

## SOLUTION OF SPARSE INDEFINITE SYSTEMS OF LINEAR EQUATIONS\*

C. C. PAIGE† AND M. A. SAUNDERS‡

**Abstract.** The method of conjugate gradients for solving systems of linear equations with a symmetric positive definite matrix  $A$  is given as a logical development of the Lanczos algorithm for tridiagonalizing  $A$ . This approach suggests numerical algorithms for solving such systems when  $A$  is symmetric but indefinite. These methods have advantages when  $A$  is large and sparse.

**1. Introduction.** Here some methods are considered for solving

$$(1.1) \quad Ax = b$$

when the  $n \times n$  real symmetric matrix  $A$  is large and sparse. Unlike matrix factorization, the methods given here for solving (1.1) regard  $A$  as an operator and only require matrix-vector products, building up  $x$  as a combination of vectors derived from a Krylov sequence. Some basic theory for different methods of this type is given in § 2.

One of the best known examples of this type is the conjugate gradients method (CGM) [2], which can be used to solve positive definite systems. Reid [11] has indicated that CGM can be very accurate and fast for several problems even though rounding errors cause it to depart significantly from its ideal path. In the experience of the present authors, CGM can be relied upon to converge ultimately, and so is very effective if regarded as an iterative method rather than an  $n$  step process. In fact CGM often gives the solution to the required accuracy in very much less than  $n$  steps.

CGM can break down for  $A$  having both positive and negative eigenvalues, and there is a need for methods which can handle large sparse indefinite symmetric matrices. Luenberger [7], [8] examined the possibility of two methods related to CGM for indefinite matrices; unfortunately his methods encountered some unresolved computational difficulties. The method of minimized iterations developed by Lanczos [5] is closely related to CGM, and Fridman [1] extended this approach to indefinite symmetric matrices. This method is good in theory, but no computational experience is mentioned in [1], and it is the experience of the present authors that the method will usually not converge on large problems.

Both the CGM and the method of minimized iterations are directly related to the very basic algorithm developed by Lanczos [4] for tridiagonalizing  $A$ , as is explained in [3]. The Lanczos process does not require  $A$  to be definite and so is a good starting point for developing algorithms for solving (1.1) with indefinite  $A$ . A description of the Lanczos process is given in § 3, and the method of conjugate gradients is developed from it in § 4. This gives computational insights

---

\* Received by the editors December 21, 1973, and in revised form September 16, 1974.

† School of Computer Science, McGill University, Montreal, Canada. This work was supported in part by National Research Council of Canada Grant A8652 and National Science Foundation Grant GJ 29988X.

‡ Applied Mathematics Division, D.S.I.R., Wellington, New Zealand. This work was supported in part by AEC Grant AT(04-3)-326 PA # 30.

into the method and leads to two new algorithms that may be used when  $A$  is indefinite; these are described in §§ 5 and 6. The method in § 6 can also be used if  $A$  is singular and (1.1) is not a consistent set of equations, and some properties of this method are developed in § 7.

When linear least squares problems are put in the form (1.1), as in [12], the symmetric matrix  $A$  will be indefinite with some zero subblocks. If these problems are large and sparse, then the new methods given here could be used. When there are no constraints the algorithms can be simplified to take advantage of the special form of  $A$ , saving storage and computation. This is described in [10], which also contains FORTRAN subroutines for both indefinite symmetric systems and the unconstrained least squares problem. However, as a better method has been found for least squares which is not a direct simplification of the symmetric case, it will be treated in a separate paper.

Computational results for the new algorithms are discussed in § 8, and these indicate that they give accurate results, often in much less than  $n$  steps. A rounding error analysis of algorithms of this type will be given in a later report.

The methods given here for symmetric indefinite systems would appear to be superior to those in [1], [7], [8], as the latter present some difficult unsettled problems when routine practical application is considered. These particular problems do not arise here, since the particular development of the algorithms from the Lanczos process allows a good understanding of their numerical properties and so some possible numerical instabilities have been avoided.

In the text upper case italic letters denote matrices, lower case italic denote vectors and lower case Greek denote scalars. The exceptions are  $c$  and  $s$  (not used as a superscript), which denote cosine and sine. The symbol  $\|\cdot\|$  denotes the 2-norm of a vector or matrix.

## 2. General theory. Given the set of equations

$$(2.1) \quad r + Ax = b, \quad Ar = 0,$$

where  $A$  is a real  $n \times n$  symmetric matrix which may be both indefinite and singular, we will consider computing various approximations to  $x$  of the form  $V_k y$ ,  $V_k \equiv [v_1, v_2, \dots, v_k]$ , where the  $v_i$  are a given set of linearly independent vectors. In particular, we will look for solutions  $x_k \equiv V_k y_k$  which give stationary values to

$$(2.2) \quad f_k(y) \equiv (AV_k y - b)^T B (AV_k y - b),$$

where  $B$  is some symmetric matrix; thus  $f_k(y)$  will be a norm of the residual if  $B$  is positive definite. Note that this is just a theoretical tool that will lead to different methods and that  $B$  will not be required explicitly.

The function  $f_k(y)$  has a stationary value at  $y_k$  if

$$(2.3) \quad V_k^T A B A V_k y_k = V_k^T A B b,$$

that is, if

$$(2.4) \quad V_k^T A B r_k = 0, \quad r_k \equiv b - A x_k,$$

and the methods to be considered will essentially try to solve (2.3). Since the second derivative of  $f_k(y)$  is  $2V_k^T ABAV_k$ , it follows that if  $ABA$  is positive definite, there is a unique  $y$  that minimizes  $f_k(y)$ . If  $ABA$  is only positive semidefinite, then the minimizing  $y$  is not necessarily unique, while if  $ABA$  is indefinite, we only have a stationary point of  $f_k(y)$ . In any case,  $x_k$  is an approximation to the solution in the sense that the residual is restricted to the nullspace of  $V_k^T AB$ , and  $V_k$  can be chosen to reduce the dimension of this nullspace with increasing  $k$ .

An obvious choice for  $B$  is  $A^m$  for some integer  $m$ , and we will restrict ourselves to this case. Choosing  $m = -2$  would essentially require a knowledge of  $x$  on the right-hand side of (2.3), and for  $m \leq -2$ , solving (2.3) would appear to require at least as much knowledge as solving the original problem. The choices  $m = -1, 0$  appear to be the most useful and will now be considered.

Case (a). Taking  $m = -1$  would give  $B = A^{-1}$ , but to allow for the more general case of singular  $A$ , we take  $B = A^-$ , where  $A^-$  is a generalized inverse of  $A$  such that  $AA^-$  is the orthogonal projector onto  $\mathcal{R}(A)$ , the range of  $A$ . With this choice, we have from (2.1)

$$AA^-b = AA^-r + AA^-Ax = Ax,$$

and (2.3) becomes

$$(2.5) \quad V_k^T AV_k y_k = V_k^T Ax = V_k^T b - V_k^T r,$$

which cannot be solved directly for  $y_k$  unless a value for  $V_k^T r$  is known. We will consider only the case  $V_k^T r = 0$ , so the method will be applicable if  $r = 0$  or  $v_i \in \mathcal{R}(A)$ ,  $i = 1, \dots, k$ , and then

$$(2.6) \quad V_k^T AV_k y_k = V_k^T b, \quad x_k = V_k y_k$$

gives a stationary value of (2.2) with  $B \equiv A^-$ . If the columns of  $V_k$  span  $\mathcal{R}(A)$ , then we have the least squares solution of minimum length.

Case (b). Taking  $m = 0$  gives  $B = I$ , and we can minimize  $\|r_k\|$  by solving

$$(2.7) \quad V_k^T A^2 V_k u_k = V_k^T Ab, \quad x_k = V_k u_k,$$

where  $y$  has been replaced by  $u$  to avoid confusion with (2.6). Furthermore, if  $v_1, \dots, v_k$  span  $\mathcal{R}(A)$ , then  $x_k$  is the minimum length least squares solution of (2.1). We will call methods based on (2.7) minimum residual methods. A possible danger with these is that if  $A$  is poorly conditioned for solutions of equations, then the condition of the problem (2.7) can be much worse. Values of  $m > 0$  would lead to more poorly conditioned problems still and will not be examined here.

**3. The Lanczos vectors.** If the vectors  $v_1, \dots, v_k$  in §2 are computed by the Lanczos algorithm [4], then some important and computationally useful simplifications result. In particular, algorithms arise which are useful for large sparse matrices: for example the method of conjugate gradients.

The initial vector we will use in the Lanczos algorithm will be

$$(3.1) \quad v_1 = b/\beta_1, \quad \beta_1 \equiv \|b\|;$$

there are indications that this choice, and possibly  $v_1 = Ab/\|Ab\|$ , are the most computationally viable ones for solving large problems of the form (2.1).

If we have an initial approximation  $x_0$  to  $x$  in (2.1), then we change the problem to

$$r + Ag = r_0 \equiv b - Ax_0, \quad Ar = 0, \quad x = x_0 + g,$$

and proceed as before. With the choice of  $v_1$  in (3.1), we restrict ourselves to the case of  $r = 0$  in § 2, Case (a), though there is no such restriction on Case (b).

A satisfactory computational variant of the Lanczos algorithm [9] has as its  $j$ th step, defining  $v_0 \equiv 0$ ,

$$(3.2) \quad \beta_{j+1}v_{j+1} = Av_j - \alpha_jv_j - \beta_jv_{j-1}, \quad \alpha_j = v_j^T Av_j$$

with  $\beta_{j+1} \geq 0$  chosen so that  $\|v_{j+1}\| = 1$ . After the  $k$ th step,

$$(3.3) \quad AV_k = V_k T_k + \beta_{k+1}v_{k+1}e_k^T, \quad T_k \equiv \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & & & \ddots & \\ & & & & \beta_k & \alpha_k \end{bmatrix},$$

$$V_k^T V_k = I \equiv [e_1, \dots, e_k], \quad V_k^T v_{k+1} = 0.$$

The process will be terminated at the first zero  $\beta_j$ , so from now on we can assume that  $\beta_j \neq 0, j = 1, \dots, k$ .

From (3.3) and (3.1), we have  $V_k^T AV_k = T_k$  and  $V_k^T b = \beta_1 e_1$ , and with this choice of vectors in § 2 Case (a), (2.6) becomes

$$(3.4) \quad T_k y_k = \beta_1 e_1, \quad x_k^c = V_k y_k,$$

where the superscript  $c$  indicates that it is the solution that would be obtained using the method of conjugate gradients, as will now be explained.

**4. Derivation of the conjugate gradients method from the Lanczos process.** The conjugate gradients method [2] can be developed in a straightforward manner from the Lanczos process. The purpose of giving this development here is to divide the conjugate gradients method into separate computational algorithms whose numerical properties are more clearly understood; this leads to some new and useful methods.

If  $A$  is positive definite then so is  $T_k = V_k^T AV_k$  in (3.4), and hence the Cholesky factorization

$$(4.1) \quad T_k = \mathcal{L}_k \mathcal{D}_k \mathcal{L}_k^T$$

exists. Here  $\mathcal{D}_k$  is diagonal with positive elements, and  $\mathcal{L}_k$  is unit lower bidiagonal, and these can be developed as  $k$  increases. Unfortunately  $y_k$  in (3.4) changes fully with each increase in  $k$ , and so  $V_k y_k$  cannot be accumulated as  $k$  increases. This difficulty can be overcome if we define  $p_k \equiv \mathcal{L}_k^T y_k$  and  $C_k \equiv V_k \mathcal{L}_k^{-T}$  so that (3.4) becomes

$$(4.2) \quad \mathcal{L}_k \mathcal{D}_k p_k = \beta_1 e_1, \quad x_k^c = C_k p_k.$$

The columns of  $C_k$  can be found in ascending order by solving

$$(4.3) \quad \mathcal{L}_k C_k^T = V_k^T$$



where in the next step we compute

$$(5.6) \quad \gamma_k = (\bar{\gamma}_k^2 + \beta_{k+1}^2)^{1/2}, \quad c_k = \bar{\gamma}_k/\gamma_k, \quad s_k = \beta_{k+1}/\gamma_k.$$

In the following discussion we will use  $L_k$  to denote  $\bar{L}_k$  with  $\bar{\gamma}_k$  replaced by  $\gamma_k$ . Similarly, following (5.2) and (5.3), we define  $z_k \equiv (\zeta_1, \dots, \zeta_k)^T$  and  $W_k \equiv [w_1, \dots, w_k]$ , where  $z_k$  is found from

$$(5.7) \quad L_k z_k = \beta_1 e_1,$$

so from (5.4) and (5.6)

$$(5.8) \quad \zeta_k = \bar{\gamma}_k \bar{\zeta}_k / \gamma_k = c_k \bar{\zeta}_k;$$

finally from (5.2) and the form of  $Q_{k,k+1}$  in (5.5) we have

$$(5.9) \quad [\bar{w}_k, v_{k+1}] \begin{bmatrix} c_k & s_k \\ s_k & -c_k \end{bmatrix} = [w_k, \bar{w}_{k+1}], \quad \text{with } \bar{w}_1 \equiv v_1.$$

Although the rotation matrices  $Q_{i,i+1}$  allow an easy description of the decomposition of  $T_k$ , Professor B. Parlett pointed out that a saving can be made by using the fast Givens transformations introduced by Gentleman [13], [14]. Equations (5.1) and (5.2) can be combined to give

$$(\bar{L}_k^T, \bar{W}_k^T) = Q_k(T_k, V_k^T),$$

showing that  $\bar{L}_k$  and  $\bar{W}_k$  are obtained by transforming  $(T_k, V_k^T)$  to upper trapezoidal form. By using fast transformations to do this, we obtain diagonal  $\bar{D}_k$ , lower triangular  $\tilde{L}_k$ , and  $\tilde{W}_k$ , such that

$$Q_k(T_k, V_k^T) = \bar{D}_k(\tilde{L}_k^T, \tilde{W}_k^T).$$

It turns out that this use of stable 2 or 3 multiplication rotations can give a saving of about  $n$  multiplications per step over the algorithm SYMMLQ programmed in [10]. Combining the above two equations with (5.4) then gives

$$\begin{aligned} \bar{L}_k \bar{z}_k &= \tilde{L}_k \bar{D}_k \bar{z}_k = \beta_1 e_1, \\ x_k^c &= \bar{W}_k \bar{z}_k = \tilde{W}_k \bar{D}_k \bar{z}_k = \beta_1 \tilde{W}_k \tilde{L}_k^{-1} e_1. \end{aligned}$$

However, for simplicity we will continue with the description that uses ordinary rotations.

The algorithm defined by (5.1) to (5.4) should not be implemented directly, since it is wasteful to update  $x_k^c$  fully each step in (5.4), while if  $\bar{L}_k$  is singular in (5.4) then  $\bar{z}_k$  is undefined. Instead we see from (5.5) and (5.6) that  $L_k$  is nonsingular if  $\beta_{k+1} \neq 0$ , so  $z_k$  is defined in (5.7), and rather than updating  $x_k^c$  each step, we update

$$(5.10) \quad x_k^L = W_k z_k = x_{k-1}^L + \zeta_k w_k,$$

where  $L$  indicates we are using  $L_k$  rather than  $\bar{L}_k$ . Since (5.4) and (5.10) show that

$$(5.11) \quad x_{k+1}^c = x_k^L + \bar{\zeta}_{k+1} \bar{w}_{k+1},$$

we are always able to obtain  $x_{k+1}^c$  if it is needed. Because  $L_k$  has better condition than  $\bar{L}_k$ , solving (5.7) will probably also give better numerical results than solving (5.4).

In theory, the Lanczos iteration will stop with some  $\beta_{k+1} = 0$ , and then  $x_k^c = x_k^L = x$ , but in practice it is rare to have even a very small  $\beta_{k+1}$ , and some other stopping criterion must be used; here  $x_k^c$  and  $x_k^L$  will be different, and the one which gives the smaller residual would usually be chosen.  $x_k^c$  is often a much better approximation to  $x$  than  $x_k^L$ , and so (5.11) is usually carried out at the end of the iteration; there is no facility for doing this in the version of our algorithm described by Lawson [6], but it is included in the FORTRAN subroutine SYMMLQ in [10]. Note that  $W_k$  has orthonormal columns, so that if

$$(5.12) \quad d_k^L \equiv x - x_k^L, \quad d_k^c \equiv x - x_k^c,$$

then  $d_k^L$ , but not  $d_k^c$ , must decrease in 2-norm every step. Thus  $x_k^L$  is the best approximation to  $x$  lying in the space spanned by  $w_1, \dots, w_k$  and is monotonically increasing in size every step; apparently this space is usually not as good an approximation space as that spanned by  $w_1, \dots, w_{k-1}, \bar{w}_k$ .

For the algorithm using (5.7) and (5.10) to be theoretically well-defined it is still necessary to show that there is no possibility of  $L_k$  being singular. Now from the discussion following (2.5) and the choice of  $v_1$  in (3.1), we see that methods based on (3.4) will only be useful when  $r = 0$  in (2.1), in which case (3.2) shows that  $v_i \in \mathcal{R}(A)$ ,  $i = 1, \dots, k$ ; but the only possibility of  $L_k$  being singular is if  $\beta_{k+1} = 0$ , giving  $AV_k = V_k T_k$  in (3.3), from which we see that  $T_k$  cannot be singular. We see then that  $L_k = \bar{L}_k = T_k Q_k^T$  cannot be singular in (5.7), and therefore  $z_k = \bar{z}_k$  must be well-defined at the final step, even if  $\beta_{k+1} = 0$ .

In any practical computation, we will be interested in monitoring the size of the residual, usually to decide when to terminate, so when  $\bar{L}_k$  is nonsingular, we see from (3.1), (3.4) and (3.3), that

$$(5.13) \quad \begin{aligned} r_k^c &\equiv b - Ax_k^c = \beta_1 v_1 - AV_k y_k \\ &= \beta_1 v_1 - V_k T_k y_k - \beta_{k+1} v_{k+1} e_k^T y_k = -\beta_{k+1} \eta_{kk} v_{k+1}, \end{aligned}$$

where  $\eta_{kk}$  is the  $k$ th element of  $y_k$ . The vector  $y_k$  is not directly available in the computation, but since  $T_k = T_k^T = Q_k^T \bar{L}_k^T$ , (3.4) gives

$$(5.14) \quad \bar{L}_k^T y_k = \beta_1 Q_k e_1,$$

and from the last element on each side

$$(5.15) \quad \bar{\gamma}_k \eta_{kk} = \beta_1 s_1 s_2 \cdots s_{k-1},$$

so with (5.6)

$$(5.16) \quad r_k^c = -(\beta_1 s_1 s_2 \cdots s_k / c_k) v_{k+1}.$$

Thus  $\|r_k^c\|$  is directly available without ever forming  $x_k^c$ , and in fact, it will be shown in a later paper that when rounding errors are present, the norm of the residual using (5.16) is within  $O(\epsilon)\|A\|\|x\|$  of the true residual norm corresponding to the computed  $x_k^c$ , where  $\epsilon$  specifies the relative accuracy of floating-point computation.

A slightly longer algebraic manipulation shows that

$$(5.17) \quad r_k^L \equiv b - Ax_k^L = \gamma_{k+1} \zeta_{k+1} v_{k+1} - \epsilon_{k+2} \zeta_k v_{k+2}$$

so that

$$(5.18) \quad \|r_k^L\|^2 = \gamma_{k+1}^2 \zeta_{k+1}^2 + \varepsilon_{k+2}^2 \zeta_k^2$$

is directly available during the computation and may be used to decide whether to exit with  $x_k^c$  or  $x_k^L$ . Finally, it is theoretically interesting to compare these two approximations to  $x$ . From (5.11), (5.10) and (5.8), it follows that

$$x_k^c = x_k^L + \zeta_k(\bar{w}_k/c_k - w_k),$$

but from (5.9)  $\bar{w}_k = c_k w_k + s_k \bar{w}_{k+1}$ , giving

$$(5.19) \quad x_k^c = x_k^L + (\zeta_k s_k / c_k) \bar{w}_{k+1},$$

and since from (5.2)  $W_k^T \bar{w}_{k+1} = 0$ , we see from (5.10) that  $\bar{w}_{k+1}$  and  $x_k^L$  are orthogonal, so that

$$(5.20) \quad \|x_k^L\| \leq \|x_k^c\|.$$

For later reference, we shall call the method of this section SYMMLQ.

**6. The minimum residual method.** We will now examine the simplifications that result when we use the Lanczos vectors in the minimum residual method described in § 2 Case (b). From (3.3) we see that

$$(6.1) \quad V_k^T A^2 V_k = T_k^2 + \beta_{k+1}^2 e_k e_k^T,$$

$$(6.2) \quad V_k^T A b = \beta_1 V_k^T A v_1 = \beta_1 T_k e_1.$$

The matrix in (6.1) is pentadiagonal and at least positive semidefinite, and so could be used directly in (2.7) with the Cholesky decomposition, in a very similar manner to the method of conjugate gradients. Forming the matrix in (6.1) and then factorizing would lead to an unnecessary loss of accuracy, but fortunately there is a simpler approach.

If we carry out the orthogonal factorization in (5.5) and use (5.6) we see that

$$(6.3) \quad T_k^2 + \beta_{k+1}^2 e_k e_k^T = \bar{L}_k \bar{L}_k^T + \beta_{k+1}^2 e_k e_k^T = L_k L_k^T,$$

so that we have the Cholesky factor directly from  $T_k$ . In (2.7), we then have to solve

$$(6.4) \quad L_k L_k^T u_k = \beta_1 \bar{L}_k Q_k e_1.$$

But since from (5.5) and (5.6)

$$(6.5) \quad \bar{L}_k = L_k D_k, \quad D_k \equiv \text{diag}(1, 1, \dots, 1, c_k),$$

and while  $L_k$  is nonsingular, (6.4) gives

$$(6.6) \quad L_k^T u_k = \beta_1 D_k Q_k e_1 \equiv (\tau_1, \dots, \tau_k)^T \equiv t_k,$$

$$(6.7) \quad \tau_1 \equiv \beta_1 c_1, \quad \tau_i \equiv \beta_1 s_1 s_2 \cdots s_{i-1} c_i, \quad i = 2, \dots, k,$$

so there is minimal error in computing  $L_k^T u_k$ . Clearly  $u_k$  cannot be found until the algorithm is completed, but it is not really needed; instead we form

$$(6.8) \quad M_k \equiv [m_1, \dots, m_k] = V_k L_k^{-T}$$

column by column (cf. (4.3)), and then in (2.7)

$$(6.9) \quad x_k^M = V_k u_k = V_k L_k^{-T} L_k^T u_k = M_k t_k,$$

where  $t_k$  is developed in (6.7), and the superscript  $M$  shows this is the vector which gives the minimum residual. Again it can be seen that previous vectors need not be held, and this is ideal for very large sparse matrices.

Note that much of the ill-conditioning suggested by (2.7) has been avoided, but nevertheless, as  $k$  increases the condition number of  $L_k$  in (6.8) approaches that of  $A$ , so that if  $A$  is ill-conditioned then some of the vectors  $m_k$  arising in (6.8) could be very large and somewhat in error, leading to errors in  $x_k^M$  in (6.9). In fact, this minimum residual method has been found to suffer a little computationally on very poorly conditioned problems, whereas no such trouble has been found with the method in § 5. This is probably because the vectors  $w_i$  in (5.10) are theoretically orthonormal.

The minimum residual method could also have been derived by considering solving (3.4) using a  $QR$  factorization of  $T$ . For since from (5.1)  $T_k^T = T_k = Q_k^T \bar{L}_k^T$ , (3.4) becomes

$$(6.10) \quad \bar{L}_k^T y_k = \beta_1 Q_k e_1, \quad x_k^c = V_k (\bar{L}_k)^{-T} \bar{L}_k^T y_k,$$

and a small change to make the computation slightly faster by using (6.5) leads to

$$(6.11) \quad x_k^M = V_k L_k^{-T} (\beta_1 D_k Q_k e_1).$$

In fact,  $x_k^c$  could be found from (6.10) or via (6.11), but neither way is as accurate as the method in § 5 for the same reasons that the minimum residual method is suspect.

The minimum residual method will later be referred to as MINRES.

**7. Some properties of the minimum residual method.** The minimum residual method described in § 6 does not give as accurate results as the method in § 5 when the problem is very ill-conditioned, but it still appears to give very good results in other cases. It can also be used for inconsistent equations whereas the method as described in § 5 cannot. Furthermore, as the only way we have of deciding when to terminate the iteration is by testing the size of the residual, the method which minimizes this is likely to take fewer iterations than other methods. The other methods occasionally took a significant percentage more iterations than the minimum residual method, and so it will be of interest to examine the latter method further.

It is straightforward to compare  $x_k^M$  with  $x_k^c$ , for (3.4) and (6.8) give

$$x_k^c = V_k L_k^{-T} L_k^T y_k = M_k L_k^T y_k, \\ Q_k T_k y_k = \bar{L}_k^T y_k = \beta_1 Q_k e_1,$$

so with (6.5) and (6.6)

$$(7.1) \quad x_k^c = M_k D_k^{-1} \bar{L}_k^T y_k = M_k D_k^{-2} t_k \\ = x_k^M + \tau_k (c_k^{-2} - 1) m_k = x_k^M + \tau_k (s_k/c_k)^2 m_k.$$

Note that  $x_k^c$  can easily be obtained during the computation of  $x_k^M$ , but the reverse is not true. The  $m_k$  and  $w_k$  are related in a simple manner, for if the Lanczos process stops with  $\beta_{m+1} = 0$ , then  $AV_m = V_m T_m$ ,  $T_m = L_m Q_m$ , and  $M_m = V_m L_m^{-T}$ , so

$$(7.2) \quad AM_m = V_m T_m L_m^{-T} = V_m Q_m^T = W_m.$$

Using this result with (7.1) gives

$$(7.3) \quad r_k^M - r_k^c \equiv A(x_k^c - x_k^M) = \tau_k(s_k/c_k)^2 w_k,$$

so that with (5.16) and (6.7)

$$r_k^M \equiv b - Ax_k^M = \beta_1 s_1 s_2 \cdots s_k (s_k w_k - v_{k+1})/c_k;$$

but from (5.9)  $v_{k+1} = s_k w_k - c_k \bar{w}_{k+1}$ , so

$$(7.4) \quad r_k^M = \beta_1 s_1 s_2 \cdots s_k \bar{w}_{k+1}, \quad \|\bar{w}_{k+1}\| = 1,$$

which shows clearly how the residual norm decreases each step. Also using (5.16),

$$(7.5) \quad \|r_k^M\| = |c_k| \|r_k^c\| < \|r_k^c\|$$

except at the last step, where theoretically  $s_k = 0$ .

It is interesting to note from (5.16) and (7.4) that the sizes of the residuals  $r_k^c$  and  $r_k^M$  are given directly by the size of  $b$  and the  $LQ$  decomposition of  $T_{k+1}$ , and so are immediately available whichever of these algorithms is used. Equation (5.18) shows that  $\|r_k^L\|$  is also available if (5.7) is solved.

**8. Computational experience.** Algorithms SYMMLQ and MINRES have been programmed and tested on various systems of equations in order to obtain an impression of their numerical properties. A comparison has also been made in some cases with Reid's version 2 of the conjugate gradients method (CGM) [11]. We make the following observations.

1. On symmetric positive definite systems, SYMMLQ gives essentially the same results as CGM. For example, the problem involving the Laplacian matrix of order 4080 ( $15 \times 16 \times 17$  grid) was solved with SYMMLQ under the same conditions as described by Reid [11] for CGM, viz., single precision on the IBM System/360. A graph of  $\|x - x_k^L\|$  lagged markedly behind the curve for CGM shown in [11, Fig. 3], but SYMMLQ terminated at the same point as CGM by taking a final step from  $x_k^L$  to  $x_{k+1}^c$  as in (5.11). For test purposes all points  $x_{k+1}^c$  were computed from the points  $x_k^L$ , and the quantities  $\|x - x_k^c\|$  were seen to follow the curve in [11, Fig. 3] almost exactly.

2. Although SYMMLQ obtains the same final point as CGM, it is clear that for positive definite systems CGM is to be preferred as it is more efficient.

3. The variant of CGM described in §4 gave almost identical results for positive definite matrices as CGM in [11]. This confirms that the derivation of CGM from the Lanczos vectors and the Cholesky factorization of  $T_k$  is *computationally* similar to CGM, aside from their mathematical equivalence.

4. Algorithm MINRES has behaved well on some examples involving the 2-dimensional Laplacean matrix, giving a rapid and very smooth decrease in both  $\|r_k^M\|$  and  $\|x - x_k^M\|$ . On the other very ill-conditioned problems the estimate of  $\|r_k^M\|$  in (7.4) decreased steadily but departed from the true  $\|r_k^M\|$  and thus caused premature termination. In such cases, it was also observed that if iterations were continued, the true  $\|r_k^M\|$  stayed essentially constant while the true error  $\|x - x_k^M\|$  continued to decrease until it reached quite an acceptably low level.

5. An excellent application of SYMMLQ and MINRES is in solving symmetric systems of the form  $(A - \mu I)x = b$  in the style of inverse iteration, since if  $\mu$  is

near an interior eigenvalue  $\lambda$  of  $A$ , the matrix  $A - \mu I$  is indefinite. If  $\mu$  is sufficiently close to  $\lambda$  and  $b$  is chosen appropriately then the computed  $x$  will be a good approximation to an eigenvector of  $A$ , and in practice it appears that the number of iterations required by SYMMLQ or MINRES is very small. In [15] Ruhe and Wiberg used CGM when  $A - \mu I$  was definite, or nearly so. Here there are no such restrictions.

6. Figure 1 illustrates the behavior of SYMMLQ on a symmetric system  $(B^2 - \mu I)x = b$  of order  $n = 50$ , where  $\mu = \sqrt{3}$  is not near an eigenvalue of  $B^2$  but was chosen to make the system indefinite. The matrix  $B$  is tridiagonal with typical nonzero row elements  $(-1, 2, -1)$ , so that  $B^2$  is pentadiagonal with typical row  $(1, -4, 6, -4, 1)$ . Computation was performed on a Burroughs B6700 with floating-point precision  $\varepsilon = 8^{-12} = 1.455 \times 10^{-11}$ .

Estimates of the size of the residual vectors  $r_k^L$ ,  $r_k^C$  and  $r_k^M$  are all available from SYMMLQ, and these were used to give estimates of  $\log_{10} \|r_k^L\|$ ,  $\log_{10} \|r_k^C\|$ ,  $\log_{10} \|r_k^M\|$  which are plotted in Fig. 1 against iteration number  $k$ . Of interest is the sharp reduction in residual obtained by taking a final step from  $x_{32}^L$  to the CGM point  $x_{33}^C$  (see dotted line in Fig. 1). Note also the sharp jumps in  $\|r_k^C\|$  at  $k = 11$  and 24. These indicate regions of instability in the CGM sequence  $x_k^C$ ,

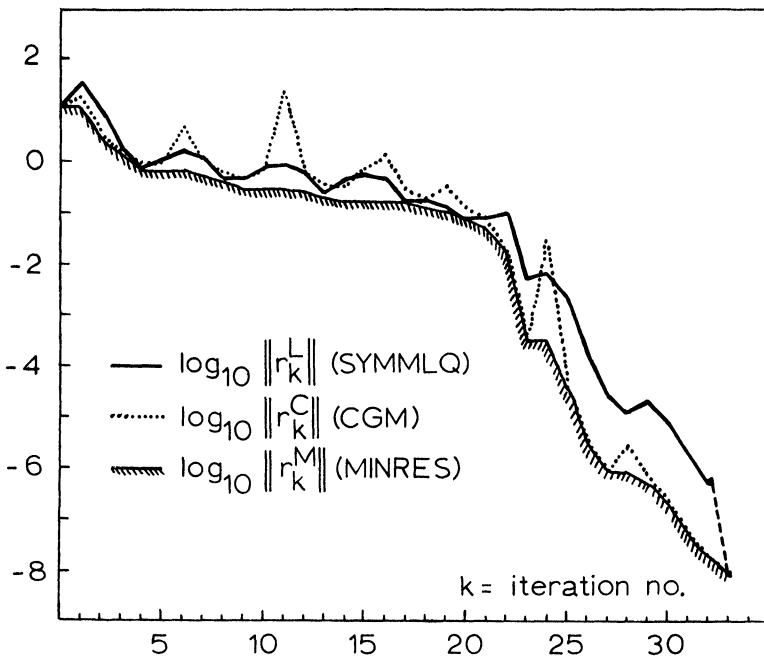


FIG. 1. Solution of an indefinite symmetric system of equations  $(B^2 - \mu I)x = b$ , using subroutine SYMMLQ

- Notes
1. Dimension of system is  $n = 50$ ;  $\mu$  is not close to an eigenvalue of  $B^2$ .
  2.  $r_k^L$ ,  $r_k^C$ ,  $r_k^M$  are residual vectors for iteration paths taken by algorithms SYMMLQ, CGM, MINRES, respectively. Estimates of the norms of these quantities are all computed by subroutine SYMMLQ.
  3. Note large jumps in the size of  $\|r_k^C\|$ , reflecting intermediate near-singularities which would cause the standard method of conjugate gradients to break down.

as described more fully in the next section, and if the standard method of conjugate gradients were used to compute the points  $x_k^c$ , it is to be expected that the iterates  $\|r_k^c\|$  would diverge from the path shown.

The final residual norm obtained was  $7.83 \times 10^{-9}$ , while the computed estimate of  $\|r_{3,3}^c\|$  was  $7.65 \times 10^{-9}$ . This illustrates that the computed estimate of  $\|r_k^c\|$  remains a good measure of the residual for the *computed* point  $x_k^c$ , in spite of the fact that the computed points are significantly different from those that would be obtained with exact computation. The same is true of the computed estimate of  $\|r_k^L\|$ .

**9. Summary.** We can now distinguish two reasons why the method of conjugate gradients (CGM) may fail to solve the symmetric system  $Ax = b$  if  $A$  is not positive (or negative) definite. Recall that CGM attempts to compute a sequence of approximations to  $x_k^c$  which satisfy

$$(9.1) \quad x_k^c = V_k T_k^{-1} \beta_1 e_1, \quad k = 1, 2, \dots, m$$

for some  $m > 0$ , where  $V_k^T V_k = I_k$ ,  $V_k^T A V_k = T_k$  and the matrices  $T_k$  are tridiagonal, with  $T_{k+1}$  having  $T_k$  as its  $k \times k$  principal submatrix. Recall also that CGM effectively computes the Cholesky factorization of each  $T_k$ . The basic problem we must contend with is the following:

If  $A$  is indefinite, it is possible for some  $T_k$  to be singular or nearly singular ( $k < m$ ), even if  $T_m$  is well conditioned.

Now if  $T_k$  is nearly singular it happens that the Cholesky factorization of  $T_j$  is poorly determined numerically for all  $j > k$ . Even more seriously, if  $T_k$  is singular the corresponding point  $x_k^c$  in (9.1) is not properly defined. Thus we see that CGM's use of (9.1) is doubly doomed to failure.

The main features of algorithms SYMMLQ and MINRES can now be put into perspective. First of all, the orthogonal factorization  $T_k = \bar{L}_k Q_k$  is well defined regardless of any near-singularities in  $T_j$  for  $j \leq k$ . In fact, as (5.11) and (5.19) show, we could compute the CGM sequence of points using

$$(9.2) \quad x_k^c = x_{k-1}^c + (\bar{\zeta}_k - \zeta_{k-1} s_{k-1} / c_{k-1}) \bar{w}_k$$

without the aid of the Cholesky factorization; but the more fundamental difficulty remains that  $x_k^c$  does not exist if  $T_k$ , and hence  $\bar{L}_k$  are singular. In such cases  $\bar{\zeta}_k$  in (9.2) is undefined.

Secondly, then, instead of using  $\bar{L}_k$  to compute the CGM sequence  $x_k^c$ , we define two new sequences  $x_k^L$  and  $x_k^M$  in terms of a matrix  $L_k$  which is the  $k \times k$  principal submatrix of  $\bar{L}_{k+1}$  and is guaranteed to be nonsingular. By this means, we effectively step around any irrelevant intermediate singularities in the CGM sequence (9.1). Some near-singularities are shown by the peaks in  $\|r_k^c\|$  in Fig. 1. We see from (5.6) and (7.5) that  $\|r_k^c\| = \|r_k^M\| \cdot |\gamma_k|/|\bar{\gamma}_k|$  so we will get a large jump in  $\|r_k^c\|$  when  $T_k$  is nearly singular but  $A$  is not.

Finally we note that the CGM points  $x_k^c$  are not to be discarded completely, since at least half of them *are* well defined by (9.1). This can be seen from the fact that if both  $T_k$  and  $T_{k+1}$  are singular then so are all  $T_j$ ,  $j \geq k$ ; hence if  $A$  is nonsingular, there cannot be two singular  $T_k$ 's in a row. Thus in algorithm SYMMLQ provision is made to terminate iterations at a CGM point whenever advantageous.

**Acknowledgments.** We would like very much to thank John Reid for discussions about this work, and Gene Golub for his comments on this work and his hospitality at Stanford. We also gratefully acknowledge use of the computing facilities at the Stanford Linear Accelerator Center where earlier versions of our programs were developed.

## REFERENCES

- [1] V. M. FRIDMAN, *The Method of minimum iterations with minimum errors for a system of linear algebraic equations with a symmetrical matrix*, U.S.S.R. Computational Math. and Math. Phys., 3 (1963), pp. 362–363.
- [2] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.
- [3] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Blaisdell, New York, 1964, pp. 139–141.
- [4] 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), pp. 255–282.
- [5] ———, *Solution of systems of linear equations by minimized iterations*, *Ibid.*, 49 (1952), pp. 33–53.
- [6] C. L. LAWSON, *Sparse matrix methods based on orthogonality and conjugacy*, Tech. Memo. 33–627, Jet Propulsion Laboratory, Pasadena, Calif., 1973.
- [7] D. G. LUENBERGER, *Hyperbolic pairs in the method of conjugate gradients*, SIAM J. Appl. Math., 17 (1969), pp. 1263–1267.
- [8] ———, *The conjugate residual method for constrained minimization problems*, this Journal, 7 (1970), pp. 390–398.
- [9] C. C. PAIGE, *Computational variants of the Lanczos method for the eigenproblem*, J. Inst. Math. Appl. 10 (1972), pp. 373–381.
- [10] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of equations and least squares problems*, Tech. Rep. CS 399, Computer Science Dept., Stanford Univ., Stanford, Calif., 1973.
- [11] J. K. REID, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, Proc. of a Conference on Large Sparse Sets of Linear Equations, Academic Press, New York, 1971, pp. 231–254.
- [12] I. H. SIEGEL, *Deferment of computation in the method of least squares*, Math. Comput., 19 (1965), pp. 329–331.
- [13] W. M. GENTLEMAN, *Least squares computations by Givens transformations without square roots*, J. Inst. Math. Appl., 12 (1973), pp. 329–336.
- [14] S. HAMMARLING, *A note on modifications to the Givens plane rotation*, *Ibid.*, 13 (1974), pp. 215–218.
- [15] A. RUHE AND T. WIBERG, *The method of conjugate gradients used in inverse iteration*, BIT, 12 (1972), pp. 543–554.