Numer. Math. 60, 315–339 (1991)

Numerische Mathematik

QMR: a quasi-minimal residual method for non-Hermitian linear systems*

Roland W. Freund[1,2] **and Noël M. Nachtigal**[3]

[1] RIACS, Mail Stop Ellis Street, NASA Ames Research Center, Moffett Field, CA 94035, USA
[2] Institut für Angewandte Mathematik und Statistik, Universität Würzburg, W-8700 Würzburg, Federal Republic of Germany
[3] Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Received February 19, 1991

Summary. The biconjugate gradient (BCG) method is the "natural" generalization of the classical conjugate gradient algorithm for Hermitian positive definite matrices to general non-Hermitian linear systems. Unfortunately, the original BCG algorithm is susceptible to possible breakdowns and numerical instabilities. In this paper, we present a novel BCG-like approach, the quasi-minimal residual (QMR) method, which overcomes the problems of BCG. An implementation of QMR based on a look-ahead version of the nonsymmetric Lanczos algorithm is proposed. It is shown how BCG iterates can be recovered stably from the QMR process. Some further properties of the QMR approach are given and an error bound is presented. Finally, numerical experiments are reported.

Mathematics Subject Classification (1991): 65F10

1 Introduction

The solution of large sparse systems of linear equations

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

is one of the most frequently encountered tasks in numerical computations. For example, such systems arise from finite difference or finite element approximations to partial differential equations. For Hermitian positive definite coefficient matrices A, the classical conjugate gradient method (CG hereafter) of Hestenes and Stiefel [11] is one of the most powerful iterative schemes for solving (1.1), especially when combined with a preconditioning technique. For general non-Hermitian matrices, the situation is less satisfactory. An ideal CG-type method for solving

* This work was supported in part by DARPA via Cooperative Agreement NCC 2-387 between NASA and the Universities Space Research Association (USRA).
Offprint requests to: R.W. Freund (USA address)

non-Hermitian linear systems would have features similar to the classical CG algorithm. It would produce approximate solutions to (1.1) which:

(i) are characterized by a minimization property over Krylov subspaces generated by A;
(ii) can be computed with little work and low storage requirements per iteration.

Unfortunately, for general non-Hermitian matrices, there are no CG-type algorithms which fulfill both requirements (i) and (ii); this was proved by Faber and Manteuffel [2]. Instead, most CG-type methods for non-Hermitian linear systems satisfy either (i) or (ii).

In the first category, the most successful scheme is the generalized minimal residual algorithm (GMRES) by Saad and Schultz [21]. It fulfills (i), but not (ii), since work and storage per iteration grow linearly with the iteration number. Consequently, in practice, one cannot afford to run the full algorithm and it is necessary to use restarts. For difficult problems, this often results in very slow convergence.

In the second category, the archetype is the biconjugate gradient algorithm (BCG hereafter) due to Lanczos [13]. At least in the generic case, BCG is based on simple three-term recurrences, which keep work and storage requirements constant at each iteration. 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. Furthermore, in the BCG algorithm, breakdowns — more precisely, divisions by 0 — may occur. In finite precision arithmetic, such exact breakdowns are very unlikely; however, near-breakdowns may occur, leading to numerical instabilities in subsequent iterations. Recently, two modifications of BCG, namely CGS [22] and Bi-CGSTAB [24], have been proposed. However, while these methods seem to work well in many cases, they do not address the problem of breakdowns, and thus they too, like BCG, are susceptible to instabilities. In exact arithmetic, both CGS and Bi-CGSTAB break down every time BCG does.

In this paper, we present a novel BCG-like approach for general nonsingular non-Hermitian linear systems (1.1), the quasi-minimal residual algorithm (QMR hereafter), which overcomes the problems of BCG. The method uses a look-ahead variant of the nonsymmetric Lanczos process to generate basis vectors for the Krylov subspaces induced by A. The look-ahead Lanczos approach was first proposed by Taylor [23] and Parlett et al. [18]. For the QMR method, we use the implementation of the look-ahead Lanczos process recently developed by Freund et al. [6, 7]. Using the Lanczos basis, the actual QMR iterates are then defined by a relaxed version of (i), namely a quasi-minimal residual property. This approach was first proposed by Freund [5] for the special case of linear systems with complex symmetric coefficient matrices $A = A^T$.

The QMR method can be implemented using only short recurrences and hence it still satisfies the requirement (ii). The quasi-minimal residual property ensures that QMR, unlike BCG, converges smoothly; moreover, existing BCG iterates can also be easily and stably recovered from the QMR process. Finally, for the QMR method, it is possible to obtain error bounds which are essentially the same as the standard bounds for GMRES. To the best of our knowledge, this is the first convergence result for a BCG-like algorithm.

The outline of this paper is as follows. In Sect. 2, the implementation of the look-ahead Lanczos algorithm derived in [6, 7] is briefly outlined. In Sect. 3, we describe the basic idea of the QMR approach. In Sect. 4, details of an implementation of the QMR algorithm are given. In Sect. 5, we discuss the connection of QMR with BCG. In Sect. 6, a convergence theorem for QMR is derived. In Sect. 7, we show how to incorporate preconditioning into the QMR method and describe two preconditioners which we have used for our numerical tests. In Sect. 8, we present numerical examples. Finaly, in Sect. 9, we make some concluding remarks.

Throughout the paper, all vectors and matrices, unless otherwise stated, are assumed to be complex. As usual, $M^T = (m_{ji})$ and $M^H = (\bar{m}_{ji})$ denote the transpose and the conjugate transpose, respectively, of the marix $M = (m_{ij})$. We use $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$ for the largest and smallest singular value of M, respectively. The vector norm $\|x\| = \sqrt{x^H x}$ is always the Euclidean norm and $\|M\| = \sigma_{\max}(M)$ is the corresponding matrix norm. The set of eigenvalues of a square matrix M is denoted by $\lambda(M)$. 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. Furthermore, it is always assumed that A is a complex, in general non-Hermitian, $N \times N$ matrix.

Finally, one more note. In our formulations of BCG and of the nonsymmetric Lanczos algorithm, we use A^T rather than A^H. This was a deliberate choice in order to avoid complex conjugation of the scalars in the recurrences; the algorithms can be formulated equally well in either terms.

2 The look-ahead Lanczos algorithm

Given two nonzero starting vectors $v_1 \in \mathbb{C}^N$ and $w_1 \in \mathbb{C}^N$, the classical nonsymmetric Lanczos method [12], [25, pp. 388–394] generates two sequences of vectors $v_1, v_2, \ldots, v_n$ and $w_1, w_2, \ldots, w_n$, $n = 1, 2, \ldots$, such that

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

and

$$w_j^T v_l = \begin{cases} 0 & \text{if } j \neq l\,, \\ d_j \neq 0 & \text{if } j = l\,. \end{cases} \tag{2.2}$$

The actual construction of each new pair v_{n+1} and w_{n+1} of Lanczos vectors is based on two simple three-term recurrences. If

$$w_{n+1}^T v_{n+1} = 0\,, \tag{2.3}$$

the Lanczos algorithm needs to be terminated, since (2.3) would lead to a division by 0 at the next iteration. In exact arithmetic, (2.3) will occur after a finite number, $m = n(\leqq N)$, of steps. If (2.3) is caused by $v_{m+1} = 0$ or $w_{m+1} = 0$, then $K_m(v_1, A)$ is A-invariant or $K_m(w_1, A^T)$ is A^T-invariant, respectively, and, by (2.1), the Lanczos

process has constructed a basis for this invariant subspace. This is referred to as *regular termination.* Unfortunately, it can also happen that (2.3) is satisfied with $v_{m+1} \neq 0$ and $w_{m+1} \neq 0$. In this case, the Lanczos algorithm stops before an invariant subspace has been found. This is referred to as *serious breakdown* [25, p. 389].

It is the possibility of serious breakdowns, or, in finite precision arithmetic, of *near-breakdowns*, i.e.,

$$w_{n+1}^T v_{n+1} \approx 0, \qquad w_{n+1} \not\approx 0, \qquad v_{n+1} \not\approx 0 ,$$

that has brought the classical nonsymmetric Lanczos algorithm into discredit. However, there are so-called *look-ahead* [23, 18] variants of the Lanczos process which allow to skip — except in the very special case of an incurable breakdown [23] — over those iterations in which the standard algorithm would break down. We refer the reader to [9, 10, 17] for a detailed theory of the look-ahead Lanczos process. In [6, 7], we have developed a robust implementation of the look-ahead Lanczos algorithm, which we briefly sketch here; for details and the actual FORTRAN code, see [6, 7].

Like the classical process, the look-ahead Lanczos algorithm generates two sequences of vectors $v_1, v_2, \ldots, v_n$ and $w_1, w_2, \ldots, w_n$, $n = 1, 2, \ldots$, which satisfy (2.1). Possible breakdowns are prevented by relaxing the biorthogonality condition (2.2) whenever a breakdown (exact or near) would occur. More precisely, for each fixed $n = 1, 2, \ldots$, the Lanczos vectors $v_1, \ldots, v_n$ and $w_1, \ldots, w_n$ generated by the look-ahead algorithm can be grouped into $k = k(n)$ blocks

$$V_l = [v_{n_l}\, v_{n_l+1} \ldots v_{n_{l+1}-1}], \qquad W_l = [w_{n_l}\, w_{n_l+1} \ldots w_{n_{l+1}-1}] ,$$

$$l = 1, 2, \ldots, k-1 , \tag{2.4}$$

$$V_k = [v_{n_k}\, v_{n_k+1} \ldots v_n], \qquad W_k = [w_{n_k} w_{n_k+1} \ldots w_n] ,$$

where

$$1 = n_1 < n_2 < \ldots < n_l < \ldots < n_k \leqq n < n_{k+1} .$$

The blocks are constructed such that, instead of (2.2), we have

$$W_j^T V_l = \begin{cases} 0 & \text{if } j \neq l , \\ D_l & \text{if } j = l , \end{cases} \quad j, l = 1, 2, \ldots, k , \tag{2.5}$$

where

$$\begin{gathered} D_l \text{ is nonsingular, } l = 1, 2, \ldots, k-1, \text{ and} \\ D_k \text{ is nonsingular if } n = n_{k+1} - 1 . \end{gathered} \tag{2.6}$$

In the sequel, we denote by

$$h_l = n_{l+1} - n_l, \quad l = 1, 2, \ldots, k-1, \quad \tilde{h}_k = n - n_k$$

the number of vectors in each block. The first vectors v_{n_l} and w_{n_l} in each block are called *regular*, the remaining vectors are called *inner*. The kth block is called *complete* if $n = n_{k+1} - 1$; in this case, at the next step $n + 1$, a new block is started

with the regular vectors $v_{n_{k+1}}$ and $w_{n_{k+1}}$. Otherwise, if $n < n_{k+1} - 1$, the kth block is *incomplete* and at the next step, the Lanczos vectors v_{n+1} and w_{n+1} are added to the kth block as inner vectors.

With these preliminaries, the basic structure of the look-ahead Lanczos algorithm is as follows.

Algorithm 2.1 (sketch of the look-ahead Lanczos process).

0) *Choose* $v_1, w_1 \in \mathbb{C}^N$ *with* $\|v_1\| = \|w_1\| = 1$;
Set $V_1 = v_1$, $W_1 = w_1$, $D_1 = W_1^T V_1$;
Set $n_1 = 1, k = 1$, $v_0 = w_0 = 0$, $V_0 = W_0 = \varnothing$, $\rho_1 = \xi_1 = 1$;
For $n = 1, 2, \ldots$:

1) *Decide whether to construct* v_{n+1} *and* w_{n+1} *as regular or inner vectors and go to* 2) *or* 3), *respectively*;

2) (*Regular step.*) *Compute*

$$\begin{aligned} \tilde{v}_{n+1} &= Av_n - V_k D_k^{-1} W_k^T A v_n - V_{k-1} D_{k-1}^{-1} W_{k-1}^T A v_n , \\ \tilde{w}_{n+1} &= A^T w_n - W_k D_k^{-T} V_k^T A^T w_n - W_{k-1} D_{k-1}^{-T} V_{k-1}^T A^T w_n , \end{aligned} \tag{2.7}$$

set $n_{k+1} = n + 1$, $k = k + 1$, $V_k = W_k = \varnothing$, *and go to* 4);

3) (*Inner step.*) *Compute*

$$\begin{aligned} \tilde{v}_{n+1} &= Av_n - \zeta_{n-n_k} v_n - (\eta_{n-n_k}/\rho_n) v_{n-1} - V_{k-1} D_{k-1}^{-1} W_{k-1}^T A v_n , \\ \tilde{w}_{n+1} &= A^T w_n - \zeta_{n-n_k} w_n - (\eta_{n-n_k}/\xi_n) w_{n-1} - W_{k-1} D_{k-1}^{-T} V_{k-1}^T A^T w_n ; \end{aligned} \tag{2.8}$$

4) *Compute* $\rho_{n+1} = \|\tilde{v}_{n+1}\|$ *and* $\xi_{n+1} = \|\tilde{w}_{n+1}\|$;
If $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$, *stop*;
Otherwise, set

$$\begin{aligned} v_{n+1} &= \tilde{v}_{n+1}/\rho_{n+1}, \qquad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1} , \\ V_k &= [V_k \, v_{n+1}], \qquad W_k = [W_k \, w_{n+1}], \qquad D_k = W_k^T V_k . \end{aligned} \tag{2.9}$$

If only regular steps 2) are performed, all blocks have size $h_l = 1$ and Algorithm 2.1 reduces to the classical Lanczos process. Therefore, the strategy for the decision in step 1) should be such that regular steps are performed whenever possible and blocks of size $h_l > 1$ are built only to avoid exact or near-breakdowns. In [6], we proposed a practical procedure for the decision in step 1) based on three different checks. Clearly, for a regular step, it is necessary that D_k is nonsingular. Therefore, one of the checks monitors the size of $\sigma_{\min}(D_k)$ relative to the roundoff unit. The other two checks monitor the size of the components along the two previous blocks of vectors V_k and V_{k-1} respectively W_k and W_{k-1} in (2.7). A regular step is performed only if these terms do not dominate the components Av_n respectively $A^T w_n$ in the new Krylov subspaces. These checks are necessary to ensure linear independence of the Lanczos vectors. We refer the reader to [6] for details.

At each step, Algorithm 2.1 requires the computation of Euclidean norms of two vectors, $\tilde{v}_{n+1}$ and $\tilde{w}_{n+1}$, of length N. In addition, inner products of vectors of length N occur in the terms $W_{k-1}^T A v_n$, $V_{k-1}^T A^T w_n$, $W_k^T V_k$, and, if a regular step is performed, in $W_k^T A v_n$ and $V_k^T A^T w_n$. However, as shown in [6], all these inner

products can be generated stably based on the actual computation of only two inner products per iteration. Therefore, Algorithm 2.1 requires the same number of inner products per iteration as the standard nonsymmetric Lanczos process.

Note that, in (2.8), the inner recurrence coefficients ζ_j, η_j, $j = 0, 1, \ldots, \eta_0 = 0$, are still arbitrary. For example, Chebyshev iteration [14] would be a possible choice for these coefficients. On the other hand, in practice, the blocks built by the look-ahead Lanczos algorithm are usually small and the choice $\zeta_j \equiv \eta_j \equiv 0$ is feasible.

Next, we list some properties of Algorithm 2.1 which will be used in the sequel. First, in view of (2.9), we have

$$\|v_n\| = \|w_n\| = 1, \quad n = 1, 2, \ldots . \tag{2.10}$$

It is convenient to introduce the notation

$$\begin{aligned} V^{(n)} &= [v_1 v_2 \ldots v_n] \qquad (= [V_1 V_2 \ldots V_k]) , \\ W^{(n)} &= [w_1 w_2 \ldots w_n] \qquad (= [W_1 W_2 \ldots W_k]) . \end{aligned} \tag{2.11}$$

Hence, by (2.1),

$$\begin{aligned} K_n(v_1, A) &= \{ V^{(n)} z \mid z \in \mathbb{C}^N \} , \\ K_n(w_1, A^T) &= \{ W^{(n)} z \mid z \in \mathbb{C}^N \} . \end{aligned} \tag{2.12}$$

Moreover, the recursions for the v's in (2.7) and (2.8) can be rewritten in matrix formulation as follows:

$$A V^{(n)} = V^{(n)} H^{(n)} + [0 \ldots 0 \; \tilde{v}_{n+1}] . \tag{2.13}$$

Here,

$$H^{(n)} := \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \ldots & 0 \\ \gamma_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_k \\ 0 & \ldots & 0 & \gamma_k & \alpha_k \end{bmatrix} \tag{2.14}$$

is a $n \times n$ block tridiagonal matrix with blocks of the form

$$\alpha_l = \begin{bmatrix} * & * & 0 & \ldots & 0 & * \\ \rho_{n_l+1} & * & \ddots & \ddots & \vdots & \vdots \\ 0 & \rho_{n_l+2} & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & * & * \\ \vdots & & \ddots & \ddots & * & * \\ 0 & \ldots & \ldots & 0 & \rho_{n_{l+1}-1} & * \end{bmatrix},$$

$$\gamma_l = \begin{bmatrix} 0 & \ldots & 0 & \rho_{n_l} \\ \vdots & \ddots & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & \ldots & 0 \end{bmatrix} . \tag{2.15}$$

The blocks β_l are in general full matrices. Furthermore, for $l = 1, \ldots, k-1$, the matrices α_l, β_l, and γ_l are of size $h_l \times h_l$, $h_{l-1} \times h_l$, and $h_l \times h_{l-1}$, respectively. The matrices α_k, β_k, and γ_k corresponding to the current block k are of size $\tilde{h}_k \times \tilde{h}_k$, $h_{k-1} \times \tilde{h}_k$, and $\tilde{h}_k \times h_{k-1}$, respectively. Here $\tilde{h}_k = h_k$ if the kth block is complete.

In view of (2.14) and (2.15), $H^{(n)}$ is an upper Hessenberg matrix with subdiagonal elements

$$\rho_j > 0, \quad j = 2, 3, \ldots, n . \tag{2.16}$$

In exact arithmetic, the stopping criterion in step 4) of Algorithm 2.1 will be satisfied after at most N steps — except in a very special situation. Recall that in order to close the current block k, it is necessary that $W_k^T V_k$ is nonsingular. However, in general, it cannot be excluded that Algorithm 2.1 produces infinite blocks V_k and W_k of nonzero Lanczos vectors such that $W_k^T V_k$ is the infinite zero matrix. This is called an *incurable* breakdown [23]; such a breakdown is very rare and does not present a problem in practice. Furthermore, even in the case of an incurable breakdown, the look-ahead Lanczos process still yields information on the spectrum of A, as Taylor [23] showed in his Mismatch Theorem (see also [9, 17]). For later use, we summarize the termination properties of the look-ahead Lanczos process in the following proposition.

Proposition 2.2. *There is a "termination index" $m \leqq N$ such that, in exact arithmetic, Algorithm 2.1 will either stop in step $n = m$ with $\rho_{m+1} = 0$ or $\xi_{m+1} = 0$, or, starting with the regular vectors v_{m+1} and w_{m+1}, an incurable breakdown will occur. If $\rho_{m+1} = 0$ or $\xi_{m+1} = 0$, then $v_1, \ldots, v_m$ or $w_1, \ldots, w_m$ span the A-invariant subspace $K_m(v_1, A)$ or the A^T-invariant subspace $K_m(w_1, A^T)$, respectively. Moreover, in all cases,*

$$\lambda(H^{(m)}) \subseteq \lambda(A) . \tag{2.17}$$

3 The quasi-minimal residual approach

In this section, we describe the basic idea of the QMR approach for the solution of linear systems (1.1). From now on, it is always assumed that A is nonsingular.

Given any initial guess $x_0 \in \mathbb{C}^N$ for the exact solution $A^{-1}b$ of (1.1), we will construct iterates x_n, $n = 1, 2, \ldots$, such that

$$x_n \in x_0 + K_n(r_0, A) . \tag{3.1}$$

Let $r_n = b - Ax_n$ denote the residual vector corresponding to the nth iterate x_n, and set

$$\rho_0 = \|r_0\|, \qquad v_1 = r_0/\rho_0 . \tag{3.2}$$

Let $v_1, v_2, \ldots, v_n$ be the right Lanczos vectors generated by Algorithm 2.1, with the normalized initial residual v_1 as one of the two starting vectors. From (2.12), the v's span $K_n(r_0, A)$, and hence we have the parametrization

$$x_n = x_0 + V^{(n)} z, \qquad z \in \mathbb{C}^n , \tag{3.3}$$

for all possible iterates (3.1). Note that the second starting vector, $w_1 \in \mathbb{C}^N$, is still unspecified. Due to the lack of a criterion for the choice of w_1, one usually sets $w_1 = v_1$ in practice.

Next, letting

$$H_e^{(n)} := \begin{bmatrix} H^{(n)} \\ \rho_{n+1}(e_n^{(n)})^T \end{bmatrix}, \qquad e_n^{(n)} = [0 \ldots 0\ 1]^T \in \mathbb{R}^n , \tag{3.4}$$

we can rewrite (2.13) as

$$AV^{(n)} = V^{(n+1)} H_e^{(n)} . \tag{3.5}$$

From (3.2) and (3.5), the residual vectors corresponding to (3.3) satisfy

$$r_n = r_0 - AV^{(n)} z = r_0 - V^{(n+1)} H_e^{(n)} z = V^{(n+1)} (\rho_0 e_1^{(n+1)} - H_e^{(n)} z) , \tag{3.6}$$

where $e_1^{(n+1)} = [1\ 0 \ldots 0]^T \in \mathbb{R}^{n+1}$. Next, we introduce an $(n+1) \times (n+1)$ diagonal weight matrix

$$\Omega^{(n)} = \operatorname{diag}(\omega_1, \omega_2, \ldots, \omega_{n+1}), \quad \omega_j > 0, \quad j = 1, \ldots, n+1 , \tag{3.7}$$

to serve as a free parameter that can be used to modify the scaling of the problem. With it, (3.6) reads

$$\begin{aligned} r_n &= V^{(n+1)} (\Omega^{(n)})^{-1} \Omega^{(n)} (\rho_0 e_1^{(n+1)} - H_e^{(n)} z) \\ &= V^{(n+1)} (\Omega^{(n)})^{-1} (d^{(n)} - \Omega^{(n)} H_e^{(n)} z), \quad \text{with} \quad d^{(n)} = \omega_1 \rho_0 e_1^{(n+1)} . \end{aligned} \tag{3.8}$$

Ideally, we would like to choose $z \in \mathbb{C}^n$ in (3.8) such that $\| r_n \|$ is minimal. However, since in general $V^{(n+1)}$ is not unitary, this would require $O(Nn^2)$ work, which is too expensive. We will instead minimize just the Euclidean norm of the bracketed terms in (3.8), i.e., we will choose $z = z^{(n)} \in \mathbb{C}^n$ as the solution of the least squares problem

$$\| d^{(n)} - \Omega^{(n)} H_e^{(n)} z^{(n)} \| = \min_{z \in \mathbb{C}^n} \| d^{(n)} - \Omega^{(n)} H_e^{(n)} z \| . \tag{3.9}$$

By (2.16), (3.4), and (3.7), $H_e^{(n)}$ and $\Omega^{(n)} H_e^{(n)}$ are $(n+1) \times n$ matrices with full column rank n. This guarantees that the solution $z^{(n)}$ of (3.9) is unique and hence, via (3.3), defines a unique nth iterate x_n. In view of the minimization property (3.9), we refer to this iteration scheme as the *quasi-minimal residual* (QMR) method. Clearly, the QMR iterates still depend on the choice of the weights ω_j in (3.7). In our numerical experiments, the simplest scaling

$$\omega_j = 1, \quad j = 1, 2, \ldots , \tag{3.10}$$

gave satisfactory results. Recall from (2.10) that all the columns of $V^{(n+1)}$ are unit vectors. Hence, the scaling (3.10) ensures that all basis vectors v_j/ω_j, $j = 1, \ldots, n+1$, in the representation (3.8) of r_n have the same Euclidean length; this is a "natural" requirement. However, better strategies for choosing $\Omega^{(n)}$ might be possible, and therefore we have formulated the QMR approach with a general scaling matrix $\Omega^{(n)}$.

For the solution of the least squares problem (3.9), we use the standard approach (see, e.g., [8, Chapter 6]) based on a QR decomposition of $\Omega^{(n)} H_e^{(n)}$:

$$\Omega^{(n)} H_e^{(n)} = (Q^{(n)})^{\mathrm{H}} \begin{bmatrix} R^{(n)} \\ 0 \end{bmatrix}. \tag{3.11}$$

Here, $Q^{(n)}$ is a unitary $(n+1)\times(n+1)$ matrix, and $R^{(n)}$ is a nonsingular upper triangular $n \times n$ matrix. Inserting (3.11) in (3.9) yields

$$\begin{aligned} \min_{z \in \mathbb{C}^n} \| d^{(n)} - \Omega^{(n)} H_e^{(n)} z \| &= \min_{z \in \mathbb{C}^n} \left\| (Q^{(n)})^{\mathrm{H}} \left(Q^{(n)} d^{(n)} - \begin{bmatrix} R^{(n)} \\ 0 \end{bmatrix} z \right) \right\| \\ &= \min_{z \in \mathbb{C}^n} \left\| Q^{(n)} d^{(n)} - \begin{bmatrix} R^{(n)} \\ 0 \end{bmatrix} z \right\|. \end{aligned}$$

Hence, $z^{(n)}$ is given by

$$z^{(n)} = (R^{(n)})^{-1} t^{(n)}, \quad \text{where} \quad t^{(n)} = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix}, \quad \begin{bmatrix} t^{(n)} \\ \tilde{\tau}_{n+1} \end{bmatrix} = Q^{(n)} d^{(n)}. \tag{3.12}$$

Furthermore, we have

$$\| d^{(n)} - \Omega^{(n)} H_e^{(n)} z^{(n)} \| = |\tilde{\tau}_{n+1}|. \tag{3.13}$$

We conclude this section by summarizing the basic structure of the QMR algorithm.

Algorithm 3.1 (QMR algorithm).

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $r_0 = b - Ax_0$, $\rho_0 = \|r_0\|$, $v_1 = r_0/\rho_0$;
Choose $w_1 \in \mathbb{C}^N$ *with* $\|w_1\| = 1$;
For $n = 1, 2, \ldots$:
1) *Perform the nth iteration of the look-ahead Lanczos Algorithm* 2.1;
This yields matrices $V^{(n)}$, $V^{(n+1)}$, $H_e^{(n)}$ *which satisfy* (3.5);
2) *Update the QR factorization* (3.11) *of* $\Omega^{(n)} H_e^{(n)}$ *and the vector* $t^{(n)}$ *in* (3.12);
3) *Compute*

$$x_n = x_0 + V^{(n)} (R^{(n)})^{-1} t^{(n)}; \tag{3.14}$$

4) *If* x_n *has converged, stop.*

4 Implementation details

In this section, we give some of the details for the actual implementation of steps 2), 3), and 4) of the QMR Algorithm 3.1. In particular, it is shown that the QMR iterates x_n can be computed with short recurrences. This approach for updating the iterates x_n is based on a technique which was first used by Paige and Saunders [16] in connection with their SYMMLQ and MINRES algorithms for real symmetric matrices.

First, note that the QR decomposition (3.11) of $\Omega^{(n)}H_e^{(n)}$ can be computed by means of n Givens rotations, taking advantage of the fact that $\Omega^{(n)}H_e^{(n)}$ is an upper Hessenberg matrix. Hence, the unitary factor in (3.11) in of the form

$$Q^{(n)} = G_n \begin{bmatrix} G_{n-1} & 0 \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} G_1 & 0 \\ 0 & I_{n-1} \end{bmatrix} \tag{4.1}$$

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

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

Recall that, in view of (2.14), $\Omega^{(n)}H_e^{(n)}$ is block tridiagonal. Therefore, the upper triangular factor in (3.11) is of the form

$$R^{(n)} = \begin{bmatrix} \delta_1 & \varepsilon_2 & \theta_3 & 0 & \ldots & 0 \\ 0 & \delta_2 & \varepsilon_3 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \delta_3 & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \theta_k \\ \vdots & & & \ddots & \ddots & \varepsilon_k \\ 0 & \ldots & \ldots & \ldots & 0 & \delta_k \end{bmatrix} \tag{4.2}$$

where the blocks δ_l and ε_l are of the same size as the blocks α_l and β_l, respectively, in (2.14). Moreover, the diagonal blocks δ_l are nonsingular upper triangular matrices. Clearly, a QR decomposition based on unitary matrices (4.1) limits fill-in to the row above each block β_l in (2.14). Hence each of the blocks θ_l in (4.2) has possible nonzero entries only in its last row.

Next, we note that the decompositon (3.11) is easily updated from the factorization of $\Omega^{(n-1)}H_e^{(n-1)}$ at the previous step $n-1$. Indeed, to obtain $R^{(n)}$, one only needs to compute its last column,

$$[\mu_1 \ldots \mu_n]^T = R^{(n)} e_n^{(n)} , \tag{4.3}$$

and append it to $R^{(n-1)}$. This is done by first multiplying the last column of $\Omega^{(n)}H_e^{(n)}$ by the previous Givens rotations; by (2.14), this last column has zero entries in positions $1, 2, \ldots, n_G$ where

$$n_G = \begin{cases} \max(n_{k-1} - 1, 1) & \text{if } v_n \text{ is an inner vector} , \\ \max(n_{k-2} - 1, 1) & \text{if } v_n \text{ is a regular vector} . \end{cases}$$

Therefore, only the Givens rotations with indices $n_G, n_G + 1, \ldots, n-1$ have to be

applied, and, by setting

(4.4)
$$\begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{n-1} \\ \mu \\ \nu \end{bmatrix} = \begin{bmatrix} G_{n-1} & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} G_{n-2} & 0 \\ 0 & I_2 \end{bmatrix} \cdots \begin{bmatrix} G_{n_G} & 0 \\ 0 & I_{n-n_G} \end{bmatrix} \Omega^{(n)} H_e^{(n)} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

we obtain the desired vector (4.3), except for its last component μ_n. It remains to multiply (4.4) by a suitably chosen Givens rotation G_n which zeros out the last element $\nu = \omega_{n+1}\rho_{n+1}$. To achieve this, set

(4.5)
$$\begin{aligned} c_n &= \frac{|\mu|}{\sqrt{|\mu|^2 + |\nu|^2}}, \quad \bar{s}_n = c_n \frac{\nu}{\mu}, \quad \text{if } \mu \neq 0, \\ c_n &= 0, \qquad\qquad \bar{s}_n = 1, \qquad \text{if } \mu = 0, \end{aligned}$$

and finally one gets $\mu_n = c_n\mu + s_n\nu$. For later use, we notice that

(4.6)
$$|s_n\mu_n| = \omega_{n+1}\rho_{n+1},$$

which is readily verified using (4.5). The vector $t^{(n)}$ in (3.12) is updated by setting

$$\begin{bmatrix} t^{(n)} \\ \tilde{\tau}_{n+1} \end{bmatrix} = G_n \begin{bmatrix} t^{(n-1)} \\ \tilde{\tau}_n \\ 0 \end{bmatrix}.$$

Clearly, $t^{(n)}$ differs from $t^{(n-1)}$ only in its last two entries which are given by

(4.7)
$$\tau_n = c_n\tilde{\tau}_n \qquad \text{and} \qquad \tilde{\tau}_{n+1} = -\bar{s}_n\tilde{\tau}_n.$$

Next, we turn to the computation of the QMR iterates x_n in (3.14). We define vectors p_j via

(4.8)
$$P^{(n)} = [p_1 p_2 \ldots p_n] = V^{(n)}(R^{(n)})^{-1}.$$

Then, with (3.14) and (3.12), it follows that

$$x_n = x_{n-1} + p_n\tau_n.$$

It remains to show how to compute p_n. In analogy to the partitioning of $V^{(n)}$ in (2.4) and (2.11), we group the columns of $P^{(n)}$ into blocks

(4.9)
$$P^{(n)} = [P_1 P_2 \ldots P_k].$$

With (4.8), (4.2), and (4.9), one obtains the relation

$$P_k = (V_k - P_{k-1}\varepsilon_k - P_{k-2}\theta_k)\delta_k^{-1},$$

and thus p_n can be updated via short recurrences.

Finally, for step 4) of Algorithm 3.1, a convergence criterion is needed. We stop the QMR iteration as soon as

$$\|r_n\| \leqq tol \cdot \|r_0\| ; \tag{4.10}$$

here *tol* is a suitable tolerance, e.g., $tol = 10^{-6}$. In the QMR algorithm described so far, neither the residual vectors r_n nor their norms $\|r_n\|$ are generated explicitly. However, in part a) of the next proposition, we derive an upper bound for $\|r_n\|$ which is available at no extra cost. In our implementation, the convergence criterion is checked for this upper bound, (4.11), rather than $\|r_n\|$. Once this test is satisfied, we switch to checking (4.10) for the true residual norm $\|r_n\|$. Typically, this is necessary only in the last one or two iterations, since (4.11) is a good upper bound for $\|r_n\|$ (see Sect. 8 for examples).

The residual vector itself can be easily updated at the expense of one additional SAXPY per iteration, based on the recursion given in part b) of the following proposition.

Proposition 4.1. *For* $n = 1, 2, \ldots$:

a)

$$\|r_n\| \leqq \|r_0\| \sqrt{n+1}\, |s_1 s_2 \cdots s_{n-1} s_n| \max_{j=1,\ldots,n+1} (\omega_1/\omega_j) ; \tag{4.11}$$

b)

$$r_n = |s_n|^2 r_{n-1} + \frac{c_n \tilde{\tau}_{n+1}}{\omega_{n+1}} v_{n+1} . \tag{4.12}$$

Proof. By taking norms in (3.8) and with (3.13), we obtain

$$\|r_n\| \leqq \|V^{(n+1)}\| \cdot \|(\Omega^{(n)})^{-1}\| \cdot |\tilde{\tau}_{n+1}| . \tag{4.13}$$

Now, from (2.10) and (2.11), $V^{(n+1)}$ has $n+1$ columns of Euclidean norm 1, and this implies

$$\|V^{(n+1)}\| \leqq \sqrt{n+1} . \tag{4.14}$$

Furthermore, by (3.7),

$$\|(\Omega^{(n)})^{-1}\| \leqq \max_{j=1,\ldots,n+1} (1/\omega_j) . \tag{4.15}$$

Finally, by (4.7),

$$|\tilde{\tau}_{n+1}| = |\tilde{\tau}_1| \cdot |s_1 s_2 \cdots s_{n-1} s_n| , \tag{4.16}$$

where, in view of (3.12), (3.8), and (3.2),

$$\tilde{\tau}_1 = \|r_0\| \omega_1 . \tag{4.17}$$

By combining (4.13–17), one obtains the inequlity (4.11).

Now we turn to part b). By inserting $z = z^{(n)}$ from (3.12) in (3.8) and using (3.11), one obtains

$$r_n = \tilde{\tau}_{n+1} y_{n+1} , \tag{4.18}$$

where

$$y_{n+1} = V^{(n+1)}(\Omega^{(n)})^{-1}(Q^{(n)})^{\mathrm{H}} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} .$$

From (4.1), one readily verifies that two successive vectors y_{n+1} and y_n in (4.18) are connected by

$$y_{n+1} = -s_n y_n + \frac{c_n}{\omega_{n+1}} v_{n+1} . \tag{4.19}$$

Finally, by inserting (4.19) in (4.18) and using the second relation in (4.7), we arrive at (4.12). □

5 The connection between QMR and BCG

In this section, we are concerned with the connection between QMR and BCG. In particular, it is shown that BCG iterates can be easily recovered from the QMR process.

In the BCG approach, one aims at computing iterates x_n which are characterized by the Galerkin type condition

$$w^{\mathrm{T}}(b - Ax_n) = 0 \quad \text{for all} \quad w \in K_n(\tilde{r}_0, A^{\mathrm{T}}), \quad x_n \in x_0 + K_n(r_0, A) . \tag{5.1}$$

(see, e.g., [19]). Here, $\tilde{r}_0 \in \mathbb{C}^N$ is any nonzero vector. Usually, one sets $\tilde{r}_0 = r_0$. In the classical BCG algorithm [13, 4], the iterates (5.1) are generated as follows.

Algorithm 5.1 (BCG).

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $q_0 = r_0 = b - Ax_0$;
Choose $\tilde{r}_0 \in \mathbb{C}^N$, $\tilde{r}_0 \neq 0$, *and set* $\tilde{q}_0 = \tilde{r}_0$;
For $n = 1, 2, \ldots$:
1) *Compute* $\delta_n = \tilde{r}_{n-1}^{\mathrm{T}} r_{n-1} / \tilde{q}_{n-1}^{\mathrm{T}} A q_{n-1}$ *and set* $x_n = x_{n-1} + \delta_n q_{n-1}$;
Set $r_n = r_{n-1} - \delta_n A q_{n-1}$ *and* $\tilde{r}_n = \tilde{r}_{n-1} - \delta_n A^{\mathrm{T}} \tilde{q}_{n-1}$;
2) *Compute* $\rho_n = \tilde{r}_n^{\mathrm{T}} r_n / \tilde{r}_{n-1}^{\mathrm{T}} r_{n-1}$;
Set $q_n = r_n + \rho_n q_{n-1}$ *and* $\tilde{q}_n = \tilde{r}_n + \rho_n \tilde{q}_{n-1}$;
3) *If* $r_n = 0$ *or* $\tilde{r}_n = 0$, *stop.*

BCG is closely related to the classical nonsymmetric Lanczos algorithm. Indeed (see, e.g., [19]), for $n = 1, 2, \ldots$,

$$r_{n-1} = \phi_n v_n, \quad \phi_n \in \mathbb{C}, \phi_n \neq 0, \qquad \text{and} \qquad \tilde{r}_{n-1} = \psi_n w_n, \quad \psi_n \in \mathbb{C}, \psi_n \neq 0 , \tag{5.2}$$

where v_n and w_n denote the vectors generated by the standard Lanczos process with starting vectors

$$v_1 = r_0 / \| r_0 \| \qquad \text{and} \qquad w_1 = \tilde{r}_0 / \| \tilde{r}_0 \| . \tag{5.3}$$

Unfortunately, like the Lanczos algorithm, BCG is also susceptible to breakdowns and numerical instabilities. Obviously, Algorithm 5.1 breaks down prematurely, if

$$\tilde{q}_{n-1}^{\mathrm{T}} A q_{n-1} = 0, \qquad \tilde{r}_{n-1} \neq 0, \qquad r_{n-1} \neq 0\,, \tag{5.4}$$

or

$$\tilde{r}_{n-1}^{\mathrm{T}} r_{n-1} = 0, \qquad \tilde{r}_{n-1} \neq 0, \qquad r_{n-1} \neq 0\,, \tag{5.5}$$

occurs. We will refer to (5.4) and (5.5) *as breakdown of the first* and *second kind*, respectively. In general, Galerkin iterates (5.1) need not exist for every n. This is the cause of the breakdown of the first kind. Indeed, one can show that (5.4) occurs if no BCG iterate x_n exists. Breakdowns of the second kind have a different cause: by (5.2), (5.5) is equivalent to a serious breakdown in the classical nonsymmetric Lanczos process.

Next, we rewrite the Galerkin condition (5.1) in terms of the look-ahead Lanczos Algorithm 2.1, started with the initial vectors (5.3). This yields a formulation of the BCG approach for which breakdowns of the second kind, except for ones caused by an incurable breakdown in the look-ahead Lanczos process, cannot occur. In analogy to (3.3), we use the parametrization

$$x_n = x_0 + V^{(n)} u^{(n)}, \quad u^{(n)} \in \mathbb{C}^n\,, \tag{5.6}$$

for the BCG iterates. Then, by (2.13), the corresponding residual vector satisfies

$$\begin{aligned} r_n = b - A x_n &= V^{(n)}(f^{(n)} - H^{(n)} u^{(n)}) - (u^{(n)})_n \tilde{v}_{n+1}\,, \\ \text{with} \quad f^{(n)} &= [\rho_0\, 0 \ldots 0]^{\mathrm{T}} \in \mathbb{R}^n\,. \end{aligned} \tag{5.7}$$

By inserting (5.7) in (5.1) and using (2.12), it follows that the iterate (5.6) satisfies (5.1) if, and only if,

$$(W^{(n)})^{\mathrm{T}} V^{(n)}(f^{(n)} - H^{(n)} u^{(n)}) = (u^{(n)})_n (W^{(n)})^{\mathrm{T}} \tilde{v}_{n+1}\,. \tag{5.8}$$

To simplify the discussion of (5.8), we will attempt to recover the BCG iterate only when the current block k in Algorithm 2.1 is complete. Therefore, in the sequel, it is always assumed that $n = n_{k+1} - 1$. This ensures that, in view of (2.5–6), the linear system (5.8) reduces to

$$H^{(n)} u^{(n)} = f^{(n)}\,, \tag{5.9}$$

from which we can now derive a simple criterion for the existence of the nth BCG iterate.

In the following proposition, superscripts BCG and QMR are used to distinguish iterates and residuals of the BCG and QMR approaches. The remaining notation is as introduced in Sects. 3 and 4.

Proposition 5.2. *Let $n = n_{k+1} - 1$, $k = 0, 1, \ldots$. Then, the following three conditions are equivalent:*

(i) *the BCG iterate x_n^{BCG} defined by* (5.1) *exists;*
(ii) *$H^{(n)}$ is nonsingular;*
(iii) *$c_n \neq 0$.*

Moreover, if x_n^{BCG} *exists, then*

$$x_n^{\mathrm{BCG}} = x_n^{\mathrm{QMR}} + \frac{\tau_n |s_n|^2}{c_n^2} p_n , \tag{5.10}$$

$$\| r_n^{\mathrm{BCG}} \| = \| r_0 \| \cdot |s_1 s_2 \cdots s_{n-1} s_n| \frac{\omega_1}{\omega_{n+1} c_n} . \tag{5.11}$$

Proof. Clearly, an nth BCG iterate exists if, and only if, the linear system (5.9) has a solution. From (5.7) and (2.14–16), the extended coefficient matrix $[f^{(n)} H^{(n)}]$ of (5.9) is an upper triangular matrix whose diagonal elements are all nonzero, and thus it has full row rank n. Consequently, (5.9) has a solution if, and only if, $H^{(n)}$ is nonsingular. This shows the equivalence of (i) and (ii).

Next, using (3.11), (3.4), and (4.1), one readily verifies that

$$Q^{(n-1)} \Omega^{(n-1)} H^{(n)} = \begin{bmatrix} I_{n-1} & 0 \\ 0 & c_n \end{bmatrix} R^{(n)} . \tag{5.12}$$

This relation implies that (ii) and (iii) are equivalent.

Now assume $c_n \neq 0$. From (5.9) and (5.12) it follows that

$$u^{(n)} = (R^{(n)})^{-1} \begin{bmatrix} I_{n-1} & 0 \\ 0 & 1/c_n \end{bmatrix} Q^{(n-1)} \Omega^{(n-1)} f^{(n)} . \tag{5.13}$$

Recalling the definitions of $d^{(n)}$ and $f^{(n)}$ in (3.8) and (5.7), and using (3.12), we can rewrite (5.13) as follows:

$$u^{(n)} = z^{(n)} + (R^{(n)})^{-1} \begin{bmatrix} 0 \\ \tilde{\tau}_n / c_n - \tau_n \end{bmatrix} . \tag{5.14}$$

By comparing (5.6) and (5.14) with (3.3) and (3.12), and by using (4.8), we obtain the relation

$$x_n^{\mathrm{BCG}} = x_n^{\mathrm{QMR}} + \left(\frac{\tilde{\tau}_n}{c_n} - \tau_n \right) p_n$$

which, by (4.7), is just (5.10). By inserting (5.9) in (5.7), it follows that

$$r_n^{\mathrm{BCG}} = - (u^{(n)})_n \tilde{v}_{n+1} . \tag{5.15}$$

From (5.15), (5.14), and (3.12), we obtain

$$\| r_n^{\mathrm{BCG}} \| = \frac{\| \tilde{v}_{n+1} \|}{c_n} \left| \frac{\tilde{\tau}_n}{\mu_n} \right| \quad \text{where } \mu_n = (R^{(n)})_{n,n} . \tag{5.16}$$

In view of (4.6),

$$\| \tilde{v}_{n+1} \| = \rho_{n+1} = \frac{|s_n \mu_n|}{\omega_{n+1}} . \tag{5.17}$$

Then, by inserting (5.17), (4.16), and (4.17) in (5.16), we get (5.11), and this concludes the proof. □

Remark 5.3. Proposition 5.2 shows that existing BCG iterates can be recovered easily from the QMR process. By (5.11), $\|r_n^{\mathrm{BCG}}\|$ can be computed at no extra cost from quantities which are generated in the QMR Algorithm 3.1 anyway. In particular, one may monitor $\|r_n^{\mathrm{BCG}}\|$ during the course of the QMR iteration, and compute x_n^{BCG} via (5.10) whenever the actual BCG iterate is desired.

Remark 5.4. CGS [22] and Bi-CGSTAB [24] are modifications of the BCG Algorithm 5.1. In many cases, these algorithms have better convergence properties than BCG. However, neither CGS nor Bi-CGSTAB addresses the problem of breakdowns. Indeed, one can show that, in exact arithmetic, CGS as well as Bi-CGSTAB break down every time BCG does. Numerical examples for that will be given in Sect. 8.

6 A convergence theorem

In this section, we derive bounds for the QMR residuals which are essentially the same as the standard bounds for GMRES. To the best of our knowledge, this is the first convergence result for a BCG-like algorithm for general non-Hermitian matrices.

Let m denote the termination index of the look-ahead Lanczos Algorithms 2.1, as introduced in Proposition 2.2. We remark that, in exact arithmetic, the QMR Algorithm 3.1 will also terminate in step $n = m$. For a diagonalizable matrix M, we denote by

$$\kappa(M) = \min_{X\,:\,X^{-1}MX \text{ diagonal}} \|X\| \cdot \|X^{-1}\|$$

the condition number of the eigenvalue problem of M. Furthermore, we denote by Π_n the set of all complex polynomials of degree at most n.

With these notations, the main result of this section can be formulated as follows.

Theorem 6.1. *Suppose that the $m \times m$ matrix $H^{(m)}$ generated by m steps of the look-ahead Lanczos Algorithm* 2.1 *is diagonalizable, and set*

$$H = \Omega^{(m-1)} H^{(m)} (\Omega^{(m-1)})^{-1} . \tag{6.1}$$

Then, for $n = 1, 2, \ldots, m-1$, the residual vectors of the QMR *Algorithm* 3.1 *satisfy*

$$\|r_n\| \leqq \|r_0\| \kappa(H) \sqrt{n+1}\, \varepsilon^{(n)} \max_{j=1,\ldots,n+1} (\omega_1/\omega_j) , \tag{6.2}$$

where

$$\varepsilon^{(n)} = \min_{P \in \Pi_n\,:\,P(0)=1} \max_{\lambda \in \lambda(A)} |P(\lambda)| . \tag{6.3}$$

Moreover, if Algorithm 2.1 *terminates with $\rho_{m+1} = 0$, then $x_m = A^{-1}b$ is the exact solution of $Ax = b$.*

Proof. Using (4.13–15), (3.13), (3.8–9), and (3.2), one readily verifies that

$$\|r_n\| \leqq \|r_0\| \sqrt{n+1}\, \vartheta_n \max_{j=1,\ldots,n+1} (\omega_1/\omega_j) ,$$

where ϑ_n is given by

$$\vartheta_n = \min_{z \in \mathbb{C}^n} \|e_1^{(n+1)} - \Omega^{(n)} H_e^{(n)} z\| , \qquad e_1^{(n+1)} = [1\ 0\ \ldots\ 0]^T \in \mathbb{R}^{n+1} . \tag{6.4}$$

Therefore, for the proof of (6.2), it remains to show that

$$\vartheta_n \leqq \kappa(H)\, \varepsilon^{(n)} . \tag{6.5}$$

In the following, let $n \in \{1, 2, \ldots, m-1\}$ be arbitrary, but fixed. By

$$H^{(m)} = \begin{bmatrix} H_e^{(n)} & * \\ 0 & * \end{bmatrix}$$

and (6.1), we have

$$H \begin{bmatrix} z \\ 0 \end{bmatrix} = \begin{bmatrix} \Omega^{(n)} H_e^{(n)} (\Omega^{(n-1)})^{-1} z \\ 0 \end{bmatrix} \quad \text{for all} \quad z \in \mathbb{C}^n . \tag{6.6}$$

Recall that $H^{(m)}$, and therefore also H, is an upper Hessenberg matrix with nonzero subdiagonal elements. This implies that

$$\left\{ \begin{bmatrix} z \\ 0 \end{bmatrix} \middle| z \in \mathbb{C}^n \right\} = \{ P(H) e_1^{(m)} \mid P \in \Pi_{n-1} \} . \tag{6.7}$$

Using (6.6–7), we can rewrite (6.4) as follows:

$$\vartheta_n = \min_{z \in \mathbb{C}^n} \left\| e_1^{(m)} - H \begin{bmatrix} z \\ 0 \end{bmatrix} \right\| = \min_{P \in \Pi_n : P(0) = 1} \| P(H) e_1^{(m)} \| . \tag{6.8}$$

$H^{(m)}$ is assumed diagonalizable, so by (6.1) H is also diagonalizable, and by expanding $e_1^{(m)}$ into any set of eigenvectors of H, we deduce from (6.8) that

$$\vartheta_n \leqq \kappa(H) \min_{P \in \Pi_n : P(0) = 1} \max_{\lambda \in \lambda(H)} |P(\lambda)| . \tag{6.9}$$

By (6.1) and (2.17), we have $\lambda(H) = \lambda(H^{(m)}) \subseteq \lambda(A)$, and thus (6.9) is equivalent to the desired inequality (6.5).

Finally, we need to show that $x_m = A^{-1} b$, if Algorithm 2.1 terminates with $\rho_{m+1} = 0$. For $n = m$ and $\rho_{m+1} = 0$, the least squares problem (3.9) reduces to a linear system with coefficient matrix $\Omega^{(m-1)} H^{(m)}$. Since A is nonsingular, by (2.17), this linear system is nonsingular, and hence it can be solved exactly. Therefore, $r_m = 0$ and this concludes the proof. □

Recall (cf. Proposition 2.2) that, in exact arithmetic, it can also happen that the QMR algorithm terminates with $\rho_{m+1} \neq 0$. In this case, one restarts the QMR method, using the last available QMR iterate as the new initial guess. Theorem 6.1 shows that x_{m-1} is a good choice. However, the finite termination property of the look-ahead Lanczos Algorithm 2.1 is usually lost in finite precision arithmetic. In

particular, situations where the QMR algorithm needs to be restarted are very rare in practice.

We remark that for the "natural" scaling $\omega_j \equiv 1$, the bound (6.2) simplifies somewhat.

Next, we contrast the bounds (6.2) for QMR with the standard bounds [21] for GMRES. Assume that A is a diagonalizable matrix. Then, the residuals r_n^{GMRES} generated by the GMRES algorithm (without restarts) satisfy

$$\|r_n^{\mathrm{GMRES}}\| \leqq \|r_0\| \kappa(A) \varepsilon^{(n)}, \quad n = 1, 2, \ldots,$$

where, as before, $\varepsilon^{(n)}$ is given by (6.3). Hence, up to the slow growing factor $\sqrt{n+1}$ in (6.2) and different constants, the error bounds for QMR and GMRES are essentially the same.

In general, simple upper bounds for (6.3) are known only for special cases. For example, assume that the eigenvalues are contained in an ellipse in the complex plane which does not contain the origin:

$$\lambda(A) \subset \mathscr{E}, \quad 0 \notin \mathscr{E}.$$

Let $f_1 \neq f_2$ denote the two foci of $\mathscr{E}$. The ellipse can be represented in the form

$$\mathscr{E} = \left\{ \lambda \in \mathbb{C} \,\middle|\, |\lambda - f_1| + |\lambda - f_2| \leqq \frac{|f_1 - f_2|}{2} \left(r + \frac{1}{r} \right) \right\} \quad \text{with} \quad r > 1.$$

Moreover, let R be the unique solution of

$$\frac{1}{2}\left(R + \frac{1}{R} \right) = \frac{|f_1| + |f_2|}{|f_1 - f_2|}, \quad R > 1.$$

It can be shown that $0 \notin \mathscr{E}$ implies $R > r$. Then, by [3, Theorem 2], we have the following upper bound for (6.3):

$$\varepsilon^{(n)} \leqq \frac{r^n + 1/r^n}{R^n + 1/R^n}, \quad n = 1, 2, \ldots.$$

7 Preconditioning

As for other conjugate gradient type methods, when solving realistic problems, it is crucial to combine the QMR algorithm with an efficient preconditioning technique. In this section, we show how to incorporate preconditioners into the QMR algorithm. Also, we briefly describe two preconditioning techniques.

Let M be a given nonsingular $N \times N$ matrix which approximates in some sense the coefficient matrix A of the linear system (1.1), $Ax = b$. Moreover, assume that M is decomposed in the form

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

Instead of solving the original system (1.1), we apply the QMR algorithm to the equivalent linear system

$$A'y = b', \quad \text{where } A' = M_1^{-1} A M_2^{-1}, \quad b' = M_1^{-1}(b - Ax_0), \quad y = M_2(x - x_0).$$

Here x_0 denotes some initial guess for the solution of $Ax = b$. The iterates y_n and residual vectors $r'_n = b' - A'y_n$ for the preconditioned system $A'y = b'$ are transformed back into the corresponding quantities for the original system by setting

$$x_n = x_0 + M_2^{-1} y_n \quad \text{and} \quad r_n = M_1 r'_n . \tag{7.2}$$

For the special cases $M_1 = I$ or $M_2 = I$ in (7.1) one obtains right or left preconditioning, respectively.

Using (7.2), the QMR Algorithm 3.1 combined with preconditioning can be sketched as follows:

Algorithm 7.1 (QMR approach with preconditioning).

0) *Choose* $x_0 \in \mathbb{C}^N$ *and set* $r'_0 = M_1^{-1}(b - Ax_0)$, $\rho_0 = \|r'_0\|$, $v_1 = r'_0/\rho_0$, $y_0 = 0$; *Choose* $w_1 \in \mathbb{C}^N$ *with* $\|w_1\| = 1$;

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

1) *Perform the nth iteration of the look-ahead Lanczos Algorithm* 2.1 (*applied to* A'); *This yields matrices* $V^{(n)}$, $V^{(n+1)}$, $H_e^{(n)}$ *which satisfy* $A'V^{(n)} = V^{(n+1)}H_e^{(n)}$;
2) *Update the QR factorization* (3.11) *of* $\Omega^{(n)}H_e^{(n)}$ *and the vector* $t^{(n)}$ *in* (3.12);
3) *Compute* $y_n = V^{(n)}(R^{(n)})^{-1}t^{(n)}$;
4) *If* y_n *has converged, compute* $x_n = x_0 + M_2^{-1}y_n$, *and stop.*

In the case of right or left preconditioning, Algorithm 7.1 simplifies somewhat. In general, however, for the QMR algorithm applied to a preconditioned system, one has to be able to compute $M_1^{-1}z$, $M_1^{-T}z$, $M_2^{-1}z$, and $M_2^{-T}z$, for arbitrary vectors z.

For the numerical examples in Sect. 8, we have used SSOR and ILUT(k) preconditioning.

- **SSOR**

The SSOR preconditioner is based on a decomposition of the matrix A into a nonsingular diagonal matrix D, a strictly lower triangular matrix L, and a strictly upper triangular matrix U, such that $A = D + L + U$. D might have to be block diagonal to ensure it is nonsingular. The preconditioner matrix M is given by

$$M = (D + L)D^{-1}(D + U) .$$

For our experiments, we have used SSOR as a right or left preconditioner.

- **ILUT(k)**

The Incomplete LU decomposition is based on the LU decomposition of the coefficient matrix A into a unit lower triangular matrix L and an upper triangular matrix U. The full LU decomposition of A would result in factors L and U which, in general, have far more nonzero elements than A. The incomplete LU factorization aims to reduce this additional fill-in in the factors L and U.

In ILUT(k), we use a strategy due to Saad [20] for dropping nonzero elements which would fill-in L and U. Each row of L and U is constructed subject to the restriction that only a small amount of fill-in, k more elements for each, is allowed beyond the number of elements of A already present in that row (in the lower and

upper part, respectively). Furthermore, elements which are deemed to make only an insignificant contribution to the decomposition are also dropped. For example, this means that if $nmax_L$ is the maximum number of elements allowed for some row of L, n_L is the actual number of elements of that row computed by the elimination process, and *ctol* is the cutoff tolerance, then the algorithm orders the n_L elements in decreasing order of magnitude, and keeps only up to min $(n_L, nmax_L)$ elements, or until the elements reach the level *ctol*, whichever cutoff comes first. The resulting matrices L and U can be used either as $M_1 = L$ and $M_2 = U$ in (7.1), or in $M_2 = LU$ respectively $M_1 = LU$ for right respectively left preconditioning.

The variant of ILU used is different from the standard one. For a Hermitian matrix A, the standard ILU preconditioner [15] preserves the sparsity structure of the matrix, i.e., for $k = 0$, the preconditioner matrices have nonzero elements only in those locations where A itself has nonzero elements. In [15] it is shown that this strategy does produce a good preconditioner, provided that A is a Hermitian M-matrix. For a general non-Hermitian matrix, there is no reason to preserve the sparsity structure of A. Instead, the ILUT(k) variant discards elements subject only to the constraints of fill-in and size, without regard to the sparsity structure of A. However, this does mean that if A is Hermitian, we do not recover the standard ILU preconditioner.

8 Numerical experiments

We have performed extensive numerical experiments with the QMR algorithm and the other iterative methods mentioned in this paper. In this section, we present a few typical results of these experiments. Further numerical results are reported in [7] and, for the case of complex symmetric matrices, in [5],

For our test runs, we always chose $x_0 = 0$ as initial guess. The iteration was stopped as soon as the convergence criterion (4.10) (with $tol = 10^{-6}$) was satisfied. We always used the QMR algorithm with no scaling, i.e., $\Omega^{(n)} \equiv I_{n+1}$ in (3.7). For GMRES [21], work and storage per iteration step n grow linearly with n and hence one needs to use restarts. In the sequel, GMRES(n_0) refers to GMRES restarted after every n_0 iterations. A typical value for the restart parameter is $n_0 = 20$; work and storage per iteration are then roughly comparable for GMRES and QMR. Furthermore, unless stated otherwise, the BCG iterates were always obtained from the QMR process via (5.10), rather than from the BCG Algorithm 5.1. Examples 1, 3, and 4 were run using right preconditioning. We also tried left preconditioning, and in all cases, the number of iterations needed to converge was roughly the same. In all our experiments with QMR, the look-ahead Lanczos Algorithm 2.1 performed almost exclusively regular steps and only few blocks of size $h_l > 1$ were built. The biggest block that occurred was of size $h_l = 4$. For all examples, we report the exact numbers of blocks of size 2, 3, and 4. Finally, the convergence plots show the relative residual norms $\|r_n\|/\|r_0\|$ (on the vertical axis) versus the iteration number n (on the horizontal axis).

All examples were run on a Cray-2 at the NASA Ames Research Center.

Example 1. We consider the partial differential equation

$$(8.1) \quad -\Delta u + \gamma\left(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y} + z\frac{\partial u}{\partial z}\right) + \beta u = f \quad \text{on} \quad (0,1)\times(0,1)\times(0,1)\,,$$

with Dirichlet boundary conditions. We discretize (8.1) 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. For our experiments, we have chosen the parameters $\beta = -250$ and $\gamma = 40$. Note that this choice guarantees that the cell Reynolds number is smaller than one, and hence centered differences yield a stable discretization of (8.1). The right-hand side was chosen such that the vector of all ones is the exact solution of the linear system. QMR was run with identical starting vectors $v_1 = w_1$. This example was run with SSOR preconditioning. In Fig. 1, we show the convergence curves for QMR (solid line), BCG (dotted line), GMRES (19) (dash-dotted line), and GMRES (20) (dashed line). As the plot indicates, the convergence curve for QMR is rather smooth; we also see the typical oscillations in the BCG convergence curve. Also, we see that GMRES, while being optimal as long as it is not restarted, loses its edge once it is restarted, and furthermore its behavior after being restarted can be quite sensitive to the length n_0 of the restart cycle. Finally, we note that in the course of the QMR run, 4 blocks of size 2 were built.

Example 2. This example was taken from the Harwell-Boeing set of sparse test matrices [1]. It is the fifth matrix from the SHERMAN collection, called SHERMAN 5. It comes from a fully implicit black oil simulator on a $16 \times 23 \times 3$ grid, with three unknowns per grid point. The order of the matrix is 3312, and it has 20793 nonzero elements. The right-hand side b was chosen as a vector with random entries from a normal distribution with mean 0.0 and variance 1.0. Again, QMR was run with identical starting vectors $v_1 = w_1$.

If no preconditioning is used, this linear system is very difficult to solve by iterative methods. As Fig. 2 shows, QMR needs almost 1500 iterations to converge. On the other hand, GMRES(100) (dash-dotted line), even with an unrealistically large restart value $n_0 = 100$, does not converge within a reasonable number of iterations. Figure 2 also shows the relative residual norms (dots) of the BCG iterates as obtained from the QMR algorithm. During the QMR run, the

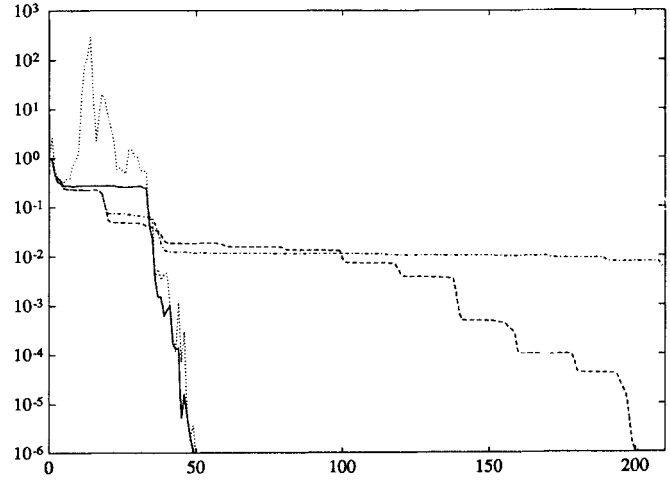

Fig. 1. Convergence curves for Example 1, right SSOR

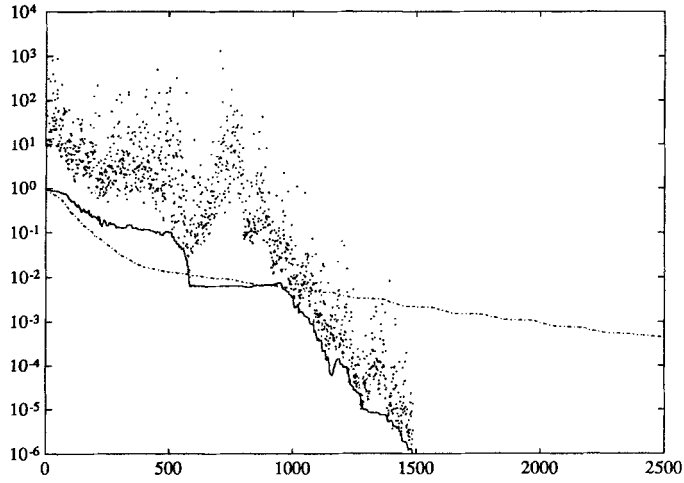

Fig. 2. Convergence curves for Example 2, no preconditioning

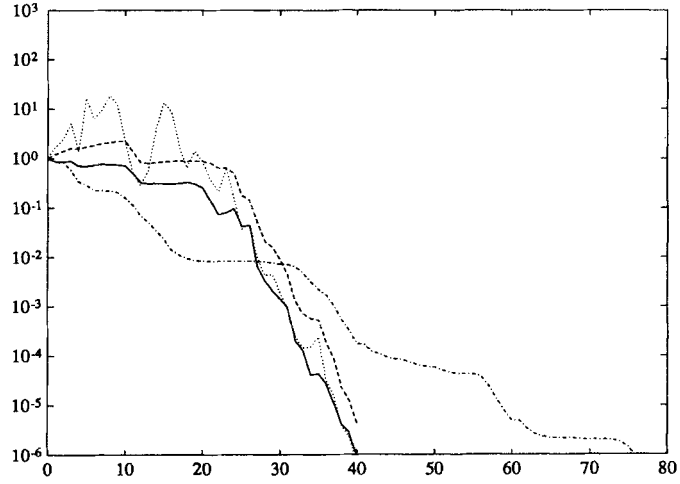

Fig. 3. Convergence curves for Example 2, right SSOR

underlying look-ahead Lanczos algorithm built 49 blocks of size 2, 7 blocks of size 3, and 1 block of size 4.

Next, we ran the SHERMAN 5 problem with SSOR and ILUT(0) preconditioning (cf. Sect. 7). For ILUT(0) the cutoff tolerance $ctol = 0.001$ was chosen; this choice leads to approximate factors L and U which together have 19899 nonzero elements. In Fig. 3 (for SSOR) and Fig. 4 (for ILUT(0)), we show the convergence curves for QMR (solid line), BCG (dotted line), and GMRES(20) (dash-dotted line). We have also plotted (dashed line) the upper bound (4.11) for the QMR residuals. Clearly, these bounds are very close to the true QMR residuals. In Fig. 4, we have also added the convergence curve (lower dash-dotted line) for a full GMRES run without restart to the one for GMRES(20). A comparison of this curve with the QMR curve shows that the quasi-minimal residual property is very

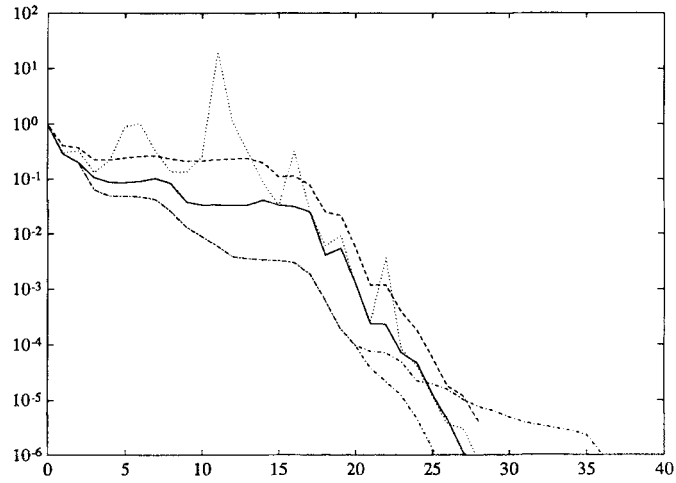

Fig. 4. Convergence curves for Example 2, right ILUT(0)

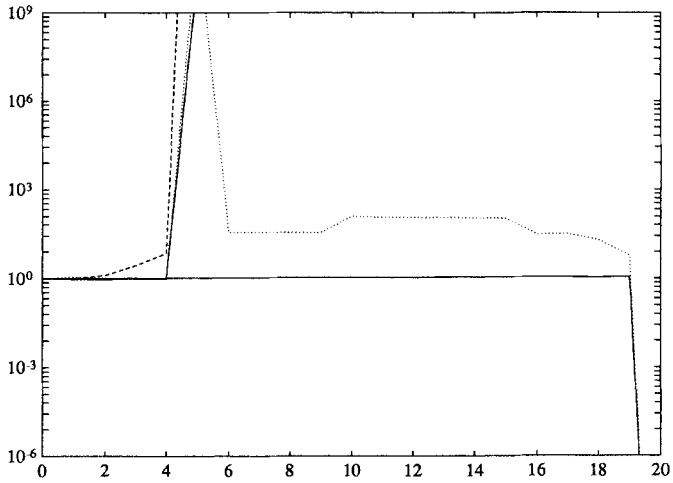

Fig. 5. Convergence curves for Example 3

close to the true minimal residual property of GMRES. Finally, we note that, for both SSOR and ILUT(0), only blocks of size 1 were built during the QMR run.

Examples 3 and 4. Next, we present two examples which were constructed such that a breakdown of the first kind (Example 3) respectively of the second kind (Example 4) occurs (cf. Sect. 5). In both cases the matrix A is of order $N = 20$. For Example 3, the matrix H_5 generated by 5 steps of the Lanczos process is singular and thus, in view of Proposition 5.2, x_5^{BCG} does not exist. Consequently, BCG, CGS, and Bi-CGSTAB all break down in step 5 (cf. Remark 5.4). Example 4 was constructed such that an exact breakdown occurs in step 2 of the standard Lanczos algorithm. This then leads to a breakdown of BCG, CGS, and Bi-CGSTAB in step 3. In Figs. 5 and 6, we show the convergence curves for QMR (solid line), BCG

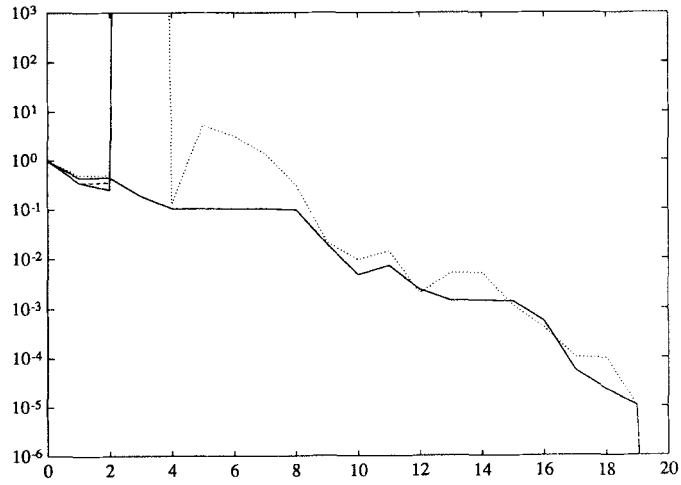

Fig. 6. Convergence curves for Example 4

(dotted line), CGS (dashed line), and Bi-CGSTAB (solid line becoming vertical at step 5 respectively 3). In both cases, the BCG iterates were computed by the BCG Algorithm 5.1 before the breakdown occurs and via the QMR process after the breakdown. Finally, we note that, for both examples, 2 blocks of size 2 were built during the QMR run.

9 Concluding remarks

In this paper, we have proposed a robust iterative solver, the QMR algorithm, for general nonsingular non-Hermitian linear systems. The method uses a recently proposed [6, 7] robust implementation of the look-ahead Lanczos algorithm to generate basis vectors for the Krylov subspaces $K_n(r_0, A)$. The QMR iterates are characterized by a quasi-minimal residual property over $K_n(r_0, A)$. Both the look-ahead Lanczos algorithm and the computation of the actual QMR iterates can be implemented using only short recurrences. The QMR approach is closely related to the BCG algorithm; however, unlike BCG, the QMR algorithm has smooth convergence curves and good numerical properties. Furthermore, we have derived bounds for the QMR residuals which are essentially the same as the standard bounds for GMRES. To the best of our knowledge, this is the first convergence result for a BCG-like algorithm for general non-Hermitian matrices.

FORTRAN 77 implementations of the QMR method and the underlying look-ahead Lanczos algorithm are available electronically from the authors (na. freund@na-net. ornl. gov or na. nachtigal@na-net. ornl. gov).

References

1. Duff, I.S., Grimes, R.G., Lewis, J.G. (1989): Sparse matrix test problems. ACM Trans. Math. Softw. **15**, 1–14
2. Faber, V., Manteuffel, T. (1984): Necessary and sufficient conditions for the existence of a conjugate gradient method. SIAM J. Numer. Anal. **21**, 352–362

3. Fischer, B., Freund, R.W. (1990): On the constrained Chebyshev approximation problem on ellipses. J. Approx. Theory **62**, 297–315
4. Fletcher, R. (1976): Conjugate gradient methods for indefinite systems. In: G.A. Watson, ed., Numerical Analysis Dundee 1975, pp. 73–89. Lecture Notes in Mathematics 506. Springer, Berlin Heidelberg New York
5. Freund, R.W. (1989): Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices. Technical Report 89.54, RIACS, NASA Ames Research Center
6. Freund, R.W., Gutknecht, M.H., Nachtigal, N.M. (1990): An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part I. Technical Report 90.45, RIACS, NASA Ames Research Center
7. Freund, R.W., Nachtigal, N.M. (1990): An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part II. Technical Report 90.46, RIACS, NASA Ames Research Center
8. Golub, G.H., Van Loan, C.F. (1983): Matrix computations. The Johns Hopkins University Press, Baltimore
9. Gutknecht, M.H. (1990): A completed theory of the unsymmetric Lanczos process and related algorithms, Part I. IPS Research Report No. 90-10, Zürich
10. Gutknecht, M.H. (1990): A completed theory of the unsymmetric Lanczos process and related algorithms, Part II. IPS Research Report No. 90-16, Zürich
11. Hestenes, M.R., Stiefel, E. (1952): Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**, 409–436
12. Lanczos, C. (1950): An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Natl. Bur. Stand. **45**, 255–282
13. Lanczos, C. (1952): Solution of systems of linear equations by minimized iterations. J. Res. Natl. Bur. Stand. **49**, 33–53
14. Manteuffel, T.A. (1977): The Tchebychev iteration for nonsymmetric linear systems. Numer. Math. **28**, 307–327
15. Meijerink, J.A., van der Vorst, H.A. (1977): An iterative solution for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comp. **31**, 148–162
16. Paige, C.C., Saunders, M.A. (1975): Solution of sparse indefinite systems of linear equations. SIAM J. Numer. Anal. **12**, 617–629
17. Parlett, B.N. (1990): Reduction to tridiagonal form and minimal realizations. Preprint, Berkeley
18. Parlett, B.N., Taylor, D.R., Liu, Z.A. (1985): A look-ahead Lanczos algorithm for unsymmetric matrices. Math. Comp. **44**, 105–124
19. Saad, Y. (1982): The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems. SIAM J. Numer. Anal. **19**, 485–506
20. Saad, Y. (1990): SPARSKIT: a basic tool kit for sparse matrix computations. Technical Report 90.20, RIACS, NASA Ames Research Center
21. Saad, Y., Schultz, M.H. (1986): GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869
22. Sonneveld, P. (1989): CGS, a fast Lanczos-type solver for nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **10**, 36–52
23. Taylor, D.R. (1982): Analysis of the look ahead Lanczos algorithm. Ph.D. Dissertation, University of California, Berkeley
24. van der Vorst, H.A. (1990): Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. Preprint, Utrecht
25. Wilkinson, J.H. (1965): The Algebraic Eigenvalue Problem. Oxford University Press, Oxford