A QUASI-MINIMAL RESIDUAL SQUARED ALGORITHM FOR NON-HERMITIAN LINEAR SYSTEMS

ROLAND W. FREUND* AND TEDD SZETO†

Abstract. Freund and Nachtigal have recently proposed the quasi-minimal residual algorithm (QMR) for solving general nonsingular non-Hermitian linear systems. The QMR method is based on the nonsymmetric Lanczos process and thus, like the latter, requires matrix-vector products with the coefficient matrix of the linear system, as well as with its transpose. In this paper, a variant of QMR is developed that does not involve the transpose of the coefficient matrix. Since its iterates are derived by squaring the residual polynomials of the standard QMR method, the proposed transpose-free scheme is called the quasi-minimal residual squared algorithm (QMRS). Results of numerical experiments are reported.

Key words. linear systems, non-Hermitian matrices, quasi-minimal residual method, biconjugate gradient algorithm, conjugate gradients squared

AMS(MOS) subject classifications. 65F10, 65N20

1. Introduction. One of the most frequently encountered tasks in numerical computations is the solution of systems of linear equations

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

Often, the coefficient matrix A is large, but sparse; for example, this is the case for linear systems arising in the numerical treatment of partial differential equations. A natural way to exploit the sparsity of A in the solution process is to use iterative techniques that involve the coefficient matrix A only in the form of matrix-vector products. One of the most powerful schemes of this type is the classical conjugate gradient algorithm (CG) of Hestenes and Stiefel [12], which is a method for the case of Hermitian positive definite coefficient matrices A. The "natural" extension of CG for linear systems (1.1) with general nonsingular coefficient matrices A is the biconjugate gradient method (BCG), which goes back to Lanczos [14]. Unfortunately, there are two problems with the original BCG algorithm: (i) typically, the convergence behavior is rather erratic with large oscillations in the residual norm; (ii) breakdowns—more precisely, division by 0—may occur.

* RIACS, Mail Stop T041–5, NASA Ames Research Center, Moffett Field, California 94035. The research of this author was supported by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association.

† Department of Mathematics, University of California, Los Angeles, California 90024. The research of this author was supported in part by the Office of Naval Research under contract N00014-90-J-1695 and by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association.

Recently, Freund [4, 6], for the special case of complex symmetric matrices, and Freund and Nachtigal [10] for the case of general non-Hermitian linear systems, have proposed a BCG-like method, the quasi-minimal residual algorithm (QMR), that overcomes both problems (i) and (ii) of BCG. The QMR iterates are defined by a quasi-minimization of the residual norm, which leads to smooth convergence curves. Moreover, by using look-ahead techniques, the QMR algorithm avoids possible breakdowns.

Both the BCG and the QMR algorithm requires per iteration one matrix-vector product with the coefficient matrix A, and one product with its transpose A^T. This is a disadvantage for certain applications, where A^T is not readily available. Freund and Zha [11] have shown that, in principle, the transpose can always be eliminated by choosing special starting vectors. However, this approach is practical only for special cases, as, e.g., complex symmetric linear systems [4, 6]; in general, there is a need for transpose-free BCG-like schemes. The first method of this type was proposed by Sonneveld [16]. His conjugate gradients squared algorithm (CGS) is derived from BCG, by rewriting BCG in terms of polynomials and then squaring the BCG residual polynomials. Recently, various other transpose-free BCG-type algorithms have been proposed; we refer the reader to [8] for an overview. Two of these methods are quasi-minimal residual approaches. The first is Freund's transpose-free QMR algorithm (TFQMR) [5], which uses basis vectors from CGS to generate iterates characterized by a quasi-minimal residual property. We note that the TFQMR iterates are different from the ones produced by standard QMR. The second transpose-free QMR-type algorithm was proposed by Chan, de Pillis, and Van der Vorst [1]. Their method is mathematically equivalent to standard QMR without look-ahead, but it produces the QMR iterates using matrix-vector products with A only.

In this paper, we propose a new transpose-free method, the quasi-minimal residual squared algorithm (QMRS), based on standard QMR without look-ahead. Similar to the derivation of CGS from BCG, the QMRS scheme is obtained by squaring the QMR residual polynomials. Like standard QMR, the resulting QMRS algorithm converges smoothly, in contrast to CGS, which typically amplifies the erratic convergence behavior of BCG.

The remainder of this paper is organized as follows. In § 2, we briefly review the standard QMR algorithm. In § 3, we show how the iterates of QMR without look-ahead can be obtained directly from classical BCG. In § 4, we then derive the QMRS algorithm. In § 5, a few results of numerical experiments are reported. Finally, in § 6, we make some concluding remarks.

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, M^T and M^H denote the transpose and conjugate transpose of a matrix M, respectively. The vector norm $\|x\| = \sqrt{x^H x}$ is always the Euclidean norm. We denote by

$$\mathcal{P}_n := \{\varphi(\lambda) \equiv \sigma_0 + \sigma_1\lambda + \cdots + \sigma_n\lambda^n \;\Big|\; \sigma_0, \sigma_1, \ldots, \sigma_n \in \mathbb{C}\}$$

the set of all complex polynomials of degree at most n. We use the notation

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

for the nth Krylov subspace of $\mathbb{C}^N$ generated by $c \in \mathbb{C}^N$ and the $N \times N$ matrix B. Note that

$$K_n(c, B) = \{\varphi(B)c \,\big|\, \varphi \in \mathcal{P}_{n-1}\}. \tag{1.2}$$

2. The QMR Algorithm. We are interested in solving linear systems (1.1), where from now on it is always assumed that A is a nonsingular, in general non-Hermitian, $N \times N$ matrix. In the following, $x_0 \in \mathbb{C}^N$ is an arbitrary initial guess for the solution of (1.1) and $r_0 := b - Ax_0$ is the corresponding residual vector.

The standard QMR algorithm is based on a look-ahead variant of the classical nonsymmetric Lanczos process [13] that has been developed by Freund, Gutknecht, and Nachtigal [9]. Starting with $v_1 = r_0/\|r_0\|$ and an arbitrary nonzero vector $w_1 \in \mathbb{C}^N$, the look-ahead Lanczos algorithm generates two sequences of vectors $v_1, v_2, \ldots$, and $w_1, w_2, \ldots$, such that, for $n = 1, 2, \ldots$,

$$\begin{aligned} \operatorname{span}\{v_1, v_2, \ldots, v_n\} &= K_n(r_0, A), \\ \operatorname{span}\{w_1, w_2, \ldots, w_n\} &= K_n(w_1, A^T), \end{aligned} \tag{2.1}$$

and the two sequences are block biorthogonal:

$$\left(W^{(j)}\right)^T V^{(k)} = 0 \quad \text{for all} \quad j \neq k, \quad j, k = 1, 2, \ldots, l(n). \tag{2.2}$$

The matrices $W^{(j)}$ and $V^{(k)}$ in (2.2) are obtained by partitioning the first n Lanczos vectors $v_1, \ldots, v_n$ and $w_1, \ldots, w_n$ into blocks according to

$$\begin{aligned} V_n &:= [\, v_1 \quad v_2 \quad \cdots \quad v_n \,] = [\, V^{(1)} \quad V^{(2)} \quad \ldots \quad V^{(l(n))} \,] \\ \text{and} \quad [\, w_1 \quad w_2 \quad \cdots \quad w_n \,] &= [\, W^{(1)} \quad W^{(2)} \quad \ldots \quad W^{(l(n))} \,], \end{aligned} \tag{2.3}$$

respectively. More precisely, the matrices $V^{(k)}$ and $W^{(k)}$ are both of order $N \times h_k$, and their columns are just the Lanczos vectors corresponding to the kth look-ahead step. Here, h_k denotes the length of the kth look-ahead step. Moreover, in (2.3), $l(n)$ denotes the number of look-ahead steps that have been performed after n steps of the Lanczos process. We remark that the algorithm performs mostly standard Lanczos steps, i.e., look-ahead steps of size $h_k = 1$. Typically, only few true look-ahead steps, i.e., steps of size $h_k > 1$, occur, and—except for contrived examples—their size is usually 2, 3, or at most 4. Finally, the crucial point of the Lanczos process is that the vectors satisfying (2.1) and (2.2) can be

generated by means of short recurrences. For the vectors $v_1, \ldots, v_n, v_{n+1}$, these recurrence relations can be written compactly in matrix form as follows:

$$AV_n = V_{n+1} H_n^{(e)}. \tag{2.4}$$

Here $H_n^{(e)}$ is an $(n+1) \times n$ upper Hessenberg matrix, which is also block tridiagonal with square blocks of order h_k on the diagonal. Furthermore, $H_n^{(e)}$ always has full rank n. A relation similar to (2.4) also holds for the vectors $w_1, \ldots, w_{n+1}$. For further details of the look-ahead Lanczos algorithm, we refer the reader to [9].

From now on, we assume that the Lanczos vectors v_j are normalized to have unit length:

$$\|v_j\| = 1, \quad j = 1, 2, \ldots . \tag{2.5}$$

Then, using the quantities V_{n+1} and $H_n^{(e)}$ generated by means of the look-ahead Lanczos algorithm, the standard QMR algorithm produces iterates

$$x_n = x_0 + V_n z_n, \tag{2.6}$$

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

$$\|d_1^{(n+1)} - \Omega_{n+1} H_n^{(e)} z_n\| = \min_{z \in \mathbb{C}^n} \|d_1^{(n+1)} - \Omega_{n+1} H_n^{(e)} z\|. \tag{2.7}$$

Here

$$d_1^{(n+1)} = \omega_1 \|r_0\| \cdot [\, 1 \quad 0 \quad \cdots \quad 0 \,]^T \in \mathbb{R}^{n+1}, \tag{2.8}$$

and

$$\Omega_{n+1} = \mathrm{diag}(\omega_1, \omega_2, \ldots, \omega_{n+1}), \quad \omega_j > 0, \quad j = 1, 2, \ldots, n+1, \tag{2.9}$$

is a still arbitrary scaling matrix. Usually, one chooses unit weights

$$\omega_j \equiv 1 \quad \text{for all} \quad j. \tag{2.10}$$

We remark that, by (2.6) and (2.4), the residual vector corresponding to x_n is given by

$$r_n = V_{n+1} \Omega_{n+1}^{-1} (d_1^{(n+1)} - \Omega_{n+1} H_n^{(e)} z_n). \tag{2.11}$$

Hence, in view of (2.7), the QMR iterates are characterized by a minimization of the second factor in (2.11); this is just the quasi-minimal residual property. Note that, by (2.5), the scaling (2.10) is very natural, in the sense that all columns of $V_{n+1} \Omega_{n+1}^{-1}$ in the representation (2.11) of r_n are treated equally. Finally, we remark that the QMR iterates x_n can be easily updated from step to step. Due to the block tridiagonal structure of $H_n^{(e)}$, this update can be implemented with only short recurrences; see [10] for details.

From now on, we will only consider a simplified version of QMR, namely QMR based on the standard Lanczos algorithm without look-ahead. Note that then $h_k = 1$, $V^{(k)} = v_k$, $W^{(k)} = w_k$ for all k, and $H_n^{(e)}$ is a scalar tridiagonal matrix.

3. An Implementation of QMR Based on BCG. In this section, we derive a new implementation of the QMR algorithm without look-ahead, using the connection of QMR and the classical BCG algorithm.

It is well-known (see, e.g., [10, Section 5]) that the BCG iterates x_n^{BCG} satisfy

$$x_n^{\mathrm{BCG}} = x_0 + V_n \tilde{z}_n, \tag{3.1}$$

where $\tilde{z}_n$ is the solution of the linear system

$$H_n \tilde{z}_n = \|r_0\| e_1^{(n)}, \quad \text{with} \quad e_1^{(n)} := [1 \quad 0 \quad \cdots \quad 0]^T \in \mathbb{R}^n. \tag{3.2}$$

Here H_n denotes the $n \times n$ matrix obtained from the $(n+1) \times n$ Lanczos matrix $H_n^{(e)}$ by deleting the last row. Multiplying (3.2) by the nonsingular scaling matrix Ω_n and using (2.8), we see that the linear system (3.2) is equivalent to

$$\Omega_n H_n \tilde{z}_n = d_1^{(n)}. \tag{3.3}$$

Note that the coefficient matrix $\Omega_n H_n \tilde{z}_n = d_1^{(n)}$ of (3.3) is an upper Hessenberg matrix, and the right-hand side is a multiple of the first unit vector $e_1^{(n)}$. Furthermore, the linear system just consists of the first n equations of the least squares problem (2.7).

Freund has proved a more general lemma that shows how the solutions z_n and z_{n-1} of two successive Hessenberg least squares problems of the type (2.7) are connected with the solution $\tilde{z}_n$ of the corresponding leading square Hessenberg linear system of the type (3.3). By applying his result [5, Lemma 4.1] to (2.7) and (3.3), we obtain the relation

$$z_n = (1 - c_n^2) \begin{bmatrix} z_{n-1} \\ 0 \end{bmatrix} + c_n^2 \tilde{z}_n, \quad n = 1, 2, \ldots, \tag{3.4}$$

where the coefficient c_n is given by

$$\vartheta_n = \frac{1}{\tau_{n-1}} \|d_1^{(n+1)} - \Omega_{n+1} H_n^{(e)} \tilde{z}_n\|, \quad c_n = \frac{1}{\sqrt{1 + \vartheta_n^2}}. \tag{3.5}$$

Moreover, the τ_n's can be updated from step to step, by setting

$$\tau_n = \tau_{n-1} \vartheta_n c_n, \quad \text{where, for } n = 1, \quad \tau_0 := \omega_1 \|r_0\|. \tag{3.6}$$

We remark that c_n is just the cosine of the nth Givens rotation in an QR decomposition of the upper Hessenberg matrix $\Omega_{n+1} H_n^{(e)}$ (see [5, Section 4]).

The norm in the first equation in (3.5) is directly connected with the norm of the BCG residual vector, and it holds

$$\|d_1^{(n+1)} - \Omega_{n+1} H_n^{(e)} \tilde{z}_n\| = \omega_{n+1} \|r_n^{\mathrm{BCG}}\|. \tag{3.7}$$

To verify (3.7), note that, from (3.3) and (2.4), we have

$$d_1^{(n+1)} - \Omega_{n+1} H_n^{(e)} \tilde{z}_n = \zeta_n \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}^T, \quad \zeta_n \in \mathbb{C}, \tag{3.8}$$

and

$$r_n^{\mathrm{BCG}} = V_{n+1} \Omega_{n+1}^{-1} (d_1^{(n+1)} - \Omega_{n+1} H_n^{(e)} \tilde{z}_n) = \frac{\zeta_n}{\omega_{n+1}} v_{n+1}. \tag{3.9}$$

The relation (3.7) then follows by taking norms in (3.8) and (3.9) and by using that, by (2.5), $\|v_{n+1}\| = 1$.

Next, we show how the QMR iterates can be updated by means of quantities from the BCG algorithm. In the following, let $n \geq 1$. With (2.6), (3.4), and (3.1), we obtain that

$$\begin{aligned} x_n &= x_0 + (1 - c_n^2) V_{n-1} z_{n-1} + c_n^2 V_n \tilde{z}_n, \\ &= x_{n-1} - c_n^2 x_{n-1} + c_n^2 x_n^{\mathrm{BCG}}. \end{aligned} \tag{3.10}$$

Now, recall that, in the standard BCG algorithm (see Algorithm 3.1 below), one generates direction vectors q_{n-1} and then obtains the next BCG iterate by an update of the form

$$x_n^{\mathrm{BCG}} = x_{n-1}^{\mathrm{BCG}} + \alpha_{n-1} q_{n-1}, \quad \text{where} \quad \alpha_{n-1} \in \mathbb{C}. \tag{3.11}$$

Furthermore, we set

$$\hat{p}_n := x_n - x_{n-1}. \tag{3.12}$$

In [10, Proposition 5.2], it was shown that

$$x_n^{\mathrm{BCG}} = x_n + \frac{1 - c_n^2}{c_n^2} \hat{p}_n. \tag{3.13}$$

Using (3.12), (3.10), (3.11), (3.13) (with n replaced by $n-1$), and the relation $\vartheta_{n-1}^2 = 1/c_{n-1}^2 - 1$ (see (3.5)), it follows that

$$\begin{aligned} \hat{p}_n &= -c_n^2 x_{n-1} + c_n^2 (x_{n-1}^{\mathrm{BCG}} + \alpha_{n-1} q_{n-1}), \\ &= c_n^2 (x_{n-1}^{\mathrm{BCG}} - x_{n-1}) + c_n^2 \alpha_{n-1} q_{n-1}, \\ &= c_n^2 \vartheta_{n-1}^2 \hat{p}_{n-1} + c_n^2 \alpha_{n-1} q_{n-1}. \end{aligned} \tag{3.14}$$

We note that $\hat{p}_1 = c_1^2 \alpha_0 q_0$, and thus formula (3.14) remains valid for $n = 1$, if one sets $\vartheta_0 := 0$.

By means of the relations (3.12), (3.14), and (3.5)–(3.7), we can update the QMR iterates, provided that the BCG quantities $\|r_n^{\mathrm{BCG}}\|$, α_{n-1}, and q_{n-1} are available. Therefore,

by adding the updates (3.12), (3.14), and (3.5)–(3.7), to the classical BCG algorithm [14, 3], we obtain the following implementation of QMR without look-ahead.

ALGORITHM 3.1. (QMR without look-ahead from BCG)

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $x_0^{\mathrm{QMR}} = x_0^{\mathrm{BCG}} = x_0$.
Set $q_0 = r_0^{\mathrm{BCG}} = r_0 = b - Ax_0$, $\hat{p}_0 = 0$, $\tau_0 = \omega_1\|r_0\|$, $\vartheta_0 = 0$.
Choose $\tilde{r}_0 \in \mathbb{C}^N$, $\tilde{r}_0 \neq 0$, *and set* $\tilde{q}_0 = \tilde{r}_0$, $\rho_0 = \tilde{r}_0^T r_0$.

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

1) *Set* $\sigma_{n-1} = \tilde{q}_{n-1}^T A q_{n-1}$.
If $\sigma_{n-1} = 0$: *stop. Otherwise, compute*

$$\alpha_{n-1} = \rho_{n-1}/\sigma_{n-1}, \quad r_n^{\mathrm{BCG}} = r_{n-1}^{\mathrm{BCG}} - \alpha_{n-1} A q_{n-1}, \quad \tilde{r}_n = \tilde{r}_{n-1} - \alpha_{n-1} A^T \tilde{q}_{n-1}. \tag{3.15}$$

If BCG *iterates are desired, set*

$$x_n^{\mathrm{BCG}} = x_{n-1}^{\mathrm{BCG}} + \alpha_{n-1} q_{n-1}.$$

2) *Compute*

$$\vartheta_n = \frac{\omega_{n+1}\|r_n^{\mathrm{BCG}}\|}{\tau_{n-1}}, \quad c_n = \frac{1}{\sqrt{1+\vartheta_n^2}}, \quad \tau_n = \tau_{n-1}\vartheta_n c_n,$$
$$\hat{p}_n = c_n^2 \vartheta_{n-1}^2 \hat{p}_{n-1} + c_n^2 \alpha_{n-1} q_{n-1}, \quad x_n^{\mathrm{QMR}} = x_{n-1}^{\mathrm{QMR}} + \hat{p}_n.$$

3) *If* $\rho_{n-1} = 0$: *stop. Otherwise, compute*

$$\rho_n = \tilde{r}_n^T r_n^{\mathrm{BCG}}, \quad \beta_n = \rho_n/\rho_{n-1},$$
$$q_n = r_n^{\mathrm{BCG}} + \beta_n q_{n-1}, \quad \tilde{q}_n = \tilde{r}_n + \beta_n \tilde{q}_{n-1}. \tag{3.16}$$

We remark that, except for the additional updates in step 2), this algorithm is just classical BCG.

4. A QMR Squared Algorithm. In view of (1.2), the residual vector $r_n = b - Ax_n$ of any iterate $x_n \in x_0 + K_n(r_0, A)$ is of the form $r_n = \varphi(A)r_0$, where $\varphi \in \mathcal{P}_n$ and $\varphi(0) = 1$. In particular, we have

$$r_n^{\mathrm{BCG}} = \varphi_n(A)r_0, \quad \text{where} \quad \varphi_n \in \mathcal{P}_n, \quad \varphi_n(0) = 1. \tag{4.1}$$

From now on, φ_n always denotes the nth BCG residual polynomial defined by (4.1). Sonneveld's CGS algorithm [16] generates iterates x_n^{CGS} with residual vectors that are obtained by "squaring" (4.1). More precisely, x_n^{CGS} is defined by

$$x_n^{\mathrm{CGS}} \in x_0 + K_{2n}(r_0, A) \quad \text{and} \quad r_n^{\mathrm{CGS}} = b - Ax_{2n}^{\mathrm{CGS}} = (\varphi_n(A))^2 r_0. \tag{4.2}$$

In this section, we derive the QMRS algorithm by squaring the QMR algorithm.

We consider the QMR algorithm without look-ahead, where—at the moment—we still allow arbitrary weights $\omega_j > 0$ in (2.9). Let ϕ_n be the polynomial corresponding to the nth QMR residual vector, i.e.,

$$r_n^{\mathrm{QMR}} = \phi_n(A)r_0, \quad \text{where} \quad \phi_n \in \mathcal{P}_n, \quad \phi_n(0) = 1. \tag{4.3}$$

In analogy to (4.2), we then define the nth iterate x_n^{QMRS} of the quasi-minimal residual squared algorithm (QMRS) by

$$x_n^{\mathrm{QMRS}} \in x_0 + K_{2n}(r_0, A) \quad \text{and} \quad r_n^{\mathrm{QMRS}} = b - Ax_{2n}^{\mathrm{QMRS}} = (\phi_n(A))^2 r_0. \tag{4.4}$$

Note that (4.4) can be rewritten in the form

$$x_n^{\mathrm{QMRS}} = \hat{\phi}_n(A)r_0, \quad \text{where} \quad \hat{\phi}_n(\lambda) \equiv (1/\lambda)(1 - \phi_n^2(\lambda)). \tag{4.5}$$

It remains to derive an actual implementation for computing x_n^{QMRS}.

First, we recall from [10, Proposition 4.1] that the QMR and BCG residuals are connected as follows:

$$r_n^{\mathrm{QMR}} = (1 - c_n^2)r_{n-1}^{\mathrm{QMR}} + \nu_n r_n^{\mathrm{BCG}}. \tag{4.6}$$

Here c_n is the cosine of the nth Givens rotation in an QR decomposition of $\Omega_{n+1}H_n^{(e)}$, and c_n is identical to the quantity defined in (3.5). By rewriting (4.6), by means of (4.3) and (4.1), in terms of polynomials, we obtain

$$\phi_n(\lambda) \equiv \mu_n \phi_{n-1}(\lambda) + \nu_n \varphi_n(\lambda), \tag{4.7}$$

where

$$\mu_n := 1 - c_n^2 \quad \text{and} \quad \nu_n = c_n^2. \tag{4.8}$$

Note that the second equation in (4.8) follows by setting $\lambda = 0$ in (4.7) and by using the fact that $\phi_n(0) = \phi_{n-1}(0) = \varphi_n(0) = 1$. Next, we remark that, in analogy to (4.5), we have

$$x_n^{\mathrm{CGS}} = \hat{\varphi}_n(A)r_0, \quad \text{where} \quad \hat{\varphi}_n(\lambda) \equiv (1/\lambda)(1 - \varphi_n^2(\lambda)). \tag{4.9}$$

Using (4.7) and the fact that, by (4.8), $\mu_n + \nu_n = 1$, one readily verifies that the polynomials $\hat{\phi}_n$, $\hat{\phi}_{n-1}$, and $\hat{\varphi}_n$ in (4.5) and (4.9) are connected as follows:

$$\hat{\phi}_n(\lambda) \equiv \mu_n^2 \hat{\phi}_{n-1}(\lambda) + 2\mu_n\nu_n(1/\lambda)(1 - \phi_{n-1}(\lambda)\varphi_n(\lambda)) + \nu_n^2 \hat{\varphi}_n(\lambda). \tag{4.10}$$

Then, with (4.5) and (4.9), it follows from (4.10) that

$$x_n^{\mathrm{QMRS}} = \mu_n^2 x_{n-1}^{\mathrm{QMRS}} + 2\mu_n\nu_n f_n + \nu_n^2 x_n^{\mathrm{CGS}}, \tag{4.11}$$

where

$$f_n := A^{-1}(I - \phi_{n-1}(A)\varphi_n(A))r_0. \tag{4.12}$$

By means of (4.11), the QMRS iterates can be generated from the standard CGS algorithm, provided that we can also compute the scalar μ_n and the vector f_n.

First, we show how to update f_n. By (4.1) and the second relation in (3.15), one has

$$\varphi_n(\lambda) \equiv \varphi_{n-1}(\lambda) - \alpha_{n-1}\lambda\psi_{n-1}(\lambda), \tag{4.13}$$

where $\psi_{n-1} \in \mathcal{P}_{n-1}$ is given by

$$q_{n-1} = \psi_{n-1}(A)r_0. \tag{4.14}$$

For later use, we note the relation

$$\psi_n(\lambda) \equiv \varphi_n(\lambda) + \beta_n\psi_{n-1}(\lambda), \tag{4.15}$$

which follows by rewriting the update for q_n in (3.16) by means of (4.1) and (4.14) in terms of polynomials. Using (4.7), (4.9) (both with n replaced by $n-1$), and (4.13), one verifies that

$$\begin{aligned}(1/\lambda)(1 - \phi_{n-1}(\lambda)\varphi_n(\lambda)) \equiv{}& \mu_{n-1}(1/\lambda)(1 - \phi_{n-2}(\lambda)\varphi_{n-1}(\lambda)) \\ &+ \nu_{n-1}\hat{\varphi}_{n-1}(\lambda) + \alpha_{n-1}\psi_{n-1}(\lambda)\phi_{n-1}(\lambda).\end{aligned} \tag{4.16}$$

With (4.12) and (4.9), it follows from (4.16) that

$$f_n = \mu_{n-1}f_{n-1} + \nu_{n-1}x_{n-1}^{\mathrm{CGS}} + \alpha_{n-1}s_{n-1}, \tag{4.17}$$

where

$$s_{n-1} := \psi_{n-1}(A)\phi_{n-1}(A)r_0. \tag{4.18}$$

We note that (4.17) remains valid for $n = 1$, if one sets $\mu_0 = \nu_0 = 0$. To show how to update s_n, we set

$$y_n := \phi_{n-1}(A)\varphi_n(A)r_0, \quad t_n := \phi_n(A)\varphi_n(A)r_0, \quad g_n := \psi_{n-1}(A)\varphi_n(A)r_0. \tag{4.19}$$

Then, with (4.4, (4.7), (4.13), (4.15), (4.18), and (4.19), one readily verifies that

$$\begin{aligned} y_n &= t_{n-1} - \alpha_{n-1} A s_{n-1}, \quad t_n = \nu_n r_n^{\mathrm{CGS}} + \mu_n y_n, \\ s_n &= t_n + \beta_n(\nu_n g_n + \mu_n s_{n-1}). \end{aligned} \tag{4.20}$$

The vector g_n also occurs in the CGS process, and its update is part of the standard CGS algorithm.

In remains to show how to obtain μ_n. From (4.8) and (3.5), we have

$$\nu_n = \frac{1}{1+\eta_n} \quad \text{and} \quad \mu_n = \eta_n \nu_n, \quad \text{where} \quad \eta_n := \vartheta_n^2. \tag{4.21}$$

Furthermore, in view of (3.5) and (3.7), the quantity η_n is given by

$$\eta_n = \frac{1}{\xi_{n-1}} \omega_{n+1}^2 \|r_n^{\mathrm{BCG}}\|^2, \quad \text{where} \quad \xi_{n-1} := \tau_{n-1}^2. \tag{4.22}$$

Note that, by (3.6) and (4.21), the ξ_n's can be updated as follows

$$\xi_n = \xi_{n-1} \mu_n. \tag{4.23}$$

So far, we have not specified how to select the weight ω_{n+1} in (4.22). Recall from (2.10) that the natural choice is $\omega_{n+1} = 1$. However, the computation of η_n in (4.22) would then require $\|r_n^{\mathrm{BCG}}\|$, which is not available in squared algorithms. Instead, we set

$$\omega_{n+1} = \frac{\sqrt{\|r_n^{\mathrm{CGS}}\| \cdot \|r_0\|}}{\|r_n^{\mathrm{BCG}}\|}, \tag{4.24}$$

so that (4.22) reduces to

$$\eta_n = \frac{\|r_n^{\mathrm{CGS}}\| \cdot \|r_0\|}{\xi_{n-1}}. \tag{4.25}$$

Note that, for the choice (4.24), the relations (4.1) and (4.2) suggests that

$$\omega_{n+1}^2 = \|(\varphi_n(A))^2 r_0\| \cdot \|r_0\| / \|\varphi_n(A) r_0\|^2 \approx 1.$$

In summary, we have shown that the QMRS algorithm corresponding to the weights (4.24) can be implemented with short vector updates. One only has to add the updates (4.11), (4.17), (4.20), (4.21), (4.23), and (4.25) to the standard CGS algorithm [16]. The resulting QMRS is as follows.

ALGORITHM 4.1. (QMRS corresponding to the weights (4.24))

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $x_0^{\mathrm{QMRS}} = x_0^{\mathrm{CGS}} = x_0$.

Set $s_0 = t_0 = p_0 = u_0 = r_0^{\mathrm{CGS}} = r_0 = b - Ax_0$, $v_0 = Ap_0$, $f_0 = 0$,

$\xi_0 = \|r_0\|^2$, $\eta_0 = 0$, $\mu_0 = \nu_0 = 0$.

Choose $\tilde{r}_0 \in \mathbb{C}^N$, $\tilde{r}_0 \neq 0$, *and set* $\tilde{q}_0 = \tilde{r}_0$, $\rho_0 = \tilde{r}_0^T r_0$.

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

1) *Set* $\sigma_{n-1} = \tilde{r}_0^T v_{n-1}$.

If $\sigma_{n-1} = 0$: *stop. Otherwise, compute*

$$\alpha_{n-1} = \rho_{n-1}/\sigma_{n-1}, \quad g_n = u_{n-1} - \alpha_{n-1} v_{n-1},$$
$$f_n = \mu_{n-1} f_{n-1} + \nu_{n-1} x_{n-1}^{\mathrm{CGS}} + \alpha_{n-1} s_{n-1},$$
$$x_n^{\mathrm{CGS}} = x_{n-1}^{\mathrm{CGS}} + \alpha_{n-1}(u_{n-1} + g_n), \quad r_n^{\mathrm{CGS}} = r_{n-1}^{\mathrm{CGS}} - \alpha_{n-1} A(u_{n-1} + g_n),$$
$$\eta_n = \frac{\|r_n^{\mathrm{CGS}}\| \cdot \|r_0\|}{\xi_{n-1}}, \quad \nu_n = \frac{1}{1+\eta_n}, \quad \mu_n = \eta_n \nu_n, \quad \xi_n = \xi_{n-1}\mu_n,$$
$$x_n^{\mathrm{QMRS}} = \mu_n^2 x_{n-1}^{\mathrm{QMRS}} + 2\mu_n \nu_n f_n + \nu_n^2 x_n^{\mathrm{CGS}}.$$

2) *If* $\rho_{n-1} = 0$: *stop. Otherwise, compute*

$$\rho_n = \tilde{r}_0^T r_n^{\mathrm{CGS}}, \quad \beta_n = \rho_n/\rho_{n-1},$$
$$u_n = r_n^{\mathrm{CGS}} + \beta_n g_{n-1}, \quad p_n = u_n + \beta_n(g_n + \beta_n p_{n-1}), \quad v_n = Ap_n,$$
$$y_n = t_{n-1} - \alpha_{n-1} A s_{n-1}, \quad t_n = \nu_n r_n^{\mathrm{CGS}} + \mu_n y_n,$$
$$s_n = t_n + \beta_n(\nu_n g_n + \mu_n s_{n-1}).$$

We remark that Algorithm 4.1 requires 3 matrix-vector products with A per iteration. This is the same as for the transpose-free implementation of the standard QMR algorithm proposed in [1]. On the other hand, both the CGS algorithm and Freund's TFQMR algorithm [5] involve only 2 matrix-vector products with A per iteration.

5. Numerical Examples. In this section, we present numerical results for two examples. For both examples, we have compared the QMRS Algorithm 4.1, Freund's TFQMR algorithm [5], and CGS. For the second example, we have also included standard QMR. The figures show the relative residual norms $\|r_n\|/\|r_0\|$ plotted versus the iteration number n, for QMRS (solid line), TFQMR (dashed line), and CGS (dotted line). For the second example, the QMR curve (dash-dotted line) is also depicted.

Our first example is the SHERMAN5 matrix from the Harwell-Boeing set of sparse test matrices [2]. This is a matrix of order 3312, and it has 20793 nonzero elements. This example

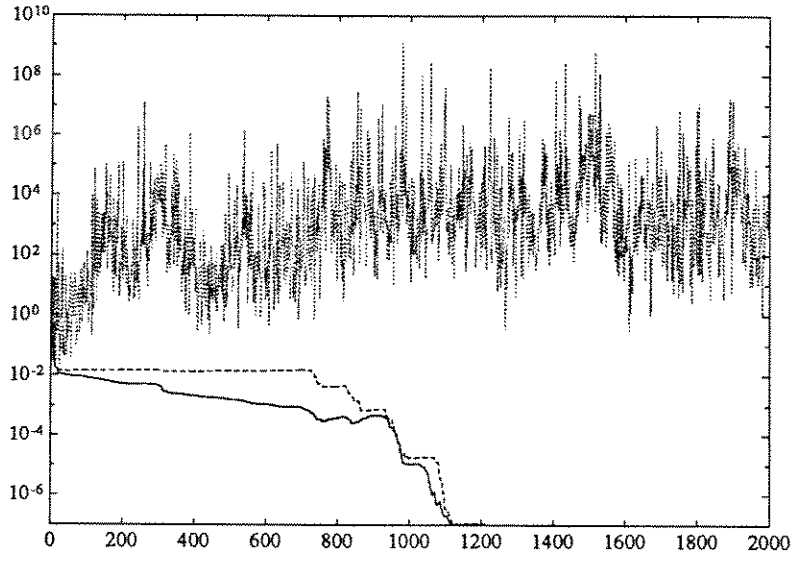

FIG. 5.1. *Convergence curves for example 1*

was run without preconditioning. The convergence curves are shown in Figure 1. Note that CGS does diverge, while QMRS and TFQMR converge.

The second example is a linear system arising from the partial differential equation

$$-\Delta u + 40\left(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y} + z\frac{\partial u}{\partial z}\right) - 240u = f \quad \text{on} \quad (0,1)\times(0,1)\times(0,1). \tag{5.1}$$

We discretize (5.1) 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. This example was run with a variant of an ILU preconditioner due to Saad [15]. The convergence curves are shown in Figure 2.

6. Concluding Remarks. We have shown that the standard QMR algorithm, without look-ahead and with a specific choice of the weighting matrix, can be squared. An implementation of the resulting QMRS algorithm has been presented. Like the transpose-free standard QMR algorithm in [1], the proposed QMRS requires 3 matrix-vector products with A per iteration. This is one multiplication more than in CGS and Freund's TFQMR [5], which both involve only 2 matrix-vector products with A per iteration.

We remark that, at present, none of the proposed transpose-free QMR-type methods, except for Freund and Zha's approach [11] for special cases, incorporates look-ahead, and in

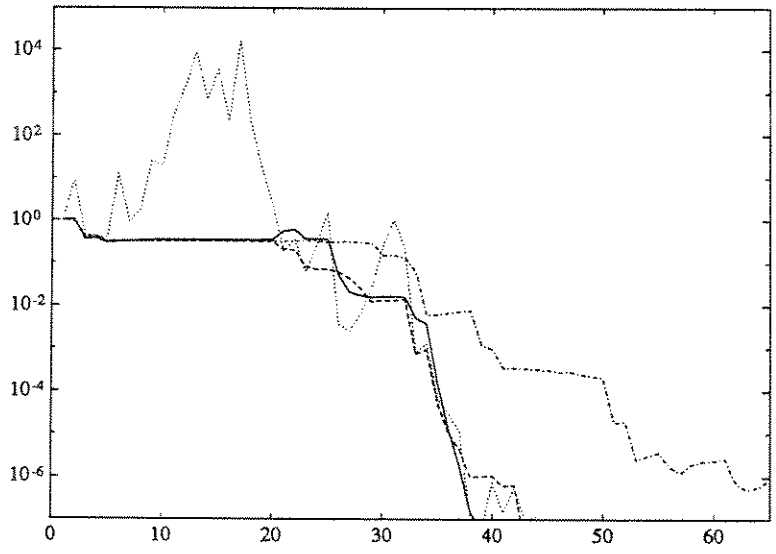

FIG. 5.2. *Convergence curves for example 2*

exact arithmetic, they all breakdown whenever BCG does. A look-ahead version of TFQMR is presently developed by Freund and Nachtigal [7].

REFERENCES

[1] 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*, Technical Report CAM 91-17, Department of Mathematics, University of California, Los Angeles, CA, October 1991.

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

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

[4] R.W. FREUND, *Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices*, RIACS Technical Report 89.54, NASA Ames Research Center, Moffett Field, CA, December 1989.

[5] R.W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, RIACS Technical Report 91.18, NASA Ames Research Center, Moffett Field, CA, September 1991.

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

[7] R.W. FREUND AND N.M. NACHTIGAL, *TFQMR: a maximal subspace look-ahead QMR algorithm for non-Hermitian linear systems*, RIACS Technical Report, NASA Ames Research Center, Moffett Field, CA, in preparation.

[8] R.W. FREUND, G.H. GOLUB, AND N.M. NACHTIGAL, *Iterative solution of linear systems*, Technical Report 91.21, RIACS, NASA Ames Research Center, CA, November 1991, Acta Numerica, 1992, to appear.

[9] 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, CA, April 1991, SIAM J. Sci. Stat. Comput., to appear.

[10] 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.

[11] R.W. FREUND AND H. ZHA, *Simplifications of the nonsymmetric Lanczos process and a new algorithm for solving indefinite symmetric linear systems*, Technical Report, RIACS, NASA Ames Research Center, Moffett Field, CA, 1992.

[12] M.R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., 49 (1952), pp. 409–436.

[13] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Natl. Bur. Stand., 45 (1950), pp. 255–282.

[14] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 33–53.

[15] Y. SAAD, *SPARSKIT: a basic tool kit for sparse matrix computations*, Technical Report 90.20, RIACS, NASA Ames Research Center, Moffett Field, CA, May 1990.

[16] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36–52.