TRANSPOSE-FREE FORMULATIONS OF LANCZOS-TYPE METHODS FOR NONSYMMETRIC LINEAR SYSTEMS*

TONY F. CHAN†, LISETTE DE PILLIS‡ AND HENK VAN DER VORST§

Abstract. We present a transpose-free version of the nonsymmetric scaled Lanczos procedure. It generates the same tridiagonal matrix as the classical algorithm, using two matrix-vector products per iteration without accessing A^T. We apply this algorithm to obtain a transpose-free version of the Quasi-Minimal Residual method of Freund and Nachtigal [11] (without look-ahead), which requires three matrix-vector products per iteration. We also present a related transpose-free version of the Bi-Conjugate Gradients algorithm.

Keywords: Lanczos algorithm, Quasi-Minimal Residual algorithm, Bi-Conjugate Gradients algorithm, Nonsymmetric Linear Systems, Krylov Subspace Methods.

AMS Subject Classification: 65F10, 65N20

1. Introduction.

Finding the solution of large nonsymmetric linear systems through iterative schemes is an important and fundamental problem in numerical linear algebra, and one in which there have been many recent exciting developments. One common approach is to generalize the very successful Conjugate Gradient method which is designed for symmetric positive definite matrices. There are two major classes of methods within this category, both intimately tied to related methods for computing eigenvalues. The first class of methods is based on the Arnoldi procedure (*c.f.* [13]) which computes an orthonormal basis for the Krylov subspace through a Gram-Schmidt process, reducing the original matrix A to a smaller upper-Hessenberg matrix H. The most well-known example in this class is the GMRES method [19], which minimizes the residual within this Krylov subspace. The second class of methods is based on the nonsymmetric Lanczos procedure (*c.f.* [13]), which reduces A to tridiagonal form through a three term recurrence. Among the members of this class are the Bi-Conjugate Gradient method (Bi-CG) [7, pp. 73-89], the Conjugate Gradient Squared method (CGS) [22], Quasi-Minimal Residual methods (QMR) [9], [11] and Bi-CGSTAB methods [23], [17], [3], [20].

One of the major advantages of the Lanczos-based methods are their short recurrences, as compared to the full orthogonalization required by the Arnoldi procedure. The short recurrences are more efficient with respect to both computational work and storage. However, there are several potential disadvantages to the Lanczos-based methods. First, none of the methods in this class minimizes the residual in any *fixed* norm.

* THIS VERSION: September 18, 1997. ORIGINAL VERSION: SEPT. 20, 1991.

† Dept. of Mathematics, UCLA, Los Angeles, CA 90095-1555, USA. This author was partially supported by the Office of Naval Research under contract N00014-90-J-1695, by the Department of Energy under contract DE-FG03-87ER25037, by the National Science Foundation under contract ASC 9003002, and by the Army Research Office under contract DAAL03-91-G-0150.

‡ Dept. of Mathematics, Harvey Mudd College, Claremont, CA 91711-5990, USA. This author was supported in part by the National Science Foundation under contract NSF-DMS93-21728

§ Mathematical Institute, University of Utrecht, Budapestlaan 6, Utrecht, The Netherlands.

The result of Faber and Manteuffel [6] precludes this. Second, the Lanczos process can "break down" before the iterates have converged. Third, both A and A^T are needed for matrix-vector products whereas the Arnoldi procedure does not require A^T.

The first two potential problems do show up in practice for the Bi-CG and CGS methods in the form of irregular convergence behavior, along with the associated problems of round-off errors. These problems are partly overcome by the QMR, Bi-CGSTAB and Generalized CGS[8] methods (the latter two can still break down.) For example, the QMR method solves a reduced system in a generalized least squares sense, and avoids the "break down" via a look-ahead strategy. Additionally, round-off problems are discussed in [14] and an approach to overcoming these is addressed in [21].

The purpose of our paper is to address the third disadvantage, namely the need to access A^T. There are at least three reasons for developing "transpose-free" methods:

1. Given a storage format for A, it may be difficult to perform $A^T v$ efficiently, since storing A^T explicitly would require extra storage space.
2. Transpose-free methods can be applied directly to nonlinear problems through the use of directional differencing techniques [2, 4, 24] in a Newton-type iteration. The essential idea is that matrix-vector products of the form $J(x)v$, where $J(x)$ is the Jacobian of a function $f(x)$, can be approximated by

$$J(x)v \approx \frac{(f(x+\epsilon v) - f(x))}{\epsilon},$$

where ϵ is a small parameter. It is not known how to come up with a similar trick for approximating $J^T(x)v$.
3. In some applications, neither A nor A^T are known explicitly, and matrix-vector products Av are computed by some complicated procedure. It may not be easy or convenient to modify such a procedure to compute $A^T v$. An example of this can be seen in the solution of integral equations where Av is computed by fast algorithms such as the "fast multipole algorithm" [15].

In this paper, we consider a "squared Lanczos" procedure, which generates the tridiagonal matrix of the standard nonsymmetric Lanczos procedure without accessing A^T. Such a procedure can then be used in conjunction with known algorithms for using the Lanczos tridiagonal factorization to solve linear systems. In particular, we will derive transpose-free implementations of both the QMR and the Bi-CG methods, which we will call TFiQMR and TFiBiCG. For related work on Krylov subspace methods for nonsymmetric systems, see [5].

The main idea of the "squared Lanczos" procedure is to borrow from the technique used by Sonneveld in deriving the CGS method [22]. In the standard Lanczos procedure, two sets of vectors, say v_i and w_i, are generated by two three-term recurrences. Typically, the w_i are not used in many Lanczos-based methods for solving linear systems. The main idea we need from CGS is to note that the coefficients in the recurrence for v_i (which form the tridiagonal factor) can be generated without referring to the w_i, *provided* we can generate the iterates u_i corresponding to the *squares* of the Krylov polynomial for v_i. It is relatively straightforward to derive a short recurrence for the u_i's using two matrix-vector products with A per step. However, if one also wants the original vectors v_i, which are needed for some methods including Bi-CG and QMR, it appears that either another matrix-vector product is needed per step, or we could choose to combine a factor of the squared polynomial from earlier constructed matrix-vector products. In this second case, we would have to save all the previous iteration results of Lanczos-squared. This second method would then be of little advantage over GMRES. Thus we focus on the first method mentioned, involving one extra matrix-vector product. We note that Gutknecht[16] has derived squared Bi-Conjugate Gradient methods that are somewhat similar to our squared Lanczos procedure.

Our algorithms for TFiBiCG and TFiQMR require three operations with A per iteration step. However, they may be useful not only for studying the behavior of QMR and Bi-CG when A^T is not readily available, they may also be used to build hybrid combinations of QMR or Bi-CG with CGS or Bi-CGSTAB methods. In that case the three operations with A would be natural and acceptable. For more about hybrid methods, see the work of Brezinski and Redivo-Zaglia[1].

For the sake of brevity, in this short note we have not dealt explicitly with many of the interesting questions which one may ask about these transpose-free methods. First, we have not considered the possible break-down of the Lanczos procedure, although the squaring procedure should extend to look-ahead implementations [10]. Second, although the transpose-free methods are mathematically equivalent to the original methods, we have not fully addressed the effect of round-off errors. Some preliminary numerical experiments will be presented in section 5. Thirdly, we have not explored the possibility of deriving a transpose-free version of the QMR method from other "product" methods (e.g. Bi-CGSTAB [23]) which may have better round-off properties than the squared Lanczos method presented here. Finally, we have not pursued the possibility of a QMR-squared method, which in principle is made possible by the squared Lanczos procedure. Freund and Szeto [12] derived exactly such a method based on the Bi-Conjugate Gradients algorithm. Freund [9] has also proposed a transpose-free method, TFQMR, which is essentially CGS in combination with the quasi-minimal residual method approach of QMR, and which is different from the method we put forth in this paper.

All norms in this paper are the Euclidean norm.

2. The Squared Nonsymmetric Lanczos Procedure.

We now derive a transpose-free version of the classical Lanczos algorithm. We start by squaring an unscaled version of the standard Lanczos algorithm, because the derivation is much simpler in this case. We will then derive a scaled version.

2.1. The Unscaled Version.

In the classical nonsymmetric Lanczos iteration, see [11] and [13], we generate a sequence of vectors v_i and w_i, $i = 0..n$, so that at iteration n, when setting

$$\begin{aligned} V_n &= [v_0, v_1, v_2, ..., v_i, ..., v_n] \\ W_n &= [w_0, w_1, w_2, ..., w_i, ..., w_n], \end{aligned}$$

we have

$$
\begin{aligned}
&(1) \qquad & W_n^T V_n &= D_n \\
&(2) \qquad & AV_n &= V_n T_n + v_{n+1}(e_{n+1}^{(n+1)})^T \\
&(3) \qquad & A^T W_n &= W_n T_n + w_{n+1}(e_{n+1}^{(n+1)})^T
\end{aligned}
$$

where

$$T_n = \begin{bmatrix} \alpha_0 & \beta_1 & & & \\ 1 & \alpha_1 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & \alpha_{n-1} & \beta_n \\ & & & 1 & \alpha_n \end{bmatrix},$$

D_n is an $(n+1) \times (n+1)$ diagonal matrix and $e_j^{(i)}$ denotes the j^{th} column of the $i \times i$ identity matrix. The following are the three term recurrence relationships for v_i and w_i:

$$
\begin{aligned}
&(4) \qquad & v_{i+1} &= Av_i - \alpha_i v_i - \beta_i v_{i-1} \\
&(5) \qquad & w_{i+1} &= A^T w_i - \alpha_i w_i - \beta_i w_{i-1},
\end{aligned}
$$

where

$$
\begin{aligned}
&(6) \qquad & \alpha_i &= (w_i, Av_i)/(w_i, v_i) \\
&(7) \qquad & \beta_i &= (w_i, v_i)/(w_{i-1}, v_{i-1})
\end{aligned}
$$

and $(\cdot, \cdot)$ represents the Euclidean inner product operator.

Let us now consider these recurrence relations in terms of polynomials. It is well known (see [22]) that v_i may be represented as an i-th degree polynomial in A multiplied with the initial vector v_0. Similarly, we may represent w_i with the *same* polynomial in A^T, multiplied with w_0. Thus we write

$$
\begin{aligned}
v_i &= \Phi_i(A)v_0 \\
w_i &= \Phi_i(A^T)w_0
\end{aligned}
$$

where Φ_i is a polynomial of degree i, and $\Phi_i(0) = 1$. The recurrence relation for Φ_i may now be written by substituting $\Phi_i(A)v_0$ into the recurrence relation for v_i:

$$(8) \qquad \Phi_i(A) = (A - \alpha_{i-1}I)\Phi_{i-1}(A) - \beta_{i-1}\Phi_{i-2}(A).$$

Examining the term (w_i, v_i) in equations (6) and (7), we find that by making the appropriate substitutions with polynomial representations, we may write

$$(w_i, v_i) = (\Phi_i(A^T)w_0, \Phi_i(A)v_0) = (w_0, \Phi_i^2(A)v_0).$$

We see that there is no longer the need to access A^T when calculating the coefficients α_i and β_i, as long as we know how to generate the polynomials $\Phi_i^2(A)$.

Following an argument similar to that outlined in [22], we find $\Phi_i^2(A)$ as follows:

$$\begin{aligned}
\Phi_i^2(A) &= ((A-\alpha_{i-1}I)\Phi_{i-1}(A) - \beta_{i-1}\Phi_{i-2}(A))^2 \\
&= [(A-\alpha_{i-1}I)\Phi_{i-1}(A)]^2 - 2\beta_{i-1}[(A-\alpha_{i-1}I)\Phi_{i-1}(A)]\Phi_{i-2}(A) + \beta_{i-1}^2\Phi_{i-2}^2(A) \\
&= (A-\alpha_{i-1}I)([(A-\alpha_{i-1}I)\Phi_{i-1}^2(A)] - 2\beta_{i-1}\Phi_{i-1}(A)\Phi_{i-2}(A)) + \beta_{i-1}^2\Phi_{i-2}^2(A).
\end{aligned}$$

It is clear in this recurrence for $\Phi_i^2(A)$ that we will also need a recurrence for the term $\Phi_{i-1}(A)\Phi_{i-2}(A)$. We multiply equation (8) by $\Phi_{i-1}(A)$ and find:

$$\begin{aligned}
\Phi_i(A)\Phi_{i-1}(A) &= ((A-\alpha_{i-1}I)\Phi_{i-1}(A) - \beta_{i-1}\Phi_{i-2}(A))\Phi_{i-1}(A) \\
&= (A-\alpha_{i-1}I)\Phi_{i-1}^2(A) - \beta_{i-1}\Phi_{i-1}(A)\Phi_{i-2}(A).
\end{aligned}$$

If we set $\Upsilon_{i-1}(A) = (A-\alpha_{i-1}I)\Phi_{i-1}^2(A)$, the complete recurrence relation for $\Phi_i^2(A)$ becomes

$$\begin{aligned}
\Upsilon_{i-1}(A) &= (A-\alpha_{i-1}I)\Phi_{i-1}^2(A) \\
\Phi_i^2(A) &= (A-\alpha_{i-1}I)(\Upsilon_{i-1}(A) - 2\beta_{i-1}\Phi_{i-1}(A)\Phi_{i-2}(A)) + \beta_{i-1}^2\Phi_{i-2}^2(A) \\
\Phi_i(A)\Phi_{i-1}(A) &= \Upsilon_{i-1}(A) - \beta_{i-1}\Phi_{i-1}(A)\Phi_{i-2}(A).
\end{aligned}$$

Translating back from a polynomial formulation to a vector formulation, we set

$$\begin{aligned}
z_i &= \Upsilon_i(A)v_0 \\
u_i &= \Phi_i^2(A)v_0 \\
q_i &= \Phi_i(A)\Phi_{i-1}(A)v_0
\end{aligned}$$

which then allows us to write our vector recursion formulae:

$$\begin{aligned}
\alpha_{i-1} &= \frac{(w_0, Au_{i-1})}{(w_0, u_{i-1})} \\
\beta_{i-1} &= \frac{(w_0, u_{i-1})}{(w_0, u_{i-2})} \\
z_{i-1} &= (A-\alpha_{i-1}I)u_{i-1} \\
u_i &= (A-\alpha_{i-1}I)(z_{i-1} - 2\beta_{i-1}q_{i-1}) + \beta_{i-1}^2 u_{i-2} \\
q_i &= z_{i-1} - \beta_{i-1}q_{i-1},
\end{aligned}$$

with $q_0 = 0$, $u_{-1} = 0$, and $u_0 = v_0$. This recurrence requires two matrix-vector multiplications, and does not involve A^T.

We now have a Lanczos iteration that will generate a sequence of tridiagonal matrices T_n whose eigenvalues can be used as approximations to those of A. See [13] for details.

2.2. The Scaled Version. The unscaled squared Lanczos algorithm above could potentially suffer from numerical overflow or underflow. To avoid this problem, we need to scale the vectors u_i and q_i as they are being generated. However, we still need to recover the α_i and β_i from equations (6), and (7). In this section we derive a procedure for accomplishing this.

Define $\delta_i = \|u_i\|$ and $\tilde{u}_i = u_i/\delta_i$, $\tilde{z}_i = z_i/\delta_i$ and $\tilde{q}_i = q_i/\delta_i$. Note that $\|\tilde{u}_i\| = 1$. Also define $f_i = \frac{\delta_{i-1}}{\delta_i}$. By a direct algebraic substitution, we get:

$$\alpha_{i-1} = \frac{(w_0, A\tilde{u}_{i-1})}{(w_0, \tilde{u}_{i-1})}$$

$$\beta_{i-1} = f_{i-1}^{-1}\frac{(w_0, \tilde{u}_{i-1})}{(w_0, \tilde{u}_{i-2})}$$

$$\tilde{u}_i = f_i[(A - \alpha_{i-1}I)(\tilde{z}_{i-1} - 2\beta_{i-1}\tilde{q}_{i-1}) + f_{i-1}\beta_{i-1}^2\tilde{u}_{i-2}]$$

$$\tilde{q}_i = f_i(\tilde{z}_{i-1} - \beta_{i-1}\tilde{q}_{i-1}).$$

Since $\|\tilde{u}_i\| = 1$, we see that

$$f_i = \|(A - \alpha_{i-1}I)(\tilde{z}_{i-1} - 2\beta_{i-1}\tilde{q}_{i-1}) + f_{i-1}\beta_{i-1}^2\tilde{u}_{i-2}\|^{-1}.$$

Thus, we have all the information we need to generate α_i and β_i using only the scaled quantities $\tilde{u}_i, \tilde{z}_i, \tilde{q}_i$.

3. Transpose-Free implementation of the QMR Method : TFiQMR. We now turn to the Quasi-minimal residual algorithm developed by Freund and Nachtigal [11]. We embed this in our squared Lanczos algorithm in order to solve the nonsymmetric linear system $Ax = b$.

First let us define $\tilde{V}_n \equiv [\tilde{v}_0, ..., \tilde{v}_i, ..., \tilde{v}_n] = V_n D_n$, where $D_n = diag(d_0, ..., d_i, ..., d_n)$ with $d_i = \|v_i\|$. Note that $\|\tilde{v}_i\| = 1$. Now we rewrite equation (2) as

$$A\tilde{V}_n = \tilde{V}_{n+1}H_e^{(n+1)},$$

where

$$H_e^{(n+1)} = \begin{bmatrix} \tilde{T}_n \\ d_{n+1}(e_{n+2}^{(n+2)})^T \end{bmatrix},$$

and $\tilde{T}_n = D_n T_n D_n^{-1}$.

Given an initial guess x_0 for the exact solution of $Ax = b$, we construct iterates x_n such that

$$x_n = x_0 + \tilde{V}_{n-1}g_n, \tag{9}$$

where $g_n \in \Re^n$.

We let $r_n = b - Ax_n$ be the residual vector corresponding to the n^{th} iterate x_n, and set $\tilde{v}_0 = \frac{r_0}{\|r_0\|}$.

The residual vectors corresponding to equation (9) satisfy

$$
\begin{aligned}
r_n &= r_0 - A\tilde{V}_{n-1} g_n \\
&= r_0 - \tilde{V}_n H_e^{(n)} g_n \\
&= \tilde{V}_n (\|r_0\| e_1^{(n+1)} - H_e^{(n)} g_n). \qquad (10)
\end{aligned}
$$

Ideally, we would like to choose g_n in equation (10) so that $\|r_n\|$ is minimal. However, since in general $\tilde{V}_n$ is not unitary, this would require $\mathcal{O}(Nn^2)$ work (where N is the dimension of the matrix A), which is too expensive. Instead, the main idea of the QMR algorithm is to minimize only the Euclidean norm of the bracketed term in equation (10). Thus, we choose g_n as the solution of the least squares problem

$$
\|e_1^{(n+1)} \|r_0\| - H_e^{(n)} g_n\| \;=\; \min_{g \in \Re^n} \|e_1^{(n+1)} \|r_0\| - H_e^{(n)} g\|.
$$

It can be shown that the solution g_n is unique, and hence defines a unique n^{th} iterate x_n. In view of this minimization property, the method is referred to as the "Quasi-minimal residual" method. We note that the QMR method as defined in [11] allows an arbitrary diagonal scaling of $H_e^{(n)}$ but we shall leave that out in our discussions for the sake of simplicity.

For the solution of the least squares problem, the standard approach is used, see [13], based on a QR decomposition of $H_e^{(n)}$. Let

$$
H_e^{(n)} = (Q_n)^T \begin{bmatrix} R_n \\ 0 \end{bmatrix}
$$

be the QR decomposition, where Q_n is a unitary $(n+1) \times (n+1)$ matrix, and R_n is a nonsingular upper triangular $n \times n$ matrix. Next we let

$$
\begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \\ \tilde{\tau}_{n+1} \end{bmatrix} = Q_n \|r_0\| e_1^{(n+1)}, \qquad t_n = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix}. \qquad (11)
$$

Our quasi-minimal residual solution x_n is then given by

$$
x_n = x_0 + \tilde{V}_n R_n^{-1} t_n.
$$

For practical implementation of this scheme, there is an efficient recurrence scheme for updating x_n [11, pp. 11-14], which eliminates the need to store all the vectors $\tilde{v}_i$.

In the original QMR method, the vectors $\tilde{v}_i$ and the corresponding $\tilde{T}_n$ are generated via a scaled version of the Lanczos algorithm [11]:

$$
\gamma_i \tilde{v}_{i+1} = A\tilde{v}_i - \tilde{\alpha}_i \tilde{v}_i - \tilde{\beta}_i \tilde{v}_{i-1}
$$

where the scaling factor γ_i is chosen at each iteration so that $\|\tilde{v}_{i+1}\| = 1$. If we wish to embed the QMR solution method in our squared Lanczos algorithm, we will have to generate the vectors $\tilde{v}_i$ in some way. Since the squared Lanczos algorithm produces α_i and β_i, we can generate the unscaled vectors v_i by (4), at the expense of one additional matrix-vector product, and then scale these vectors to obtain $\tilde{v}_i$. However, such a procedure would also suffer from potential numerical overflow and underflow. Thus we now derive a method for generating the $\tilde{v}_i$ directly from the α_i and β_i which are generated by the squared Lanczos algorithm.

Substituting $v_i = d_i \tilde{v}_i$ into equation (4), we get

$$\gamma_i \tilde{v}_{i+1} = A\tilde{v}_i - \alpha_i \tilde{v}_i - \beta_i \tilde{v}_{i-1} \gamma_{i-1}^{-1},$$

where $\gamma_i = d_{i+1}/d_i$. Since $\|\tilde{v}_{i+1}\| = 1$, we get the following updating formula for γ_i:

$$\gamma_i = \|A\tilde{v}_i - \alpha_i \tilde{v}_i - \beta_i \tilde{v}_{i-1} \gamma_{i-1}^{-1}\|.$$

Defining $\tilde{\beta}_i = \beta_i \gamma_{i-1}^{-1}$, $\tilde{T}_n$ is then given by

$$\tilde{T}_n = \begin{bmatrix} \alpha_0 & \tilde{\beta}_1 & & & \\ \gamma_0 & \alpha_1 & \tilde{\beta}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-2} & \alpha_{n-1} & \tilde{\beta}_n \\ & & & \gamma_{n-1} & \alpha_n \end{bmatrix},$$

and $d_{n+1} = \gamma_n$. Note that $\tilde{T}_n$ is computed directly without computing the d_i's explicitly.

We now summarize the above procedures in the following pseudo-code for a scaled transpose-free QMR method based on the squared Lanczos algorithm. In this code we have incorporated the updating procedure to generate x_n.

ALGORITHM TFiQMR for solving $Ax = b$ given x_0:

(1) Initializations:

(a) Set $r_0 = b - Ax_0$; $\tilde{v}_0 = \tilde{u}_0 = r_0/\|r_0\|$. Choose arbitrary w_0 (e.g. $w_0 = \tilde{v}_0$).

(b) Set $\tilde{u}_{-1} = \tilde{v}_{-1} = p_{-1} = p_{-2} = \tilde{q}_0 = 0$;

(c) Set $\theta = \epsilon = \beta_0 = \tilde{\beta}_0 = 0$; $\sigma_0 = 1$; $\tilde{\tau}_1 = \|r_0\|$;

(2) Calculate the squared Lanczos Vectors:

For $i = 1, 2, 3, ...$ do:

(a) $\rho_{i-1} = w_0^T \tilde{u}_{i-1}$; $y = A\tilde{u}_{i-1}$; $\alpha_{i-1} = \tilde{\delta} = w_0^T y/\rho_{i-1}$;

(b) $\tilde{z}_{i-1} = y - \alpha_{i-1}\tilde{u}_{i-1}$;

(c) If $(i > 1)$ $\beta_{i-1} = \rho_{i-1}/(f_{i-1}\rho_{i-2})$;

(d) $\hat{u}_i = (A - \alpha_{i-1}I)(\tilde{z}_{i-1} - 2\beta_{i-1}\tilde{q}_{i-1}) + f_{i-1}\beta_{i-1}^2\tilde{u}_{i-2}$; $f_i = 1/\|\hat{u}_i\|$; $\tilde{u}_i = f_i\hat{u}_i$;

(e) $\tilde{q}_i = f_i(\tilde{z}_{i-1} - \beta_{i-1}\tilde{q}_{i-1})$;

(f) If $(i > 1)$ $\tilde{\beta}_{i-1} = \beta_{i-1}/\gamma_{i-2}$; $\tilde{\epsilon} = \tilde{\beta}_{i-1}$;

(g) $\hat{v}_i = (A - \alpha_{i-1}I)\tilde{v}_{i-1} - \tilde{\beta}_{i-1}\tilde{v}_{i-2}$; $\gamma_{i-1} = \|\hat{v}_i\|$; $\tilde{v}_i = \hat{v}_i/\gamma_{i-1}$;

(3) Update QR-factorization (see [11]):

(a) If $(i > 2)$ $\theta = s_{i-2}\tilde{\beta}_{i-1}$; $\tilde{\epsilon} = -c_{i-2}\tilde{\beta}_{i-1}$;

(b) If $(i > 1)$ $\epsilon = -c_{i-1}\tilde{\epsilon} + s_{i-1}\alpha_{i-1}$; $\tilde{\delta} = -s_{i-1}\tilde{\epsilon} - c_{i-1}\alpha_{i-1}$;

(c) If $(|\gamma_{i-1}| > |\tilde{\delta}|)$ $\kappa = -\tilde{\delta}/\gamma_{i-1}$; $s_i = 1/\sqrt{1+\kappa^2}$; $c_i = s_i\kappa$;
Else $\kappa = -\gamma_{i-1}/\tilde{\delta}$; $c_i = 1/\sqrt{1+\kappa^2}$; $s_i = c_i\kappa$;

(d) $\delta = -c_i\tilde{\delta} + s_i\gamma$;

(4) Update solution and bound on residual norm (see [11]):

(a) $\tau_i = -c_i\tilde{\tau}_i$; $\tilde{\tau}_{i+1} = -s_i\tilde{\tau}_i$;

(b) $p_i = (\tilde{v}_{i-1} - \theta p_{i-2} - \epsilon p_{i-1})/\delta$;

(c) $x_i = x_{i-1} + \tau_i p_i$;

(d) $\sigma_i = \sigma_{i-1}|s_i|$; (Note: $\|r_i\| \le \|r_0\|\sigma_i\sqrt{i+1}$);

(e) If (x_i has converged) exit.

End For.

4. Transpose-Free implementation of the Bi-CG Method : TFiBiCG. In this section, we present a transpose-free version of Bi-CG [7]. It is well known that the Bi-CG algorithm is related to the Lanczos algorithm. Therefore, in principle the squared Lanczos algorithm in Section 2 can be used to derive a transpose-free Lanczos method for solving $Ax = b$ that is mathematically equivalent to Bi-CG, in a way similar to the derivation of TFiQMR. However, it is also possible to directly obtain a transpose-free Bi-CG algorithm. To do this, we use CGS to generate the $\Phi_i^2(A)$ polynomials, and then extract the necessary Bi-CG iterates from CGS. Essentially, we carry out the CGS iteration, and then add three new iteration lines to extract the Bi-CG iterates from the data generated. The algorithm follows. Our notation here follows that in [23].

ALGORITHM TFiBiCG for solving $Ax = b$ given x_0:

(1) Initialization:

(a) $r_0 = b - Ax_0$; $\hat{r_0}$ arbitrary;

(b) $p_0^{\mathrm{BCG}} = 0; \rho_0 = 1;$ ← New

(c) $p_0^{\mathrm{CGS}} = 0; q_0 = 0;$

(2) For $i = 1, 2, \ldots$ do:

(a) $\rho_i = (\hat{r_0}, r_{i-1}^{\mathrm{CGS}})$

(b) $\beta = \frac{\rho_i}{\rho_{i-1}}$

(c) $u = r_{i-1}^{\mathrm{CGS}} + \beta q_{i-1}$

(d) $p_i^{\mathrm{CGS}} = u + \beta(q_{i-1} + \beta p_{i-1}^{\mathrm{CGS}})$

(e) $p_i^{\mathrm{BCG}} = r_i^{\mathrm{BCG}} + \beta p_{i-1}^{\mathrm{BCG}}$ ← New

(f) $v = Ap_i^{\mathrm{CGS}}$

(g) $\alpha = \frac{\rho_i}{(\hat{r_0}, v)}$

(h) $q_i = u - \alpha v$

(i) $\tilde{u} = u + q_i$

(j) $r_i^{\mathrm{CGS}} = r_{i-1}^{\mathrm{CGS}} - \alpha A\tilde{u}$

(k) $x_i^{\mathrm{BCG}} = x_{i-1}^{\mathrm{BCG}} + \alpha p_i^{\mathrm{BCG}}$ ← New

(l) $r_i^{\mathrm{BCG}} = r_{i-1}^{\mathrm{BCG}} - \alpha A p_i^{\mathrm{BCG}}$ ← New

End for

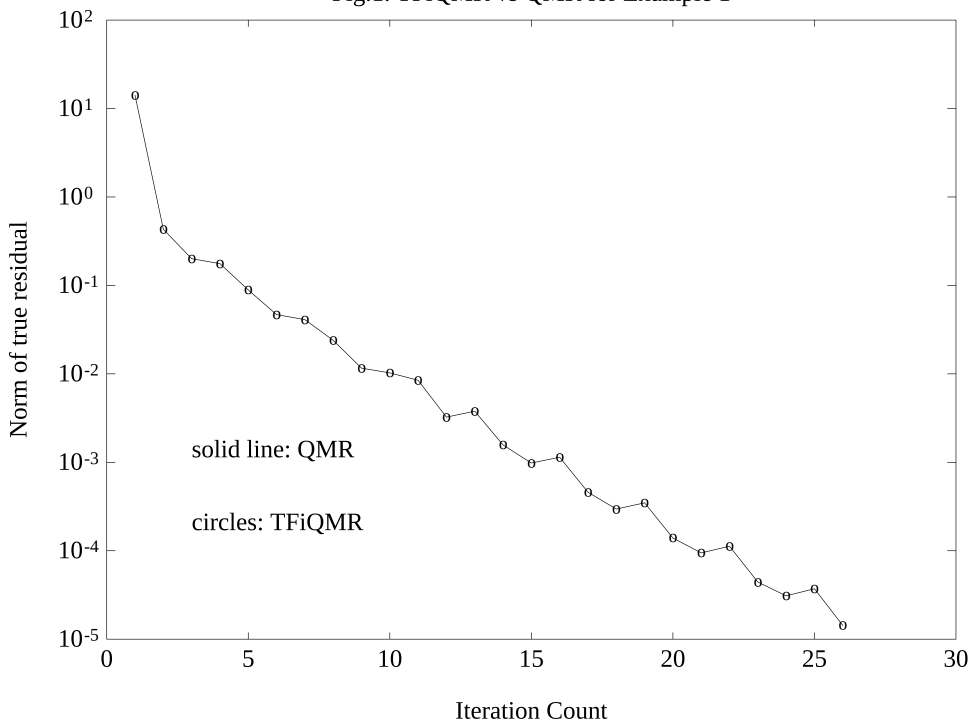
In the above algorithm, the Bi-CG iterates are obtained from the CGS iterates at the cost of one extra matrix-vector multiplication per iteration. There is no computation involving A^T.

We point out here that if CGS is converging smoothly, there is no advantage to extracting the Bi-CG iterates. However, it is not difficult to construct examples in which CGS will not converge well at all, whereas Bi-CG will [23]. This can happen in practical situations when the algorithm is used in combination with Modified ILU preconditioning. In many cases, this leads to fast convergence of one of the extreme Ritz values, and correspondingly, to loss of biorthogonality in the Bi-CG scheme. Squaring the corresponding polynomial in that case would not be advisable.

Another example for which we would be interested in a transpose-free Bi-CG algorithm is the case when one is solving nonlinear problems with Newton's method. In the final stages of the Newton process, say when quadratic convergence has set in, CGS is often seen to behave erratically. In these cases, one often does not have access to the transpose of the operator, so this transpose-free variant of Bi-CG can be useful.

5. Numerical Experiments. We present several numerical examples using TFiQMR. Since this algorithm is mathematically equivalent to the original QMR algorithm, our main purpose is to see whether TFiQMR behaves the same as QMR in finite precision arithmetic. Unless otherwise stated, we use $w_0 = \tilde{v}_0$ and $x_0 = 0$. The tests were run on several different machines, each with machine precision about 10^{-16}.

Fig.1. TFiQMR vs QMR for Example 1



Example 1: The first example is the following 200×200 matrix from [17]:

$$A = \begin{bmatrix} 2 & 1 & & & & \\ 0 & 2 & 1 & & & \\ 1 & 0 & 2 & 1 & & \\ & 1 & 0 & 2 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots \end{bmatrix},$$

and the right hand side is chosen to be $b = (1, ..., 1)^T$. In Figure 1, we plot the norm of the residual versus the iteration number for both QMR and TFiQMR. We can see that the TFiQMR residual norms are indistinguishable from those of QMR. In fact, we have

$$\|x_{20}^{QMR} - x_{20}^{TFiQMR}\| / \|x_{20}^{QMR}\| < 2 \times 10^{-14}$$

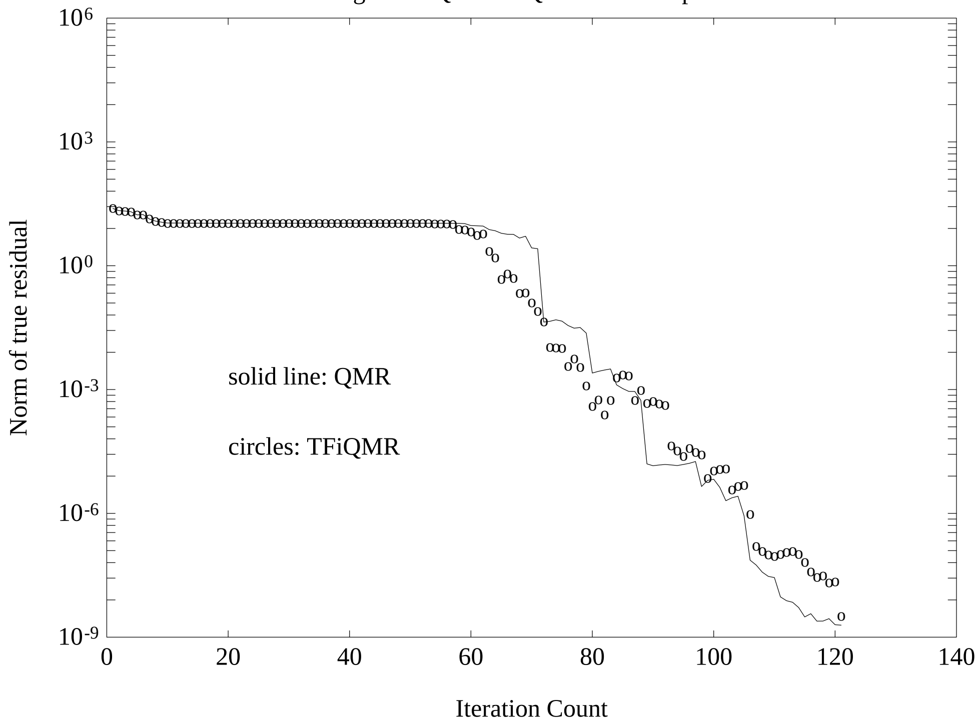
at iteration 20. Thus, the TFiQMR iterates seem to agree with those of QMR to almost full machine precision for this example.

Example 2: This example is taken from Example 3 of [23]. The system comes from a second order central difference discretization of the partial differential equation:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + 0.5(\frac{\partial (au)}{\partial x} + a\frac{\partial u}{\partial x}) = 1,$$

over the unit square with $a = 20e^{3.5(x^2+y^2)}$, Dirichlet boundary conditions and mesh size $h = 1/101$ $(10, 000$ unknowns). A standard ILU preconditioner is used[18]. In Figure 2, we plot the residual generated by QMR and TFiQMR. Again, the two methods behave quite similarly, although after about 60 iterations, there is a noticeable difference.

Fig.2. TFiQMR vs QMR for Example 2



Example 3: This example is taken from Example 6.1 of [9]. The system comes from a second order central difference discretization of the partial differential equation:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + \gamma(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y}) + \beta u = f,$$

over the unit square with Dirichlet boundary conditions. No preconditioner is used. For this example, we use mesh size $h = 1/32$ (961 unknowns), $\gamma = 50$ and $\beta = 25$. The numerical results are shown in Figure 3. Once again, TFiQMR is indistinguishable from QMR up to 60 iterations, after which the difference is noticeable but both methods converge.

Example 4: This example is the same as Example 3, except we use mesh size $h = 1/64$ (3969 unknowns), $\gamma = 100$ and $\beta = 100$. The results are shown in Figure 4. We see that TFiQMR and QMR agree to about 150 iterations after which TFiQMR converges at a much slower rate than QMR. We also ran the problem with w_0 chosen randomly from a normal distribution with mean 0 and variance 1, as was the case in [9]. In this case, QMR converged (reduced the norm of the residual by a factor of 10^{-6}) in about 270 iterations and TFiQMR converged in 420 iterations. We are not sure how round-off errors and the choice of w_0 affect the behavior of either method or how representative this example is.

From the above numerical examples, we see that in finite precision, TFiQMR behaves quite similarly to QMR in many cases, but in some cases it can be inferior. More tests and analysis are needed to fully understand this issue.

Acknowledgment: We acknowledge generous help from Charles Tong of Sandia Laboratory for proof-reading and providing the numerical results for Example 3 and 4.

Fig.3. TFiQMR vs QMR for Example 3

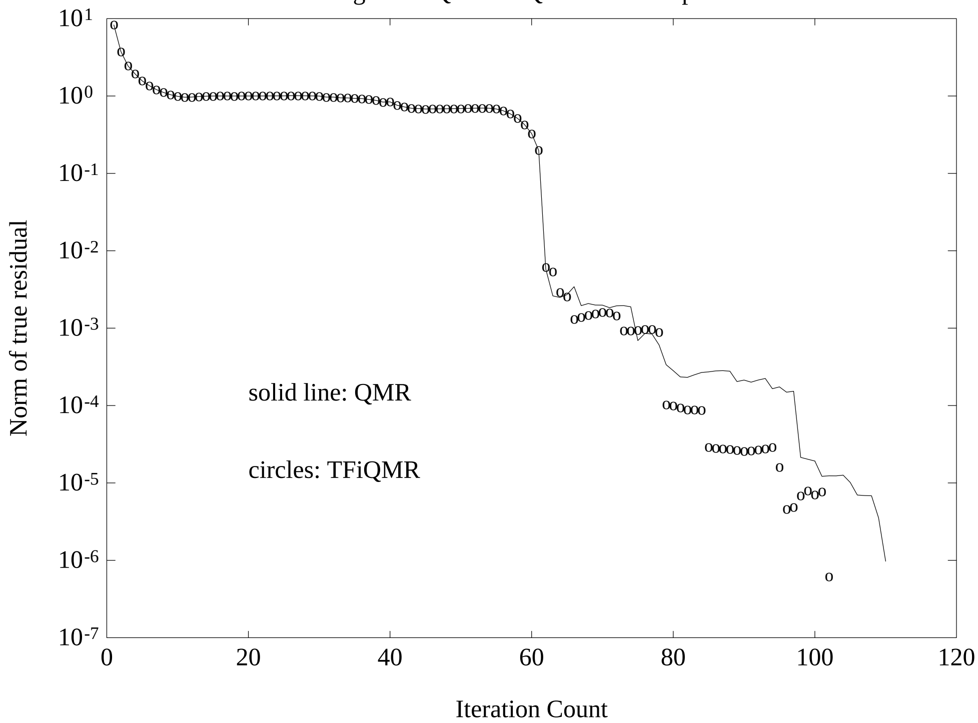
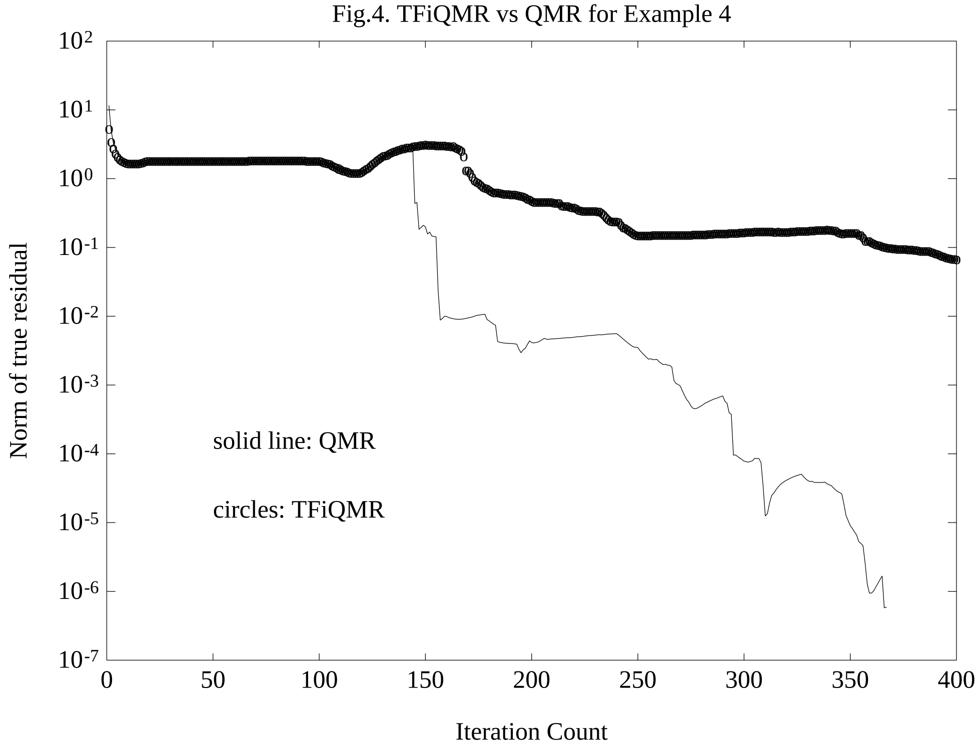


Fig.4. TFiQMR vs QMR for Example 4



REFERENCES

[1] C. Brezinski and M. Redivo-Zaglia, *Hybrid Procedures for Solving Linear Systems*, Numer. Math., 67 (1994), pp. 1–19.

[2] P. Brown and Y. Saad, *Hybrid Krylov Methods for Nonlinear Systems of Equations*, SIAM J. Sci. Stat. Comp., 11 (1990), pp. 450–481.

[3] T. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, and C. Tong, *A Quasi-Minimal Residual Variant of the Bi-CGSTAB Algorithm for Nonsymmetric Systems*, SIAM J. Sci. Comput., 15 (1994), pp. 338–347.

[4] T. Chan and K. Jackson, *The Use of Iterative Linear Equation Solvers in Codes for Large Systems of Stiff IVP's for ODE's*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 378–417.

[5] T. Chronopoulos and S. Ma, *On Squaring Krylov Subspace Iterative Methods for Nonsymmetric Linear Systems*, Tech. Rep. 89-67, Computer Science Department, University of Minnesota, 1989.

[6] V. Faber and T. Manteuffel, *Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method*, SIAM J. Numer. Anal., 21 (1984), pp. 352–362.

[7] R. Fletcher, *Conjugate Gradient Methods for Indefinite Systems*, no. 506 in Lecture Notes in Mathematics, Springer-Verlag, 1976.

[8] D. Fokkema, G. Sleijpen, and H. Van der Vorst, *Generalized Conjugate Gradient Squared*, J. of Comp. and Appl. Math., 71 (1994), pp. 125–146.

[9] R. Freund, *A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems*, SIAM J. Sci. Comp., 14 (1993), pp. 470–482.

[10] R. Freund, M. Gutknecht, and N. Nachtigal, *An Implementation of the Look-Ahead Lanczos Algorithm for non-Hermitian Matrices*, SIAM J. Sci. Comp., 13 (1992), pp. 137–158.

[11] R. Freund and N. Nachtigal, *QMR: A Quasi-Minimal Residual Method for Non-Hermitian Linear Systems*, Numer. Math., 60 (1991), pp. 315–339.

[12] R. Freund and T. Szeto, *A Transpose-Free Quasi-Minimal Residual Squared Algorithm for Non-Hermitian Linear Systems*, in Advances in Computer Methods for Partial Differential Equations – VII, R. Vichnevetsky, D. Knight, and G. Richter, eds., IMACS, 1992, pp. 258–264.

[13] G. Golub and C. van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

[14] A. Greenbaum, *Estimating the Attainable Accuracy of Recursively Computed Residual Methods*, SIAM J. Matrix Anal., 18 (1997), pp. 535–551.

[15] L. Greengard and V. Rokhlin, *A Fast Algorithm for Particle Simulations*, J. Comp. Phys., 73 (1987), p. 325.

[16] M. Gutknecht, *The Unsymmetric Lanczos Algorithms and Their Relations to Pade Approximation, Continued Fractions and the QD Algorithms*, in Proceedings of the Copper Mountain Conference on Iterative Methods, April 1990. Also available at http://www.scsc.ethz.ch/ mhg/pub/CopperMtn90.ps.Z and CopperMtn90-7.ps.Z.

[17] ———, *Variants of BiCGStab for Matrices with Complex Spectrum*, SIAM J. Sci. Comp., 14 (1993), pp. 1020–1033.

[18] J. Meijerink and H. Van der Vorst, *An Iterative Solution Method for Linear Systems of Which The Coefficient Matrix is a Symmetric M-Matrix*, Math. Comp., 31 (1977), pp. 148–162.

[19] Y. Saad and M. Schultz, *GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.

[20] G. Sleijpen and D. Fokkema, *BICGSTAB(ℓ) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum*, ETNA, 1 (1993), pp. 11–32.

[21] G. Sleijpen and H. Van der Vorst, *Reliable Updated Residuals in Hybrid Bi-CG Methods*, Computing, 56 (1996), pp. 141–163.

[22] P. Sonneveld, *CGS: A Fast Lanczos-Type Solver For Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comp., 10 (1989), pp. 36–52.

[23] H. Van der Vorst, *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*, SIAM J. Sci. Comp., 13 (1992), pp. 631–644.

[24] L. Wigton, D. Yu, and N. Young, *GMRES Acceleration of Computational Fluid Dynamics Codes*, in AIAA Conference, Denver, CO, 1985.