrix T_n in (2.8) by means of a squared transpose-free Lanczos process. Once T_n is computed, the standard Lanczos vectors $\{v_j\}_{j=1}^{n+1}$ are obtained by running the recurrences summarized in the first identity in (2.8). Finally, the QMR iterates are then computed as in the standard QMR algorithm, based on their definition (2.13)–(2.14). Like QMRS, the resulting transpose-free QMR scheme also requires three matrix-vector multiplications per step, and thus it is not competitive with TFQMR. Furthermore, the underlying squared Lanczos process appears to be less robust than the standard Lanczos algorithm, and there are cases where this transpose-free QMR algorithm converges much slower than standard QMR or where it even diverges.

We have defined the TFQMR and QMRS methods in the general setting of the look-ahead Lanczos process. However, actual implementations were only given for TFQMR and QMRS without look-ahead, and it cannot be excluded that exact breakdowns or near-breakdowns occur in both algorithms. We remark that, in exact arithmetic, the conditions for an exact breakdown in CGS, TFQMR, or QMRS are equivalent. To obtain robust implementations of these three methods, it is crucial to incorporate look-ahead steps that allow to skip over those iterations in which exact or near-breakdowns would occur. It is an open question, whether such look-ahead versions of CGS and TFQMR can be implemented with short vector updates and with only two matrix-vector multiplications per iteration. We remark that the problem of avoiding exact breakdowns in the CGS algorithm was studied in [1].

It was observed that Krylov subspace iterations based on the Lanczos process are typically numerically more robust than their transpose-free counterparts. While it is certainly attractive to eliminate the need for matrix-vector products with the transpose, it appears that the use of the transpose enhances the numerical stability of Lanczos-based Krylov subspace iterations. This phenomenon is not at all understood, and we feel that more research is needed to clarify the role of the transpose.

Acknowledgment. The author would like to thank Tedd Szeto for his help with the numerical examples in § 8.2 and § 8.3.

REFERENCES

[1] C. Brezinski and H. Sadok, *Avoiding breakdown in the CGS algorithm*, Numer. Algorithms, 1 (1991), pp. 199–206.

[2] J. R. Bunch and B. N. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8 (1971), pp. 639–655.

[3] R. H. Chan and G. Strang, *Toeplitz equations by conjugate gradients with circulant preconditioner*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 104–119.

[4] T. F. Chan, L. de Pillis, and H. A. van der Vorst, *A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems*, Tech. Report CAM 91-17, Department of Mathematics, University of California, Los Angeles, California, October 1991.

[5] P. Concus, G. H. Golub, and D. P. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in *Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds.*, Academic Press, New York, 1976, pp. 309–332.

[6] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Softw., 15 (1989), pp. 1–14.

[7] S. C. Eisenstat, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 1–4.

[8] V. Faber and T. Manteuffel, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal., 21 (1984), pp. 352–362.

[9] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in *Proc. Dundee Conference on Numerical Analysis, 1975, Lecture Notes in Mathematics 506, G. A. Watson, ed.*, Springer, Berlin, 1976, pp. 73–89.

[10] R. W. Freund, *Pseudo-Ritz values for indefinite Hermitian matrices*, Tech. Report 89.33, RIACS, NASA Ames Research Center, Moffett Field, California, August 1989.

[11] ——————, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 425–448.

[12] ——————, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Stat. Comput. (to appear).

[13] ——————, *Quasi-kernel polynomials and their use in non-Hermitian matrix iterations*, J. Comput. Appl. Math. (to appear).

[14] ——————, *Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations*, in *Numerical Methods of Approximation Theory, D. Braess and L. L. Schumaker, eds.*, Birkäuser, Basel, 1992 (to appear).

[15] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica, 1 (1992), pp. 57–100.

[16] 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. Stat. Comput. (to appear).

[17] R. W. Freund and M. Hochbruck, *On the use of two QMR algorithms for solving singular systems and applications in Markov chain modeling*, Tech. Report 91.25, RIACS, NASA Ames Research Center, Moffett Field, California, December 1991.

[18] 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.

[19] ——————, *An implementation of the QMR method based on coupled two-term recurrences*, Numerical Analysis Manuscript 92–06, AT&T Bell Laboratories, Murray Hill, New Jersey, July 1992.

[20] R. W. Freund and T. Szeto, *A quasi-minimal residual squared algorithm for non-Hermitian linear systems*, Tech. Report 91.26, RIACS, NASA Ames Research Center, Moffett Field, California, December 1991.

[21] R. W. Freund and H. Zha, *A look-ahead algorithm for the solution of general Hankel systems*, Numer. Math. (to appear).

[22] ——————, *Simplifications of the nonsymmetric Lanczos process and a new algorithm for solving symmetric indefinite linear systems*, Tech. Report, RIACS, NASA Ames Research Center, Moffett Field, California, 1992.

[23] P. E. Gill, W. Murray, D. B. Ponceleón, and M. A. Saunders, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 292–311.
[24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 2nd ed., 1989.
[25] M. D. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows*, Academic Press, San Diego, 1989.
[26] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 594–639.
[27] M. R. Hestenes and E. L. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.
[28] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.
[29] W. D. Joubert and D. M. Young, *Necessary and sufficient conditions for the simplification of generalized conjugate-gradient algorithms*, Linear Algebra Appl., 88/89 (1987), pp. 449–485.
[30] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[31] 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.
[32] ———, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.
[33] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[34] 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.
[35] Y. Saad, *ILUT: a dual threshold incomplete LU factorization*, Tech. Report UMSI 92/38, University of Minnesota Supercomputer Institute, Minneapolis, Minnesota, March 1992.
[36] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[37] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36–52.
[38] E. L. Stiefel, *Kernel polynomials in linear algebra and their numerical applications*, U.S. National Bureau of Standards, Applied Mathematics Series, 49 (1958), pp. 1–22.
[39] V. V. Voevodin, *The question of non-self-adjoint extension of the conjugate gradients method is closed*, USSR Comput. Math. and Math. Phys., 23 (1983), pp. 143–144.
[40] M. H. Wright, *Interior methods for constrained optimization*, Acta Numerica, 1 (1992), pp. 341–407.