Implementation Details of the Coupled QMR Algorithm*

Roland W. Freund
AT&T Bell Laboratories
600 Mountain Avenue, Room 2C–420
Murray Hill, New Jersey 07974–0636

Noël M. Nachtigal
RIACS, Mail Stop T041–5
NASA Ames Research Center
Moffett Field, California 94035–1000

Abstract

The original quasi-minimal residual method (QMR) relies on the three-term look-ahead Lanczos process to generate basis vectors for the underlying Krylov subspaces. However, empirical observations indicate that, in finite precision arithmetic, three-term vector recurrences are less robust than mathematically equivalent coupled two-term recurrences. Therefore, we recently proposed a new implementation of the QMR method based on a coupled two-term look-ahead Lanczos procedure. In this paper, we describe implementation details of this coupled QMR algorithm, and we present results of numerical experiments.

*This research was performed at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, California 94035, and it was supported by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association.

1 Introduction

Recently, we proposed a new Krylov subspace iteration, the quasi-minimal residual algorithm (QMR) [5], for solving general nonsingular non-Hermitian systems of linear equations

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

The QMR method has two main ingredients: the look-ahead Lanczos process, and a quasi-minimal residual condition. The look-ahead Lanczos algorithm is used to generate—with low work and storage requirements—basis vectors for the underlying Krylov subspaces. Furthermore, look-ahead techniques are used to avoid possible breakdowns in the classical Lanczos algorithm [7], except for so-called incurable breakdowns. Once the Lanczos basis is constructed, the quasi-minimal residual property is used to select the QMR iterates from the Krylov subspaces. As was shown in [5], the QMR iterates are always well defined, and the quasi-minimal residual condition leads to a smooth and nearly monotone convergence behavior. In addition, thanks to the quasi-minimal residual property, it is possible to prove convergence results for the QMR algorithm. The result is a method with several desirable numerical and theoretical properties.

In the original QMR algorithm, the look-ahead Lanczos method used generates the basis vectors for the Krylov subspaces by means of three-term recurrences. It has been observed that, in finite precision arithmetic, vector iterations based on three-term recursions are usually less robust than mathematically equivalent coupled two-term vector recurrences. Therefore, in [6], we proposed a new implementation of the QMR algorithm, based on a coupled two-term recurrence formulation of the Lanczos algorithm. Together with the derivation, we discussed in [6] the properties of the new implementation, and presented numerical results showing that the new method is more robust than the original QMR algorithm. In this paper, we discuss in more detail the implementation of the new algorithm; in particular, we focus on the implementation of the coupled two-term version of the look-ahead Lanczos algorithm. We also give several new numerical examples.

The outline of the paper is as follows. In Section 2, we recall the three-term look-ahead Lanczos process that was proposed in [3]. Then, in Section 3, we review the coupled two-term look-ahead Lanczos algorithm that was proposed in [6], and in Section 4 we give implementation details of the new algorithm. In Section 5, we briefly recall how the QMR approach can be combined with the coupled Lanczos algorithm to obtain a new implementation of the QMR method, and in Section 6, we report results of numerical experiments with this new QMR algorithm. Finally, in Section 7, we make some concluding remarks.

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

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

the nth Krylov subspace of $\mathbb{C}^N$ generated by $c \in \mathbb{C}^N$ and the $N \times N$ matrix B. Finally, it is always assumed that A is an $N \times N$ matrix, singular or nonsingular.

2 The three-term look-ahead Lanczos algorithm

The Lanczos process is a method that builds basis vectors for two Krylov subspaces, with low work and storage requirements. Given two starting vectors, v_1 and $w_1 \in \mathbb{C}^N$, the algorithm computes two sequences of vectors, $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$, such that, for $n = 1, 2, \ldots,$

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

In addition, the two sets of vectors are required to obey a biorthogonality relation. Ideally, one would like to impose the condition

$$w_n^T v_j = w_j^T v_n = 0 \quad \text{for all} \quad j < n. \tag{2.2}$$

This is done in the Lanczos process, as proposed by Lanczos in [7]. However, it turns out that it is not always possible or numerically stable to construct vectors satisfying (2.2), as *exact breakdowns* ($w_n^T v_n = 0$) or *near-breakdowns* ($w_n^T v_n$ is small in some sense) may arise. This poses a problem, since the construction of the pair v_{n+1} and w_{n+1} obeying (2.2) involves division by $w_n^T v_n$. As a remedy, one relaxes the biorthogonality condition, requiring instead that, in case of a breakdown, the relations (2.2) hold only for some range of j up to, but not equal to, n. This leads to so-called *look-ahead* Lanczos algorithms, which skip over the exact and near-breakdowns. The original QMR algorithm is based on a look-ahead Lanczos method proposed by Freund, Gutknecht, and Nachtigal [3], which we briefly review next.

Like the classical algorithm, the look-ahead Lanczos algorithm [3] generates vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ with (2.1). In addition, they satisfy the biorthogonality relation

$$W_n^T V_n = D_n, \tag{2.3}$$

where D_n is a block diagonal matrix whose structure is discussed below, and

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

This means that some of the vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ do in fact satisfy the full biorthogonality (2.2). These vectors are called *regular*, and they form a subsequence $\{v_{n_j}\}_{j=1}^l$ and $\{w_{n_j}\}_{j=1}^l$, where

$$1 =: n_1 < n_2 < \cdots < n_l \le n < n_{l+1}, \quad l := l(n). \tag{2.4}$$

All vectors that are not regular are called *inner*. The regular vectors are used to partition $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ into blocks. By convention, one defines blocks $V^{(j)}$, of size $N \times h_j$, containing the regular vector v_{n_j} and all inner vectors—if any—between v_{n_j} and $v_{n_{j+1}}$:

$$V^{(j)} = [\, v_{n_j} \quad v_{n_j+1} \quad \cdots \quad].$$

A look-ahead step is then defined in terms of building such a block. Hence the integer l in (2.4) is just the number of look-ahead steps taken during the first n steps of the Lanczos algorithm. Moreover, h_j is called the length of the jth look-ahead step. The structure of the sequence $\{w_j\}_{j=1}^n$ parallels that of the sequence $\{v_j\}_{j=1}^n$, so that V_n and W_n can be written as

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

and D_n in (2.3) is given by

$$D_n = \mathrm{diag}(D^{(1)}, D^{(2)}, \ldots, D^{(l)}), \quad D^{(j)} := (W^{(j)})^T V^{(j)}.$$

The choice of whether to build a regular or an inner vector is determined at each step, based on the particular look-ahead strategy used. It should also be pointed out that, even though the look-ahead Lanczos algorithm can handle most breakdowns, there remains a class of breakdowns, the so-called *incurable breakdowns*, which cannot be cured by look-ahead techniques. However, incurable breakdowns occur only in very particular circumstances, and they do not pose a problem in practice. Finally, since the scaling of the Lanczos vectors is not determined by (2.1) and (2.3), the look-ahead algorithm scales the vectors to have unit length:

$$\|v_n\| = \|w_n\| = 1, \quad n = 1, 2, \ldots \,. \tag{2.5}$$

The main point of the look-ahead Lanczos process is that vectors satisfying (2.1) and (2.3) can be constructed by means of short block three-term recurrences. These recurrences can be written compactly as

$$AV_n = V_{n+1} H_n, \tag{2.6}$$

$$A^T W_n = W_{n+1} \Gamma_{n+1}^{-1} H_n \Gamma_n, \tag{2.7}$$

where H_n is an $(n+1) \times n$ block tridiagonal matrix,

$$\Gamma_n := \mathrm{diag}(\gamma_1, \gamma_2, \ldots, \gamma_n), \quad \text{where} \quad \gamma_j := \begin{cases} 1, & \text{if} \quad j = 1, \\ \gamma_{j-1}\rho_j/\xi_j, & \text{if} \quad j > 1, \end{cases} \tag{2.8}$$

is a diagonal scaling matrix with positive diagonal entries, and ρ_j and ξ_j are scale factors used to ensure that v_j and w_j, respectively, obey the scaling (2.5). Since the recurrences used to build v_{n+1} and w_{n+1} are short, the look-ahead algorithm has low work and storage requirements, making it an attractive method for building bases for Krylov subspaces. However, it has been observed that, in finite precision arithmetic, three-term vector recurrences are less robust than mathematically equivalent coupled two-term recurrences. This was our motivation in proposing in [6] a different implementation of the look-ahead Lanczos algorithm, based on coupled two-term recurrences. Next, we review this algorithm.

3 The coupled two-term look-ahead Lanczos process

The coupled two-term Lanczos process is an alternate way of generating the Lanczos basis vectors[1]. The algorithm generates, in addition to the Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$,

[1]The discussion that follows will not cover all the details and will not justify all the statements made. For full details, we refer the reader to [6].

a second set of basis vectors, $\{p_j\}_{j=1}^n$ and $\{q_j\}_{j=1}^n$, such that, for $n = 1, 2, \ldots,$

$$\begin{aligned} \operatorname{span}\{p_1, p_2, \ldots, p_n\} &= K_n(v_1, A), \\ \operatorname{span}\{q_1, q_2, \ldots, q_n\} &= K_n(w_1, A^T). \end{aligned}$$

For simplicity, we will sometimes refer to the Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ as the V–W sequence, and to the auxiliary vectors $\{p_j\}_{j=1}^n$ and $\{q_j\}_{j=1}^n$ as the P–Q sequence. The four sets of basis vectors are generated using coupled two-term recurrences of the form:

$$\begin{aligned} V_n &= P_n U_n, & A P_n &= V_{n+1} L_n, \\ W_n &= Q_n \Gamma_n^{-1} U_n \Gamma_n, & A^T Q_n &= W_{n+1} \Gamma_{n+1}^{-1} L_n \Gamma_n, \end{aligned}$$

Here,

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

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

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

and Γ_n is the diagonal matrix defined in (2.8). As was shown in [6], the matrices L_n and U_n define a factorization of the block tridiagonal Hessenberg matrix H_n generated by the three-term look-ahead Lanczos algorithm,

$$H_n = L_n U_n. \tag{3.1}$$

In addition, it is possible to reduce L_n and U_n to block bidiagonal matrices, by constructing the basis vectors p_n and q_n so as to be block A-biorthogonal. Here, similar to the V–W sequence, the vectors p_n and q_n are also constructed using look-ahead techniques. For example, we again have blocks $P^{(j)}$,

$$P^{(j)} = [\, p_{m_j} \quad p_{m_j+1} \quad \cdots \quad],$$

where p_{m_j} is called regular, the other vectors in the block are called inner, and the indices m_j satisfy

$$1 =: m_1 < m_2 < \cdots < m_k < n \leq m_{k+1}, \quad k := k(n).$$

The regular vectors p_{m_j} satisfy the A-biorthogonality condition

$$q_i^T A p_{m_j} = 0 \quad \text{for all} \quad i < m_j, \tag{3.2}$$

while the inner vectors satisfy only a relaxed version of this condition. Once again, the structure of Q_n parallels that of P_n, and the A-biorthogonality of the two sets of basis vectors can be written as:

$$Q_n^T A P_n = E_n = \operatorname{diag}(E^{(1)}, E^{(2)}, \ldots, E^{(k)}), \quad E^{(j)} := (Q^{(j)})^T A P^{(j)}. \tag{3.3}$$

Before we consider the implementation details, let us briefly discuss an outline of the algorithm. At each iteration, the process consists of the following four basic steps.

Algorithm 3.1 (Overview of the coupled algorithm with look-ahead)

1) *Decide whether to construct p_n and q_n as regular or inner vectors.*
2) *Compute p_n and q_n as either regular or inner vectors.*
3) *Decide whether to construct v_{n+1} and w_{n+1} as regular or inner vectors.*
4) *Compute v_{n+1} and w_{n+1} as either regular or inner vectors.*

Steps 1) and 3) are the basis of the look-ahead strategy, and they each consist of three checks. Recall from (2.6) and (2.7) that the Lanczos vectors v_{n+1} and w_{n+1} can be obtained from the previous Lanczos vectors by a block three-term recurrence. Similarly, it is possible to show that the vectors p_n and q_n also have a block three-term recurrence, of the form

$$AP_{n-1} = P_n G_{n-1} \quad \text{and} \quad A^T Q_{n-1} = Q_n \Gamma_n^{-1} G_{n-1} \Gamma_{n-1}, \tag{3.4}$$

where

$$G_{n-1} := U_n L_{n-1}. \tag{3.5}$$

The look-ahead strategy for the two pairs of sequences is then similar, and was first proposed in [3]. In Step 1), the algorithm checks whether:

$$\sigma_{\min}(E^{(k)}) \geq eps,$$

$$n(A)\|p_n\| \geq \sum_{i=m_{k-1}}^{n-1} \left|(U_n L_{n-1})_{i,n-1}\right| \|p_i\|, \tag{3.6}$$

$$n(A)\|q_n\| \geq \sum_{i=m_{k-1}}^{n-1} \frac{\gamma_{n-1}}{\gamma_i} \left|(U_n L_{n-1})_{i,n-1}\right| \|q_i\|. \tag{3.7}$$

Here, eps is machine epsilon, and $n(A)$ is an estimate for the norm of A. The vectors p_n and q_n are built as regular vectors only if all three of the above checks are satisfied. Likewise, in Step 3), the algorithm checks whether:

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

$$n(A) \geq \sum_{i=n_{l-1}}^{n} |(L_n U_n)_{in}|, \tag{3.8}$$

$$n(A) \geq \sum_{i=n_{l-1}}^{n} \frac{\gamma_n}{\gamma_i} |(L_n U_n)_{in}|, \tag{3.9}$$

and again, the vectors v_{n+1} and w_{n+1} are built as regular vectors only if all three of the checks are passed. The motivation for these checks can be found in [3, 6]. Here, we will only note that the look-ahead strategy proposed will build regular vectors in preference to inner vectors, and thus it will take as few look-ahead steps as possible.

Once the decisions in Steps 1) and 3) are taken, the next vectors p_n and q_n, and v_{n+1} and w_{n+1}, are built in Steps 2) and 4). For p_n and q_n, let $k^\star$ denote the number of the row of the first possible nonzero element in the nth column of U_n. It can be shown that

$$k^\star = \max\{j \mid 1 \leq j \leq k \ \text{ and } \ m_j \leq \max\{1, n_l - 1\}\}. \tag{3.10}$$

From (3.3), both the regular and the inner vectors have the same coefficients $U_{m_{k^\star}:m_k-1,n}$, given by[2]

$$\begin{aligned} U_{m_{k^\star}:m_k-1,n} = {} & (\operatorname{diag}(E^{(k^\star)}, E^{(k^\star+1)}, \ldots, E^{(k-1)}))^{-1} \\ & \cdot [\, Q^{(k^\star)} \quad Q^{(k^\star+1)} \quad \cdots \quad Q^{(k-1)} \,]^T A v_n. \end{aligned}$$

For the regular vectors, the coefficients $U_{m_k:n-1,n}$ are determined by the condition (3.2),

$$U_{m_k:n-1,n} = (E^{(k)})^{-1} (Q^{(k)})^T A v_n, \tag{3.11}$$

while for the inner vectors, the coefficients $U_{m_k:n-1,n}$ are arbitrary:

$$u_{in} = \zeta_{in} \in \mathbb{C}, \quad \text{for} \quad i = m_k, m_k + 1, \ldots, n-1. \tag{3.12}$$

This completes the computation of the recurrence coefficients for p_n and q_n, and the vectors are then computed from

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

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

For v_{n+1} and w_{n+1}, let $l^\star$ denote the number of the row of the first possible nonzero element in the nth column of L_n. It can be shown that

$$l^\star = \max\{j \mid 1 \leq j \leq l \ \text{ and } \ n_j \leq m_k\}. \tag{3.13}$$

[2] We denote $x_{i:j} = [\, x_i \quad x_{i+1} \quad \cdots \quad x_j \,]^T$.

Again, by (2.3), both the regular and the inner vectors have the same coefficients $L_{n_{l^\star}:n_l-1,n}$, given by

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

For the regular vectors, the coefficients $L_{n_l:n,n}$ are determined by the condition (2.2),

$$L_{n_l:n,n} = (D^{(l)})^{-1}(W^{(l)})^T A p_n, \tag{3.14}$$

while for the inner vectors, the coefficients $L_{n_l:n,n}$ are arbitrary:

$$l_{in} = \eta_{in} \in \mathbb{C} \quad \text{for} \quad i = n_l, n_l+1, \ldots, n. \tag{3.15}$$

Once the recurrence coefficients for v_{n+1} and w_{n+1} are computed, the vectors are constructed by scaling the vectors

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

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

to have unit length.

4 Implementation details

We now turn to a detailed description of the implementation of Algorithm 3.1. If the coupled two-term Lanczos process is run without any look-ahead steps, it will require two inner products at each step in order to compute all the coefficients of the recurrence formulas. Hence, the goal for the look-ahead implementation is to also require only two inner products per step for all the recurrence coefficients. Recall that the look-ahead strategy (3.6) and (3.7) for the P–Q sequence and the normalization (2.5) require a total of four norm computations, so that the implementation will require two inner products and four norms per iteration.

To begin, let us introduce the auxiliary matrices

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

whose columns are needed in (3.14) and in (3.11). In addition, we will make use of the following symmetry relations from [6]:

$$D_n \Gamma_n = (D_n \Gamma_n)^T, \tag{4.1}$$

$$E_n \Gamma_n = (E_n \Gamma_n)^T, \tag{4.2}$$

$$F_n \Gamma_n = (\tilde{F}_n \Gamma_n)^T. \tag{4.3}$$

The nth iteration of the implementation will update the matrices D_{n-1}, E_{n-1}, F_{n-1}, L_{n-1}, U_{n-1}, P_{n-1}, Q_{n-1}, V_n, and W_n, to D_n, E_n, F_n, L_n, U_n, P_n, Q_n, V_{n+1}, and W_{n+1}, respectively. We first list an outline of the algorithm as we have implemented it.

Algorithm 4.1 (Coupled algorithm with look-ahead)

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

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

1) *Update* D_{n-1} *to* D_n.
2) *Determine* $k^\star$ *from* (3.10):

$$k^\star = \max\left\{ j \mid 1 \le j \le k \ \text{ and } \ m_j \le \max\{1, n_l - 1\} \right\}.$$

3) *Compute* $F_{n,1:n-1}$ *from* (4.4) *below, using* L_{n-1} *and* $D_{n,1:n}$.
   Then compute $\tilde{F}_{1:n-1,n}$ *from* (4.3).
4) *Check whether* $E^{(k)}$ *is nonsingular*:

$$\texttt{innerp} = \sigma_{\min}(E^{(k)}) < eps.$$

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

$$\begin{aligned} U_{m_i:m_{i+1}-1,n} &= (E^{(i)})^{-1}(Q^{(i)})^T A v_n \\ &= (E^{(i)})^{-1}\tilde{F}_{m_i:m_{i+1}-1,n}, \quad i = k^\star, \ldots, k-1. \end{aligned}$$

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

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

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

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

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

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

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

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

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

9) *Build and check the coefficient* $G_{m_k:n-1,n}$. *If* `innerp`, *go to* 11).

10) *Set* $m_{k+1} = n$, $k = k + 1$, *and go to* 12).

11) *Choose the inner recurrence coefficients* u_{in}, $i = m_k, \ldots, n-1$, *and build* p_n *and* q_n *as inner vectors:*

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

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

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

12) *If* $\|p_n\| = 0$, *or* $\|q_n\| = 0$, *then stop.*

13) *Compute* $A^T q_n$.

14) *Update* E_{n-1} *to* E_n.

15) *Determine* $l^\star$ *from* (3.13):

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

16) *Compute* $F_{1:n,n}$ *from* (4.5) *below, using* E_n *and* U_n.

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

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

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

$$\begin{aligned} L_{n_i:n_{i+1}-1,n} &= (D^{(i)})^{-1} (W^{(i)})^T Ap_n \\ &= (D^{(i)})^{-1} F_{n_i:n_{i+1}-1,n}, \quad i = l^\star, \ldots, l-1. \end{aligned}$$

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

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

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

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

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

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

20) *Build* $H_{n_l:n,n}$ *and check the coefficient* $H_{n_{l-1}:n,n}$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

25) *Set*

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

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

$$\begin{aligned} F_{n-1} &= W_{n-1}^T A P_{n-1} = W_{n-1}^T V_n L_{n-1} \\ &= D_{n-1} L_{1:n-1,1:n-1} + l_{n,n-1} D_{1:n-1,n} \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}, \end{aligned} \tag{4.4}$$

the remainder of the last column of D_n is computed from D_{n-1}, F_{n-1}, and L_{n-1}. The last row of D_n is obtained by symmetry, using (4.1).

Step 7. We build $G_{m_k:n-1,n-1}$, which would be the coefficient of the $P^{(k)}$ and $Q^{(k)}$ blocks in the three-term recurrences (3.4) for p_n and q_n. Using (3.5), one has

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

The coefficient $G_{m_{k-1}:m_k-1,n-1}$ has already been built as part of Step 9) at the previous iteration. We then check (3.6)–(3.7), and set `innerp` to `TRUE` if at least one of the two checks fails.

Step 9. We build $G_{m_k:n-1,n}$, which would be the coefficient of the $P^{(k)}$ and $Q^{(k)}$ blocks in the three-term recurrences (3.4) for p_{n+1} and q_{n+1}. It is straightforward to show that

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

Moreover, we have

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

We then check a subset of (3.6)–(3.7), and set `innerp` to `TRUE` if at least one of the two checks fails.

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

$$F_n = W_n^T A P_n = \Gamma_n U_n^T \Gamma_n^{-1} Q_n^T A P_n, \tag{4.5}$$

the remainder of the last row of E_n is computed from E_{n-1}, $F_{1:n,1:n-1}$, and U_n. The last column of E_n is obtained by symmetry, using (4.2).

Step 20. We build $H_{n_l:n,n}$, which would be the coefficient of the $V^{(l)}$ and $W^{(l)}$ blocks in the three-term recurrences (2.6) and (2.7) for v_{n+1} and w_{n+1}. Using (3.1), one has

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

The coefficient $H_{n_{l-1}:n_l-1,n}$ has already been built as part of Step 22) at the previous iteration. We then check (3.8)–(3.9), and set `innerv` to `TRUE` if at least one of the two checks fails.

Step 22. We build $H_{n_l:n,n+1}$, which would be the coefficient of the $V^{(l)}$ and $W^{(l)}$ blocks in the three-term recurrences (2.6) and (2.7) for v_{n+2} and w_{n+2}. It is straightforward to show that

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

Moreover, we have

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

We then check a subset of (3.8)–(3.9), and set `innerv` to `TRUE` if at least one of the two checks fails.

We remark that the checks in steps 9) and 22) are actually slightly relaxed versions of (3.8)–(3.9), and (3.6)–(3.7), respectively, since the indices checked are only a subset of the full range appearing in (3.8)–(3.9) and (3.6)–(3.7). We also note that the algorithm above requires minimal inputs from the user. Recall that *eps* in steps 4) and 17) is machine epsilon. Furthermore, the estimate $n(A)$ for the norm of the matrix can be updated dynamically, as was done in [3].

The coupled Lanczos Algorithm 4.1 requires per iteration the computation of two inner products and four vector norms. We conclude this section by noting that, in Algorithm 4.1, the choice of the inner recurrence coefficients (3.12) and (3.15) is arbitrary. In our implementation of the algorithm, we have used

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

for the inner vector recurrence coefficients.

5 The coupled two-term QMR algorithm

We now consider the quasi-minimal residual approach and briefly outline how it can be combined with the coupled two-term look-ahead Lanczos algorithm of Section 3 to obtain a new implementation of the QMR method. We note that the QMR algorithm was proposed in [5] for the case of nonsingular linear systems (1.1). It was later shown by Freund and Hochbruck [4] that the algorithm can also be applied to singular systems, and that it always generates well-defined iterates. However, these iterates converge to a meaningful solution only for the special case of consistent systems with coefficient matrices A of index 1. Here, we consider the QMR method for the general case of $N \times N$ linear systems, with singular or nonsingular coefficient matrices.

The QMR algorithm belongs to the family of Krylov subspace methods. Let $x_0 \in \mathbb{C}^N$ be an initial guess for the solution of (1.1), and $r_0 = b - Ax_0$ the corresponding initial residual, of length $\rho_1 = \|r_0\|$. Choosing $v_1 = r_0/\rho_1$ as the starting right Lanczos vector for Algorithm 4.1, and w_1 with $\|w_1\| = 1$ as an arbitrary starting left Lanczos vector, one obtains the four basis sets, V_n, W_n, P_n, and Q_n, of which the ones of interest are V_n and P_n, related by:

$$V_n = P_n R_n \quad \text{and} \quad AP_n = V_{n+1} L_n.$$

Once the basis vectors are constructed, the nth QMR iterate is selected from the shifted Krylov subspace $x_0 + K_n(r_0, A)$ as

$$x_n = x_0 + P_n y_n, \tag{5.1}$$

where $y_n \in \mathbb{C}^n$ is defined by the quasi-minimal residual condition

$$\|f_{n+1} - L_n y_n\| = \min_{y \in \mathbb{C}^n} \|f_{n+1} - L_n y\|. \tag{5.2}$$

This is an $(n+1) \times n$ least-squares problem, where

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

and we have used the normalization (2.5) of the Lanczos vectors; otherwise, the least-squares problem above also involves a diagonal scaling matrix.

Note that, by setting

$$z_n = (R_n)^{-1} y_n,$$

and inserting in (5.2), one obtains the equivalent least-squares problem

$$\|f_{n+1} - L_n R_n z_n\| = \min_{z \in \mathbb{C}^n} \|f_{n+1} - L_n R_n z\|,$$

which is exactly the least-squares problem solved by the QMR algorithm based on the three-term Lanczos process. Thus, the QMR iterates (5.1) are, in exact arithmetic, identical to the iterates of the original QMR algorithm [5]. However, as was shown in [6], in finite precision arithmetic, the coupled QMR algorithm is more robust than the three-term recurrence version.

Like the original QMR algorithm, the new implementation has a number of desirable properties. Since they are equivalent, all the theoretical properties of the three-term recurrence QMR algorithm carry over to the coupled version. The new method also shows a smooth and nearly monotone convergence curve. At all times during the iteration, an upper bound for the QMR residual norm is available at no extra cost, and, as the examples will show, this upper bound is a very good indicator of the convergence of the method. As in the original version, the QMR iterate has a short update formula, involving only one direction vector that can be updated with a two-term recurrence. This is slightly cheaper than in the old method, where the corresponding search directions for the update of the iterates had a three-term recurrence. Finally, the new version seems to be significantly more robust than the old version, though no theoretical results are available to explain the differences. For a full discussion of the properties of the two QMR algorithms, see [5, 6].

6 Numerical experiments

In this section, we present a few numerical examples with the new implementation of the QMR algorithm. All these examples were run either on a Cray–2 at the NASA Ames Research Center or on a Cray Y–MP at AT&T Bell Laboratories, with a machine epsilon of about 5.0E−29.

In the plots below, we always show two curves, the computed scaled residual norm $\|r_n\|/\|r_0\|$ (solid line) and the residual norm upper bound (dotted line). Recall that the upper bound is available at each step at no extra cost.

Example 6.1. This example is taken from [2]. Here we consider the differential equation

$$Lu = f \quad \text{on} \quad (0,1)\times(0,1)\times(0,1), \tag{6.1}$$

where

$$\begin{aligned} Lu := & -\frac{\partial}{\partial x}\left(e^{-xy}\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(e^{xy}\frac{\partial u}{\partial y}\right) - \frac{\partial}{\partial z}\left(e^{xy}\frac{\partial u}{\partial z}\right) \\ & +50(x+y+z)\frac{\partial u}{\partial x} + \left(\frac{1}{1+x+y+z} - 250\right)u, \end{aligned}$$

with Dirichlet boundary conditions $u = 0$. The right-hand side f was chosen so that

$$u = (1-x)(1-y)(1-z)(1-e^{-x})(1-e^{-y})(1-e^{-z})$$

is the exact solution of (6.1). We discretized (6.1) using centered differences on a $40\times 40\times 40$ grid with mesh size $h = 1/41$. This leads to a linear system of order $N = 64000$ with 438400 nonzero entries. The starting vector w_1 was a random vector, and the initial guess x_0 was zero. The example was run with a right SSOR preconditioner [1], with $\omega = 1.0$. The algorithm stagnated at 1.6E−25 after 119 steps. For the V–W sequence, it built 4 blocks of size 2, and forced a block closure once. For the P–Q sequence, it built 2 blocks of size 2, and forced a block closure once.

Example 6.2. This example was provided to us by V. Venkatakrishnan [10], from the Numerical Aerodynamic Simulation Group at the NASA Ames Research Center. It comes from an unstructured 2-D Euler solver, and it corresponds to the system at the beginning of time-stepping. The linear system is of order $N = 62424$ with 1717792 nonzero elements. The right-hand side b and the starting vector w_1 were both random vectors, while the initial guess x_0 was zero. Once again, the example was run with a right SSOR preconditioner, with $\omega = 1.0$. The algorithm was stopped once it reached 1.0E−20, after 107 steps. For the V–W sequence, it built 3 blocks of size 2, and 1 block of size 3. For the P–Q sequence, it built no blocks, but forced a block closure once.

Example 6.3. This example was taken from [8], where McQuain and his collaborators investigated the applicability of iterative methods to linear systems arising in circuit simulation. In particular, they studied the solution of systems involving the Jacobians that arise when a homotopy algorithm is applied to the computation of the DC operating point of a circuit.

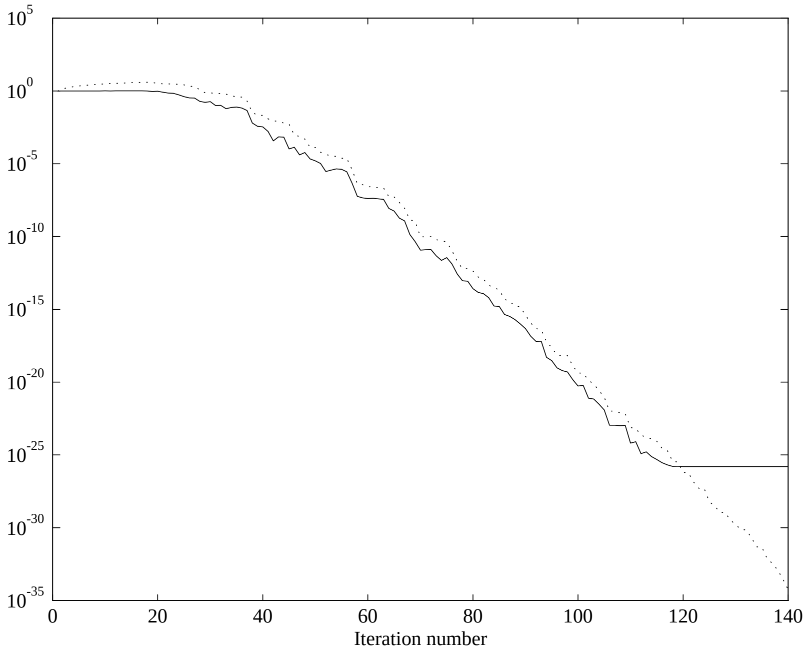

Figure 1: Convergence curves for Example 6.1.

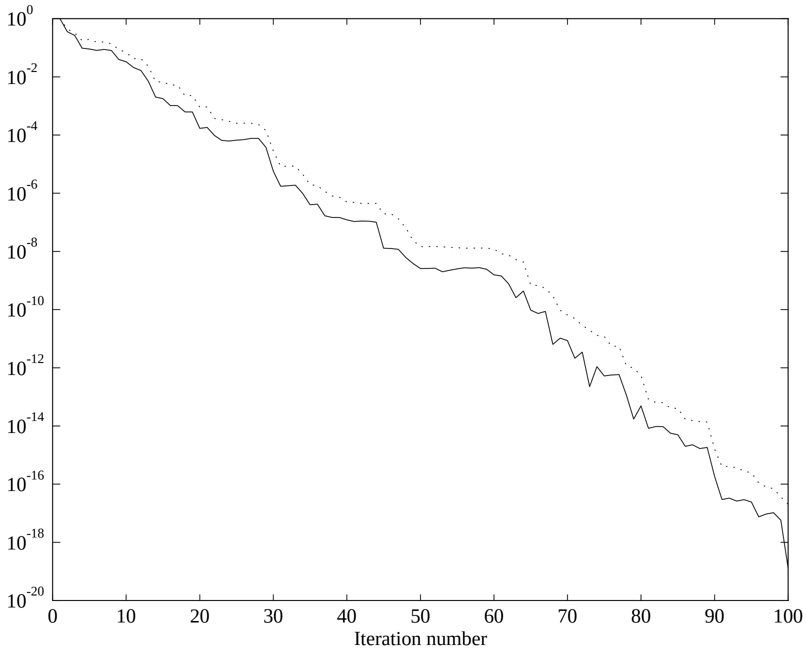

Figure 2: Convergence curves for Example 6.2.

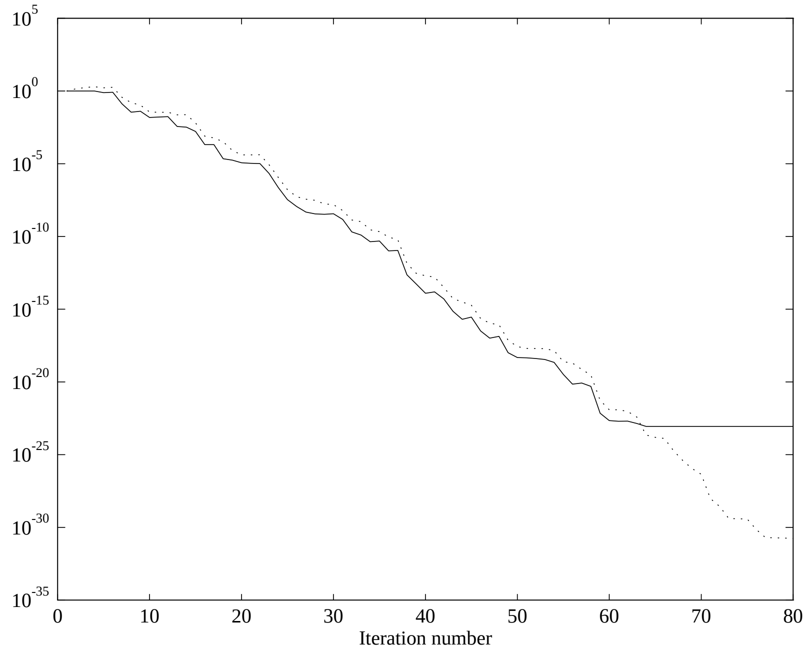

Figure 3: Convergence curves for Example 6.3.

While these linear systems seem to be rather difficult, they are not intractable. In the example, a Jacobian of order $N = 1853$ with 8994 nonzero elements was considered; it is the first Jacobian from the IS7B sequence discussed in Section 5.3 of [8]. The right-hand side b was obtained by moving the last column of the original rectangular $n \times (n+1)$ Jacobian to the other side. The starting guess x_0 was zero, and the starting vector w_1 was set $w_1 = v_1 = b$. The example was run with the variant described in [5] of Saad's ILUT preconditioner [9], with no additional fill-in allowed and a drop tolerance of 0.001, which generated a matrix L with 4766 nonzero elements, and a matrix U with 5034 nonzero elements. The algorithm stagnated at 8.7E$-$23 after 67 steps. It built no look-ahead blocks.

The examples shown illustrate several points. As already noted, the coupled QMR algorithm has a rather smooth and almost monotone convergence behavior. It makes available a residual norm upper bound that is a very good indicator of the convergence of the method. Finally, while it is possible to construct examples with arbitrary look-ahead structure, numerical experience seems to indicate that, on the average, the look-ahead strategy does not build many look-ahead blocks. Indeed, the vast majority of the steps taken by the coupled two-term look-ahead Lanczos process are regular steps; furthermore, from the look-ahead steps of size greater than 1 taken, the majority are blocks of size 2.

7 Concluding remarks

We have presented details of an implementation of a new look-ahead algorithm for constructing Lanczos vectors based on coupled two-term recurrences instead of the usual three-term recurrences. We then discussed a new implementation of the quasi-minimal residual algorithm, using the coupled process to build the basis for the Krylov subspace. While the theoretical results derived for the original algorithm carry over to the new one, the latter was shown in examples to have better numerical properties.

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

Acknowledgements

We would like to thank V. Venkatakrishnan and W. McQuain for providing the matrices for Examples 6.2 and 6.3.

References

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

[2] D. Baxter, J. Saltz, M. Schultz, S. Eisenstat, and K. Crowley, *An experimental study of methods for parallel preconditioned Krylov methods*, Technical Report RR–629, Department of Computer Science, Yale University, New Haven, Connecticut, 1988.

[3] R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput. **14** (1993), to appear.

[4] R.W. Freund and M. Hochbruck, *On the use of two* QMR *algorithms for solving singular systems and applications in Markov chain modeling*, Journal of Numerical Linear Algebra with Applications **2** (1993), to appear.

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

[6] R.W. Freund and N.M. Nachtigal, *An implementation of the* QMR *method based on coupled two-term recurrences*, Technical Report 92.15, RIACS, NASA Ames Research Center, Moffett Field, California, June 1992.

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

[8] W.D. McQuain, C.J. Ribbens, L.T. Watson, and R.C. Melville, *Preconditioned iterative methods for sparse linear algebra problems arising in circuit simulation*, Technical Report 92–07, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, March 1992.

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

[10] V. Venkatakrishnan and T.J. Barth, *Unstructured grid solvers on the* iPSC/860, Parallel Computational Fluid Dynamics, Rutgers, New Jersey, May 1992.