Software for simplified Lanczos and QMR algorithms

Roland W. Freund

AT&T Bell Laboratories, Room 2C–420, 600 Mountain Avenue, Murray Hill, NJ 07974–0636, USA

Noël M. Nachtigal

Mathematical Sciences Section, Oak Ridge National Laboratory, P.O. Box 2008, Building 6012, Oak Ridge, TN 37831–6367, USA

Abstract

The nonsymmetric Lanczos process simplifies when applied to J-symmetric and J-Hermitian matrices, and work and storage requirements are roughly halved compared to the general case. In this paper, we describe FORTRAN-77 implementations of simplified versions of the look-ahead Lanczos algorithm and of the quasi-minimal residual (QMR) method, which is a Lanczos-based iterative procedure for the solution of linear systems. These implementations of the simplified algorithms complete our software package QMRPACK, which so far contained only codes for Lanczos and QMR algorithms for general matrices. We describe in some detail the use of two routines, one for the solution of linear systems, and the other for eigenvalue computations. We present examples that lead to J-symmetric and J-Hermitian matrices. Results of numerical experiments are reported.

Keywords. Lanczos process; quasi-minimal residual iteration; linear system; eigenvalue computation; J-symmetric matrix; J-Hermitian matrix; look-ahead; three-term recurrence; coupled two-term recurrence.

1. Introduction

The nonsymmetric Lanczos process [30] is a powerful tool for many large-scale matrix computations, including the solution of large linear systems, the computation of approximate eigenvalues of large matrices, and the computation of low-order Padé approximations to high-order transfer functions; see, e.g., [5, 8, 12, 14] and the references given there. However, the Lanczos algorithm in its original form [30] is susceptible to potential exact or near breakdowns caused by division by zero or a near-zero number, respectively. In the early 1980s, Taylor [40] and Parlett, Taylor, and Liu [37] showed how the problem of exact and near breakdowns can be overcome by using so-called *look-ahead* techniques. In recent years, there has been a surge of interest in the Lanczos process with look-ahead, and a number of new look-ahead Lanczos algorithms have been proposed; see, e.g., [12, 14] and the references given there. One of the principal goals of our own work on look-ahead Lanczos algorithms was the development of public-domain software for the algorithms we proposed. This has led to the software package QMRPACK [20], which is available from NETLIB [6]. QMRPACK contains FORTRAN-77 implementations of the three-term look-ahead Lanczos algorithm developed jointly with Gutknecht [15], of the coupled two-term Lanczos algorithm proposed in [19], and two implementations of the quasi-minimal residual (QMR) method [18, 19] for solving linear systems based on the three-term, respectively the coupled two-term, look-ahead Lanczos algorithm. Implementations of no-look-ahead versions of all these algorithms are also included in QMRPACK.

Lanczos already pointed out that his algorithm simplifies and that work and storage are roughly halved, when applied to Hermitian or symmetric matrices; see [32, p. 176]. The resulting algorithms are the Hermitian Lanczos process and the complex symmetric Lanczos algorithm, respectively. For the Hermitian Lanczos process, breakdowns can be excluded, and this algorithm has been widely used and extensively studied; see, e.g., [4]. The complex symmetric Lanczos algorithm suffers from the same breakdown problem as the general Lanczos process, and it has received less attention, despite the fact that complex symmetric matrices arise frequently in practice; see, e.g., [9].

Freund and Zha [11, 13, 22] observed that the look-ahead Lanczos algorithm also simplifies for matrices that are J-Hermitian or J-symmetric with respect to inner products induced by general nonsingular, not necessarily positive-definite matrices J. There are important special cases that lead to J-Hermitian or J-symmetric matrices. Examples include Hamiltonian matrices, Toeplitz systems, and symmetric, but highly indefinite linear systems with symmetric indefinite preconditioners.

The purpose of this paper is to describe FORTRAN-77 implementations of simplified versions of the three-term and coupled two-term Lanczos algorithms and the associated QMR algorithms for general J-symmetric and J-Hermitian matrices. These implementations of the simplified algorithms represent counterparts to the algorithms already contained in our software package QMRPACK, and their addition now completes QMRPACK.

The paper is organized as follows. In Section 2, we briefly review the Lanczos process and the QMR method. In Section 3, the general concept of simplified Lanczos algorithms is discussed, and a few examples are given. In Section 4, we describe implementation issues for the simplified algorithms. In Section 5, we consider one of the algorithms, QMR-from-BCG, in more detail. In Section 6, we give a list of the simplified algorithms we have implemented, and in Section 7, we describe example drivers for two of the algorithms. In Section 8, results of numerical experiments are reported. Finally, in Section 9, we make some concluding remarks.

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

$$K_n(M, b) := \operatorname{span}\{\, b, Mb, M^2 b, \ldots, M^{n-1} b \,\}$$

for the nth *Krylov subspace* of $\mathbb{C}^N$ generated by the matrix $M \in \mathbb{C}^{N \times N}$ and the vector $b \in \mathbb{C}^N$.

2. A brief review of the Lanczos and QMR algorithms

In this section, we briefly review the three-term look-ahead Lanczos algorithm proposed in [15], its coupled two-term version developed in [19], and the QMR method [18, 19]. For a detailed description of these algorithms, we refer the reader to the original papers [15, 18, 19].

In the following, $A \in \mathbb{C}^{N\times N}$ is always assumed to be a given $N \times N$ matrix.

2.1. *The Lanczos algorithm*

Let $v_1,\, w_1 \in \mathbb{C}^N$ be any two vectors different from the zero vector. Starting with v_1 and w_1, the look-ahead Lanczos algorithm generates two sequences of vectors,

$$\{v_i\}_{i=1}^n \quad \text{and} \quad \{w_i\}_{i=1}^n, \qquad n = 1, 2, \ldots, \tag{1}$$

such that

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

and the two sequences are block biorthogonal:

$$w_i^T v_n = w_n^T v_i = 0 \quad \text{for all} \quad i = 1, 2, \ldots, n_{l(n)} - 1. \tag{2}$$

Here, for each $n = 1, 2, \ldots$, the index $l = l(n)$ is the integer defined by $n_l \le n \le n_{l+1} - 1$, where

$$1 = n_1 < n_2 < \cdots < n_j < \cdots \tag{3}$$

is an ascending sequence of integers that is used to keep track of the look-ahead steps. More precisely, $n = n_j$ and $n = n_{j+1} - 1$ mark the beginning and the end, respectively, of the jth look-ahead step, and $h_j = n_{j+1} - n_j$ is the *length* of the jth look-ahead step. A *true* jth look-ahead step with length $h_j > 1$ is taken if $w_{n_j}^T v_{n_j} = 0$ or $w_{n_j}^T v_{n_j} \approx 0$, to avoid an exact or near breakdown that would occur due to the necessary division by $w_{n_j}^T v_{n_j}$ in the classical no-look-ahead Lanczos algorithm [30]. In particular, if $h_j = 1$ for all j, then no true look-ahead steps are taken at all, and the look-ahead Lanczos algorithm reduces to the classical Lanczos process. In this case, $n_{l(n)} = n$, and the relation (2) just reduces to the standard biorthogonality condition of the Lanczos vectors.

The Lanczos vectors (1) can be generated by means of three-term or coupled two-term block recurrences. First, we consider the three-term algorithm. We denote by

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

the matrices whose columns are the first n vectors of the two sequences (1), respectively. For $j = 1, 2, \ldots, l = l(n)$, we denote by

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

the submatrices of V_n and W_n, respectively, containing the Lanczos vectors from the jth look-ahead step. Moreover, we set

$$D^{(j)} = \left(W^{(j)}\right)^T V^{(j)}.$$

If $n+1 = n_{l+1}$, then the next pair of Lanczos vectors v_{n+1} and w_{n+1} are constructed as so-called *regular* vectors that are biorthogonal to all previously generated vectors. This construction is done by means of the following three-term block recurrences:

$$
\begin{aligned}
v_{n+1}\rho_{n+1} &= Av_n - V^{(l)}\left(D^{(l)}\right)^{-1}\left(W^{(l)}\right)^T Av_n - V^{(l-1)}\left(D^{(l-1)}\right)^{-1}\left(W^{(l-1)}\right)^T Av_n, \\
w_{n+1}\eta_{n+1} &= A^T w_n - W^{(l)}\left(D^{(l)}\right)^{-T}\left(V^{(l)}\right)^T A^T w_n - W^{(l-1)}\left(D^{(l-1)}\right)^{-T}\left(V^{(l-1)}\right)^T A^T w_n.
\end{aligned}
\tag{4}
$$

If $n+1 < n_{l+1}$, then the vectors v_{n+1} and w_{n+1} are constructed as so-called *inner* vectors that are biorthogonal to all vectors of the previous blocks $V^{(j)}$ and $W^{(j)}$, $j = 1, 2, \ldots, l-1$, but they are, in general, not biorthogonal to the vectors of the current blocks $V^{(l)}$ and $W^{(l)}$. These inner vectors are generated as follows:

$$
\begin{aligned}
v_{n+1}\rho_{n+1} &= Av_n - v_n h_{nn} - V^{(l-1)}\left(D^{(l-1)}\right)^{-1}\left(W^{(l-1)}\right)^T Av_n, \\
w_{n+1}\eta_{n+1} &= A^T w_n - w_n \tilde{h}_{nn} - W^{(l-1)}\left(D^{(l-1)}\right)^{-T}\left(V^{(l-1)}\right)^T A^T w_n.
\end{aligned}
\tag{5}
$$

In (5), the coefficients h_{nn} and $\tilde{h}_{nn}$ are free parameters. In our implementation of the algorithm, we use a choice based on a heuristic to enhance the numerical stability of the recurrences (5). Furthermore, in (4) and (5), $\rho_{n+1} \neq 0$ and $\eta_{n+1} \neq 0$ are scaling factors to normalize the Lanczos vectors. In our look-ahead algorithms, we normalize the Lanczos vectors to have unit length:

$$
\|v_n\| = \|w_n\| = 1 \quad \text{for all} \quad n = 1, 2, \ldots . \tag{6}
$$

Our strategy to determine the necessity of look-ahead steps is based on the normalization (6). However, for the algorithms without look-ahead, scaling the Lanczos vectors to unit length is not crucial. In fact, in one of the no-look-ahead algorithms, QMR-from-BCG [21], we use a different normalization in order to save work.

The recurrence relations (4) and (5) for all Lanczos vectors $\{v_i\}_{i=1}^{n+1}$ and $\{w_i\}_{i=1}^{n+1}$, which are generated in the course of the first n steps of the algorithm, can be summarized compactly as follows:

$$
\begin{aligned}
AV_n &= V_{n+1} H_n^{(e)}, \\
A^T W_n &= W_{n+1} \tilde{H}_n^{(e)}.
\end{aligned}
\tag{7}
$$

Here,

$$
H_n^{(e)} = \begin{bmatrix} \mu_1 & \nu_2 & 0 & \cdots & 0 \\ \iota_2 & \mu_2 & \nu_3 & \ddots & \vdots \\ 0 & \iota_3 & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & \nu_l \\ \vdots & \ddots & \ddots & \iota_l & \mu_l \\ 0 & \cdots & \cdots & 0 & \iota_{l+1} \end{bmatrix} \quad \text{and} \quad \tilde{H}_n^{(e)} = \begin{bmatrix} \tilde{\mu}_1 & \tilde{\nu}_2 & 0 & \cdots & 0 \\ \tilde{\iota}_2 & \tilde{\mu}_2 & \tilde{\nu}_3 & \ddots & \vdots \\ 0 & \tilde{\iota}_3 & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & \ddots & \tilde{\nu}_l \\ \vdots & \ddots & \ddots & \tilde{\iota}_l & \tilde{\mu}_l \\ 0 & \cdots & \cdots & 0 & \tilde{\iota}_{l+1} \end{bmatrix} \tag{8}
$$

are block-tridiagonal matrices of size $(n+1) \times n$ that are also upper Hessenberg. The matrices (8) are connected by a diagonal scaling:

$$
\tilde{H}_n^{(e)} = \Gamma_{n+1}^{-1} H_n^{(e)} \Gamma_n,
$$

where

$$\Gamma_i = \operatorname{diag}(\gamma_1, \gamma_2, \ldots, \gamma_i), \quad i = 1, 2, \ldots,$$

are nonsingular diagonal matrices. In (8), the jth diagonal blocks μ_j and $\tilde{\mu}_j$ of $H_n^{(e)}$ and $\tilde{H}_n^{(e)}$, respectively, are matrices of size $h_j \times h_j$, $j = 1, 2, \ldots, l$, and the sizes of the sub- and superdiagonal blocks ι_j, ν_j, $\tilde{\iota}_j$, and $\tilde{\nu}_j$ are accordingly. In particular, if no true look-ahead steps occur, then the matrices $H_n^{(e)}$ and $\tilde{H}_n^{(e)}$ are just the scalar tridiagonal matrices generated by the classical no-look-ahead Lanczos algorithm.

For the coupled two-term version of the Lanczos process, the three-term recurrences summarized in (7) are split up into coupled two-term recursions:

$$\begin{aligned} V_n &= P_n U_n, & A P_n &= V_{n+1} L_n, \\ W_n &= Q_n \Gamma_n^{-1} U_n \Gamma_n, & A^T Q_n &= W_{n+1} \Gamma_{n+1}^{-1} L_n \Gamma_n. \end{aligned} \tag{9}$$

Here, L_n is a lower block-bidiagonal matrix of size $(n+1) \times n$ and U_n is an upper block-bidiagonal matrix of size $n \times n$. The matrices L_n and U_n are the factors of block-LU decompositions,

$$H_n^{(e)} = L_n U_n \quad \text{and} \quad \tilde{H}_n^{(e)} = \Gamma_{n+1}^{-1} L_n U_n \Gamma_n, \tag{10}$$

of the block-tridiagonal matrices (8). Furthermore, in (9),

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

are $N \times n$ matrices whose columns are a second set of basis vectors

$$\{p_i\}_{i=1}^n \quad \text{and} \quad \{q_i\}_{i=1}^n. \tag{11}$$

The vectors (11) span the same spaces as the Lanczos vectors (1), i.e.,

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

However, instead of the block-biorthogonality condition (2), the basis vectors (11) satisfy the following block-A-biorthogonality condition:

$$q_i^T A p_n = q_n^T A p_i = 0 \quad \text{for all} \quad i = 1, 2, \ldots, m_{k(n)} - 1.$$

Here, for each $n = 1, 2, \ldots$, the index $k = k(n)$ is the integer defined by $m_k \le n \le m_{k+1} - 1$, where

$$1 = m_1 < m_2 < \cdots < m_j < \cdots$$

is an ascending sequence of integers that is used to keep track of the look-ahead steps in the construction of the vectors (11). We remark that these look-ahead steps are different from the look-ahead steps that occur in the construction of the Lanczos vectors (1). In fact, due to these two types of look-ahead, the coupled two-term recurrences are somewhat more complicated than the recursions (4)–(5) of the three-term version; see [19]. On the other hand, these additional complications are justified, since the coupled two-term recurrences are often considerably more stable than the three-term recursions.

2.2. The QMR method

We now consider linear systems,

$$Ax = b, \tag{12}$$

with $A \in \mathbb{C}^{N \times N}$ as coefficient matrix and $b \in \mathbb{C}^N$. The QMR method uses the look-ahead Lanczos process to compute a sequence of approximate solutions, x_n, $n = 1, 2, \ldots$, for (12).

Let $x_0 \in \mathbb{C}^N$ be an arbitrary initial guess for the solution of (12), and set $r_0 = b - Ax_0$. We choose $v_1 = r_0/\rho_1$, where $\rho_1 \neq 0$ is a scaling factor, and an arbitrary vector $w_1 \in \mathbb{C}$, $w_1 \neq 0$, as starting vectors for the Lanczos process. When the three-term Lanczos algorithm is used, the nth QMR iterate x_n is defined as follows:

$$x_n = x_0 + V_n z_n, \tag{13}$$

where z_n is the unique solution of the least-squares problem

$$\left\| f_{n+1} - \Omega_{n+1} H_n^{(e)} z_n \right\| = \min_{z \in \mathbb{C}^n} \left\| f_{n+1} - \Omega_{n+1} H_n^{(e)} z \right\|. \tag{14}$$

Here, Ω_{n+1} and f_{n+1} are given by

$$\Omega_{n+1} = \operatorname{diag}(\|v_1\|, \|v_2\|, \ldots, \|v_{n+1}\|) \quad \text{and} \quad f_{n+1} = \begin{bmatrix} \|v_1\|\rho_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{n+1}.$$

The residual vector $r_n = b - Ax_n$ of the iterate (13) satisfies the relation

$$r_n = \left(V_{n+1} \Omega_{n+1}^{-1} \right) \left(f_{n+1} - \Omega_{n+1} H_n^{(e)} z_n \right). \tag{15}$$

Thus the choice (14) of z_n means a quasi-minimization of the residual norm in the sense that the Euclidean norm of the second factor in the representation (15) of r_n is minimal.

When the coupled two-term Lanczos algorithm is used, the nth QMR iterate x_n is expressed in terms of P_n and L_n. In fact, using the first relations in (9) and (10), respectively, and setting $y_n = U_n z_n$, it follows from (13) and (14) that

$$x_n = x_0 + P_n y_n,$$

where y_n is the unique solution of the least-squares problem

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

2.3. Eigenvalue approximations

The Lanczos process can also be employed to compute approximations of the eigenvalues of the matrix $A \in \mathbb{C}^{N \times N}$. Here, both initial Lanczos vectors v_1 and w_1 can be chosen as arbitrary nonzero vectors, and typically, random vectors are used. After n steps, the Lanczos algorithm has generated the $(n+1) \times n$ matrix $H_n^{(e)}$. We denote by H_n the $n \times n$ matrix that is obtained by deleting the last row of $H_n^{(e)}$. The eigenvalues of H_n are used as approximations to the eigenvalues of A. One motivation for this choice is as follows. Rewriting the first relation in (7) in terms of H_n, we obtain

$$AV_n = V_n H_n + [\,0 \quad \cdots \quad 0 \quad v_{n+1}\rho_{n+1}\,],$$

which suggests that the eigenvalues of H_n in some sense approximate the eigenvalues of A. In fact, if $v_{n+1}\rho_{n+1} = 0$ were the zero vector, then all eigenvalues of H_n would also be eigenvalues of A.

3. Simplification

In this section, we review the general concept of simplified Lanczos algorithms, and we describe some examples for which simplification is practical.

3.1. The general concept

Let $J \in \mathbb{C}^{N\times N}$ be a nonsingular matrix. A matrix $A \in \mathbb{C}^{N\times N}$ is called *J-symmetric* if

$$A^T J = JA.$$

A matrix $A \in \mathbb{C}^{N\times N}$ is called *J-Hermitian* if

$$A^H J = JA.$$

We note that, for any $N \times N$ matrix A, there exists a nonsingular $N \times N$ matrix J so that A is J-symmetric; see, e.g., [28, Chap. 3.2]. However, determining such a matrix J is in general not practical, and the really useful cases of J-symmetric matrices are those for which J is known explicitly. A few such cases are discussed in Section 3.2 below. The J-Hermitian property is less general. In fact, there exists a nonsingular $N \times N$ matrix J such that A is J-Hermitian if, and only if, A is similar to A^H; see, e.g., [25, Sec. I.3.2].

Freund and Zha [11, 13, 22] showed that, for J-symmetric and J-Hermitian matrices A, the Lanczos process can be simplified by choosing the second Lanczos starting vector w_1 properly. Recall that, for the QMR method and for Lanczos-based eigenvalue approximations, the vector w_1 is still arbitrary.

First, we consider the case that A is J-symmetric. Here, we set

$$w_1 = \xi_1 J v_1, \quad \text{where} \quad \xi_1 \in \mathbb{C}, \quad \xi_1 \neq 0.$$

Then, it follows that, for all $n = 1, 2, \ldots$,

$$w_n = \xi_n J v_n, \quad \text{for some} \quad \xi_n \in \mathbb{C}, \quad \xi_n \neq 0; \tag{16}$$

see [22]. The scalars ξ_n, $n = 1, 2, \ldots$, depend on the particular normalization used in the Lanczos process. They are connected to the scaling factors ρ_{n+1} and η_{n+1} in the recurrences (4) and (5) as follows:

$$\xi_{n+1} = \xi_n \frac{\rho_{n+1}}{\eta_{n+1}}, \quad n = 1, 2, \ldots .$$

In view of (16), the Lanczos process simplifies for J-symmetric matrices A in that we only need to run the first recursion in (4), respectively (5), to generate the right Lanczos vectors $v_1, v_2, \ldots$, while the left Lanczos vectors $w_1, w_2, \ldots$ are obtained via (16). In particular, the resulting simplified Lanczos procedure does not require matrix-vector multiplications with A^T. We remark that, by (16), the block-biorthogonality relation (2) for the general case now reduces to the Galerkin-type condition

$$v_n^T J v_i = 0, \quad i = 1, 2, \ldots, n_{l(n)} - 1, \tag{17}$$

for the simplified Lanczos process for J-symmetric matrices. The associated simplified QMR algorithm for J-symmetric matrices can also be interpreted as a quasi-minimal residual smoothing applied to the vectors v_n defined by the Galerkin-type condition (17); cf. [18, Proposition 4.1] and [42].

When the coupled two-term algorithm is used, then the additional set of basis vectors (11) are connected by a relation analogous to (16), namely,

$$q_n = \zeta_n J p_n, \quad \text{for some} \quad \zeta_n \in \mathbb{C}, \quad \zeta_n \neq 0. \tag{18}$$

Thus, again, we only need to run the recursions that generate the vectors $p_1, p_2, \ldots$, and we can obtain the vectors $q_1, q_2, \ldots$, via (18).

In the case of a J-Hermitian matrix A, we set

$$w_1 = \xi_1 \overline{J v_1}, \quad \text{where} \quad \xi_1 \in \mathbb{C}, \quad \xi_1 \neq 0.$$

In [22], it is shown that then, for all $n = 1, 2, \ldots$,

$$w_n = \xi_n \overline{J v_n}, \quad \text{for some} \quad \xi_n \in \mathbb{C}, \quad \xi_n \neq 0. \tag{19}$$

Moreover, when the coupled two-term algorithm is used, we also have

$$q_n = \zeta_n \overline{J p_n}, \quad \text{for some} \quad \zeta_n \in \mathbb{C}, \quad \zeta_n \neq 0. \tag{20}$$

Therefore, for J-Hermitian matrices A, the Lanczos process again simplifies, and the recurrences for w_n (and q_n) can be replaced by (19) (and (20)).

3.2. Examples

We now discuss three examples for which the crucial matrix J is known explicitly and thus simplification becomes practical.

Our first example is real linear systems,

$$K x' = c, \tag{21}$$

with symmetric, but indefinite nonsingular coefficient matrices $K \in \mathbb{R}^{N \times N}$. The standard conjugate gradient-type algorithms for the solution of such systems are Paige and Saunders' SYMMLQ and MINRES [35]. However, SYMMLQ and MINRES can only be used in conjunction with symmetric positive-definite preconditioners, which is a rather unnatural restriction when the matrix K is *highly* indefinite with both many positive and many negative eigenvalues. On the other hand, by using the simplification concept, one can design a symmetric QMR algorithm for the solution of (21) that can be combined with symmetric indefinite preconditioners; see [16, 22]. Moreover, the symmetric QMR method has about the same work and storage requirements as SYMMLQ and MINRES, however, it usually converges in considerably fewer iterations; see the numerical results in Section 8. The basic idea is as follows. Let $M = M^T \in \mathbb{R}^{N\times N}$ be an arbitrary nonsingular matrix that we want to use as a preconditioner for the system (21). To allow for left, right, or two-sided preconditioning, we write the preconditioner M in the form $M = M_1 \cdot M_2$. We stress that M_1 and M_2 are arbitrary matrices such that

$$M = M_1 M_2 = M_2^T M_1^T = M^T, \tag{22}$$

and that M_1 and M_2 need not be the transpose of each other. Then, instead of (21), we solve the equivalent preconditioned system

$$Ax = b, \quad \text{where} \quad A = M_1^{-1} K M_2^{-1}, \quad x = M_2 x', \quad \text{and} \quad b = M_1^{-1} c. \tag{23}$$

Note that $M_1 = I$ for right preconditioning, and $M_2 = I$ for left preconditioning. We now set

$$J = M_1^T M_2^{-1}. \tag{24}$$

Using $K = K^T$ and (22), one readily verifies that

$$A^T J = M_2^{-T} K M_2^{-1} = M_1^T M_2^{-1} M_1^{-1} K M_2^{-1} = JA.$$

Hence, the matrix A is J-symmetric, and the Lanczos process simplifies. We remark that this simplification of the Lanczos process was also exploited by Parlett and Chen [36] in a somewhat different context, namely the computation of eigenvalues of symmetric indefinite pencils.

A second example is the iterative solution of linear systems,

$$Tx' = c, \tag{25}$$

with persymmetric coefficient matrices $T \in \mathbb{C}^{N\times N}$, using persymmetric preconditioners. Here, an $N \times N$ matrix T is called *persymmetric* if it is symmetric with respect to the antidiagonal, i.e.,

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

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

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

see, e.g., [38]. An important special case of persymmetric matrices is Toeplitz matrices; recall that a matrix T is said to be *Toeplitz* if it is of the form $T = [\, t_{j-k} \,]_{j,k=1,2,\ldots,N}$. In fact, the most important case of (25) are systems with Toeplitz coefficient matrices. Such systems can be efficiently preconditioned with circulant matrices, which are special Toeplitz matrices and thus persymmetric; see, e.g., [3]. Let $M = M_1 M_2$ be an arbitrary nonsingular persymmetric $N \times N$ matrix that we wish to use as a preconditioner for (25), with M_1 and M_2 applied from the left and right, respectively. The resulting preconditioned coefficient matrix is

$$A = M_1^{-1} T M_2^{-1}. \tag{26}$$

Using the persymmetry of T and M, one easily verifies that the matrix A is J-symmetric for

$$J = M_1^T J_N M_2^{-1}. \tag{27}$$

Hence, the Lanczos process and QMR simplify for persymmetric linear systems with persymmetric preconditioners.

Our third example is the Hamiltonian eigenvalue problem. An $N \times N$ matrix A is called *Hamiltonian* if $N = 2s$ is even and A is of the form

$$A = \begin{bmatrix} F & G \\ H & -F^T \end{bmatrix}, \quad \text{where} \quad F,\ H = H^T,\ G = G^T \in \mathbb{R}^{s\times s}. \tag{28}$$

The problem of computing eigenvalues of Hamiltonian matrices arises in important applications, notably in control theory; see, e.g., [34]. Let $\mathbf{i} = \sqrt{-1}$, let I_s denote the $s \times s$ identity matrix, and set

$$J = \mathbf{i} \begin{bmatrix} 0 & I_s \\ -I_s & 0 \end{bmatrix}. \tag{29}$$

Using the definition (28), one readily verifies that any Hamiltonian matrix A is J-Hermitian. Therefore, the Lanczos process simplifies for Hamiltonian matrices. Note that, due to the factor $\mathbf{i}$, the matrix J has purely imaginary entries, while A itself is a real matrix. However, recall that the normalization factors ξ_n in (16) are still arbitrary, and by choosing purely imaginary ξ_n's, all quantities generated by the Lanczos process can be kept real. Finally, we remark that, for the simplified Lanczos process for Hamiltonian matrices, look-ahead is crucial. More precisely, as Freund and Mehrmann [17] showed, the Hamiltonian structure (28) of A alone implies that the classical Lanczos algorithm would break down in every second step. As a result, the generic situation for Hamiltonian matrices is that the Lanczos process always performs look-ahead steps of length $h_j = 2$, i.e., the look-ahead indices (3) are given by $n_j = 2j-1$. In addition, occasional breakdowns unrelated to the Hamiltonian structure can occur, and then a look-ahead step of length $h_j = 2h$, $h > 2$, is performed; see [17] for details.

4. Implementation issues

In this section, we discuss two issues that come up in the design of codes for simplified Lanczos and QMR algorithms for J-symmetric and J-Hermitian matrices.

The first issue concerns the number of matrix-vector multiplications with the matrix J per Lanczos step. For the three-term algorithm, exactly one product with J is needed per step in order to obtain the left Lanczos vector w_n from v_n via (16), respectively (19) in the J-Hermitian case. For the coupled two-term algorithm, however, one has the option of using either one or two products with J per Lanczos step. Two products with J are needed when both vectors w_n and q_n are obtained from v_n and p_n, respectively, via matrix-vector multiplications with J. On the other hand, it is sufficient to compute only one matrix-vector product, $J \cdot p_n$, since the vector Jv_n can also be obtained by simple vector updates. Indeed, from the recursions summarized as $V_n = P_n U_n$ in (9) and the fact that U_n is a block-bidiagonal matrix, it follows that the vector Jv_n is just a combination of already computed vectors Jp_i. Moreover, if no true look-ahead step is performed, then only the last two vectors, Jp_n and Jp_{n-1}, are needed, and computing Jv_n requires only one SAXPY. In the case of a true look-ahead step, then a few extra SAXPYs are needed to obtain Jv_n

as a combination of previously computed vectors Jp_i. For our codes, we decided in favor of using two matrix-multiplications with J to obtain the vectors $J \cdot v_n$ and $J \cdot p_n$. One reason is that this keeps the codes somewhat simpler. Furthermore, for the applications we have in mind, the matrix J is always such that matrix-vector multiplications with J are very inexpensive. For example, a multiplication of a vector v with the matrix (29) associated with the Hamiltonian eigenvalue problem can be done by first changing the signs of the first half of v and then swapping the two halves of the resulting vector.

Another issue arises when A is the J-symmetric or J-Hermitian coefficient matrix of a preconditioned linear system, such as the preconditioned symmetric system (23). One option is to explicitly work with the preconditioned matrix A and the associated matrix J. The other option is to rewrite the simplified algorithms in terms of the original coefficient matrix of the system to be solved and the preconditioner. For example, for the case of the preconditioned symmetric system (23), this would mean to rewrite the simplified algorithm in terms of the matrix K and the preconditioning factors M_1 and M_2. This can be done by working with the vectors $\hat{v}_n := M_2^{-1} v_n$ and $\hat{w}_n := M_1^{-T} w_n$, instead of the vectors v_n and w_n corresponding to the preconditioned matrix A. The main advantage of this approach is that the resulting algorithm only requires the usual one matrix-vector product with K and one solve with each of the preconditioning factors M_1 and M_2 per iteration, while matrix-vector products with J are completely eliminated. On the other hand, there are important cases where products with the preconditioned matrix A can be computed efficiently; in Sections 4.1 and 4.2 below, we briefly discuss two such cases. This was one of the reasons why we designed QMRPACK so that the algorithms are always applied to the preconditioned matrices; see [12]. This means that the application of the preconditioner rests with the user, inside the user-provided routines for computing matrix-vector products with the preconditioned matrix A and its transpose A^T. In QMRPACK, we use *reverse communication*[1] to call these matrix-vector routines. This way, different preconditioners can easily be incorporated by the user, and no change of the solver routine itself is required. For the codes for simplified Lanczos algorithms, we decided to maintain the design principles of QMRPACK, and thus the simplified algorithms are always applied to the preconditioned matrix A. A disadvantage of this approach is that the matrix-vector products with J are no longer eliminated, and the preconditioning factors M_1 and M_2 have to be chosen carefully so that the multiplications with J are inexpensive. For example, consider the case of preconditioned symmetric systems (23) with matrix J given by (24). Any nonsingular symmetric indefinite preconditioner M can be written as

$$M = U^T D^{-1} U, \quad \text{where} \quad D = D^T \quad \text{is nonsingular.} \tag{30}$$

Furthermore, for standard preconditioners, systems with the preconditioning factor U in (30) can be solved easily, and the matrix D is block diagonal with 1×1 and 2×2 blocks. Based on (30), we now set

$$M_1 = U^T \quad \text{and} \quad M_2 = D^{-1} U.$$

By (24), it follows that

$$J = M_1^T M_2^{-1} = D.$$

Thus, matrix-vector multiplications with J are inexpensive, since D is a block-diagonal matrix with small blocks.

[1]Reverse communication is a mechanism for obtaining from the user specific information required within a subroutine by returning to the calling program. The idea of reverse communication goes back to Lawson and Krogh; see [27] and [23, Sections 3.8 and 4.4].

4.1. Eisenstat's trick for symmetric indefinite systems

By using an observation due to Eisenstat [7], matrix-vector products with the coefficient matrix A of the preconditioned symmetric system (23) can be computed very efficiently when an SSOR-type preconditioner is used. Adapting the usual SSOR preconditioner [1] to symmetric indefinite matrices, one first symmetrically permutes the original coefficient matrix K so that the permuted matrix has the following decomposition:

$$\Pi^T K \Pi = L + D + L^T. \tag{31}$$

Here, D is block diagonal with 1×1 and 2×2 blocks, L is strictly lower triangular, and Π is a permutation matrix. Based on (31), one then uses

$$M = U^T D^{-1} U, \quad \text{where} \quad U = \left(L^T + D\right) \Pi^T, \tag{32}$$

as a preconditioner, resulting in the preconditioned matrix

$$A = U^{-T} K U^{-1} D. \tag{33}$$

Of course, the permutation matrix Π should also be chosen such that the block-diagonal part D is in some sense "as large as possible"; see [16].

We now show how the matrix-vector product $A \cdot p$, for any $p \in \mathbb{R}^N$, can be computed efficiently, by using Eisenstat's trick. By (31)–(33), we have

$$\begin{aligned} A &= U^{-T} \Pi (L + D + L^T) \Pi^T U^{-1} D \\ &= (L + D)^{-1} (L + D + L^T)(L^T + D)^{-1} D \\ &= \left(I + (L + D)^{-1}(L^T + D - D)\right)(L^T + D)^{-1} D \\ &= (L^T + D)^{-1} D + (L + D)^{-1} \left(D - D(L^T + D)^{-1} D\right). \end{aligned} \tag{34}$$

Setting $\tilde{t} = Dp$ and using (34), it follows that the vector $t = A \cdot p$ can be obtained as follows:

$$\begin{aligned} \tilde{t} &= Dp, \\ \hat{t} &= (L^T + D)^{-1} \tilde{t}, \\ t &= \hat{t} + (L + D)^{-1} \left(\tilde{t} - D\hat{t}\right). \end{aligned} \tag{35}$$

Note that (35) requires two solves with the block-triangular matrices $L^T + D$ and $L + D$, together with two multiplications with D. Applying the preconditioner (32) itself involves the same two triangular solves and one of the two products with D. Consequently, by using Eisenstat's trick, the multiplication with the original coefficient matrix K is replaced by one matrix-vector product with D, which is considerably cheaper.

4.2. Toeplitz matrices with circulant preconditioners

When the coefficient matrix T of the persymmetric system (25) is a Toeplitz matrix and M is a circulant preconditioner, then matrix-vector products $A \cdot p$ with the preconditioned matrix (26) can be computed with about the same work that is required to obtain $T \cdot p$, by using observations made in [29, 41].

Let T be an $N \times N$ Toeplitz matrix, and let M be an $N \times N$ circulant matrix. The matrix M can be decomposed as follows:

$$M = F^{-1} D F. \tag{36}$$

Here, $F = \left[\exp\left(-2\pi \mathbf{i} jk/N\right)\right]_{j,k=0,1,\ldots,N-1}$ is the matrix of the discrete Fourier transform on $\mathbb{C}^N$, and D is a diagonal matrix with the eigenvalues of M on the diagonal; see, e.g., [3]. The eigenvalues of M can be computed easily using one fast Fourier transform (FFT). Based on (36), we now choose the preconditioning factors

$$M_1 = F^{-1} \quad \text{and} \quad M_2 = DF.$$

Then the preconditioned matrix (26) is given by

$$A = FTF^{-1}D^{-1}, \tag{37}$$

and the associated matrix (27) is defined as

$$J = F^{-T} J_N F^{-1} D^{-1}. \tag{38}$$

Tismenetsky [41] showed that $F^{-T} J_N F^{-1}$ is a diagonal matrix. Therefore, the matrix J in (38) is also diagonal, and matrix-vector products with J are inexpensive. Now consider the matrix A in (37). Tismenetsky [41] also showed that

$$FTF^{-1} = D_c + F^{-1}\Omega F D_s F^{-1} \Omega^{-1} F,$$

where D_c, D_s, and Ω are diagonal matrices. Inserting this relation into (37), we obtain

$$A = \left(D_c + F^{-1}\Omega F D_s F^{-1} \Omega^{-1} F\right) D^{-1}.$$

Using this representation of A, each matrix-vector product with A can be computed by means of two multiplications with F, two multiplications with F^{-1}, and five multiplications with diagonal matrices. Each matrix-vector product with F and F^{-1} can be done with one FFT of length N. Therefore, a matrix-vector product with A can be computed by essentially four FFTs of length N. This is about the same work as computing a matrix-vector product with T only, which requires two FFTs of length $2N$. Consequently, applying a circulant preconditioner involves practically no additional work.

5. Simplified QMR-from-BCG

In this section, we present a complete statement of one of the simplified QMR algorithms for linear systems with J-symmetric or J-Hermitian coefficient matrices A.

The algorithm we chose is the simplest possible implementation of the QMR method without look-ahead. This is the QMR-from-BCG algorithm proposed by Freund and Szeto [21]. The algorithm uses the coupled-two term recurrences of the classical biconjugate gradient (BCG) algorithm due to Lanczos [31] to generate the vectors v_n and p_n. By adding updates for the QMR iterates and the QMR search directions and some minor scalar operations to these BCG recursions, one obtains an algorithm that is only marginally more expensive than BCG. More precisely, the QMR-from-BCG algorithm requires one additional SAXPY per iteration.

It turns out that the simplified QMR-from-BCG algorithm for both cases, J-symmetric and J-Hermitian matrices, can be stated as one algorithm, by using the following convention. The exponent $*$ means $* = T$ if A is J-symmetric, and it means $* = H$ if A is J-Hermitian. The simplified QMR-from-BCG algorithm is then as follows.

Algorithm 5.1. (QMR-from-BCG without look-ahead for the solution of linear systems $Ax = b$ with J-symmetric ($* = T$) or J-Hermitian ($* = H$) coefficient matrices $A \in \mathbb{C}^{N \times N}$.)

0) *Choose* $x_0 \in \mathbb{C}^N$.

Compute the vectors

$$v_1 = b - Ax_0, \quad p_1 = v_1, \quad \textit{and} \quad q_1 = Jp_1.$$

Compute the scalars

$$\tau_1 = v_1^H v_1 \quad \textit{and} \quad \rho_1 = q_1^* v_1.$$

Set $d_0 = 0 \in \mathbb{C}^N$ *and* $\vartheta_0 = 0 \in \mathbb{C}$.

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

1) *Compute the matrix-vector product*

$$t_n = Ap_n.$$

2) *Compute* $\sigma_n = q_n^* t_n$.

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

3) *Set*

$$\alpha_n = \frac{\rho_n}{\sigma_n} \quad \textit{and} \quad v_{n+1} = v_n - t_n \alpha_n.$$

4) *Compute the scalars*

$$\vartheta_n = \frac{v_{n+1}^H v_{n+1}}{\tau_n}, \quad \psi_n = \frac{1}{1 + \vartheta_n}, \quad \textit{and} \quad \tau_{n+1} = \tau_n \vartheta_n \psi_n.$$

Update the vectors

$$\begin{aligned} d_n &= d_{n-1}(\psi_n \vartheta_{n-1}) + p_n(\psi_n \alpha_n), \\ x_n &= x_{n-1} + d_n. \end{aligned}$$

5) *If* x_n *has converged, then stop.*

6) *If* $\rho_n = 0$, *then stop.*

7) *Set*

$$\begin{aligned} \rho_{n+1} &= (Jv_{n+1})^* v_{n+1}, \quad \beta_n = \frac{\rho_{n+1}}{\rho_n}, \\ p_{n+1} &= v_{n+1} + p_n \beta_n, \\ q_{n+1} &= Jp_{n+1}. \end{aligned}$$

We note that, as a by-product, Algorithm 5.1 generates the upper bound

$$\|b - Ax_n\| \ \le \ \sqrt{(n+1)\tau_n}$$

for the norm of the residual vector $b - Ax_n$. This upper bound is usually very close to the true residual norm, and we use it as part of the convergence check in step 5) of Algorithm 5.1. More precisely, we monitor the upper bound, and once this upper bound is within a factor of 10 of the desired convergence tolerance, then we compute the true residual vector and its norm. Usually, this strategy requires the computation of the true residual only in the last or the last two iteration steps.

6. List of algorithms

The version of QMRPACK, as it is described in [20], contained implementations of the three-term and the coupled two-term Lanczos algorithms (both with and without look-ahead), of the QMR method based on these variants of the Lanczos process, of the QMR-from-BCG algorithm, and of Freund's transpose-free quasi-minimal residual (TFQMR) algorithm [10]. All these algorithms are for general nonsymmetric real or complex matrices. In addition, special versions for symmetric matrices of all the above algorithms, except for TFQMR, are also included in QMRPACK. We remark that, unlike the Lanczos process and QMR, TFQMR uses only matrix-vector products with A and thus it does not simplify when A is symmetric or Hermitian. QMRPACK provides codes for all the above algorithms in single and double precision, real and complex data types. Finally, we note that QMRPACK is available from NETLIB [6]. The package is distributed in the form of the compressed tar file "qmrpack.tar.Z" in the "linalg" subdirectory of NETLIB. It can be retrieved via anonymous ftp (ftp to "netlib.att.com" and then get the file "qmrpack.tar.Z" from the directory "netlib/linalg"). It is also possible to retrieve single algorithms from QMRPACK. The single algorithms are contained in the "linalg/qmr" subdirectory of NETLIB. To obtain a list of the available algorithms, send an e-mail message consisting of the single line "send index from linalg/qmr" to the address "netlib@research.att.com", or get the file "linalg/qmr/index" via anonymous ftp.

The new codes we have added to the ones already in QMRPACK are the simplified versions for J-symmetric and J-Hermitian matrices of the three-term and coupled two-term Lanczos algorithms, both with and without look-ahead, of the QMR method based on these Lanczos variants, and of the QMR-from-BCG algorithm. We also added codes for all these algorithms for Hermitian matrices; this was done to complement the codes for symmetric matrices that were already included in QMRPACK. A list of all these routines, including the ones from the original QMRPACK [20], is as follows. The solver routines have names of the form `PMALG`, where:

`P` denotes the precision and data type:

`C`	single-precision complex
`D`	double-precision real
`S`	single-precision real
`Z`	double-precision complex

`M` denotes the matrix type:

`S`	symmetric
`H`	Hermitian
`U`	unsymmetric
`P`	J-symmetric
`Q`	J-Hermitian

`ALG` denotes the algorithm:

`CPL`	QMR based on coupled two-term look-ahead Lanczos
`CPX`	QMR based on coupled two-term no-look-ahead Lanczos
`LAL`	eigenvalue solver based on three-term look-ahead Lanczos
`QBG`	QMR-from-BCG
`QMR`	QMR based on three-term look-ahead Lanczos
`QMX`	QMR based on three-term no-look-ahead Lanczos

For example, `ZPCPL` denotes the double-precision complex code for the QMR method based on coupled two-term Lanczos with look-ahead, for J-symmetric matrices.

We note that the routines with matrix type `M` = `S`, `U` are the ones that were already in QMRPACK and the routines with matrix type `M` = `H`, `P`, `Q` are the ones we added. In addition, QMRPACK also contains the routines `PUTFX`, where `P` = `C`, `D`, `S`, `Z`, with implementations of TFQMR for nonsymmetric matrices.

7. Two example drivers

In this section, we discuss two example drivers for two of the routines in more detail. The routines we consider are the double-precision real linear-systems solver `DPCPL` and the double-precision real eigenvalue solver `DPLAL` for J-symmetric matrices. We remark that our routines use reverse communication to compute all matrix-vector products with A and J on the driver level. This makes it easy for the user to incorporate his own routines for computing these matrix-vector products. Furthermore, all our routines for the algorithms themselves are completely independent of the storage schemes used for the matrices A and J.

7.1. Subroutine `DPCPL`

The subroutine `DPCPL` is the double precision real implementation of the J-symmetric QMR algorithm based on the coupled two-term look-ahead Lanczos variant proposed in [19]. Its calling sequence is:

```
CALL DPCPL (NDIM,NLEN,NLIM,MAXP,MAXV,M,MVEC,NORMS,DWK,IDX,IWK,VECS,TOL,INFO)
```

where the parameters are similar to the ones for the routine `DUCPL` described in [20]. The parameters that determine the memory requirements are the work array `VECS`, of dimensioned size `NDIM` with

4*MVEC+3 columns, and the parameters controlling the maximal size of the look-ahead blocks in the p and v sequences, MAXP and MAXV respectively. MVEC is a function of MAXP and MAXV, being at least equal to max(MAXP,MAXV) + 1. The remaining parameters are small work arrays, the convergence tolerance TOL, the upper limit NLIM on the number of iterations, and the flags INFO. All parameters are described in detail in [20]. The main differences between these codes and the nonsymmetric counterparts are the lower memory use—the main vector array VECS requires only 4*MVEC+3 columns, as opposed to 5*MVEC+3—and the reverse-communication protocol. The routine returns to caller as part of the reverse-communication process, with INFO(2) = 1 if a multiplication with A is needed, or with INFO(2) = 3 if a multiplication with J is needed. The multiplication takes as its argument column INFO(3) of VECS, and must return the result in column INFO(4) of VECS, while preserving its argument. The following pseudo-code illustrates this procedure:

```
   INFO(2) = 0
10 CALL DPCPL (NDIM,...,INFO)
   REVCOM = INFO(2)
   COLX   = INFO(3)
   COLB   = INFO(4)
   IF (REVCOM.EQ.1) THEN
      Multiply VECS(1,COLB) by A
      Store result in VECS(1,COLX)
      GO TO 10
   ELSE IF (REVCOM.EQ.3) THEN
      Multiply VECS(1,COLB) by the matrix J
      Store result in VECS(1,COLX)
      GO TO 10
   END IF
20 Check INFO(1) for convergence or errors
```

7.2. *Subroutine* DPLAL

The subroutine DPLAL is the double precision real implementation of the J-symmetric eigenvalue solver based on the three-term look-ahead Lanczos variant proposed in [15]. Its calling sequence is:

```
   CALL DPLAL (NDIM,NLEN,NLIM,MAXN,MAXV,M,NORM,DWK,IDX,IWK,HWK,VECS,INFO)
```

The calling parameters are similar to the parameters to the routine DULAL, described in [20]. Again, the main parameters of interest are the large work array VECS, and the maximum block size MAXV. The main differences again arise in the lower memory use—the main vector array VECS requires only MAXV+5 columns, as opposed to 2*MAXV+6—and in the reverse-communication protocol.

The routine returns to caller as part of the reverse-communication process, with INFO(2) = 1 if a multiplication with A is needed, or with INFO(2) = 3 if a multiplication with J is needed. The multiplication takes as its argument column INFO(3) of VECS, and must return the result in column INFO(4) of VECS, while preserving its argument. The following pseudo-code illustrates this procedure:

```
   INFO(2) = 0
```

```
10 CALL DPLAL (NDIM,...,INFO)
   REVCOM = INFO(2)
   COLX   = INFO(3)
   COLB   = INFO(4)
   IF (REVCOM.EQ.1) THEN
      Multiply VECS(1,COLB) by A
      Store result in VECS(1,COLX)
      GO TO 10
   ELSE IF (REVCOM.EQ.3) THEN
      Multiply VECS(1,COLB) by the matrix J
      Store result in VECS(1,COLX)
      GO TO 10
   END IF
20 CONTINUE
```

Finally, we note that the J-symmetric codes only have one starting vector, as the auxiliary one is generated by multiplication with J.

8. Numerical examples

In this section, we present results of numerical experiments with our implementations of the simplified algorithms. The first two examples are real symmetric indefinite linear systems with symmetric indefinite preconditioner that result in J-symmetric matrices, as described in Section 3.2. The third example is a Hamiltonian eigenvalue problem.

The first example is a real symmetric linear system (21) with K of the form

$$K = \begin{bmatrix} B & C^T \\ C & -D \end{bmatrix}, \quad B = B^T \in \mathbb{R}^m, \quad D = D^T \in \mathbb{R}^{N-m},$$

that results from a finite-element approximation of a Stokes problem (see, e.g., [26]) in two dimensions, using the stabilized discretization from [39]. The matrix K is of the order $N = 1634$ with $m = 1122$ positive and $N - m = 512$ negative eigenvalues. The matrix K is sparse with 13702 nonzero elements; its sparsity structure is shown in Figure 1. The preconditioner M was chosen as the diagonal of K, and it was applied from the right, i.e., $M_1 = I$ and $M_2 = M$. In Figure 2, we show the relative residual norm of the preconditioned system,

$$\varrho_n = \frac{\|b - Ax_n\|}{\|b - Ax_0\|}, \quad n = 1, 2, \ldots, \tag{39}$$

for the simplified QMR-from-BCG Algorithm 5.1 (solid line) and MINRES [35] (dotted line). Recall that MINRES requires a positive-definite preconditioner, and we used the diagonal matrix $|M|$ consisting of the absolute values of the diagonal elements of K as the preconditioner. Consequently, QMR and MINRES are actually applied to two different linear systems, namely $KM^{-1}x = b$ and $K|M|^{-1}x = b$, respectively. Notice that QMR converges almost twice as fast as MINRES. We stress that both algorithms have essentially the same work and storage requirements.

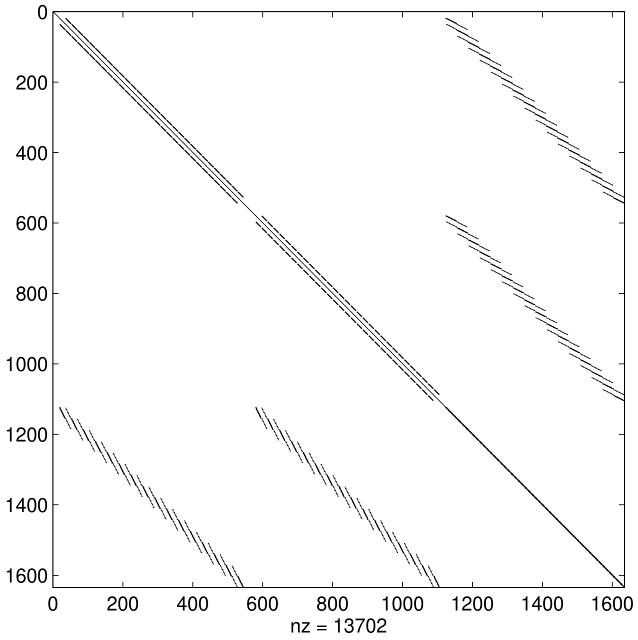

Fig. 1. Sparsity structure of the matrix K for Stokes example.

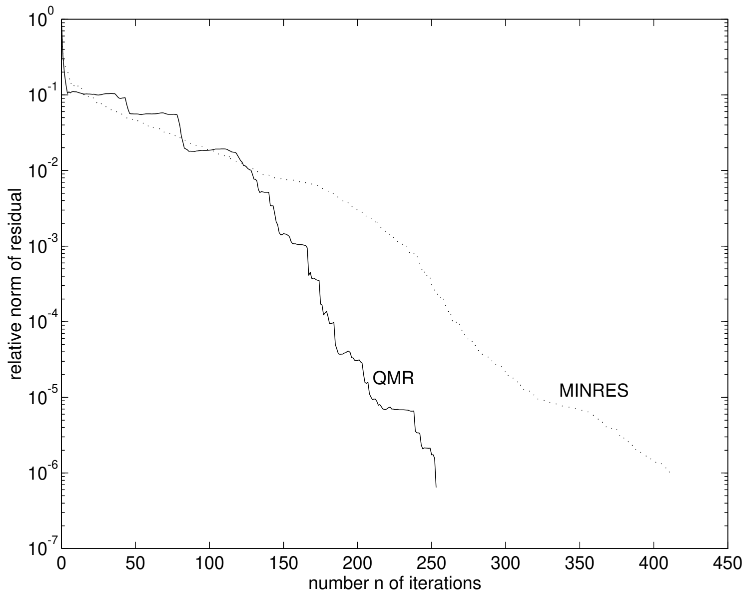

Fig. 2. Convergence curves for Stokes example.

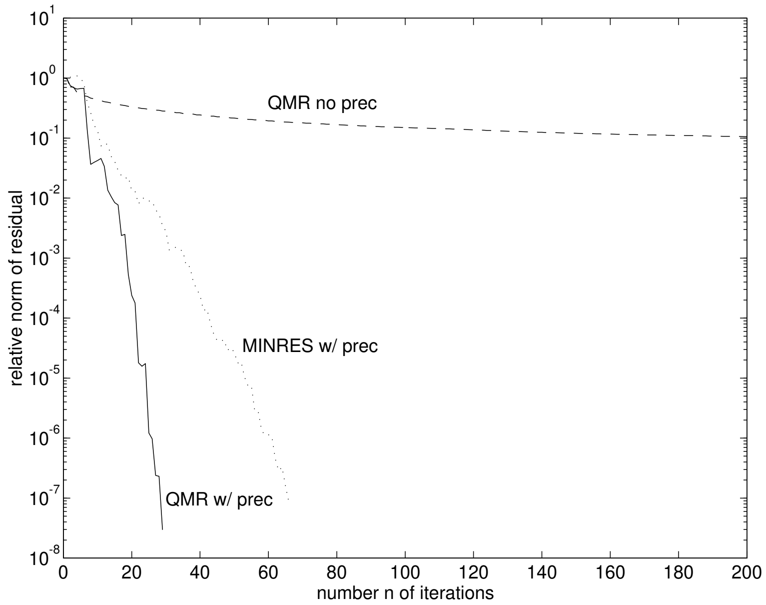

Fig. 3. Convergence curves for KKT system.

The second example is a linear system (21) with a nonsingular symmetric indefinite coefficient matrix A of the form

$$K = \begin{bmatrix} B & C^T \\ C & 0 \end{bmatrix}, \quad B = B^T \in \mathbb{R}^m.$$

This system arose in connection with the Karush-Kuhn-Tucker optimality conditions for an optimization problem, and we will refer to it as a KKT system. The order of K is $N = 1000$, and the upper left block B is a positive-definite 600×600 matrix. Moreover, B is a banded matrix with bandwidth 11, and C is a sparse matrix with an irregular sparsity structure. The matrix K is highly indefinite with 600 positive and 400 negative eigenvalues. For this system we obtained a symmetric indefinite preconditioner by computing a Bunch-Parlett factorization [2] of the form

$$M = LDL^T \tag{40}$$

of an approximation M to K. Here L is a sparse unit lower triangular matrix and D is a nonsingular block diagonal matrix with 1×1 and 2×2 blocks. We ran the simplified QMR-from-BCG Algorithm 5.1 with right preconditioner M, and we compared it to MINRES with a positive-definite right preconditioner $\hat{M}$. Following a proposal made in [24], we used the positive-definite matrix $\hat{M} = L\hat{D}L^T$ that is obtained from (40) by constructing a block diagonal matrix $\hat{D}$ whose eigenvalues are just the absolute values of those of the indefinite matrix D. The resulting relative residual norms for simplified QMR (solid line) and MINRES (dotted line) are shown in Figure 3. In this example, QMR converges more than twice as fast as MINRES.

If no preconditioning is used, then QMR and MINRES are mathematically equivalent. In Figure 3, we also plotted the residual norms (39) for a run of Algorithm 5.1 without preconditioning (dashed line). Note that the method does not converge in a reasonable number of iterations.

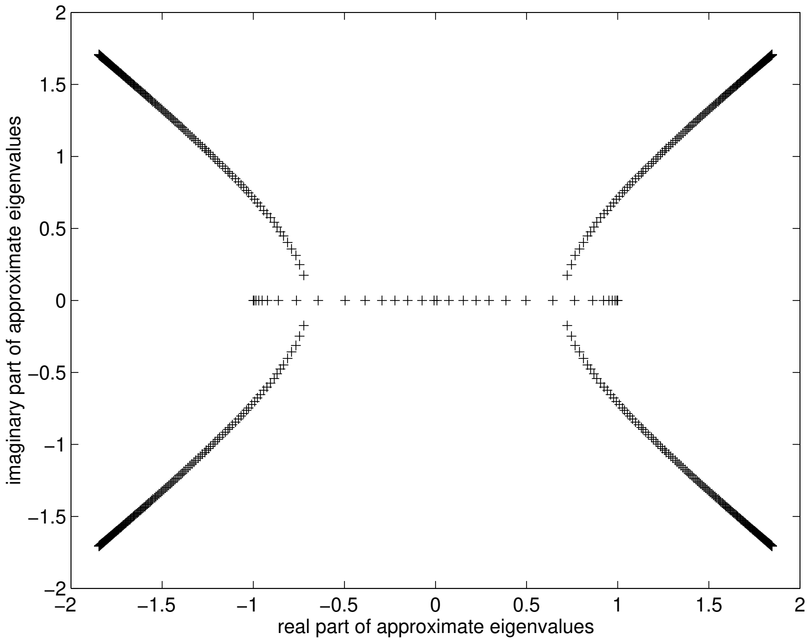

Fig. 4. Approximate eigenvalues for Hamiltonian matrix.

We remark that all Krylov-subspace iterations typically converge very slowly for highly indefinite coefficient matrices.

We also refer the reader to [16] for further results of numerical experiments with linear systems from optimization problems. In [16], Freund and Jarre use the simplified QMR-from-BCG Algorithm 5.1 to solve the symmetric indefinite linear systems that arise in each iteration of a primal-dual interior-point method for linear programs.

Our last example is a Hamiltonian eigenvalue problem, taken from [33]. This is a family of Hamiltonian matrices A of size $N = 2(2v-1)$, $v = 1, 2, \ldots$, that arises in the context of position and velocity control for a string of high-speed vehicles. Moreover, the submatrix F in the partitioning (28) is tridiagonal, and the submatrices F and H in (28) are diagonal. We used the case $v = 2501$ for our test, resulting in a Hamiltonian matrix A of order $N = 10002$. In Figure 4, we show the approximate eigenvalues obtained from the simplified three-term look-ahead Lanczos algorithm after $n = 600$ steps. We remark that these approximate eigenvalues very well match the true spectrum of A.

9. Concluding remarks

Just as is the case of symmetric and Hermitian matrices, the Lanczos process can be simplified for J-symmetric and J-Hermitian matrices. By choosing the left initial Lanczos vector as a J-multiple of the right initial Lanczos vector (respectively the conjugate thereof in the J-Hermitian case), all the left Lanczos vectors are, up to scaling, just J-multiples (respectively conjugate J-multiples) of the right Lanczos vectors. Consequently, the recurrences for generating the left Lanczos vectors can be replaced by matrix-vector multiplications with J. For the resulting simplified Lanczos process to be viable it is crucial that J is such that matrix-vector multiplications with J are inexpensive. This is the case for many important applications.

In this paper, we described software with FORTRAN-77 implementations of the simplified Lanczos process and of the Lanczos-based linear-systems solver QMR for J-symmetric and J-Hermitian matrices. These routines complete our software package QMRPACK that, so far, only contained algorithms for general unsymmetric matrices, as well as for symmetric matrices.

Acknowledgements

The research of Noël M. Nachtigal was supported in part by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy, under contract DE–AC05–840R21400 with the Martin Marietta Energy Systems, Inc.

References

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

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

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

[4] J.K. Cullum and R.A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 1, Theory* (Birkhäuser, Basel, 1985).

[5] J.K. Cullum and R.A. Willoughby, A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices, in: J. Cullum and R.A. Willoughby, eds., *Large Scale Eigenvalue Problems* (North-Holland, Amsterdam, The Netherlands, 1986) 193–240.

[6] J.J. Dongarra and E.H. Grosse, Distribution of mathematical software via electronic mail, *Comm. ACM* 30 (1987) 403–407.

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

[8] P. Feldmann and R.W. Freund, Efficient linear circuit analysis by Padé approximation via the Lanczos process, in: *Proc. EURO-DAC '94 with EURO-VHDL '94* (IEEE Computer Society Press, Los Alamitos, 1994) 170–175.

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

[10] R.W. Freund, A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, *SIAM J. Sci. Comput.* 14 (1993) 470–482.

[11] R.W. Freund, Transpose-free quasi-minimal residual methods for non-Hermitian linear systems, in: G. Golub, A. Greenbaum and M. Luskin, eds., *Recent Advances in Iterative Methods*, The IMA Volumes in Mathematics and its Applications, Vol. 60 (Springer-Verlag, New York, 1994) 69–94.

[12] R.W. Freund, The look-ahead Lanczos process for nonsymmetric matrices and its applications, in: J.D. Brown, M.T. Chu, D.C. Ellison and R.J. Plemmons, eds., *Proceedings of the Cornelius Lanczos International Centenary Conference* (SIAM, Philadelphia, 1994) 33–47.

[13] R.W. Freund, Lanczos-type algorithms for structured non-Hermitian eigenvalue problems, in: J.D. Brown, M.T. Chu, D.C. Ellison and R.J. Plemmons, eds., *Proceedings of the Cornelius Lanczos International Centenary Conference* (SIAM, Philadelphia, 1994) 243–245.

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

[15] 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) 137–158.

[16] R.W. Freund and F. Jarre, A QMR-based interior-point algorithm for solving linear programs, Numerical Analysis Manuscript 94–19, AT&T Bell Laboratories, Murray Hill, NJ, 1994.

[17] R.W. Freund and V. Mehrmann, A symplectic look-ahead Lanczos algorithm for the Hamiltonian eigenvalue problem, Numerical Analysis Manuscript, AT&T Bell Laboratories, Murray Hill, NJ, 1995.

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

[19] R.W. Freund and N.M. Nachtigal, An implementation of the QMR method based on coupled two-term recurrences, *SIAM J. Sci. Comput.* 15 (1994) 313–337.

[20] R.W. Freund and N.M. Nachtigal, QMRPACK: a package of QMR algorithms, Numerical Analysis Manuscript 94–16, AT&T Bell Laboratories, Murray Hill, NJ, 1994.

[21] R.W. Freund and T. Szeto, A transpose-free quasi-minimal residual squared algorithm for non-Hermitian linear systems, in: R. Vichnevetsky, D. Knight and G. Richter, eds., *Advances in Computer Methods for Partial Differential Equations – VII* (IMACS, New Brunswick, 1992) 258–264.

[22] R.W. Freund and H. Zha, Simplifications of the nonsymmetric Lanczos process and a new algorithm for Hermitian indefinite linear systems, Numerical Analysis Manuscript, AT&T Bell Laboratories, Murray Hill, NJ, 1995.

[23] P.E. Gill, W. Murray, S.M. Picken and M.H. Wright, The design and structure of a Fortran program library for optimization, *ACM Trans. Math. Softw.* 5 (1979) 259–283.

[24] P.E. Gill, W. Murray, D.B. Ponceleón and M.A. Saunders, Preconditioners for indefinite systems arising in optimization, *SIAM J. Matrix Anal. Appl.* 13 (1992) 292–311.

[25] I. Gohberg, P. Lancaster and L. Rodman, *Matrices and Indefinite Scalar Products* (Birkhäuser, Basel, 1983).

[26] M.D. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows* (Academic Press, San Diego, 1989).

[27] F.T. Krogh, VODQ/SVDQ/DVDQ—Variable order integrators for the numerical solution of ordinary differential equations, Subroutine Write-Up, Sec. 314, Jet Propulsion Lab., Pasadena, CA, 1969.

[28] R.A. Horn and C.R. Johnson, *Matrix Analysis* (Cambridge University Press, Cambridge, 1985).

[29] T. Huckle, Iterative methods for Toeplitz-like matrices, Manuscript SCCM–94–05, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, CA, 1994.

[30] 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.

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

[32] C. Lanczos, *Applied Analysis* (Prentice-Hall, Englewood Cliffs, NJ, 1956).

[33] A.J. Laub, A Schur method for solving algebraic Riccati equations, *IEEE Trans. Automat. Control* 24 (1979) 913–921.

[34] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem*, Lecture Notes in Control and Information Sciences 163 (Springer-Verlag, Berlin, 1991).

[35] C.C. Paige and M.A. Saunders, Solution of sparse indefinite systems of linear equations, *SIAM J. Numer. Anal.* 12 (1975) 617–629.

[36] B.N. Parlett and H.C. Chen, Use of indefinite pencils for computing damped natural modes, *Linear Algebra Appl.* 140 (1990) 53–88.

[37] B.N. Parlett, D.R. Taylor and Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, *Math. Comp.* 44 (1985) 105–124.

[38] P.A. Roebuck and S.Barnett, A survey of Toeplitz and related matrices, *Internat. J. Systems Sci.* 9 (1978) 921–934.

[39] D.J. Silvester and K. Kechkar, Stabilised bilinear-constant velocity-pressure finite elements for the conjugate gradient solution of the Stokes problem, *Comput. Methods Appl. Mech. Engrg.* 79 (1990) 71–86.

[40] D.R. Taylor, Analysis of the look ahead Lanczos algorithm, Ph.D. thesis, University of California, Berkeley, CA, 1982.

[41] M. Tismenetsky, A decomposition of Toeplitz matrices and optimal circulant preconditioning, *Linear Algebra Appl.* 154/155/156 (1991) 105–121.

[42] L. Zhou and H.F. Walker, Residual smoothing techniques for iterative methods, *SIAM J. Sci. Comput.* 15 (1994) 297–312.