SIAM J. NUMER. ANAL.
Vol. 25, No. 4, August 1988

TWO CONJUGATE-GRADIENT-TYPE METHODS FOR UNSYMMETRIC LINEAR EQUATIONS*

M. A. SAUNDERS†, H. D. SIMON‡, AND E. L. YIP§

Abstract. We propose two new conjugate-gradient-type methods for the solution of sparse unsymmetric linear systems. We present a new tridiagonalization process for unsymmetric matrices that is closely related to the Lanczos process. We use orthogonal factorizations of the tridiagonal matrix to derive the new algorithms USYMLQ and USYMQR in the same fashion as SYMMLQ and MINRES [C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM J. Numer. Anal.*, 12 (1975), pp. 617–629], for symmetric matrices. Some numerical results for the new methods and comparisons with other methods are presented.

Key words. conjugate gradients, unsymmetric sparse linear systems

AMS (MOS) subject classification. 65F10

1. Introduction. The successful application of the method of conjugate gradients for the numerical solution of sparse symmetric positive-definite linear systems, especially in connection with preconditioning, has prompted an intensive search for extensions of this method, which would perform equally well for nonsymmetric matrices (for a survey, see [3]). The three properties that make the method of conjugate gradients (CG hereafter) so useful and interesting are the finite termination property, the minimization property, and the fact that the method is based on a three-term recurrence. These properties imply that CG is guaranteed to terminate after a finite number of steps (in exact arithmetic), that some measure of the error is decreased at every step of the method, and that the computational requirements for each step are constant. Even though the finite termination property is rarely of importance in the practical applications of CG, it distinguishes CG theoretically from methods such as SOR, which do not possess this property.

Here we will present a method for general unsymmetric linear systems that enjoys all three nice properties of the conjugate-gradient algorithm. This does not contradict the results of Faber and Manteuffel [5], who answered negatively a question by Golub [6] about the existence of such a method. In contrast our method is not based on an approximation from a Krylov subspace. The new method also does not apply conjugate gradients to the normal equations. Instead, the methods are derived from a tridiagonalization process for unsymmetric matrices, which will be discussed in § 2. This tridiagonalization process reduces to the symmetric Lanczos process if it is applied to a symmetric matrix. It is distinctively different from the Lanczos biorthogonalization (or the biconjugate-gradient method) for unsymmetric matrices, which has been used for deriving methods for unsymmetric linear systems, for example, by Saad [18], [19].

The application of the new tridiagonalization process to the solution of linear systems is discussed in § 3, where we also derive some of the properties of the new

* Received by the editors August 6, 1984; accepted for publication (in revised form) July 10, 1987.

† Department of Operations Research, Stanford University, Stanford, California 94305. The work of this author was partially supported by National Science Foundation grant MCS-7926009.

‡ Boeing Computer Services, Engineering Technology Applications Division, Seattle, Washington 98124-0346. The work of this author was partially supported by the State University of New York under Research Foundation grant 431-7559A.

§ Boeing Aerospace Company, Seattle, Washington 98124-3999. The work of this author was supported by the National Aeronautics and Space Administration under contract NA51-16297.

methods. We discuss the choice of the starting vectors and the relationship to other methods in § 4.

Section 5 presents the practical implementation of the new methods. Using the LQ factorization of the tridiagonal matrix we obtain the algorithm USYMLQ in an analogous manner to the derivation of SYMMLQ from the symmetric Lanczos process by Paige and Saunders [13]. We use the QR factorization to obtain USYMQR, which can be considered as a generalization of MINRES [13]. Some numerical results are presented in § 6.

2. A tridiagonalization process for unsymmetric matrices. In this section we present a tridiagonalization process for a general unsymmetric real $n \times n$ matrix A. We consider only the real case here, and * will always mean "transpose," but the algorithm can be easily extended to complex matrices. The tridiagonalization process can be derived by constructing the following transformation of A:

$$P^*AQ = T, \tag{2.1}$$

where P and Q are orthogonal matrices and T is a tridiagonal matrix. If A is symmetric we can choose $P = Q$. Then T is symmetric as well and the symmetric Lanczos algorithm will be obtained (see [13]). We note that (2.1) is quite different from the Lanczos biorthogonalization algorithm for unsymmetric matrices (see [9], [25]), which is based on the similarity transformation

$$P^*AQ = T. \tag{2.2}$$

T is again tridiagonal, but $P^* = Q^{-1}$. Whereas (2.2) preserves the eigenvalues of A, (2.1) preserves the singular values of A, which is quite appropriate for solving linear equations. The proposed tridiagonalization process therefore can also be used for determining the singular values of large matrices. This will be discussed further in § 4.

THEOREM 1. *Let A be a general $n \times n$ matrix. There exist orthogonal matrices P and Q and a tridiagonal matrix T with positive offdiagonal elements such that*

$$P^*AQ = T. \tag{2.3}$$

Proof. Set $q_0 = 0, p_0 = 0, \beta_1 = 0, \gamma_1 = 0$. Choose arbitrary unit vectors p_1 and q_1. Then for $j = 1, 2, \cdots, n-1$ define

$$\beta_{j+1}p_{j+1} = Aq_j - \alpha_j p_j - \gamma_j p_{j-1}, \tag{2.4a}$$

$$\gamma_{j+1}q_{j+1} = A^*p_j - \alpha_j q_j - \beta_j q_{j-1}, \tag{2.4b}$$

with $\alpha_j = p_j^* A q_j$, where $\beta_{j+1} > 0$ and $\gamma_{j+1} > 0$ are chosen such that p_{j+1} and q_{j+1} are unit vectors. We will now show that the vectors thus constructed satisfy

$$q_j^* q_k = 0 \quad \text{for } k < j, \tag{2.5a}$$

$$p_j^* p_k = 0 \quad \text{for } k < j, \tag{2.5b}$$

$$p_j^* A q_k = 0 \quad \text{for } k < j-1, \tag{2.5c}$$

$$q_j^* A p_k = 0 \quad \text{for } k < j-1. \tag{2.5d}$$

Hence the matrices $Q = Q_n = (q_1, q_2, \cdots q_n)$, $P = P_n = (p_1, p_2, \cdots, p_n)$, and

$$T = T_n = \begin{bmatrix} \alpha_1 & \gamma_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \gamma_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \beta_{n-1} & \alpha_{n-1} & \gamma_n \\ 0 & \cdots & 0 & \beta_n & \alpha_n \end{bmatrix}$$

will satisfy (2.3).

Because of the choice of α_j we have immediately that $q_{j+1}^* q_j = 0$ and $p_{j+1}^* p_j = 0$ for $j = 1, 2, \cdots, n-1$. Also it can be verified that (2.5) is true for $j = 1$. We will show by induction that (2.5) holds in general. Suppose that (2.5) has been shown for j; then we obtain by multiplying (2.4b) with q_k^*, $k < j-1$ that

$$\begin{aligned} \gamma_{j+1} q_k^* q_{j+1} &= q_k^* A^* p_j \\ &= p_j^* A q_k = p_j^* (\beta_{k+1} p_{k+1} + \alpha_k p_k + \gamma_k p_{k-1}) = 0. \end{aligned}$$

Here we also used (2.4b) for Aq_k. Similarly, (2.5b)-(2.5d) can be obtained using induction and formulas (2.4a), (2.4b). In order to show (2.5a) for $k = j$ we compute

$$\begin{aligned} \gamma_{j+1} q_{j-1}^* q_{j+1} &= q_{j-1}^* A^* p_j - \beta_j \\ &= p_j^* (\beta_j p_j + \alpha_{j-1} p_{j-1} + \gamma_{j-1} p_{j-2}) - \beta_j \\ &= \beta_j - \beta_j = 0. \end{aligned}$$

Similarly (2.5b) can be shown for $k = j$.

Finally we have to address the question of a breakdown of formulas (2.4). It is possible that

$$A^* p_j \in \text{span}\,(q_1, q_2, \cdots, q_j) \quad \text{or} \quad A q_j \in \text{span}\,(p_1, p_2, \cdots, p_j).$$

In that case q_{j+1} (or p_{j+1}) would be zero. However, the recurrence (2.4) can be continued, if we choose an arbitrary unit vector $q_{j+1} \perp \text{span}\,(q_1, q_2, q_3, \cdots, q_j)$ and set $\gamma_{j+1} = 0$ (or choose $p_{j+1} \perp \text{span}\,(p_1, p_2, p_3, \cdots, p_j)$ and set $\beta_{j+1} = 0$). It can be shown that (2.5) also holds in this case. This completes the proof of Theorem 1. □

The proof of Theorem 1 shows that the tridiagonalization (2.3) is essentially uniquely determined by the matrix A and the choice of the starting vectors p_1 and q_1, unless breakdown occurs.

A tridiagonalization algorithm can be derived from (2.3) as follows:

ALGORITHM 1. Tridiagonalization of an Unsymmetric Matrix.

(1) Initialization.

Choose two arbitrary vectors $b \neq 0$, $c \neq 0$.

Set $p_0 = q_0 = 0$, $\beta_1 = \|b\|$, $\gamma_1 = \|c\|$, and $p_1 = b/\beta_1$, $q_1 = c/\gamma_1$.

(2) Loop

For $i = 1, 2, 3 \cdots$ do

$u \leftarrow Aq_i - \gamma_i p_{i-1}$

$v \leftarrow A^* p_i - \beta_i q_{i-1}$

$\alpha_i \leftarrow p_i^* u$

$u \leftarrow u - \alpha_i p_i$

$v \leftarrow v - \alpha_i^* q_i$

$\beta_{i+1} \leftarrow \|u\|$

$\gamma_{i+1} \leftarrow \|v\|$

if $\beta_{i+1} = 0$ or $\gamma_{i+1} = 0$ then stop

else $p_{i+1} \leftarrow u/\beta_{i+1}$

$q_{i+1} \leftarrow v/\gamma_{i+1}$

End loop

The choice of b and c, and the consequences of an early termination of the algorithm (because $\beta_i = 0$ or $\gamma_i = 0$), will be discussed later.

Let the tridiagonal matrix T_j be defined as the leading $j \times j$ submatrix of $T = T_n$, and let P_j and Q_j be matrices whose columns are the vectors p_i and q_i, i.e., $P_j = (p_1, p_2, \cdots, p_j)$, $Q_j = (q_1, q_2, \cdots, q_j)$. Then the first j steps can be written as

$$AQ_j = P_j T_j + \beta_{j+1} p_{j+1} e_j^*, \tag{2.6a}$$

$$A^* P_j = Q_j T_j^* + \gamma_{j+1} q_{j+1} e_j^*. \tag{2.6b}$$

Here e_j is a unit vector of length j, $e_j^* = (0, 0, \cdots, 1)$.

The above algorithm appears to be a natural generalization of the symmetric Lanczos algorithm to the unsymmetric case: instead of one three-term recurrence in the symmetric Lanczos algorithm we obtain here two three-term recurrences of the same type. Furthermore, for a symmetric matrix A the formulation of Algorithm 1 reduces to a version of the symmetric Lanczos algorithm. The actual implementation used in Algorithm 1 corresponds to a modified Gram-Schmidt orthogonalization with respect to the past two vectors. A detailed error analysis of the new algorithm would be beyond the scope of this paper. Because of its formal similarity, we believe that results by Paige [12] will generalize. Other results of recent investigations of the symmetric Lanczos algorithm such as the global behavior of the roundoff errors (loss of orthogonality) [12], [23], and the application of reorthogonalization methods [17], [22] probably can be carried over to the new algorithm as well.

Because of these similarities we might suspect that our new algorithm is closely related, if not identical, to the application of the symmetric Lanczos algorithm to the matrix of the normal equations. However, the distinct character of the tridiagonalization algorithm is apparent from the subspaces involved in the computation. From (2.6) it follows for $k = 1, 2, \cdots$ that

$$p_{2k} \in \text{span}\,(b, AA^*b, \cdots, (AA^*)^{k-1}b, Ac, AA^*Ac, \cdots, (AA^*)^{k-1}Ac),$$

$$p_{2k+1} \in \text{span}\,(b, AA^*b, \cdots, (AA^*)^{k}b, Ac, AA^*Ac, \cdots, (AA^*)^{k-1}Ac),$$

$$q_{2k} \in \text{span}\,(c, A^*Ac, \cdots, (A^*A)^{k-1}c, A^*b, A^*AA^*b, \cdots, (A^*A)^{k-1}A^*b),$$

$$q_{2k+1} \in \text{span}\,(c, A^*Ac, \cdots, (A^*A)^{k}c, A^*b, A^*AA^*b, \cdots, (A^*A)^{k-1}A^*b).$$

The underlying subspaces are therefore not Krylov subspaces. They can be viewed as the union of two Krylov subspaces generated with the normal equations matrices AA^* (or A^*A) with starting vectors b and Ac (or c and A^*b). If $b = c$ then these spaces can also be seen as modified Krylov subspaces in the sense that a multiplication by A is followed by a multiplication by A^*. Thus the subspaces span (Q_{2k}) and span (Q_{2k+1}) contain the Krylov subspace generated by k steps of the symmetric Lanczos algorithm applied to the normal equations. In addition they contain the space spanned by the intermediate vectors obtained if the matrix of the normal equations is not formed explicitly, but the multiplications by A and A^* are carried out in sequence.

3. Solving linear systems. We now consider the application of the tridiagonalization algorithm from the previous section to the solution of linear systems of the form

$$Ax = b, \tag{3.1}$$

where A is a general, unsymmetric $n \times n$ matrix.

Let x_0 be an initial approximation to the solution of (3.1); define $r_0 = b - Ax_0$, $\beta_1 = \|r_0\|$, $p_1 = r_0/\beta_1$; choose an arbitrary unit vector q_1, and start the tridiagonalization

algorithm. After j steps of Algorithm 1 it is possible to define an approximate solution to (3.1) as follows: let x_j^{cg} be the vector from the affine subspace $x_0+\text{span}(Q_j)$, such that the residual vector $r_j^{cg}=b-Ax_j^{cg}$ is orthogonal to span (P_j). This method falls into the class of oblique projection methods previously considered by Saad [18], [19] and Saad and Schultz [21]. It is easy to show that x_j^{cg} can be computed by:

(3.2)
$$\begin{aligned}&\text{Solve the } j\times j \text{ tridiagonal linear system } T_j h_j^{cg}=\beta_1 e_1.\\ &\text{Set } x_j^{cg}=x_0+Q_j h_j^{cg}.\end{aligned}$$

The superscript cg stands for conjugate gradients. Indeed, for a symmetric positive-definite matrix A the approximate solution computed from (3.2) is identical to the approximate solution vector computed after j steps of the conjugate-gradient method. This follows from the fact that our algorithm reduces to the symmetric Lanczos algorithm for a symmetric matrix A, and the equivalence of the Lanczos algorithm and the conjugate-gradient method for positive-definite matrices (see [13]). Formula (3.2) is the obvious generalization of the construction of a solution vector in the symmetric Lanczos algorithm (see [16]) to the unsymmetric case.

For the residual vector r_j^{cg} we obtain by using (2.6):

(3.3)
$$\begin{aligned}r_j^{cg}&=b-Ax_j^{cg}=r_0-AQ_j h_j^{cg}\\ &=\beta_1 p_1-(P_j T_j h_j^{cg}+\beta_{j+1}p_{j+1}e_j^* h_j^{cg})\\ &=-\beta_{j+1}(e_j^* h_j^{cg})p_{j+1}.\end{aligned}$$

From (3.3) we can draw two conclusions. First, it is possible to compute $\|r_j^{cg}\|$ without computing x_j^{cg}. By taking norms in (3.3) we obtain

(3.4)
$$\|r_j^{cg}\|=\beta_{j+1}|e_j^* h_j^{cg}|.$$

The practical implementation of the computation of x_j^{cg} via the LQ factorization of T_j will be discussed in § 5. We will call the resulting method USYMLQ.

Methods that are guaranteed to reduce some measure of the error at every step are sometimes preferable in practical situations to methods such as the above that satisfy only a Galerkin condition $p^*(b-Ax_j)=0$ for all vectors $p\in\text{span}(P_j)$ (see also the numerical results in [4]). We therefore want to consider a different approximate solution vector which can also be constructed easily from span (Q_j).

An approximate solution vector x_j^{cr} that minimizes the residual norm can be obtained as follows: let S_j be the $(j+1)\times j$ matrix obtained by augmenting T_j by the extra row $\beta_{j+1}e_j^*$, i.e.,

(3.5)
$$S_j=\begin{bmatrix}T_j\\ \beta_{j+1}e_j^*\end{bmatrix}.$$

Let h_j^{cr} be the solution to the least-squares problem

(3.6a)
$$\min_h \|S_j h-\beta_1 e_1\|$$

(e_1 is a $j+1$-vector), and let

(3.6b)
$$x_j^{cr}=x_0+Q_j h_j^{cr}.$$

Then we have the following theorem.

THEOREM 2. *If x_j^{cr} is determined by* (3.6), *then*

(3.7)
$$\|r_j^{cr}\|=\|b-Ax_j^{cr}\|=\min_q \|b-A(x_0+q)\|,$$

where $q\in\text{span}(Q_j)$.

Proof. If q is an arbitrary vector from span (Q_j), then $q = Q_j h$ for some $h \in R_j$. Also, from (2.6a) and (3.5) we have $AQ_j = P_{j+1} S_j$. Hence

$$\begin{aligned} \|b - A(x_0 + q)\| &= \|r_0 - AQ_j h\| \\ &= \|\beta_1 p_1 - P_{j+1} S_j h\| \\ &= \|P_{j+1}(\beta_1 e_1 - S_j h)\| \\ &= \|\beta_1 e_1 - S_j h\|, \end{aligned}$$

which by definition is minimized by the choice $h = h_j^{\mathrm{cr}}$. □

The superscript cr refers to the conjugate residual method, to which the above method reduces for symmetric A. Theorem 2 also implies that the residuals are monotonically decreasing, i.e., that for $j = 1, 2, 3, \cdots$

$$\|r_{j+1}^{\mathrm{cr}}\| \leqq \|r_j^{\mathrm{cr}}\|. \tag{3.8}$$

The least-squares problem (3.6a) is numerically best solved by an orthogonal factorization of S_j. This orthogonal factorization can be combined with an updating procedure to make x_j^{cr} directly available at each step. The resulting algorithm USYMQR for the computation of x_j^{cr} is discussed in detail in § 5.

It should be mentioned that both USYMLQ and USYMQR can simultaneously compute an approximate solution y_j to the transposed problem

$$A^* y = c. \tag{3.9}$$

Given an initial guess y_0, all that is needed is to set $\gamma_1 = \|c - A^* y\|$ and $q_1 = (c - A^* y_0)/\gamma_1$ instead of an arbitrary q_1. Then after j steps the approximate solution y_j^{cr} to (3.9) is defined by

$$\begin{aligned} &\text{Solve the least-squares problem } \tilde{S}_j h_j = \gamma_1 e_1 \text{ with } \tilde{S}_j = \begin{bmatrix} T_j^* \\ \gamma_{j+1} e_j^* \end{bmatrix}. \\ &\text{Set } y_j^{\mathrm{cr}} = y_0 + P_j h_j. \end{aligned} \tag{3.10}$$

By reversing the roles of P_j and Q_j and transposing all other quantities we can show that the associated residual norms are also monotonically decreasing. An approximate solution y_j^{cg} can be computed in the obvious way using T_j^*.

A final question to be discussed is the possible breakdown of the algorithm, i.e., the situation where $q_{j+1} = 0$ (or $p_{j+1} = 0$). The proof of Theorem 1 shows that the tridiagonalization can be continued, but we show that there is no need when solving linear systems, so that we can talk about a "lucky breakdown."

THEOREM 3. *If* $p_{j+1} = 0$, *then* USYMLQ *and* USYMQR *terminate with* $x_j^{\mathrm{cg}} = x_j^{\mathrm{cr}} = x$, *the exact solution to equation* (3.1).

Proof. If $p_{j+1} = 0$, then we also have $\beta_{j+1} = 0$ and the last row of S_j is zero. Hence the least-squares solution h_j to (3.6a) is identical to the solution of the linear system $T_j h_j = \beta_1 e_1$ as computed in (3.2). It follows that $x_j^{\mathrm{cr}} = x_j^{\mathrm{cg}}$ and hence by (3.4), $\|r_j^{\mathrm{cg}}\| = 0$, i.e., x_j is the exact solution. □

Theorem 3 guarantees that a breakdown of the type $p_{j+1} = 0$ implies that the exact solution has been found. If a breakdown with $q_{j+1} = 0$ occurs, the exact solution to the transposed system has been found.

Since the tridiagonalization algorithm terminates after at most n steps for an $n \times n$ linear system we have the following corollary.

COROLLARY. USYMLQ *and* USYMQR *determine the exact solution to* (3.1) *in at most n steps.*

Hence the algorithm USYMQR extends the three desirable properties of finite termination, a three-term recurrence, and the minimization property to the unsymmetric case. USYMLQ, in general, has no obvious minimization property; however, we show in the next section that USYMLQ does minimize a certain norm of the error for a special choice of the starting vector.

4. Choice of starting vector and relation to other methods. Theorem 3 in the previous section indicates that a good choice of the starting vector is $p_1 = q_1 = r_0/\|r_0\|$. If we choose q_1 arbitrarily we may encounter the situation that the transposed system can be solved before the original system and thus an unwanted breakdown would occur. As with the symmetric Lanczos algorithm, however, an exactly zero β_{j+1} is unlikely to occur in practice. Theorem 3 is therefore primarily of theoretical interest.

We might ask whether other choices for q_1 would lead to desirable properties of USYMLQ and USYMQR. For example, since p_1 and q_1 together with A determine T_j uniquely, we might try to determine q_1 such that T_j is symmetric. This is at least theoretically possible since we have the following.

THEOREM 4. *If* $q_1 = \sqrt{A^*A}\, A^{-1} p_1$, *then the tridiagonalization algorithm yields a symmetric matrix* T. *Furthermore, the approximate solution* x_j^{cg} *computed by* (3.6) *minimizes the* $\sqrt{A^*A}$*-norm of the error over the affine subspace* $x_0 + \text{span}\,(Q_j)$.

Proof. Let the singular value decomposition of A be given by

$$A = UDV^*, \tag{4.1}$$

where D is a diagonal matrix and U and V are orthogonal. Then define the symmetric matrices $\sqrt{A^*A} = VDV^*$ and $\sqrt{AA^*} = UDU^*$. Now consider the tridiagonalization of $\sqrt{A^*A}$ by the symmetric Lanczos algorithm with starting vector q_1. We obtain (see [15])

$$\sqrt{A^*A}\, Q_j = Q_j T_j + \beta_{j+1} q_{j+1} e_j^*, \tag{4.2}$$

with T_j symmetric. Hence,

$$VDV^* Q_j = Q_j T_j + \beta_{j+1} q_{j+1} e_j^*,$$

$$UDV^* Q_j = UV^* Q_j T_j + \beta_{j+1} UV^* q_{j+1} e_j^*,$$

$$AQ_j = P_j T_j + \beta_{j+1} p_{j+1} e_j^*,$$

where $p_k = UV^* q_k$, for $k = 1, 2, \cdots, j+1$. This is precisely (2.6a) with a symmetric T_j. In particular, we have

$$q_1 = VU^* p_1 = \sqrt{A^*A}\, A^{-1} p_1.$$

Thus because of the uniqueness of the tridiagonalization process we obtain a symmetric T_j for an arbitrary p_1, if q_1 is chosen as above.

Since $\sqrt{A^*A}$ is symmetric positive-definite the solution x_j^{cg} computed using (4.2) is the conjugate-gradient solution to

$$\sqrt{A^*A}\, x = \beta_1 q_1 = \beta_1 \sqrt{A^*A}\, A^{-1} p_1,$$

i.e., to $Ax = \beta_1 p_1$. Because of the minimization property of the conjugate-gradient algorithm the second part of the theorem follows as well. □

Obviously Theorem 4 has little practical value since the computation of $\sqrt{A^*A}\, A^{-1} p_1$ is more difficult than the simple solution of the original linear system. Theorem 4, however, does suggest two things: First, an approximation to $\sqrt{A^*A}\, A^{-1} p_1$ might be a good second starting vector, although we do not know of any method to compute such an approximation cheaply. Second, the algorithm USYMLQ contains

at least implicitly the "natural" extension of conjugate gradients to the unsymmetric case. Unfortunately, this natural extension based on the operator $\sqrt{A^*A}$ is computationally impractical.

The tridiagonalization process considered here is also closely related to the Golub-Kahan [7] bidiagonalization. Consequently, USYMLQ and USYMQR show some similarities to methods derived from the bidiagonalization process such as LSQR [14]. (For a survey of other mathematically equivalent methods, see [14].) All these methods are based, however, on applying conjugate gradients to the normal equations, whereas USYMLQ uses a different approach. The difference becomes clearer if we consider the augmented symmetric $2n \times 2n$ matrix

$$B = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}.$$

The Golub-Kahan bidiagonalization is mathematically equivalent to the symmetric Lanczos process applied to B with the starting vector $\begin{bmatrix} r_0 \\ 0 \end{bmatrix}$. Because of the special structure of the matrix and starting vector, the last n components of the odd Lanczos vectors and the first n components of the even Lanczos vectors are zero. The nonzero components of the even and of the odd Lanczos vectors thus form two sequences of orthogonal vectors which are used analogously to P and Q in USYMLQ for the computation of an approximate solution to (3.1).

In contrast, we consider in USYMLQ a starting vector of the form $u_1 = \begin{bmatrix} r_0 \\ r_0 \end{bmatrix}$. With this starting vector no similar pattern of zero entries will be encountered in the sequence of vectors $B^k u_1, k = 1, 2, \cdots$. Furthermore, our tridiagonalization algorithm is equivalent to constructing a sequence of vectors such that the top n and the bottom n components separately form sequences of orthogonal vectors. These vectors are thus different from the top and bottom half of the Lanczos vectors computed with B and u_1. So although we are using the operator B in a particular way, we do not apply conjugate gradients to the normal equations. Here we use extra information from the intermediate nonzero components in the quasi-Krylov vectors. This information is not used in the normal equations, where these components are zero. Methods based on the use of the operator B with a general starting vector are discussed by Jea and Young [8].

Another interesting observation (due to Beresford Parlett) relates some of the subspaces used here to block Krylov subspaces obtained with the symmetric block Lanczos algorithm. For example, we have

$$\text{span}\,(Q_{2k}) = \text{span}\,(F, A^*AF, \cdots, (A^*A)^{m-1}F),$$

where $F = (c, A^*b)$. Hence the same subspace as in the tridiagonalization algorithm is generated by a block Lanczos algorithm for A^*A with starting block (c, A^*b). The block Lanczos algorithm would generate a symmetric pentadiagonal matrix. It can be shown that this matrix is equal to $T_{2k}^* T_{2k}$. This relationship opens a variety of new possibilities to consider. Here we restrict ourselves to observing that there is no easy direct formula to relate the new tridiagonalization algorithm to the symmetric Lanczos algorithm on B.

It is also possible to show the following result (for a proof, see [2]).

THEOREM 5. *If m is the number of distinct singular values of A, then* USYMLQ *and* USYMQR *converge in at most* $\min(2m, n)$ *steps to the exact solution, assuming exact arithmetic.*

It should be noted that if $m < n$ this result is actually worse than a corresponding theorem for the normal equations, which would compute the exact solution in at most

m steps, with a comparable amount of work per step. This theoretical limit indicates potential disadvantages of the new algorithms in comparison to LSQR [14], which is based on the normal equations approach.

Finally we want to point out that the singular values of T_j approximate the singular values of A. It is thus conceivable to use the tridiagonalization algorithm for the computation of the singular values of large matrices in a manner analogous to [11] or [1].

5. The implementation of USYMLQ and USYMQR. The practical implementations of USYMLQ and USYMQR are based on ideas used in [13] for the derivation of the algorithms SYMMLQ and MINRES for symmetric indefinite matrices.

Let us first consider the solution of the tridiagonal system (3.2)

$$T_j h_j^{cg} = \beta_1 e_1$$

via the LQ factorization of T_j, i.e., let

$$T_j = L_j U_j, \tag{5.1}$$

where U_j is a $j \times j$ orthogonal matrix and the lower triangular matrix L_j has the form

$$L_j = \begin{bmatrix} \delta_1 & 0 & 0 & \cdots & 0 \\ \lambda_1 & \delta_2 & 0 & \cdots & 0 \\ \varepsilon_1 & \lambda_2 & \delta_3 & \cdots & 0 \\ & \ddots & \ddots & \ddots & \vdots \\ 0 & \varepsilon_{j-3} & \lambda_{j-2} & \delta_{j-1} & 0 \\ 0 & \cdots & \varepsilon_{j-2} & \lambda_{j-1} & \delta_j \end{bmatrix}.$$

The matrix U_j is a product of plane rotations designed to eliminate the superdiagonal elements $\gamma_2, \gamma_3, \cdots, \gamma_j$. The tridiagonal system becomes

$$T_j h_j^{cg} = L_j U_j h_j^{cg} = \beta_1 e_1.$$

We note that h_j^{cg} has no elements in common with h_{j-1}^{cg}. However, it is not really h_j^{cg} that we are interested in, but rather x_j^{cg} as defined by equation (3.2). If $z_j = U_j h_j^{cg}$ and $W_j = Q_j U_j^*$, then $x_j^{cg} = x_0 + W_j z_j$. From here on we can proceed as in [13], and compute z_j and the columns of W_j by simple recursions. We omit the details, which can be found in the literature [13].

The resulting algorithm USYMLQ can be implemented using only six n-vectors of storage: four for the generalized Lanczos vectors in the tridiagonalization algorithm and one each for x and w. This assumes that the right-hand side vector is overwritten and that both the multiplication of a vector by the matrix A and its transpose A^* can be performed as an update of the form $y \leftarrow y + Ax$. A total of $13n$ operations are needed per step plus the demands of the matrix vector multiplications. The tridiagonalization requires $9n$ operations and the updates to x and w require $4n$ operations. In a more careful implementation we might recompute α in two different ways, and thus obtain $14n$ operations per step.

Let us now consider the solution of the least-squares problem (3.6):

$$S_j h_j^{cr} = \beta_1 e_1.$$

Using the QR factorization of S_j, let

$$S_j = V_{j+1} \begin{bmatrix} R_j \\ 0 \end{bmatrix},$$

where V_{j+1} is a $j+1 \times j+1$ orthogonal matrix and the upper triangular matrix R_j is actually of upper trapezoidal form, with three nonzero diagonals.

The reduction of S_j to upper triangular form is achieved by a sequence of plane rotations which at each step reduce the β_{j+1} entry in S_j to zero. The orthogonal matrix V_{j+1} is the product of these rotations. Now define the $n \times j$ matrix $M_j = Q_j R_j^{-1}$. Then x_j^{cr} can be computed by a simple updating process, using the columns of M_j. The columns of M_j in turn can be computed by a three-term recurrence. Again we omit the details, which are analogous to the implementation of MINRES in [13]. The resulting algorithm is called USYMQR.

This implementation of USYMQR needs $12n$ operations per step in addition to the requirements for computing Ap and A^*q. The double Lanczos recurrence requires $9n$ operations, and the updates to the direction vector m and the solution vector x require $2n$ and n operations, respectively. Compared to USYMLQ, one extra n-vector of storage is needed for the update of m; hence the total storage requirement is seven n-vectors.

Although the operation count per step for both USYMLQ and USYMQR appears high in comparison to methods for symmetric matrices, it is comparable to other methods for unsymmetric matrices. For example, if the matrix A is the five-point discrete Laplacian, then the number of operations per step for USYMLQ is about the same as for ORTHOMIN(5) [24]. On the other hand, USYMLQ requires storage for only six vectors of length n, whereas ORTHOMIN(5) would require 13 n-vectors. Generally speaking all comparisons made here are highly hypothetical, since we do not know how the rates of convergence of these methods compare.

6. Numerical results. The following three numerical examples demonstrate that both USYMLQ and USYMQR are viable numerical methods for the solution of unsymmetric linear systems. They also shed some light on specific areas of applications for which the use of USYMLQ and USYMQR may be particularly advantageous. For Examples 1 and 2 no preconditioning was used. In Example 3 we used an incomplete LU factorization of the coefficient matrix as preconditioner. All results were obtained on a Cray-1S, which provides about 13 decimal digits of accuracy.

Example 1. This is a model problem, which allows us to study the influence of unsymmetry on the solution behavior of some iterative methods. It has been used before by Saad [18] and others (see references in [18]). The matrix A is given by the block tridiagonal matrix

$$A = \begin{bmatrix} B & -I & & & \\ -I & B & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & B & -I \\ & & & -I & B \end{bmatrix}$$

with

$$B = \begin{bmatrix} 4 & a & & & \\ b & 4 & a & & \\ & \ddots & \ddots & \ddots & \\ & & b & 4 & a \\ & & & b & 4 \end{bmatrix}$$

and $a = -1 + \delta$, $b = -1 - \delta$.

The parameter δ controls the symmetry of this family of matrices. We considered the problem where $n = 400$ and the dimension of B is 20. The components of the

solution vector were chosen to be uniformly distributed random numbers between 0 and 1. We solved the resulting linear system of equations with the methods ORTHOMIN(5), GCR(5), and LSQR [24], [4], [20], [14] as well as with USYMLQ and USYMQR. For each method the iteration was stopped when the residual norm was reduced by a factor of 10^{-6}. For various δ the results in Table 1 were obtained.

The number of iteration steps does not provide full information on the amount of computational work of these methods, since the number of operations per step varies. Therefore we also list the total number of floating-point operations in Table 2.

Obviously USYMLQ and USYMQR become more competitive the more unsymmetric the matrix becomes. However, the same is true for LSQR, and for this example LSQR is more efficient than either USYMLQ or USYMQR for all δ. On the other hand, for a symmetric matrix, all methods perform about the same, although LSQR takes twice as many steps as the other methods.

The most interesting behavior can be observed for a nearly symmetric matrix ($\delta = 0.01$). A small perturbation away from symmetry increases the number of iteration steps considerably. If the coefficient matrix is symmetric, USYMLQ and USYMQR reduce to SYMMLQ and MINRES from [13]. Then the equivalent to Theorem 5 suddenly says that the number of steps is at most $\min(m, n)$. Hence USYMLQ/SYMMLQ and USYMQR/MINRES exploit symmetry in a special way. Thus for nearly symmetric matrices both USYMLQ and USYMQR are relatively inefficient.

It is interesting to note that the break-even point between the ORTHOMIN-type methods and USYMLQ and USYMQR occurs for a value of δ for which $(A - A^*)/2$, the unsymmetric part of the matrix, becomes comparable in size to the symmetric part. A further investigation of this observation, especially the precise relation between symmetry and convergence behavior, appears to be a worthwhile research topic.

In this numerical example we did not attempt to find the optimal values for K1 and K2 such that the methods ORTHOMIN (K1) and GCR (K2) would deliver the solution with the required accuracy in a minimal number of operations. Other numerical

TABLE 1
Number of iteration steps required to achieve the requested accuracy.

δ	0.0	0.01	0.1	1.0	10.0	100.0
ORTHOMIN (5)	33	59	77	85	258	214
GCR (5)	33	59	80	76	176	>400
LSQR	70	173	185	100	71	33
USYMLQ	32	207	215	154	107	71
USYMQR	33	206	216	154	108	70

TABLE 2
Number of floating-point operations (in millions) to achieve the requested accuracy.

δ	0.0	0.01	0.1	1.0	10.0	100.0
ORTHOMIN (5)	0.31	0.57	0.75	0.83	2.55	2.11
GCR (5)	0.27	0.49	0.66	0.63	1.46	>3.33
LSQR	0.28	1.38	1.48	0.80	0.57	0.27
USYMLQ	0.30	1.90	1.98	1.42	0.99	0.66
USYMQR	0.30	1.88	1.98	1.41	0.99	0.64

results [3], [20] show that a different choice of these parameters can influence the convergence behavior of the ORTHOMIN-type methods considerably. The choice of these parameters is also very significant for the practical performance of these methods, since it determines storage requirements and the number of arithmetic operations per step. It is an advantage of USYMLQ and USYMQR (and LSQR) to be independent of the choice of any parameter.

The convergence for ORTHOMIN (K) and GCR (K) can only be shown for matrices with a positive-definite symmetric part [3], [20]. For Example 1 this was the case independent of δ. If the diagonal entries of the matrix A are set to two instead of four, the symmetric part of this matrix is no longer positive-definite. For $\delta = 1.1$ the following results are obtained: ORTHOMIN (5) and GCR (5) stagnate and are unable to achieve the requested reduction in the residual norm, whereas USYMLQ, USYMQR, and LSQR converge in 102, 101, and 86 steps, respectively. Hence these methods behave well also in the presence of an indefinite symmetric part of the matrix.

Example 2 (see [3, p. 135]). Consider the matrix arising from the discretization of the elliptic partial differential equation

$$Lu = f \qquad \text{with } Lu = -(Bu_x)_x - (Cu_y)_y + Eu_y + (Eu)_y + Fu$$

on the unit square with homogeneous boundary conditions, using a five-point centered finite difference scheme on a uniform $N \times N$ grid, with $h = (N+1)^{-1}$. Here $B(x, y) = e^{-xy}$, $C(x, y) = e^{xy}$, $E(x, y) = \theta(x+y)$, and $F(x, y) = (1+x+y)^{-1}$. We solved the resulting linear system of order 324 (i.e., $N = 18$) and obtained the following results.

Case (a). $\theta = 10.0$ with a right-hand side chosen such that the solution is $(1, 1, \cdots, 1)^T$:

ORTHOMIN (5)	stagnates after 10 steps with a relative residual of 3.32×10^{-2},
GCR (5)	stagnates after 10 steps with a relative residual of 3.48×10^{-2},
USYMLQ	reduces the residual norm by a factor of 10^{-6} after 33 steps,
USYMQR	reduces the residual norm by a factor of 10^{-6} after 34 steps.

Case (b). $\theta = 50.0$ with a right-hand side chosen as in [3].

Here all methods achieved the required reduction of the residual norm by a factor of 10^{-6}. The parameters for the ORTHOMIN-type methods were chosen such that the amount of work per step is about the same for all methods listed. The necessary number of steps was

ORTHOMIN (4)	83 steps,
GCR (9)	88 steps,
GMRES (14)	90 steps,
USYMLQ	168 steps,
USYMQR	167 steps.

Again Cases (a) and (b) can be considered as opposite extremes. In Case (a) none of the ORTHOMIN-type methods converged to the required accuracy, whereas USYMLQ and USYMQR had no difficulty computing the solution. In Case (b) USYMLQ and USYMQR took about twice as long to find the solution as the ORTHOMIN-type methods,which all performed about the same. Thus USYMLQ and USYMQR offer the advantage of guaranteed convergence at the price of a possible inefficiency for the cases where ORTHOMIN-type methods do perform well.

Example 3. Both ORTHOMIN and USYMLQ have been implemented in a code for the solution of transonic flow problems. The details of this work are reported in [2]. Here we present an extreme example of the behavior of both methods, which confirms the observations made above.

ORTHOMIN (5) and USYMLQ were used in the inner loop of an inexact Newton-type iteration for the nonlinear problem. The initial guess for the outer Newton iteration was subsonic, and the final solution corresponded to a transonic state. In Table 3 we list for a problem of order 1536 the number of inner iteration steps required by both ORTHOMIN (5) and USYMLQ in the course of one Newton iteration.

TABLE 3
Number of iteration steps in a Newton iteration for a transonic flow problem.

	ORTHOMIN (5)	USYMLQ
Newton step 1	6	27
Newton step 2	27	71
Newton step 3	stagnation	97

ORTHOMIN (5) fails in the difficult transition from subsonic to transonic flow, which occurs during Newton step 3.

Some related numerical experiments with BIORTH, a mathematically equivalent implementation of USYMQR, are reported in [10]. On some large problems derived from the discretization of three-dimensional elliptic partial differential equations BIORTH was compared to conjugate-gradients on the normal equations and to biconjugate-gradients. On these problems the USYMQR equivalent method did not perform as well as either of its competitors.

7. Conclusions. The numerical results show that USYMLQ and USYMQR are not only theoretically interesting new methods, but also provide a valuable tool for solving sparse unsymmetric linear systems. Since USYMLQ and USYMQR are working with the same subspace, we did not observe any significant differences in the convergence behavior of USYMLQ and USYMQR, as was to be expected. Comparing both methods to ORTHOMIN-type methods we observed that USYMLQ and USYMQR

- have better convergence behavior than ORTHOMIN (K) and related methods for strongly unsymmetric problems or problems with indefinite symmetric part,
- have the advantage of not requiring the choice of any parameters, but
- require multiplication by the transposed matrix,
- are less efficient than ORTHOMIN (K)-type methods for nearly symmetric problems, and problems with positive-definite symmetric part.

Unfortunately, we were not able to find a type of problem in the spectrum of unsymmetric sparse matrices for which the new methods offer clear performance advantages. For nearly symmetric problems, and problems with symmetric positive-definite part, Krylov subspace-type methods remain superior. For strongly unsymmetric problems, LSQR achieves about the same results as USYMLQ and USYMQR, but does so usually with a smaller number of operations. Further research into USYMLQ and USYMQR thus may be only of theoretical interest.

Acknowledgments. The first author is grateful to Åke Björck for his interest and for providing computing facilities for initial experiments during a visit to Linköping

University, May–June 1981. The second author would like to thank Howard Elman, Stan Eisenstat, Youcef Saad, and Martin Schultz for stimulating discussions on iterative methods. Roland Freund, Beresford Parlett, and David Young read an earlier version of this report and suggested several improvements. We also thank Nicholas Huslak for performing part of the programming work.

REFERENCES

[1] J. CULLUM, R. A. WILLOUGHBY, AND M. LANE, *A Lanczos algorithm for computing the singular values and vectors of large matrices*, SIAM J. Sci. Statist Comput., 4 (1983), pp. 197–215.

[2] F. E. EHLERS, W. H. WEATHERILL, AND E. L. YIP, *Development and application of algorithms for calculating the transonic flow about harmonically oscillating wings*, Tech. Report, The Boeing Commercial Airplane Company, Seattle, WA, October 1983.

[3] H. C. ELMAN, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. thesis, Dept. of Comp. Science, Yale University, 1982.

[4] ———, *Iterative methods for non-selfadjoint elliptic problems*, in Elliptic Problem Solvers II, G. Birkhoff and A. Schoenstadt, eds., Academic Press, New York, 1984, pp. 271–284.

[5] 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.

[6] G. H. GOLUB, SIGNUM Newsletter 16, No. 4, 1981, p. 7.

[7] G. H. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[8] K. C. JEA AND D. M. YOUNG, *On the simplification of generalized conjugate gradient methods for nonsymmetrizable linear systems*, Linear Algebra Appl., 52 (1983), pp. 399–417.

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

[10] H. MÜLLER, W. SCHÖNAUER, AND E. SCHNEPF, *Design considerations for the linear solver LINSOL on a Cyber* 205, Report, Rechenzentrum der Universität Karlsruhe, West Germany, 1985.

[11] C. C. PAIGE, *Bidiagonalization of matrices and solution of linear equations*, SIAM J. Numer. Anal., 11 (1974), pp. 197–209.

[12] ———, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, J. Inst. Math. Appl., 18 (1976), pp. 341–349.

[13] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[14] ———, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM TOMS, 8 (1982), pp. 43–71.

[15] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[16] ———, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 323–346.

[17] B. N. PARLETT AND D. SCOTT, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 217–238.

[18] Y. SAAD, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1980), pp. 105–126.

[19] ———, *Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 203–228.

[20] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[21] ———, *Conjugate gradient like algorithms for solving nonsymmetric linear systems*, Math. Comp., 44 (1985), pp. 417–424.

[22] H. D. SIMON, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–136.

[23] ———, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods*, Linear Algebra Appl., 61 (1984), pp. 101–131.

[24] P. K. W. VINSOME, *Orthomin, an iterative method for solving sparse sets of simultaneous linear equations*, Proc. Fourth Symposium of the Society of Petroleum Engineers, Los Angeles, CA, 1976, pp. 149–159.

[25] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon, Oxford, 1965.