Solution of Shifted Linear Systems by Quasi-Minimal Residual Iterations*

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

Abstract

High-order implicit methods for solving time-dependent partial differential equations and frequency response computations in control theory give rise to shifted systems of linear equations. Such systems have identical right-hand sides, and their coefficient matrices differ from each other only by scalar multiples of the identity matrix. This paper explores the use of two quasi-minimal residual iterations, the QMR and the TFQMR algorithm, for the solution of such shifted linear systems. It is shown that both algorithms can exploit the special structure, and that, for any family of shifted linear systems, the number of matrix-vector products and the number of inner products is the same as for a single linear system. Convergence results for the QMR and TFQMR algorithms are presented.

*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

Many iterative schemes for solving systems of linear equations fall into the category of *Krylov subspace methods*. Algorithms of this type use the coefficient matrix only in the form of matrix-vector products. Consequently, they are particularly well suited for the solution of large systems with sparse or structured dense coefficient matrices for which matrix-vector products can be computed cheaply. The most powerful Krylov subspace method is the classical *conjugate gradient algorithm* (CG) [16] for Hermitian positive definite systems. In recent years, there has been considerable interest in CG-like Krylov subspace methods for non-Hermitian linear systems, and a number of new algorithms were proposed. For a survey of these developments, we refer the reader to [9]. In particular, Krylov subspace methods with iterates characterized by a *quasi-minimal residual* property were developed. The first method of this type, the QMR algorithm, was proposed in [5] for the special case of complex symmetric matrices, and then extended to general non-Hermitian systems by Freund and Nachtigal [12]. The QMR algorithm is directly based on the nonsymmetric Lanczos process, and thus, like the latter, it requires matrix-vector products with both the coefficient matrix and its transpose. However, the products with the transpose only enter in the computation of certain auxiliary vectors that are needed to generate recurrence coefficients for the Lanczos process, while the vectors themselves are not used directly in the QMR algorithm. It is possible to devise quasi-minimal residual methods that do not require such auxiliary vectors and thus do not involve the transpose at all. A method of this type, the *transpose-free* QMR algorithm (TFQMR), was proposed in [7, 8]. We stress that QMR and TFQMR are not mathematically equivalent. In particular, both schemes generate different sets of iterates. The attractive feature of both algorithms, QMR and TFQMR, is that they generate *quasi-optimal* iterates with low work and storage requirements per iteration. In contrast, for non-Hermitian Krylov subspace methods that generate *optimal* iterates, such as the *generalized minimal residual algorithm* (GMRES) [21], work per iteration and storage requirements grow linearly with the iteration number. As a result, it is usually too expensive to run the full versions of these algorithms, and it is necessary to use restarted or truncated versions. For difficult problems, this often leads to very slow convergence.

In this paper, we explore the use of QMR and TFQMR for the solution of families of shifted linear systems of the form

$$(M - \sigma_j I)x^{(j)} = b, \quad j = 1, 2, \ldots, s. \tag{1.1}$$

Here $M \in \mathbb{C}^{N \times N}$ and $b \in \mathbb{C}^N$ are fixed, I is the $N \times N$ identity matrix, and $\sigma_j \in \mathbb{C}$, $j = 1, 2, \ldots, s$, are different shifts. We show that both algorithms, QMR and TFQMR, can exploit the special shift structure, and that, for any family of the type (1.1), the number of matrix-vector products and the number of inner products per iteration is the same as for a single linear system.

We remark that various authors studied the use of other Krylov subspace methods for the solution of shifted systems. Young [25] and Young and Vona [26] considered CG for the case of systems (1.1) with symmetric positive definite coefficient matrices $M - \sigma_j I$. In [4], we proposed and analyzed three different CG-type algorithms for the special class of shifted systems (1.1), where M is Hermitian and the σ_j's are general complex shifts. Finally, Datta and Saad [2] showed that the shift structure can be exploited when GMRES is used to solve general non-Hermitian systems (1.1).

Shifted linear systems (1.1) arise, for example, in the context of high-order implicit methods [1, 22] for solving time-dependent partial differential equations. Here, at each time step, one needs to solve a linear system of the form

$$\psi(M)x = \varphi(M)b, \tag{1.2}$$

where ψ and φ are polynomials with $\deg \varphi \leq \deg \psi =: s$. As pointed out in [24, 14], the solution of (1.2) can be obtained by solving a family of shifted linear systems (1.1). The idea is as follows. Let $\sigma_1, \sigma_2, \ldots, \sigma_s$ be the zeros of ψ, and assume that these zeros are all distinct. Then, expanding the rational function φ/ψ into partial fractions gives the decomposition

$$\frac{\varphi(\lambda)}{\psi(\lambda)} \equiv \mu_0 + \sum_{j=1}^{s} \frac{\mu_j}{\lambda - \sigma_j}, \tag{1.3}$$

where $\mu_0, \mu_1, \ldots, \mu_s$ are suitable scalars. Now assume that none of the poles $\sigma_1, \sigma_2, \ldots, \sigma_s$ in (1.3) coincides with an eigenvalue of M. Then, with (1.3), it follows that the solution $x := (\psi(M))^{-1}\varphi(M)b$ of (1.2) is given by

$$x = \mu_0 b + \sum_{j=1}^{s} \mu_j x^{(j)}, \quad \text{where} \quad x^{(j)} := (M - \sigma_j I)^{-1} b.$$

Therefore, solving (1.2) is equivalent to solving a family of s shifted systems of the type (1.1). We remark that systems of the form (1.2) (with $\varphi(\lambda) \equiv 1$) also arise in connection with an algorithm proposed in [2] for constructing the Luenberger observer in control theory.

Another application from control theory that leads to shifted systems (1.1) is frequency response computations, see, e.g., [18]. The transfer matrix associated with continuous-time or discrete-time state space models is of the form

$$G(\sigma) \equiv C(\sigma I - M)^{-1}B + D, \quad \sigma \in \mathbb{C}. \tag{1.4}$$

Here $M \in \mathbb{C}^{N\times N}$, $C \in \mathbb{C}^{p\times N}$, $B \in \mathbb{C}^{N\times m}$, and $D \in \mathbb{C}^{p\times m}$. Frequency response computations for continuous-time models, respectively discrete-time models, require the evaluation of $G(\sigma)$ at points $\sigma_j = i\omega_j$, $\omega_j \in \mathbb{R}$, $j = 1, 2, \ldots, s$, on the imaginary axis, respectively at points $\sigma_j = e^{i\theta_j}$, $0 \leq \theta_j < 2\pi$, $j = 1, 2, \ldots, s$, on the unit circle. Obviously, the columns of the corresponding matrices $(\sigma_j I - M)^{-1}B$ in (1.4) can be obtained by solving m families of shifted linear systems (1.1) with right hand-sides $b = b_k$, $k = 1, 2, \ldots, m$. Here the vectors b_k denote the columns of B. We remark that $m = 1$ for the important special case of single-input models.

The remainder of this paper is organized as follows. In Section 2, we briefly review the QMR and TFQMR algorithms. In Section 3, we discuss the use of QMR for shifted systems.

In Section 4, we propose a version of TFQMR for shifted systems. In Section 5, we present error bounds for both algorithms. Finally, in Section 6, we make some concluding remarks.

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, B^T and B^H denote the transpose and conjugate transpose of a matrix B, respectively. The vector norm $\|x\| := \sqrt{x^H x}$ is always the Euclidean norm, and $\|B\| := \max_{\|x\|=1} \|Bx\|$ is the associated matrix norm. For square matrices B, we denote by $\lambda(B)$ the set of eigenvalues of B. We use the notation

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

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

$$\mathcal{P}_n := \{\varphi(\lambda) \equiv \gamma_0 + \gamma_1\lambda + \cdots + \gamma_n\lambda^n \mid \gamma_0, \gamma_1, \ldots, \gamma_n \in \mathbb{C}\}$$

the set of all complex polynomials of degree at most n. We will make frequent use of the fact that

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

Finally, M is always an $N \times N$ matrix, $b \in \mathbb{C}^N$, and $\sigma_1, \sigma_2, \ldots, \sigma_s \in \mathbb{C}$. Moreover, we assume that

$$\sigma_j \notin \lambda(M) \quad \text{for all} \quad j = 1, 2, \ldots, s. \tag{1.5}$$

This guarantees that the coefficient matrices in (1.1) are all nonsingular.

2 The QMR and TFQMR algorithms

In this section, we briefly describe the QMR and TFQMR algorithms for solving single linear systems

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

The QMR algorithm is directly based on a look-ahead variant of the classical nonsymmetric Lanczos process. First, we sketch this look-ahead Lanczos algorithm.

2.1 The look-ahead Lanczos algorithm

Let $A \in \mathbb{C}^{N\times N}$, and let v_1, $w_1 \in \mathbb{C}^N$, v_1, $w_1 \neq 0$, be arbitrary vectors. Starting with v_1 and w_1, the classical nonsymmetric Lanczos algorithm [17] generates two sequences of vectors $v_1, v_2, \ldots, v_n$ and $w_1, w_2, \ldots, w_n$ that span the Krylov subspaces $K_n(v_1, A)$ and $K_n(w_1, A^T)$, respectively. Furthermore, the two sequences are biorthogonal to each other, i.e., $w_j^T v_k = 0$ if $j \neq k$. Unfortunately, *breakdowns*—triggered by division by 0—and *near-breakdowns*—triggered by division by numbers close to 0—can occur in the classical Lanczos algorithm. Parlett, Taylor, and Liu [20] were the first to propose a modified procedure that uses so-called *look-ahead* techniques to avoid breakdowns and near-breakdowns. The QMR algorithm is based on a different implementation of the look-ahead Lanczos method that was recently developed by Freund, Gutknecht, and Nachtigal [10].

Starting with v_1 and w_1, this look-ahead Lanczos algorithm generates two sequences of vectors

$$v_1, v_2, \dots, v_n \quad \text{and} \quad w_1, w_2, \dots, w_n, \quad n = 1, 2, \dots, \tag{2.2}$$

that, like the classical Lanczos process, satisfy

$$\begin{aligned} \operatorname{span}\{v_1, v_2, \dots, v_n\} &= K_n(v_1, A), \\ \operatorname{span}\{w_1, w_2, \dots, w_n\} &= K_n(w_1, A^T). \end{aligned} \tag{2.3}$$

However, the two sequences are now only block biorthogonal, i.e.,

$$(W^{(j)})^T V^{(k)} = 0 \quad \text{for all} \quad j \neq k, \quad j, k = 1, 2, \dots, l(n). \tag{2.4}$$

The matrices $W^{(j)}$ and $V^{(k)}$ in (2.4) are defined by partitioning the Lanczos vectors (2.2) into blocks, according to the look-ahead steps taken:

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

More precisely, the matrices $V^{(k)}$ and $W^{(k)}$ are both of order $N \times h_k$, and their columns are just the Lanczos vectors corresponding to the kth look-ahead step. Here h_k denotes the length of the kth look-ahead step. Moreover, $l := l(n)$ denotes the number of look-ahead steps that were taken in the course of the first n steps of the Lanczos process. We denote by n_k the index of the first Lanczos vectors v_{n_k} and w_{n_k} in the kth blocks $V^{(k)}$ and $W^{(k)}$. In view of (2.4), the vectors v_{n_k} and w_{n_k} are biorthogonal to all previous Lanczos vectors $w_1, w_2, \dots, w_{n_k-1}$ and $v_1, v_2, \dots, v_{n_k-1}$, respectively. Therefore, the first vectors in each block are called *regular*. The remaining vectors in each block are called *inner*. We stress that the algorithm performs mostly standard Lanczos steps, i.e., look-ahead steps of size $h_k = 1$ with blocks $V^{(k)}$ and $W^{(k)}$ that consist of only single Lanczos vectors. True look-ahead steps, i.e., steps of size $h_k > 1$, are only used to avoid breakdowns and near-breakdowns. Typically, except for contrived examples, only few true look-ahead steps occur, and their size is usually small, mostly $h_k = 2$. We note that, if only steps of length $h_k = 1$ are performed, then the look-ahead Lanczos algorithm reduces to the classical algorithm.

We remark that the conditions (2.3) and (2.4) determine the Lanczos vectors only up to scaling. In the sequel, we always assume that the Lanczos vectors (2.2) are scaled to have unit length:

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

The crucial point of the look-ahead Lanczos process is that vectors satisfying (2.3) and (2.4) can be constructed by means of short recurrences, which involve only vectors from the last or the last two blocks. Next, we present a sketch of this look-ahead Lanczos algorithm.

Algorithm 2.1 (Sketch of the look-ahead Lanczos algorithm [10])

0) *Choose* $v_1,\, w_1 \in \mathbb{C}^N$ *with* $\|v_1\| = \|w_1\| = 1$.
Set $V^{(1)} = v_1$, $W^{(1)} = w_1$, $D^{(1)} = (W^{(1)})^T V^{(1)}$, $n_1 = 1$, $l = 1$, $v_0 = w_0 = 0$, $V_0 = W_0 = \emptyset$, $\rho_1 = \xi_1 = 1$.

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

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

2) (Regular step) *Compute*

$$\begin{aligned}\tilde{v}_{n+1} = Av_n &- V^{(l)}(D^{(l)})^{-1}(W^{(l)})^T Av_n \\ &- V^{(l-1)}(D^{(l-1)})^{-1}(W^{(l-1)})^T Av_n,\end{aligned} \tag{2.6}$$

$$\begin{aligned}\tilde{w}_{n+1} = A^T w_n &- W^{(l)}(D^{(l)})^{-T}(V^{(l)})^T A^T w_n \\ &- W^{(l-1)}(D^{(l-1)})^{-T}(V^{(l-1)})^T A^T w_n,\end{aligned}$$

set $n_{l+1} = n + 1$, $l = l + 1$, $V^{(l)} = W^{(l)} = \emptyset$, *and go to* 4).

3) (Inner step) *Compute*

$$\begin{aligned}\tilde{v}_{n+1} = Av_n &- \zeta_n v_n - (\eta_n/\rho_n)\, v_{n-1} \\ &- V^{(l-1)}(D^{(l-1)})^{-1}(W^{(l-1)})^T Av_n,\end{aligned} \tag{2.7}$$

$$\begin{aligned}\tilde{w}_{n+1} = A^T w_n &- \zeta_n w_n - (\eta_n/\xi_n)\, w_{n-1} \\ &- W^{(l-1)}(D^{(l-1)})^{-T}(V^{(l-1)})^T A^T w_n.\end{aligned} \tag{2.8}$$

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

$$\begin{gathered}v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}, \\ V^{(l)} = [\, V^{(l)} \quad v_{n+1}\,], \quad W^{(l)} = [\, W^{(l)} \quad w_{n+1}\,], \quad D^{(l)} = (W^{(l)})^T V^{(l)}.\end{gathered}$$

We refer the reader to [10] for implementation details of Algorithm 2.1, and for a discussion of the criteria that are used for the decision in step 1).

We note that, in (2.7) and (2.8), $\zeta_n \in \mathbb{C}$ and $\eta_n \in \mathbb{C}$, $n = 0, 1, \ldots,$ are arbitrary recurrence coefficients with $\eta_{n_k} = 0$, $k = 1, 2, \ldots$. The choice of these coefficients is not overly important, since Algorithm 2.1 usually performs only few inner steps anyway. We mostly used the algorithm with $\zeta_n := 1$ and, if $n \neq n_k$, $\eta_n := 1$.

Next, we list a few properties of Algorithm 2.1 that will be used later on; for proofs, we refer the reader to [10] and the references given there. Setting

$$V_n := [\, v_1 \quad v_2 \quad \cdots \quad v_n\,] \quad \text{and} \quad V_{n+1} := [\, V_n \quad v_{n+1}\,], \tag{2.9}$$

the recurrences (2.6), respectively (2.7), for the vectors $v_1, \dots, v_n, v_{n+1}$, can be written compactly in matrix form as follows:

$$AV_n = V_{n+1} H_n. \tag{2.10}$$

Here H_n is an $(n+1) \times n$ upper Hessenberg matrix, which is also block tridiagonal with square blocks of order h_k on the diagonal. The subdiagonal elements of H_n are all nonzero, and hence

$$\operatorname{rank} H_n = n. \tag{2.11}$$

Each pair of Lanczos vectors v_n and w_n can be expressed in the form

$$v_n = \gamma_n \varphi_{n-1}(A) v_1 \quad \text{and} \quad w_n = \delta_n \varphi_{n-1}(A^T) w_1, \tag{2.12}$$

where $\varphi_{n-1} \in \mathcal{P}_{n-1}$ is of exact degree $n-1$ with leading coefficient 1, and $\gamma_n > 0$ and $\delta_n > 0$ are positive scalars. We refer to the φ_j's in (2.12) as the *look-ahead Lanczos polynomials*. Finally, we note that, in exact arithmetic, the stopping criterion in step 4) of Algorithm 2.1 will be satisfied after $L\,(\leq N)$ iterations, except in the very special situation of an incurable breakdown. The number L is called the *termination index*. An incurable breakdown occurs if Algorithm 2.1 starts to build infinite blocks $V^{(l)}$ and $W^{(l)}$ such that $(W^{(l)})^T V^{(l)}$ is the infinite zero matrix. Obviously, this is a very special situation, and we stress that incurable breakdowns do not pose a problem in practice. In the case of an incurable breakdown, we set $L := n_l - 1$, where n_l is the index of the last pair of regular vectors v_{n_l} and w_{n_l}.

We now turn to linear systems (2.1). For simplicity, from now on, we assume that the coefficient matrix A of (2.1) is nonsingular. We note that QMR and TFQMR generate well-defined iterates also for the case of singular A. A detailed discussion of QMR and TFQMR for singular systems can be found in [11].

2.2 The QMR method

Let $x_0 \in \mathbb{C}^N$ be any initial guess for the solution of (2.1), let $r_0 := b - Ax_0$ be the associated residual vector, and let $\tilde{r}_0 \in \mathbb{C}^N$ be an arbitrary nonzero vector. We now choose

$$v_1 := r_0 / \|r_0\| \quad \text{and} \quad w_1 := \tilde{r}_0 / \|\tilde{r}_0\| \tag{2.13}$$

as the starting vectors for Algorithm 2.1.

Using the matrices V_n, V_{n+1} and H_n generated by means of Algorithm 2.1, the QMR method [12] produces approximations x_n to the solution of (2.1) that are defined as follows:

$$x_n = x_0 + V_n z_n, \tag{2.14}$$

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

$$\|f_n - H_n z_n\| = \min_{z \in \mathbb{C}^n} \|f_n - H_n z\|. \tag{2.15}$$

The vector f_n is given by

$$f_n := \|r_0\| \cdot [1 \quad 0 \quad \cdots \quad 0]^T \in \mathbb{R}^{n+1}. \tag{2.16}$$

Note that, in view of (2.11), the solution z_n of (2.15) is unique. Furthermore, from (2.14), (2.9), (2.3), and (2.13), we have

$$x_n \in x_0 + K_n(r_0, A), \quad n = 1, 2, \ldots \, . \tag{2.17}$$

Using (2.14), (2.13), (2.16), and (2.10), one readily shows that the residual vector $r_n := b - Ax_n$ associated with x_n satisfies

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

Hence, in view of (2.15), the QMR iterates are characterized by a minimization of the second factor in the representation (2.18) of r_n; this is just the quasi-minimal residual property. Recall from (2.5) that the columns of V_{n+1} are scaled to be unit vectors. Therefore, the choice of z_n guarantees that all components in the representation (2.18) of r_n are treated equally in the quasi-minimization process. Other scaling strategies can also be used by simply defining z_n by a weighted least-squares problem, instead of (2.15); see [12] for details.

In the QMR algorithm, the least-squares problem (2.15) is solved by the standard approach based on a QR decomposition

$$H_n = Q_n^H \begin{bmatrix} R_n \\ 0 \end{bmatrix} \tag{2.19}$$

of H_n. Here Q_n is a unitary $(n+1) \times (n+1)$ matrix, and R_n is a nonsingular upper triangular $n \times n$ matrix. By means of (2.19), the least-squares problem (2.15) can be rewritten in the form

$$\min_{z \, \in \, \mathbb{C}^n} \|f_n - H_n z\| = \min_{z \, \in \, \mathbb{C}^n} \left\| Q_n f_n - \begin{bmatrix} R_n \\ 0 \end{bmatrix} z \right\|,$$

and thus z_n is given by

$$z_n = R_n^{-1} t_n, \quad \text{where} \quad t_n = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix} \in \mathbb{C}^n, \quad \begin{bmatrix} t_n \\ \tilde{\tau}_{n+1} \end{bmatrix} := Q_n f_n. \tag{2.20}$$

Since H_n is upper Hessenberg, the unitary matrix Q_n can be chosen as a product of n Givens rotations. The QR decomposition (2.19) can then be updated easily from step to step. In particular, Q_n is obtained from Q_{n-1} by a simple multiplication with one Givens rotation that modifies the last two rows only. This implies that the vector t_n in (2.20) differs from the previous one, t_{n-1}, only by its additional last element τ_n. Together with (2.14) and (2.20), it follows that consecutive QMR iterates are connected by the update formula

$$x_n = x_{n-1} + \tau_n p_n, \quad \text{where} \quad p_n := V_n R_n^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{2.21}$$

The basic structure of the resulting QMR algorithm is then as follows.

Algorithm 2.2 (Sketch of the QMR algorithm [12])

0) *Choose* $x_0 \in \mathbb{C}^N$.
   Set $r_0 = b - Ax_0$ *and* $v_1 = r_0/\|r_0\|$.
   Choose $\tilde{r}_0 \in \mathbb{C}^N$ *with* $\tilde{r}_0 \neq 0$, *and set* $w_1 = \tilde{r}_0/\|\tilde{r}_0\|$.

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

1) *Perform the nth iteration of the look-ahead Lanczos Algorithm* 2.1.
   This yields matrices V_n, V_{n+1}, *and* H_n *that satisfy* $AV_n = V_{n+1}H_n$.

2) *Update the* QR *factorization* (2.19) *of* H_n *and the vector* t_n *in* (2.20).

3) *Update the vector* p_n *in* (2.21).

4) *Set* $x_n = x_{n-1} + \tau_n p_n$.

5) *If* x_n *has converged*: *stop.*

The update of the vector p_n in step 3) can be implemented with only short recurrences. This is due to the block tridiagonal structure of H_n; see [12] for details. If no true look-ahead steps are taken, then H_n is even a scalar tridiagonal matrix, and the vectors p_n can be updated by simple three-term recurrences of the type

$$p_n = \varrho_n v_n - \epsilon_n p_{n-1} - \theta_n p_{n-2}, \tag{2.22}$$

where ϱ_n, ϵ_n, and θ_n are suitable scalars.

Finally, we remark that the procedure sketched in Algorithm 2.2 is not the only possible implementation of the QMR method. Recently, Freund and Nachtigal [13] devised an implementation that uses coupled two-term recurrences to generate the Lanczos vectors (2.2). The associated recurrence coefficients are just the entries of the block bidiagonal factors L_n and U_n of an LU decomposition $H_n = L_n U_n$ of the block tridiagonal matrix H_n. The QMR iterates are then obtained by solving a least-squares problem with L_n as coefficient matrix, instead of (2.15).

2.3 The TFQMR method

At each iteration step, the QMR method requires one matrix-vector multiplication with A, and one with A^T. However, only the products with A are used to advance the Krylov subspaces $K_n(r_0, A)$ from which, by (2.17), the QMR iterates are obtained. In [7, 8], we proposed a transpose-free quasi-minimal residual iteration, the TFQMR method, that exchanges each product with A^T for an additional product with A. As a result, with the same number of matrix-vector multiplications as QMR, the TFQMR method generates iterates from Krylov subspaces of twice the dimension. In this section, we sketch the TFQMR method.

Instead of one QMR iterate, the TFQMR method produces two iterates

$$x_m \in x_0 + K_m(r_0, A), \quad m = 2n-1,\, 2n, \tag{2.23}$$

in step n. Therefore, in the sequel, we will use $m = 1, 2, \dots,$ as the iteration index in our description of TFQMR. First, based on the look-ahead Lanczos polynomials given by (2.12), we define vectors

$$u_m := \begin{cases} \dfrac{\varphi_{n-1}(A)\varphi_n(A)r_0}{\|\varphi_{n-1}(A)\varphi_n(A)r_0\|}, & \text{if } \ m = 2n, \\ \dfrac{(\varphi_n(A))^2 r_0}{\|(\varphi_n(A))^2 r_0\|}, & \text{if } \ m = 2n+1, \end{cases} \tag{2.24}$$

and we set

$$U_m := [\, u_1 \quad u_2 \quad \cdots \quad u_m \,].$$

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

$$K_m(r_0, A) = \{ U_m z \mid z \in \mathbb{C}^m \}. \tag{2.25}$$

Furthermore, the vectors $u_1, u_2, \dots, u_{m+1}$ satisfy recurrences of the form

$$AU_m = U_{m+1} F_m, \tag{2.26}$$

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

$$\operatorname{rank} F_m = m. \tag{2.27}$$

Finally, note that, from (2.12), $\varphi_0(\lambda) \equiv 1$, and thus, by (2.24), $u_1 = r_0/\|r_0\|$.

Using (2.25), the iterates (2.23) can be parametrized in the form

$$x_m = x_0 + U_m z_m, \quad \text{where} \quad z_m \in \mathbb{C}^m. \tag{2.28}$$

Together with (2.26), it follows that the associated residual vector $r_m := b - Ax_m$ satisfies

$$r_m = U_{m+1}(f_m - F_m z_m), \tag{2.29}$$

where f_m is given by (2.16). In analogy to the definition (2.14)–(2.15) of the nth QMR iterate, the mth TFQMR iterate is given by (2.28) where z_m is chosen so that the Euclidean norm of the second factor in (2.29) is minimal, i.e., $z_m \in \mathbb{C}^m$ is the solution of the least-squares problem

$$\|f_m - F_m z_m\| = \min_{z \in \mathbb{C}^m} \|f_m - F_m z\|. \tag{2.30}$$

Note that, in view of (2.27), z_m is uniquely defined by (2.30).

Recall that the vectors u_m defined in (2.24) involve the polynomials (2.12) associated with the look-ahead Lanczos process. As a result, a robust implementation of TFQMR needs to have look-ahead incorporated in order to avoid breakdowns or near-breakdowns. It is an open question whether TFQMR with look-ahead can be implemented with only short vector recurrences and with only two matrix-vector multiplications by A per nth iteration. However, if no look-ahead is used, then an implementation with these features exists. Indeed, in [8], we derived an implementation of TFQMR without look-ahead that, instead of the vectors u_m, uses the search directions y_m from the *conjugate gradients squared algorithm* (CGS) [23] to update the TFQMR iterates. Like CGS, the resulting TFQMR algorithm without look-ahead can be implemented with only short vector updates and with only two matrix-vector multiplications per nth iteration. This implementation is as follows.

Algorithm 2.3 (TFQMR without look-ahead [8])

0) *Choose* $x_0 \in \mathbb{C}^N$.
Set $\hat{u}_1 = y_1 = b - Ax_0$, $v_0 = Ay_1$, $d_0 = 0$.
Set $\tau_0 = \|\hat{u}_1\|$, $\vartheta_0 = 0$, $\eta_0 = 0$.
Choose $\tilde{r}_0 \in \mathbb{C}^N$ *such that* $\rho_0 = \tilde{r}_0^T \hat{u}_1 \neq 0$.

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

1) *If* $\tilde{r}_0^T v_{n-1} = 0$: *stop. Otherwise, set*

$$\alpha_{n-1} = \rho_{n-1}/(\tilde{r}_0^T v_{n-1}), \quad y_{2n} = y_{2n-1} - \alpha_{n-1} v_{n-1}. \tag{2.31}$$

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

- *Set*

$$\hat{u}_{m+1} = \hat{u}_m - \alpha_{n-1} A y_m, \tag{2.32}$$

$$\begin{aligned}
\vartheta_m &= \|\hat{u}_{m+1}\|/\tau_{m-1}, \quad c_m = 1/\sqrt{1+\vartheta_m^2}, \\
\tau_m &= \tau_{m-1}\vartheta_m c_m, \quad \eta_m = c_m^2 \alpha_{n-1}, \\
d_m &= y_m + (\vartheta_{m-1}^2 \eta_{m-1}/\alpha_{n-1}) d_{m-1}, \\
x_m &= x_{m-1} + \eta_m d_m.
\end{aligned}$$

- *If* x_m *has converged: stop.*

3) *Set* $\rho_n = \tilde{r}_0^T \hat{u}_{2n+1}$.
If $\rho_n = 0$: *stop. Otherwise, set*

$$\begin{aligned}
\beta_n &= \rho_n/\rho_{n-1}, \quad y_{2n+1} = \hat{u}_{2n+1} + \beta_n y_{2n}, \\
v_n &= A y_{2n+1} + \beta_n (A y_{2n} + \beta_n v_{n-1}).
\end{aligned} \tag{2.33}$$

We stress that the checks in step 1) and step 3) are necessary in order to avoid breakdowns in the form of division by 0. In general, it cannot be excluded that Algorithm 2.3 terminates in step 1) or 3), before x_m has converged. In an implementation of TFQMR with look-ahead, such premature termination of the algorithm would be avoided by using look-ahead techniques to skip over those steps in which division by 0 or by a number close to 0 would occur.

For later use, we list some properties of Algorithm 2.3; for proofs, we refer the reader to [8, 23]. The vectors $\hat{u}_m$ in (2.32) are rescaled versions of (2.24). More precisely,

$$\hat{u}_m := \begin{cases} \hat{\varphi}_{n-1}(A)\hat{\varphi}_n(A)r_0, & \text{if } \ m = 2n, \\ (\hat{\varphi}_n(A))^2 r_0, & \text{if } \ m = 2n+1, \end{cases} \tag{2.34}$$

where the $\hat{\varphi}_n$'s are rescaled versions of the Lanczos polynomials:

$$\hat{\varphi}_n(\lambda) \equiv \frac{\varphi_n(\lambda)}{\varphi_n(0)}. \tag{2.35}$$

We remark that the check in step 1) of Algorithm 2.3 guarantees that $\varphi_n(0) \neq 0$ in (2.35). The polynomials (2.35) can be generated by the following coupled two-term recurrences:

$$\begin{aligned} \psi_n(\lambda) &\equiv \hat{\varphi}_n(\lambda) + \beta_n \psi_{n-1}(\lambda), \\ \hat{\varphi}_{n+1}(\lambda) &\equiv \hat{\varphi}_n(\lambda) - \alpha_n \lambda \psi_n(\lambda), \end{aligned} \tag{2.36}$$

where $n = 0, 1, \dots$. For $n = 0$, we set $\hat{\varphi}_0(\lambda) \equiv 1$, $\beta_0 = 0$, and $\psi_{-1}(\lambda) \equiv 0$ in (2.36).

3 The QMR algorithm for shifted matrices

We now return to shifted linear systems of the form (1.1). From now on, we assume that M and the shifts $\sigma_1, \sigma_2, \dots, \sigma_s$ satisfy (1.5), so that the coefficient matrices $M - \sigma_j I$ in (1.1) are all nonsingular.

In the sequel, we always assume that $j \in \{1, 2, \dots, s\}$. We use the following convention. Quantities generated by Algorithm 2.1 or 2.2 applied to the shifted matrices $A = M - \sigma_j I$ are marked by superscripts $^{(j)}$, while quantities without superscripts denote scalars and vectors generated by the look-ahead Lanczos Algorithm 2.1 applied to the unshifted matrix $A = M$. For instance, $v_n^{(j)}$ and $w_n^{(j)}$ are the Lanczos vectors produced by Algorithm 2.1 applied to $M - \sigma_j I$, and v_n and w_n are the Lanczos vectors generated by Algorithm 2.1 applied to M. Finally, we always assume that all algorithms are started with the same initial vectors, independent of j. Hence, for Algorithm 2.1, we set

$$v_1^{(j)} = v_1 \quad \text{and} \quad w_1^{(j)} = w_1, \quad j = 1, 2, \dots, s. \tag{3.1}$$

For the QMR algorithm, we set

$$x_0^{(j)} = 0, \quad r_0^{(j)} = r_0 = b, \quad \text{and} \quad \tilde{r}_0^{(j)} = \tilde{r}_0, \quad j = 1, 2, \dots, s. \tag{3.2}$$

First, we compare the quantities generated by the look-ahead Lanczos Algorithm 2.1, when applied to $A = M$ and $A = M - \sigma_j I$. Obviously, in view of (3.1), we have, for all $n = 1, 2, \ldots$,

$$\begin{aligned} K_n(v_1^{(j)}, M - \sigma_j I) &= K_n(v_1, M), \\ K_n(w_1^{(j)}, (M - \sigma_j I)^T) &= K_n(w_1, M^T), \end{aligned} \tag{3.3}$$

and hence the Krylov subspaces (2.3) underlying the Lanczos process do not depend on the shifts σ_j. Furthermore, it is easily verified that the look-ahead Lanczos Algorithm 2.1 even generates identical basis vectors for the Krylov subspaces (3.3), provided the recurrence coefficients ζ_n in (2.7) and (2.8) are shifted accordingly. More precisely, we have the following result.

Lemma 3.1 *Let v_n and w_n, respectively $v_n^{(j)}$ and $w_n^{(j)}$, be the Lanczos vectors generated by Algorithm* 2.1 *applied to $A = M$, respectively $A = M - \sigma_j I$, with recurrence coefficients ζ_n and η_n, respectively $\zeta_n^{(j)} = \zeta_n - \sigma_j$ and $\eta_n^{(j)} = \eta_n$. Then the termination indices L and $L^{(j)}$ are the same, independent of the shifts σ_j, and*

$$v_n^{(j)} = v_n \quad \textit{and} \quad w_n^{(j)} = w_n, \quad n = 1, 2, \ldots, L. \tag{3.4}$$

Furthermore, for $n = 1, 2, \ldots, L$,

$$(M - \sigma_j I) V_n = V_{n+1} H_n^{(j)}, \quad \textit{where} \quad H_n^{(j)} := H_n - \sigma_j \begin{bmatrix} I_n \\ 0 \end{bmatrix}. \tag{3.5}$$

In (3.5), *V_n and V_{n+1} are the matrices defined in* (2.9), *H_n is the upper Hessenberg matrix given by* (2.10) (*with $A = M$*), *and I_n denotes the $n \times n$ identity matrix.*

We remark that, by means of (2.12), the relations (3.4) can be rewritten in terms of the look-ahead Lanczos polynomials, and we obtain

$$\varphi_n^{(j)}(\lambda - \sigma_j) \equiv \varphi_n(\lambda), \quad n = 0, 1, \ldots . \tag{3.6}$$

We now use the QMR Algorithm 2.2, with starting vectors (3.2), to solve each of the s linear systems (1.1). In view of Lemma 3.1, all s runs can be based on only one run of the look-ahead Lanczos Algorithm 2.1 (applied to M). A sketch of the resulting QMR process for solving a family of s shifted linear systems (1.1) is then as follows.

Algorithm 3.2 (QMR algorithm for solving s shifted linear systems (1.1))

0) *For $j = 1, 2, \ldots, s$, set $x_0^{(j)} = 0$ and $r_0^{(j)} = b$.*
   Set $v_1 = b/\|b\|$.
   Choose $\tilde{r}_0 \in \mathbb{C}^N$ with $\tilde{r}_0 \neq 0$, and set $w_1 = \tilde{r}_0/\|\tilde{r}_0\|$.

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

1) *Perform the nth iteration of the look-ahead Lanczos Algorithm* 2.1 *applied to* M*. This yields matrices* V_n*,* V_{n+1}*, and* H_n *that satisfy* $MV_n = V_{n+1}H_n$*.*

2) *For all* $j = 1, 2, \ldots, s$ *for which* $x_n^{(j)}$ *has not converged yet:*

- *Update the* QR *factorization*
$$H_n^{(j)} = (Q_n^{(j)})^H \begin{bmatrix} R_n^{(j)} \\ 0 \end{bmatrix}$$
of the upper Hessenberg matrix
$$H_n^{(j)} := H_n - \sigma_j \begin{bmatrix} I_n \\ 0 \end{bmatrix}.$$

- *Update the vector*
$$\begin{bmatrix} \tau_1^{(j)} \\ \vdots \\ \tau_n^{(j)} \\ \tilde{\tau}_{n+1}^{(j)} \end{bmatrix} = Q_n^{(j)} \begin{bmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

- *Update the vector*
$$p_n^{(j)} = V_n (R_n^{(j)})^{-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

- *Compute*
$$x_n^{(j)} = x_{n-1}^{(j)} + \tau_n^{(j)} p_n^{(j)}.$$

3) *If all* $x_n^{(j)}$ *have converged: stop.*

In Algorithm 3.2, matrix-vector products with M and M^T and inner products are computed only in step 1), while the vector updates in step 2) involve only SAXPYs. Consequently, for Algorithm 3.2 for solving s shifted linear systems, the number of matrix-vector products and the number of inner products per iteration is the same as for the QMR Algorithm 2.2 for solving a single linear system. Furthermore, we remark that, in view of (2.22), the vectors $p_n^{(j)}$ can be updated by simple three-term recurrences of the type
$$p_n^{(j)} = \varrho_n^{(j)} v_n - \epsilon_n^{(j)} p_{n-1}^{(j)} - \theta_n^{(j)} p_{n-2}^{(j)},$$
provided that no true look-ahead steps are taken.

Finally, we note that, for the applications described in Section 1, the matrix M and the right-hand side b are typically real, and only the shifts σ_j in (1.1) are in general complex. Obviously, in this case, the Lanczos vectors generated within Algorithm 2.1 are all real, as long as one chooses $w_1 \in \mathbb{R}^N$ and, in (2.7) and (2.8), real recurrence coefficients ζ_n and η_n. Therefore, even in the case of complex shifts, all quantities step 1) of Algorithm 3.2 are real. In particular, all matrix-vector products and all inner products involve only real vectors.

4 The TFQMR algorithm for shifted matrices

In this section, we derive a version of the TFQMR method for solving shifted linear systems. For simplicity, we only consider TFQMR without look-ahead, as implemented in Algorithm 2.3.

In the following, we always assume that $j \in \{1, 2, \ldots, s\}$. We continue to use the convention introduced in Section 3, and we use superscripts $^{(j)}$ to mark quantities associated with the TFQMR Algorithm 2.3 applied to the jth linear system $(M - \sigma_j I)x^{(j)} = b$ in the family (1.1). Vectors and scalars without superscripts denote quantities generated by the TFQMR Algorithm 2.3 applied to the unshifted matrix $A = M$, and φ_n, $\hat{\varphi}_n$, and ψ_n are the corresponding polynomials in (2.35) and (2.36). Finally, we assume that TFQMR is started with the initial vectors (3.2).

First, we note that, in view of (3.6), the normalized Lanczos polynomials (2.35), $\hat{\varphi}_n$ and $\hat{\varphi}_n^{(j)}$, satisfy

$$\hat{\varphi}_n(\lambda) \equiv \frac{\varphi_n(\sigma_j)}{\varphi_n(0)} \cdot \frac{\varphi_n^{(j)}(\lambda - \sigma_j)}{\varphi_n^{(j)}(0)} \equiv \gamma_n^{(j)} \hat{\varphi}_n^{(j)}(\lambda - \sigma_j), \tag{4.1}$$

$$\text{where} \quad \gamma_n^{(j)} := \hat{\varphi}_n(\sigma_j) = \frac{\varphi_n(\sigma_j)}{\varphi_n(0)}. \tag{4.2}$$

In addition to (4.2), we define

$$\delta_n^{(j)} := \psi_n(\sigma_j). \tag{4.3}$$

By (2.36) (with $\lambda = \sigma_j$), the quantities (4.2) and (4.3) can be updated as follows:

$$\begin{aligned} \delta_n^{(j)} &= \gamma_n^{(j)} + \beta_n \delta_{n-1}^{(j)}, \\ \gamma_{n+1}^{(j)} &= \gamma_n^{(j)} - \alpha_n \sigma_j \delta_n^{(j)}, \end{aligned} \tag{4.4}$$

where, for $n = 0$, we set $\gamma_0^{(j)} = 1$, $\beta_0 = 0$, and $\delta_{-1}^{(j)} = 0$. We will also need the scalars

$$\zeta_n^{(j)} := \hat{\varphi}_n^{(j)}(-\sigma_j) \quad \text{and} \quad \xi_n^{(j)} := \psi_n^{(j)}(-\sigma_j). \tag{4.5}$$

Setting $\lambda = 0$ in (4.1) and since, by (2.35), $\hat{\varphi}_n(0) = 1$, we have

$$\zeta_n^{(j)} = \frac{\hat{\varphi}_n(0)}{\hat{\varphi}_n(\sigma_j)} = \frac{1}{\gamma_n^{(j)}}. \tag{4.6}$$

Note that, in view of (2.36), the quantities (4.5) satisfy the relations

$$\xi_n^{(j)} = \zeta_n^{(j)} + \beta_n^{(j)} \xi_{n-1}^{(j)}, \tag{4.7}$$

$$\zeta_{n+1}^{(j)} = \zeta_n^{(j)} + \alpha_n^{(j)} \sigma_j \xi_n^{(j)}, \tag{4.8}$$

where, for $n = 0$, we set $\xi_{-1}^{(j)} = 0$ and $\beta_0^{(j)} = 0$ in (4.7).

After these preliminaries, we now begin our derivation of the TFQMR algorithm for shifted linear system. We will show that the TFQMR iterates $x_m^{(j)}$ for all s shifted systems (1.1) can be computed by means of simple vector updates from the vectors y_m, $\hat{u}_m$, and v_n generated by the TFQMR Algorithm 2.3 applied to the unshifted matrix $A = M$. Consequently, the resulting TFQMR method for s shifted systems requires per nth iteration only two inner products of vectors in $\mathbb{C}^N$, one norm computation $\|u\|$ of a vector $u \in \mathbb{C}^N$, and two matrix-vector multiplications with M; this is the same as for the TFQMR Algorithm 2.3 applied to a single system.

First, note that, from (4.1) and (4.6), we have

$$\hat{\varphi}_n^{(j)}(M - \sigma_j I) = \zeta_n^{(j)} \hat{\varphi}_n(M) \quad \text{for all} \quad n.$$

Together with (2.34) and (3.2), it follows that

$$\begin{aligned} \hat{u}_{2n}^{(j)} &= \zeta_{n-1}^{(j)} \zeta_n^{(j)} \hat{u}_{2n}, \\ \hat{u}_{2n+1}^{(j)} &= (\zeta_n^{(j)})^2 \hat{u}_{2n+1}. \end{aligned} \tag{4.9}$$

From the second relation in (4.9), we get

$$\beta_n^{(j)} = \frac{\tilde{r}_0^T \hat{u}_{2n+1}^{(j)}}{\tilde{r}_0^T \hat{u}_{2n-1}^{(j)}} = \left(\frac{\zeta_n^{(j)}}{\zeta_{n-1}^{(j)}} \right)^2 \beta_n. \tag{4.10}$$

By rewriting (4.8), we obtain

$$\alpha_n^{(j)} = \frac{\zeta_{n+1}^{(j)} - \zeta_n^{(j)}}{\sigma_j \xi_n^{(j)}}. \tag{4.11}$$

Note that the $\zeta_n^{(j)}$'s and $\xi_n^{(j)}$'s in (4.10) and (4.11) can be updated by means of the formulas (4.4), (4.7), and (4.6), which only involve the scalars α_n and β_n from the unshifted TFQMR algorithm. Consequently, the corresponding quantities $\alpha_n^{(j)}$ and $\beta_n^{(j)}$ for the shifted systems can be obtained by means of (4.11) and (4.10), respectively, without computing extra inner products.

Next, we consider the construction of the vectors $y_m^{(j)}$ that are needed for updating the search directions $d_m^{(j)}$ in step 2) of Algorithm 2.3. If we would generate all these vectors directly as in TFQMR for single systems, then this would require the computation of the matrix-vector products $(M - \sigma_j I) y_m^{(j)}$, which appear in the updates (2.32) and (2.33). We now show that this is actually not necessary. The key observation is that, from (2.32) (with $A = M - \sigma_j I$), we have

$$(M - \sigma_j I) y_m^{(j)} = \frac{1}{\alpha_{n-1}^{(j)}} \left(\hat{u}_m^{(j)} - \hat{u}_{m+1}^{(j)} \right), \quad m = 2n-1,\, 2n,\; n = 1, 2, \dots\ . \tag{4.12}$$

Since, by (4.9), the $\hat{u}_m^{(j)}$'s are just rescaled versions of the $\hat{u}_m$'s from a TFQMR run for the unshifted matrix $A = M$, we can obtain the quantities $(M - \sigma_j I) y_m^{(j)}$ from (4.12), without computing additional matrix-vector products. Indeed, setting

$$\chi_n^{(j)} := \beta_n^{(j)} \frac{\alpha_n^{(j)}}{\alpha_{n-1}^{(j)}} \quad \text{and} \quad t_n^{(j)} := -\alpha_n^{(j)} v_n^{(j)},$$

and using (2.31), (2.33), (4.12), and (4.9), one readily verifies that the vectors $y_{2n-1}^{(j)}$ and $y_{2n}^{(j)}$ can be updated as follows:

$$\begin{aligned} y_{2n-1}^{(j)} &= (\zeta_{n-1}^{(j)})^2 \hat{u}_{2n-1} + \beta_{n-1}^{(j)} y_{2n-2}^{(j)}, \\ t_{n-1}^{(j)} &= \zeta_{n-1}^{(j)} \zeta_n^{(j)} \hat{u}_{2n} + (\chi_{n-1}^{(j)} - 1)(\zeta_{n-1}^{(j)})^2 \hat{u}_{2n-1} \\ &\qquad - \chi_{n-1}^{(j)} (\zeta_{n-2}^{(j)} \zeta_{n-1}^{(j)} \hat{u}_{2n-2} - \beta_{n-1}^{(j)} t_{n-2}^{(j)}), \\ y_{2n}^{(j)} &= y_{2n-1}^{(j)} + t_{n-1}^{(j)}. \end{aligned}$$

Finally, the resulting TFQMR algorithm for solving s shifted linear systems (1.1) can be summarized as follows.

Algorithm 4.1 (TFQMR without look-ahead for shifted linear systems (1.1))

0) *Set* $\hat{u}_1 = y_1 = b$, $v_0 = My_1$, $\hat{u}_0 = 0$.
   Choose $\tilde{r}_0 \in \mathbb{C}^N$ *such that* $\rho_0 = \tilde{r}_0^T \hat{u}_1 \neq 0$.
   Set $\beta_0 = 0$.

1) *For* $j = 1, 2, \ldots, s$, *do*:

   - *Set* $x_0^{(j)} = 0$.
   - *Set* $\tau_0^{(j)} = \|\hat{u}_1\|$, $\vartheta_0^{(j)} = 0$, $\eta_0^{(j)} = 0$.
   - *Set* $d_0^{(j)} = y_0^{(j)} = t_{-1}^{(j)} = 0$.
   - *Set* $\gamma_0^{(j)} = \zeta_0^{(j)} = \zeta_{-1}^{(j)} = \alpha_{-1}^{(j)} = 1$, $\delta_{-1}^{(j)} = \xi_{-1}^{(j)} = 0$.

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

2) *If* $\tilde{r}_0^T v_{n-1} = 0$: *stop. Otherwise, set*

$$\alpha_{n-1} = \rho_{n-1}/(\tilde{r}_0^T v_{n-1}), \quad y_{2n} = y_{2n-1} - \alpha_{n-1} v_{n-1}.$$

3) *For* $m = 2n-1,\, 2n$, *set*

$$\hat{u}_{m+1} = \hat{u}_m - \alpha_{n-1} M y_m.$$

4) *For all* $j = 1, 2, \ldots, s$ *for which* $x_m^{(j)}$ *has not converged yet*:

- *Set*

$$\begin{aligned}
\delta_{n-1}^{(j)} &= \gamma_{n-1}^{(j)} + \beta_{n-1}\delta_{n-2}^{(j)}, \\
\gamma_n^{(j)} &= \gamma_{n-1}^{(j)} - \alpha_{n-1}\sigma_j\delta_{n-1}^{(j)}, \quad \zeta_n^{(j)} = 1/\gamma_n^{(j)}, \\
\beta_{n-1}^{(j)} &= (\zeta_{n-1}^{(j)}/\zeta_{n-2}^{(j)})^2\beta_{n-1}, \\
\xi_{n-1}^{(j)} &= \zeta_{n-1}^{(j)} + \beta_{n-1}^{(j)}\xi_{n-2}^{(j)}, \\
\alpha_{n-1}^{(j)} &= (\zeta_n^{(j)} - \zeta_{n-1}^{(j)})/(\sigma_j\xi_{n-1}^{(j)}), \\
\chi_{n-1}^{(j)} &= \beta_{n-1}^{(j)}\alpha_{n-1}^{(j)}/\alpha_{n-2}^{(j)}.
\end{aligned}$$

- *Set*

$$\begin{aligned}
y_{2n-1}^{(j)} &= (\zeta_{n-1}^{(j)})^2\hat{u}_{2n-1} + \beta_{n-1}^{(j)}y_{2n-2}^{(j)}, \\
t_{n-1}^{(j)} &= \zeta_{n-1}^{(j)}\zeta_n^{(j)}\hat{u}_{2n} + (\chi_{n-1}^{(j)} - 1)(\zeta_{n-1}^{(j)})^2\hat{u}_{2n-1} \\
&\qquad -\chi_{n-1}^{(j)}(\zeta_{n-2}^{(j)}\zeta_{n-1}^{(j)}\hat{u}_{2n-2} - \beta_{n-1}^{(j)}t_{n-2}^{(j)}), \\
y_{2n}^{(j)} &= y_{2n-1}^{(j)} + t_{n-1}^{(j)}.
\end{aligned}$$

- *For $m = 2n-1,\, 2n$, set*

$$\begin{aligned}
\vartheta_m^{(j)} &= \frac{\|\hat{u}_{m+1}\|}{\tau_{m-1}^{(j)}}\,\zeta_n^{(j)} \cdot \begin{cases} \zeta_{n-1}^{(j)}, & \text{if } m = 2n-1, \\ \zeta_n^{(j)}, & \text{if } m = 2n, \end{cases} \\
c_m^{(j)} &= (1 + (\vartheta_m^{(j)})^2)^{-1/2}, \quad \tau_m^{(j)} = \tau_{m-1}^{(j)}\vartheta_m^{(j)}c_m^{(j)}, \\
d_m^{(j)} &= y_m^{(j)} + \left((\vartheta_{m-1}^{(j)})^2\eta_{m-1}^{(j)}/\alpha_{n-1}^{(j)}\right) d_{m-1}^{(j)}, \\
\eta_m^{(j)} &= (c_m^{(j)})^2\alpha_{n-1}^{(j)}, \quad x_m^{(j)} = x_{m-1}^{(j)} + \eta_m^{(j)}d_m^{(j)}.
\end{aligned}$$

5) *If all $x_m^{(j)}$ have converged: stop.*

6) *Set $\rho_n = \tilde{r}_0^T\hat{u}_{2n+1}$. If $\rho_n = 0$: stop. Otherwise, set*

$$\begin{aligned}
\beta_n &= \rho_n/\rho_{n-1}, \quad y_{2n+1} = \hat{u}_{2n+1} + \beta_n y_{2n}, \\
v_n &= My_{2n+1} + \beta_n(My_{2n} + \beta_n v_{n-1}).
\end{aligned}$$

Like the QMR Algorithm 3.2, complex quantities can be avoided in the key steps of TFQMR Algorithm 4.1 when applied to shifted linear systems (1.1) with real M and b, but complex shifts σ_j. Indeed, if $\tilde{r}_0 \in \mathbb{R}^N$, then all scalars and vectors that are updated in steps 2), 3), and 6) remain real. In particular, all matrix-vector products and inner products involve only real vectors.

5 Convergence results

The quasi-minimal residual property of the QMR and TFQMR iterates is strong enough to deduce from it convergence results for both methods. We refer the reader to [12, 6] where convergence results are derived for QMR and TFQMR for solving single linear systems. In this section, we state these results for the QMR and TFQMR algorithms for shifted linear systems (1.1). For simplicity, these results are presented under some additional weak assumptions. We stress that these assumptions can be removed, and we remark that formulations of the convergence results for the general case can be found [6].

First, we state the convergence result for the QMR Algorithm 3.2. We will need some further notation. For any $(n+1) \times n$ matrix H_n, we denote by

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

the matrix obtained from H_n by deleting its last row. Let L be the termination index of the look-ahead Lanczos Algorithm 2.1 (applied to $A = M$), as defined in Section 2.1. Let $\lambda_1, \lambda_2, \ldots, \lambda_L$ be the zeros of the last look-ahead Lanczos polynomial φ_L, and let $\tilde{H}_L$ be the last $L \times L$ Lanczos matrix defined by (2.10) (with $A = M$) and (5.1). It is well known (see, e.g., [10] and the references given there) that

$$\{\lambda_1, \lambda_2, \ldots, \lambda_L\} = \lambda(\tilde{H}_L) \subseteq \lambda(M). \tag{5.2}$$

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

$$D_H := \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_L). \tag{5.3}$$

Recall from Lemma 3.1 that L is also the termination index for all shifted matrices $M - \sigma_j I$, $j = 1, 2, \ldots, s$. Moreover, the last Lanczos matrices $\tilde{H}_L^{(j)}$ associated with the shifted systems are given by

$$\tilde{H}_L^{(j)} = H_L - \sigma_j I_L \quad \text{for all} \quad j = 1, 2, \ldots, s. \tag{5.4}$$

From (5.4) and (5.2), we have

$$\{\lambda_1 - \sigma_j, \lambda_2 - \sigma_j, \ldots, \lambda_L - \sigma_j\} = \lambda(\tilde{H}_L^{(j)}). \tag{5.5}$$

Moreover, by (5.4) and (5.3), there exists a nonsingular matrix $X \in \mathbb{C}^{L \times L}$ such that

$$X^{-1} \tilde{H}_L^{(j)} X = D_H - \sigma_j I_L \quad \text{for all} \quad j = 1, 2, \ldots, s. \tag{5.6}$$

We stress that X does not depend on j.

By applying the QMR convergence result in [6, Theorem 5] to each of the single systems $(M - \sigma_j I) x^{(j)} = b$, and by using (5.5) and (5.6), we obtain the following theorem for the QMR Algorithm 3.2 for shifted systems.

Theorem 5.1 *Let L be the termination index of the look-ahead Lanczos Algorithm* 2.1 (*applied to $A = M$*), *and assume that the zeros $\lambda_1, \lambda_2, \ldots, \lambda_L$ of φ_L are all distinct. Let D_H be the diagonal matrix defined in* (5.3). *Then, for $n = 1, 2, \ldots, L-1$ and all $j = 1, 2, \ldots, s$, the residual vectors $r_n^{(j)} := b - (M - \sigma_j I)x_n^{(j)}$ of the* QMR *Algorithm* 3.2 *for shifted linear systems* (1.1) *satisfy*

$$\frac{\|r_n^{(j)}\|}{\|b\|} \leq c_H \ \sqrt{n+1} \min_{\varphi \in \mathcal{P}_n \colon \varphi(0)=1} \ \max_{\lambda \in \lambda(M)} |\varphi(\lambda - \sigma_j)|,$$
$$\textit{where} \quad c_H := \min_{X \colon X^{-1}\tilde{H}_L X = D_H} \|X\| \cdot \|X^{-1}\|.$$

Next, we turn to the TFQMR method for shifted linear systems. It can be shown that if look-ahead is used, then TFQMR terminates at iteration step $m = 2L$. Let $\tilde{F}_{2L}$ be the corresponding last TFQMR matrix as defined by (2.26) and (5.1). From the results in [6], it follows that

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

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

$$\begin{aligned} J_F &:= \operatorname{diag}(J(\lambda_1), J(\lambda_2), \ldots, J(\lambda_L)), \\ &\text{where} \quad J(\lambda_k) := \begin{bmatrix} \lambda_k & 1 \\ 0 & \lambda_k \end{bmatrix} \quad \text{for all} \quad k. \end{aligned} \tag{5.7}$$

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

Theorem 5.2 *Let L be the termination index of the look-ahead Lanczos Algorithm* 2.1 (*applied to $A = M$*), *and assume that the zeros $\lambda_1, \lambda_2, \ldots, \lambda_L$ of φ_L are all distinct. Let J_F be the Jordan matrix defined in* (5.7). *Then, for $m = 1, 2, \ldots, 2L-1$ and all $j = 1, 2, \ldots, s$, the residual vectors $r_m^{(j)} := b - (M - \sigma_j I)x_m^{(j)}$ of the* TFQMR *method with look-ahead for shifted linear systems* (1.1) *satisfy*

$$\frac{\|r_m^{(j)}\|}{\|b\|} \leq c_F \ \sqrt{m+1} \min_{\varphi \in \mathcal{P}_m \colon \varphi(0)=1} \ \max_{\lambda \in \lambda(M)} (|\varphi(\lambda - \sigma_j)| + |\varphi'(\lambda - \sigma_j)|),$$
$$\textit{where} \quad c_F := \min_{X \colon X^{-1}\tilde{F}_{2L} X = J_F} \|X\| \cdot \|X^{-1}\|.$$

6 Concluding remarks

Families of linear systems with identical right-hand sides and shifted coefficient matrices arise in important applications. In this paper, we have derived special versions of two quasi-minimal residual iterations, the QMR and the TFQMR method, for solving such shifted linear systems. Both proposed algorithms take advantage of the special shift structure, and for any family of shifted systems, the number of matrix-vector products and the number of inner product is the same as for a single linear system.

Iterative methods for solving single linear systems are usually combined with a preconditioner to obtain faster convergence. Unfortunately, standard preconditioning techniques, such as incomplete factorization [19], destroy the special structure when they are applied to shifted linear systems. The only technique we are aware of that allows to preserve the shift structure is polynomial preconditioning. In [4], we studied polynomial preconditioning for the special class of linear systems (1.1) with Hermitian M and complex shifts. It turns out that, in this case, the optimal polynomial preconditioner is given by a suitable Chebyshev iteration [15] applied to the unshifted matrix M. Finally, we remark that Eiermann, Li, and Varga [3] developed a theory for polynomial preconditioning for asymptotically optimal Krylov subspace iterations for general non-Hermitian linear systems. In particular, their theory can also be used to analyze polynomial preconditioners for shifted linear systems.

Acknowledgements

I would like to thank Stratis Gallopoulos for pointing out some important references.

References

[1] J.C. Cavendish, W.E. Culham, and R.S. Varga, *A comparison of Crank-Nicolson and Chebyshev rational methods for numerically solving linear parabolic equations*, J. Comput. Phys. **10** (1972), 354–368.

[2] B.N. Datta and Y. Saad, *Arnoldi methods for large Sylvester-like observer matrix equations, and an associated algorithm for partial spectrum assignment*, Linear Algebra Appl. **154–156** (1991), 225–244.

[3] M. Eiermann, X. Li, and R.S. Varga, *On hybrid semiiterative methods*, SIAM J. Numer. Anal. **26** (1989), 152–168.

[4] R.W. Freund, *On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices*, Numer. Math. **57** (1990), 285–312.

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

[6] R.W. Freund, *Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations*, in: Numerical Methods of Approximation Theory, Vol. 9 (D. Braess and L.L. Schumaker, eds.), Birkäuser, Basel, 1992, pp. 77–95.

[7] R.W. Freund, *Transpose-free quasi-minimal residual methods for non-Hermitian linear systems*, AT&T Numerical Analysis Manuscript 92–07, Bell Laboratories, Murray Hill, New Jersey, 1992.

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

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

[10] 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.

[11] 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.

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

[13] 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, 1992.

[14] E. Gallopoulos and Y. Saad, *Efficient parallel solution of parabolic equations: implicit methods on the Cedar multicluster*, in: Proc. Fourth SIAM Conf. Parallel Processing for Scientific Computing (J. Dongarra, P. Messina, D.C. Sorensen, and R.G. Voigt, eds.), SIAM, Philadelphia, 1990, pp. 251–256.

[15] G.H. Golub and R.S. Varga, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods*, Numer. Math. **3** (1961), 147–168.

[16] M.R. Hestenes and E.L. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards **49** (1952), 409–436.

[17] 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.

[18] A.J. Laub, *Numerical linear algebra aspects of control design computations*, IEEE Trans. Automat. Contr. **AC-30** (1985), 97–108.

[19] J.A. Meijerink and H.A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp. **31** (1977), 148–162.

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

[21] Y. Saad and M.H. Schultz, GMRES: *a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput. **7** (1986), 856–869.

[22] W.L. Seward, G. Fairweather, and R.L. Johnston, *A survey of higher-order methods for the numerical integration of semidiscrete parabolic problems*, IMA J. Numer. Anal. **4** (1984), 375–425.

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

[24] R.A. Sweet, *A parallel and vector variant of the cyclic reduction algorithm*, SIAM J. Sci. Statist. Comput. **9** (1988), 761–765.

[25] D.M. Young, *The search for "high-level" parallelism for iterative sparse linear systems solvers*, in: Parallel Supercomputing: Methods, Algorithms and Applications (G.F. Carey, ed.), John Wiley, Chichester, 1989, pp. 89–105.

[26] D.M. Young and B.R. Vona, *On the use of rational iterative methods for solving large sparse linear systems*, Appl. Numer. Math. **10** (1992), 261–278.