SIAM J. NUMER. ANAL.
Vol. 11, No. 1, March 1974

BIDIAGONALIZATION OF MATRICES AND SOLUTION OF LINEAR EQUATIONS*

C. C. PAIGE†

Abstract. An algorithm given by Golub and Kahan [2] for reducing a general matrix to bidiagonal form is shown to be very important for large sparse matrices. The singular values of the matrix are those of the bidiagonal form, and these can be easily computed. The bidiagonalization algorithm is shown to be the basis of important methods for solving the linear least squares problem for large sparse matrices. Eigenvalues of certain 2-cyclic matrices can also be efficiently computed using this bidiagonalization.

1. Introduction. The singular value decomposition of an $m \times n$ matrix A is

$$A = X\Sigma Y^H, \tag{1.1}$$

where X and Y are unitary matrices and Σ is a rectangular diagonal $m \times n$ matrix with nonnegative real diagonal entries, the singular values of A.

Golub and Kahan [2, (2.4)] suggest a particular method for producing a bidiagonal matrix with the same singular values as A. This method is related to the Lanczos method of minimized iterations for tridiagonalizing a symmetric matrix [5], and like that method is ideally suited for large sparse matrices. The equivalent bidiagonalization of a general matrix and tridiagonalization of a symmetric matrix using Householder matrices is more stable but cannot be applied to many large sparse problems because of storage and time considerations.

Despite a certain numerical instability it is known that a particular variant of the Lanczos tridiagonalization algorithm still gives extremely useful results [7], that is, the eigenvalues and eigenvectors are given as accurately as could be hoped, but the convergence can be slower than the theoretical rate, and often several computed eigenvalues can represent just one true eigenvalue. The bidiagonalization algorithm of Golub and Kahan will be seen to be numerically equivalent to this variant of the Lanczos method applied to a certain matrix, and so will have the same sort of stability. Thus singular values will be given accurately, but with uncertain multiplicity. The bidiagonalization algorithm and its properties are given in § 2 more fully than in [2]. Section 2 also indicates how eigenvalues of 2-cyclic matrices may be found efficiently using this approach.

The Lanczos tridiagonalization of symmetric matrices is the basis of the conjugate gradients method of solution of equations [6], [4]; and Reid [8] has done much good work to show that the conjugate gradients method not only works, but is one of the best available for several large sparse problems. The reason it works the way it does is directly related to the performance of the Lanczos algorithm.

In a similar manner it will be shown here that the bidiagonalization algorithm is the basis for two methods of solving the linear least squares problem. These methods are remarkably simple and ideally suited to general large sparse problems, requiring very little computation per step, or storage. They derive their

* Received by the editors July 11, 1972.

† School of Computer Science, McGill University, Montreal, Canada. This work was done while Visiting Research Associate at Stanford University under National Science Foundation Grant GJ-29988X.

accuracy and convergence characteristics from those of the bidiagonalization algorithm. Finally, the relevant singular values of the problem can be found, as well as the solution; these are important for understanding the accuracy of the solution. The two solution of equations methods are developed and examined in §§ 3 and 4.

2. The bidiagonalization algorithm.

Here $\|\cdot\|$ will denote the l_2-norm, that is,

$$\|u\|^2 = (u, u) = u^H u.$$

The algorithm suggested by Golub and Kahan [2] for bidiagonalizing A will now be described in detail.

For a given $m \times n$ matrix A and a given initial vector u_1 with $\|u_1\| = 1$, the method produces m-dimensional vectors $u_1, u_2, \cdots$, and n-dimensional vectors $v_1, v_2, \cdots$, as follows: For $i = 1, 2, \cdots$,

$$\alpha_i v_i = A^H u_i - \beta_i v_{i-1}, \qquad \beta_1 v_0 \equiv 0, \tag{2.1}$$

$$\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i, \tag{2.2}$$

where the real scalars α_i and β_{i+1} are chosen to be nonnegative and such that $\|u_{i+1}\| = \|v_i\| = 1$. Suppose α_i, β_{i+1} are nonzero for $i = 1, 2, \cdots, k$. Then the above process is fully defined for k steps, and if

$$\begin{aligned} U &\equiv [u_1, u_2, \cdots, u_k], \\ V &\equiv [v_1, v_2, \cdots, v_k], \end{aligned} \qquad L \equiv \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \cdot & \cdot & \\ & & \beta_k & \alpha_k \end{bmatrix}, \tag{2.3}$$

then (2.1) and (2.2) may be rewritten

$$A^H U = V L^H, \qquad AV = UL + \beta_{k+1} u_{k+1} e_k^H, \tag{2.4}$$

where e_k is the last column of the $k \times k$ unit matrix I_k, so that

$$U^H A V = U^H U L + \beta_{k+1} U^H u_{k+1} e_k^H = L V^H V. \tag{2.5}$$

If α_{k+1} is also nonzero and one more step of (2.1) alone is executed, with $\tilde{U} \equiv [U, u_{k+1}]$, $\tilde{L} \equiv [L^H, \beta_{k+1} e_k]^H$, then

$$A^H \tilde{U} = V \tilde{L}^H + \alpha_{k+1} v_{k+1} e_{k+1}^H, \qquad AV = \tilde{U} \tilde{L}. \tag{2.6}$$

Therefore,

$$V^H A^H \tilde{U} = V^H V \tilde{L}^H + \alpha_{k+1} V^H v_{k+1} e_{k+1}^H = \tilde{L}^H \tilde{U}^H \tilde{U}. \tag{2.7}$$

It is now easy to show by induction that

$$U^H U = V^H V = I_k. \tag{2.8}$$

This is certainly true for $k = 1$, so suppose it is true for some $k \geqq 1$. Then from (2.5), $U^H u_{k+1} = 0$, and from (2.7), $V^H v_{k+1} = 0$, and so (2.8) is true for $k + 1$.

This orthogonality ensures that the process will curtail for some value k no greater than the minimum of m and n with either $\beta_{k+1} u_{k+1} = 0$ in (2.4) or

$\alpha_{k+1}v_{k+1} = 0$ in (2.6). As a result there are two possible final states of the algorithm, the first with the $k \times k$ matrix L:

$$A^H U = VL^H, \quad AV = UL, \quad U^H U = V^H V = I_k, \tag{2.9}$$

and the second with the $(k + 1) \times k$ matrix $\tilde{L}$:

$$A^H \tilde{U} = V\tilde{L}^H, \quad AV = \tilde{U}\tilde{L}, \quad \tilde{U}^H \tilde{U} = I_{k+1}, \quad V^H V = I_k. \tag{2.10}$$

To understand which termination will occur, consider the singular value decomposition of $\tilde{L}$ in (2.10). $\tilde{L}$ has linearly independent columns so

$$\tilde{L} = PMQ^H, \quad P^H P = PP^H = I_{k+1}, \quad Q^H Q = QQ^H = I_k, \tag{2.11}$$

where M is $(k + 1) \times k$ and has elements

$$m_{11}, \cdots, m_{kk} > 0; \quad m_{ij} = 0 \quad \text{for } i \neq j.$$

Thus

$$A^H \tilde{U} P = VQM^H, \qquad AVQ = \tilde{U}PM.$$

Therefore,

$$AA^H \tilde{U} P = \tilde{U}PMM^H, \qquad A^H AVQ = VQM^H M, \tag{2.12}$$

and so when (2.10) results, A has at least k nonzero singular values $m_{11}, \cdots, m_{kk}$, and at least one zero singular value with the last column of $\tilde{U}P$ being the singular vector that lies in $\mathcal{N}(A^H)$, the null space of A^H.

Now if for $i = 1$ it is true that $u_i \in \mathcal{R}(A)$, the range of A, then it can be seen from (2.2) that this is true for all i. But $\mathcal{N}(A^H)$ is the orthogonal complement of $\mathcal{R}(A)$, so if $u_1 \in \mathcal{R}(A)$, then all u_i are orthogonal to $\mathcal{N}(A^H)$, and from the previous paragraph this means that (2.10) cannot be the final result, and so (2.9) must result. Next note that if (2.9) is the final result, then since L is nonsingular $U = AVL^{-1}$, so that $u_i \in \mathcal{R}(A)$ for all i; thus if $u_1 \notin \mathcal{R}(A)$ then (2.9) cannot follow and (2.10) must be the final result.

Note that if A has rank $r = m$, then $u_1 \in \mathcal{R}(A)$ necessarily, and (2.9) follows. If $r < m$, then (2.10) will result unless u_1 is a linear combination of some of the columns of X in (1.1). Both final states give k nonzero singular values of A, so the process must curtail with $k \leqq r$.

If the process halts with $k < r$, then it can be continued in an obvious way until r orthogonal vectors $v_1, \cdots, v_r$ are obtained [2]. Then if $u_1 \in \mathcal{R}(A)$, the result (2.9) can be written

$$A = ULV^H, \qquad U^H U = V^H V = I_r. \tag{2.13}$$

Otherwise the result follows from (2.10):

$$A = \tilde{U}\tilde{L}V^H, \quad \tilde{U}^H \tilde{U} = I_{r+1}, \quad V^H V = I_r. \tag{2.14}$$

Such continuation need not be considered here.

We shall now relate the bidiagonalization algorithm to the Lanczos reduction of a certain symmetric matrix to tridiagonal form. Defining

$$W = [w_1, w_2, \cdots, w_{2k}] \equiv \begin{pmatrix} u_1 & 0 & u_2 & 0 & \cdots & u_k & 0 \\ 0 & v_1 & 0 & v_2 & & 0 & v_k \end{pmatrix},$$

(2.15)
$$B = \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix}, \quad T \equiv \begin{pmatrix} 0 & \alpha_1 & & & & \\ \alpha_1 & 0 & \beta_2 & & & \\ & \beta_2 & 0 & \alpha_2 & & \\ & & \cdot & \cdot & \cdot & \\ & & & & \alpha_k & 0 \end{pmatrix}$$

it can be seen for example that (2.4) is mathematically equivalent to

(2.16) $$BW = WT + \beta_{k+1} w_{2k+1} e_{2k}^T, \quad W^H W = I_{2k}, \quad W^H w_{2k+1} = 0,$$

which is the result of applying $2k$ steps of the Lanczos process to B using as initial vector w_1, the first column of W. Note that computation and vector storage have effectively been halved by taking full advantage of the structure of B. A direct comparison of this algorithm with the successful variant of the Lanczos algorithm in [7] applied to the matrix B and initial vector w_1 shows that the two methods are also computationally equivalent.

In practice neither algorithm will stop with either α_k or β_k zero, but for large enough k many of the eigenvalues of T in (2.15) will approximate eigenvalues of B to almost machine precision. Now if λ is an eigenvalue of B such that

$$\begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \lambda \begin{pmatrix} y \\ z \end{pmatrix},$$

then

$$A^H A z = \lambda^2 z,$$

so that $\sigma = |\lambda|$ is a singular value of A, and several singular values of A will be given to almost machine precision by computing the eigenvalues of T. However, with the permutation matrix

$$P \equiv [e_1, e_{k+1}, e_2, e_{k+2}, \cdots, e_k, e_{2k}],$$

$$T = P^T \begin{pmatrix} 0 & L \\ L^T & 0 \end{pmatrix} P,$$

so that the singular values of L are the moduli of eigenvalues of T, and thus we need only compute the singular values of L. These can be accurately computed using the QR-like algorithm given in [3].

Finally, if σ is a singular value of A, then $\gamma \pm \sigma$ are eigenvalues of the 2-cyclic matrix

$$\begin{pmatrix} \gamma I & A \\ A^H & \gamma I \end{pmatrix},$$

so eigenvalues of such matrices can be computed efficiently just by using the bidiagonalization algorithm on A, and this is particularly important for large sparse A, which arise from elliptic partial differential equations. This result is parallel to that in [9], where Reid shows how the computation and storage can be halved when the conjugate gradients algorithm is applied to such 2-cyclic matrices.

In practice the bidiagonalization algorithm performed just as expected, the largest singular values being given with remarkable accuracy very quickly, while the small ones (which correspond to the middle eigenvalues of B in (2.15)) were slower to converge, especially when there were several very close, very small ones. If the process was carried far enough, the large singular values would often appear several times.

3. Solution of the linear least squares problem. I. Given the $m \times n$ matrix A and an m-vector b, the problem is

$$\text{minimize} \quad \|Ax - b\| \tag{3.1}$$

or equivalently, find x and r such that

$$r + Ax = b, \qquad A^H r = 0. \tag{3.2}$$

Any such x is called *a least squares solution.* The x which also minimizes $\|x\|$ is called *the minimum least squares solution.* The minimum least squares solution is the unique solution orthogonal to $\mathcal{N}(A)$, and so will have the form $x = A^H y$. Thus if y is any solution of

$$A^H A A^H y = A^H b, \tag{3.3}$$

then $x = A^H y$ is the minimum least squares solution.

In order to motivate the methods that are about to be introduced for solving such problems, suppose that $AV = UL$ as in (2.9), and the representation $x = Vy$ holds; then since $0 = r^H AV = r^H UL$ and L is nonsingular, $U^H r = 0$. Also

$$b = r + Ax = r + AVy = r + ULy; \tag{3.4}$$

thus y and x may be found from

$$Ly = U^H b, \qquad x = Vy. \tag{3.5}$$

It is now necessary to show that for a certain choice of u_1 in (2.1) and (2.2) such a representation of x is possible. We must then consider which of (2.9) and (2.10) results and what can be done in the latter case.

The most convenient choice of initial vector is the one to be considered here:

$$u_1 = b/\beta_1, \qquad \beta_1 \equiv \|b\|; \tag{3.6}$$

this is convenient because $U^H b = \beta_1 e_1$ is then known a priori; this will be discussed again in § 5. The equations of the form (3.1) or (3.2) now separate into two possible classes with corresponding slightly different methods of solution. First, if the equation $Ax = b$ is *compatible*, that is if $r = 0$ in (3.2), then $b \in \mathscr{R}(A)$ and so $u_1 \in \mathscr{R}(A)$ in (3.6), and from § 2 it follows that (2.9) must result. This compatible case will be treated now in full; the case when $r \neq 0$ will be examined later in this paper.

Here

$$Ax = b = \beta_1 u_1, \quad A^H U = VL^H, \quad AV = UL, \quad U^H U = V^H V = I, \tag{3.7}$$

and let $x = Vy + w$, where $V^H w = 0$. Then $Ax = ULy + Aw = \beta_1 u_1$; but $U^H Aw = LV^H w = 0$, so $Ly = \beta_1 e_1$ and thus $Aw = 0$, meaning that the solution of minimum norm is

$$x = Vy, \tag{3.8}$$

since from (3.7), $Vy = A^H UL^{-H} y \in \mathscr{R}(A^H)$, and so is orthogonal to $\mathscr{N}(A)$.

The elements η_i of y are found from $Ly = \beta_1 e_1$, that is,

$$\eta_1 = \beta_1/\alpha_1, \qquad \eta_{i+1} = -(\beta_{i+1}/\alpha_{i+1})\eta_i. \tag{3.9}$$

The method is an attractive one because the 2-norm of the error is minimized in the ith step over all possible approximate solutions which are combinations of $v_1, \cdots, v_i$. It is an excellent algorithm if A is a large sparse matrix, since a means of computing $u := Av - u$ and storage for 3 vectors is just about all that is needed. What is more, the residual is available at each step with no extra computation, for if $y_i \equiv (\eta_1, \cdots, \eta_i, 0, \cdots, 0)^H$, then the residual r_i after the ith step is

$$\begin{aligned} r_i &= b - AVy_i = \beta_1 u_1 - ULy_i = -\beta_{i+1}\eta_i u_{i+1} \\ &= \eta_i(\alpha_i u_i - Av_i) = -\beta_i \eta_{i-1} u_i - \eta_i A v_i \\ &= r_{i-1} - \eta_i A v_i \end{aligned} \tag{3.10}$$

where use has been made of (2.2) and the expression for η_{i+1} in (3.9). From this expression it can be seen that residuals at different steps are orthogonal.

This is not a new method as will be shown in the next paragraph. Its importance here is the very simple derivation from the very natural bidiagonalization algorithm (2.1) and (2.2), so that the connection between the method and the relevant singular values of A is laid bare. As is well known, these singular values are of vital importance in the solution of linear equations.

In fact, the method (3.9) is equivalent to Craig's method in the computational form described by Faddeev and Faddeeva [1, (23), p. 405]. In Craig's algorithm take $x_0 = 0, a_i = 1/\alpha_i^2, b_i = [\alpha_{i+1}\eta_{i+1}]^2/[\alpha_i\eta_i]^2, g_i = \alpha_i^2\eta_i v_i$, and $r_i = -\beta_{i+1}\eta_i u_{i+1}$, and the algorithm here will result. Thus Craig's method is seen to apply to any compatible system of linear algebraic equations. It has been pointed out in [1] that Craig's method is mathematically equivalent to applying the method of conjugate gradients [4] to

$$AA^H y = b, \qquad x = A^H y. \tag{3.11}$$

The purpose in using (3.9) rather than conjugate gradients applied to (3.11) is to avoid any suggestion of deterioration in the condition of the problem to be solved.

The possibility that now has to be examined is the one where r is nonzero in (3.2), so that $u_1 = b/\beta_1 \notin \mathscr{R}(A)$. Section 2 then shows that (2.10) must be the final state of the bidiagonalization procedure. Thus solve

$$r + Ax = b = \beta_1 u_1, \qquad A^H r = 0 \tag{3.12}$$

using

$$A^H\tilde{U} = V\tilde{L}^H, \quad AV = \tilde{U}\tilde{L}, \quad \tilde{U}^H\tilde{U} = I_{k+1}, \quad V^HV = I_k. \tag{3.13}$$

Suppose

$$x = Vy + w \quad \text{with} \quad V^Hw = 0; \tag{3.14}$$

then $r + \tilde{U}\tilde{L}y + Aw = \beta_1u_1$, but $\tilde{U}^HAw = \tilde{L}V^Hw = 0$ so $\tilde{L}y = \beta_1e_1 - \tilde{U}^Hr$ giving $r + Aw = \tilde{U}\tilde{U}^Hr$; thus $w^HA^Hr + w^HA^HAw = \|Aw\|^2 = w^HA^H\tilde{U}\tilde{U}^Hr = 0$, since $w^HA^H\tilde{U} = 0$ from the above.

Thus $Aw = 0$, and a smaller solution is obtained by taking $x = Vy$ in (3.14). But from (2.1), $v_i \in \mathcal{R}(A^H)$ for all i, and so this x is orthogonal to $\mathcal{N}(A)$ and is therefore the minimum least squares solution. The equations of interest are now

$$r + \tilde{U}\tilde{L}y = \beta_1u_1, \quad \tilde{L}y = \beta_1e_1 - \tilde{U}^Hr, \quad r = \tilde{U}\tilde{U}^Hr, \tag{3.15}$$

but since r is unknown a priori the ith element η_i of y apparently cannot be found at the same time as u_i and v_i are produced—unlike (3.9). However, suppose

$$\gamma \equiv u_1^Hr, \quad t \equiv \begin{pmatrix} \tau_1 \\ \tau_2 \\ \cdot \\ \cdot \\ \tau_{k-1} \end{pmatrix}, \quad \begin{pmatrix} 1 \\ t \\ \tau_k \end{pmatrix} \equiv \tilde{U}^Hr/\gamma; \tag{3.16}$$

then since from (3.13) and (3.12)

$$\tilde{L}^H\tilde{U}^Hr = V^HA^Hr = 0, \tag{3.17}$$

it follows on dividing by γ that

$$\bar{L}\begin{pmatrix} t \\ \tau_k \end{pmatrix} = -\alpha_1e_1, \tag{3.18}$$

where

$$\bar{L} \equiv \begin{pmatrix} \beta_2 & & & \\ \alpha_2 & \beta_3 & & \\ & \cdot & \cdot & \cdot \\ & & \alpha_k & \beta_{k+1} \end{pmatrix}.$$

This can be solved, giving

$$\tau_i = -\tau_{i-1}\alpha_i/\beta_{i+1}, \qquad i = 1, \cdots, k, \qquad \text{where } \tau_0 \equiv 1. \tag{3.19}$$

Thus if γ were known a priori, u_i^Hr would be available in the ith step and so η_i could be found. On the other hand, from (3.15) and (3.17),

$$\|r\|^2 = \beta_1r^Hu_1 = \beta_1\gamma \tag{3.20}$$

so if γ were known a priori, then $\|r\|$ would be known a priori, which seems unlikely.

Suppose now that vectors z and w are found from

$$Lz = \beta_1e_1, \qquad Lw = \begin{pmatrix} 1 \\ t \end{pmatrix}; \tag{3.21}$$

then taking

$$y = z - \gamma w \tag{3.22}$$

gives

$$Ly = \beta_1 e_1 - U^H r \tag{3.23}$$

so that

$$x = Vy = Vz - \gamma V w \tag{3.24}$$

and Vz and Vw may be formed as the bidiagonalization proceeds. In the last step γ may be found from the last element of the middle equation in (3.15), equation (3.22), and (3.16).

$$\beta_{k+1}\eta_k = \beta_{k+1}(\zeta_k - \gamma\omega_k) = -\gamma\tau_k ,$$

i.e.,

$$\gamma = \beta_{k+1}\zeta_k/(\beta_{k+1}\omega_k - \tau_k) \tag{3.25}$$

where ζ_i, ω_i are the ith elements of z, w respectively. The solution x is then given by (3.24).

If Vz and γVw were large and nearly equal, then cancellation in (3.22) would mean numerical instability. However, from (3.21), $\|z\| \leqq \beta_1/\sigma_k$ where σ_k is the smallest singular value of L, that is, the smallest relevant (nonzero) singular value of A. Thus the subtraction in (3.22) can at most introduce an (additional) error in x of $\beta_1 O(\varepsilon)/\sigma_k$, where ε is the machine precision. Such an error size is to be expected from the condition of the problem, and so the step (3.24) appears acceptable.

With this approach the algorithm for solving either compatible or incompatible systems of equations can be written as follows:

```
        $\tau_0 := 1;\quad \omega_0 := 0;\quad \zeta_0 := -1;\quad vz := 0;\quad vw := 0;$
        $\beta_1 u_1 := b;$
        $\alpha_1 v_1 := A^H u_1;\qquad i := 0;$
general step:
            $i := i + 1;$
            $\zeta_i := -\zeta_{i-1}\beta_i/\alpha_i;$                $vz := vz + \zeta_i v_i;$
            $\omega_i := (\tau_{i-1} - \beta_i\omega_{i-1})/\alpha_i;$      $vw := vw + \omega_i v_i;$
            $\beta_{i+1}u_{i+1} := Av_i - \alpha_i u_i;$
if $\beta_{i+1} = 0$, then (solution $:= vz$; residual $:= 0$; stop);
                $\tau_i := -\tau_{i-1}\alpha_i/\beta_{i+1};$
                $\alpha_{i+1}v_{i+1} := A^H u_{i+1} - \beta_{i+1}v_i;$
if $\alpha_{i+1} = 0$, then
        ($\gamma := \beta_{i+1}\zeta_i/(\beta_{i+1}\omega_i - \tau_i)$; solution $:= vz - \gamma vw$; stop);
        go to general step;
```

As before α_i, β_i are chosen so that $\|v_i\| = \|u_i\| = 1$. This requires only 3 vectors of length n, i.e., v, vz, vw, and 1 vector of length m, i.e., u; or if it were known a priori that $\gamma = 0$, then vw would not be needed.

It can be shown by using (3.13), (3.15), and (3.16), that after the ith step the residual is

$$r_i = -\beta_{i+1}(\zeta_i - \gamma\omega_i)u_{i+1} + \gamma \sum_{j=1}^{i} \tau_{j-1}u_j$$

but this is not available until γ is known at the end of the process. However, if $r = 0$ in (3.12) then $\gamma = 0$ from (3.16) and so $-\beta_{i+1}\zeta_i u_{i+1}$ is the residual after the ith step; this is just (3.10).

In practice the stopping criteria would be almost useless, and in fact it could happen that no α_i or β_i was even small. The hope is that both ζ_i and ω_i become negligible, and that γ in (3.25) attains a stable value.

4. Solution of the linear least squares problem. II.

The bidiagonalization algorithm can be applied with A^H interchanged with A in (2.1) and (2.2), and then a different algorithm for solution of equations will result. However, since the present bidiagonalization algorithm has been analyzed so fully in § 2, it will be retained, and the alternative solution of equations algorithm will be derived from applying this same bidiagonalization algorithm to the problem

$$r + A^H x = b, \qquad Ar = 0, \tag{4.1}$$

where, as before, A is an $m \times n$ matrix, and b is given.

The most convenient choice of initial vector here is

$$u_1 = Ab/\beta_1, \qquad \beta_1 = \|Ab\|. \tag{4.2}$$

With this choice $u_1 \in \mathscr{R}(A)$, and from the discussion in § 2 the final result of the bidiagonalization must be

$$A^H U = VL^H, \qquad AV = UL, \qquad U^H U = V^H V = I_k. \tag{4.3}$$

Suppose now that

$$x = Uy + f, \qquad U^H f = 0.$$

Then $V^H A^H f = L^H U^H f = 0$, and

$$A^H x = A^H U y + A^H f = VL^H y + A^H f = b - r;$$

therefore,

$$L^H y = V^H(b - r). \tag{4.4}$$

Thus $b - r = VV^H(b - r) + A^H f$, giving with (4.3),

$$\beta_1 u_1 = Ab = A(b - r) = ULV^H(b - r) + AA^H f, \tag{4.5}$$

and on multiplying by U^H,

$$\beta_1 e_1 = LV^H(b - r) \tag{4.6}$$

since $U^H AA^H f = LV^H A^H f = 0$. Finally substituting (4.6) in (4.5) gives $AA^H f = 0$.

Thus $f \in \mathcal{N}(A^H)$, and since Uy is orthogonal to $\mathcal{N}(A^H)$, $x = Uy$ is the minimum least squares solution if y can be found from (4.4). From (4.4) and (4.6) it follows that

$$L^H y = z, \quad Lz = \beta_1 e_1, \quad z \equiv V^H(b - r). \tag{4.7}$$

Clearly y cannot be found until the bidiagonalization is complete, but we are really only interested in finding Uy, so

$$x = Uy = UL^{-H}L^H y = Wz, \qquad W \equiv UL^{-H}, \tag{4.8}$$

and W can be computed a column at a time as the algorithm progresses, by solving

$$LW^H = U^H. \tag{4.9}$$

In this algorithm

$$A^H W = A^H UL^{-H} = V$$

from (4.3), and so is orthonormal; what is more, if $x_i \equiv Wz_i$, $z_i^T \equiv (\zeta_1, \cdots, \zeta_i, 0, \cdots, 0)$, then $r_i \equiv b - A^H x_i = b - A^H UL^{-H} z_i = b - Vz_i$ which can easily be updated as the computation progresses. Without computing r_i the algorithm can be written as follows. Remember that α_i cannot in theory be zero.

```
               $i := 1;$
               $\beta_1 u_1 := Ab;$        $\alpha_1 v_1 := A^H u_1;$
               $w_1 := u_1/\alpha_1;$
               $\zeta_1 := \beta_1/\alpha_1;$        $x := \zeta_1 w_1;$
general step: $i := i + 1;$
               $\beta_i u_i := Av_{i-1} - \alpha_{i-1}u_{i-1};$   if $\beta_i = 0$ then stop;
               $\alpha_i v_i := A^H u_i - \beta_i v_{i-1};$
               $w_i := (u_i - \beta_i w_{i-1})/\alpha_i;$
               $\zeta_i := -(\beta_i/\alpha_i)\zeta_{i-1};$      $x := x + \zeta_i w_i;$
               go to general step;
```

As usual the α_i and β_i are chosen so that $\|v_i\| = \|u_i\| = 1$. This algorithm requires storage space for the 3 vectors u, w, x of dimension m, and v of dimension n.

This is apparently a new algorithm; its advantages are its simplicity and economy and its direct relation to the bidiagonalization (2.1) and (2.2), so that the singular values relevant to the problem can be found easily from L. Good approximations to the singular values can also be found after only a fraction of the full number of steps.

If this is used computationally, then it is unlikely that either β_i or α_i will ever become negligible, but in practice the ζ_i eventually do, and so the computation can be curtailed, giving as accurate a solution as could be hoped for. A possible test is to check that $Ar = A(A^H x - b)$ is negligible. Computational experience shows that unless the condition of the problem is too bad for the machine precision, the algorithm will converge, but on problems with very close, very small singular values, far more than n steps may be required.

5. Comments and conclusions. The very natural bidiagonalization algorithm of Golub and Kahan [2], described here in (2.1) and (2.2), appears to be of significant theoretical importance. First, it allows singular values and vectors of A to be obtained easily, for instance by finding the singular value decomposition of the bidiagonal matrix L using the QR-like method given in [3]. This was the original motivation for deriving the algorithm. Now since the singular values are of such great importance in solving linear equations, it is not, in retrospect, so surprising that this bidiagonalization turns out to be very basically related to several methods for solving such equations. Two solutions of equations algorithms have been developed here using the bidiagonalization algorithm as a basis; these have the important advantage that the relevant singular values can be found if needed.

A very important point that does not appear to be generally realized is that the choice of initial vector for solution of equations in algorithms such as these is computationally all important. Choices other than those in (3.6) and (4.2) are perfectly satisfactory in theory, and just require small alterations in the algorithms involving one additional inner product. However, the ones given seem to be the only ones that can be relied upon to give convergence in practice. Note that when (3.6) is used it is assumed in the algorithm that $U^H b = \beta_1 e_1$, and yet it is known that this is not usually the case, that is, orthogonality is lost. If the true value of $U^H b$ is used in the algorithm instead of $\beta_1 e_1$, it is found in practice that the algorithm will not converge once orthogonality is lost. A similar result holds for (4.2).

This does not mean that another initial approximation to the solution cannot be used. Suppose $r + Ax = b$, $A^H r = 0$, and x_0 is a good approximation to x; then if $r_0 \equiv b - Ax_0$, it is clear we now want to solve

$$r + Ag = r_0, \qquad A^H r = 0,$$

so that $x = x_0 + g$. The method for solving this new problem can now start with the necessary initial vector, e.g., $\beta_1 u_1 = r_0$ in (3.6). Note here that if $x_0 = y_0 + z_0$ with $z_0 \in \mathcal{N}(A)$, $y_0 \in \mathcal{R}(A^H)$, then $x = y + z_0$ with $y \in \mathcal{R}(A^H)$, so that x is not the minimum least squares solution for nonzero z_0.

The computational importance of all the algorithms suggested here lies in their application to very large problems with a sparse matrix A. In this case the computation per step and storage is about as small as could be hoped, and theoretically the number of steps will be no greater than the minimum dimension of A. In practice, however, the number of steps required for a given accuracy can be much less or much greater than the expected number, depending, among other things, on the computer precision and the actual singular values. Rounding errors usually cause orthogonality of the vectors u_i and of the vectors v_i in (2.1) and (2.2) to be lost, but the singular values and vectors can still be accurately obtained, and the solution of the given equations can be accurately computed. This is an exact parallel to the computational performance of the Lanczos method for finding eigenvalues of symmetric matrices [7], even as far as often obtaining several computed singular values of A corresponding to one actual singular value.

Although the bidiagonalization algorithm has been extensively tested, only the solution of equations algorithm in § 4 has been tested, it being more straightforward than the extension of Craig's algorithm given in § 3. The suggestions on computational technique come from experience with the algorithm in § 4 and other algorithms closely related to these here.

The methods described here are for a general matrix A; if A is Hermitian, then these methods will be inefficient. The Lanczos algorithm takes advantage of the symmetry of A for finding eigenvalues (singular values), while the method of conjugate gradients is suited to positive definite systems of equations. A future paper by Paige and Saunders will show how advantage may be taken of symmetry for efficiently solving systems of equations with an indefinite symmetric matrix by using the vectors from the Lanczos process. The method given in that paper will also be applied to certain symmetric matrices arising from least squares problems, and in some cases will lead to computational algorithms very similar to the ones here. Since such particular examples will be included in that paper, and since their performances parallel that of the algorithms here, no computational examples need be included here.

An attempt has been made here to show the bidiagonalization algorithm introduced by Golub and Kahan is a very basic algorithm, and of great practical importance for large sparse matrices. The same can be said of the original Lanczos algorithm for tridiagonalizing Hermitian matrices; in fact, it could be argued that this latter algorithm is more basic, one reason being that the former can be derived in a simple manner from the tridiagonalization algorithm applied to the matrix B in (2.15).

Acknowledgments. My thanks to Gene Golub for making my stay at Stanford such a pleasant one, and also to him and Mike Saunders for discussions, ideas, and insight on this work.

I would like to pay tribute to George Forsythe, by whose courtesy I visited Stanford, and whose personality has left so many clear and strong marks on that place and the people working there. It is my regret that I arrived too late to meet him.

REFERENCES

[1] D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*, W. H. Freeman, London, 1963.

[2] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, this Journal, 2 (1965), pp. 205–224.

[3] G. Golub and C. Reinsch, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403–420.

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

[5] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Ibid., 45 (1950), pp. 255–282.

[6] ———, *Solution of systems of linear equations by minimized iterations*, Ibid., 49 (1952), pp. 33–53.

[7] C. C. Paige, *Computational variants of the Lanczos method for the eigenproblem*, J. Inst. Math. Appl., 10 (1972), pp. 373–381.

[8] J. K. REID, *On the method of conjugate graduents for the solution of large sparse systems of linear equations*, Proc. Conference on large sparse sets of linear equations, Academic Press, New York, 1971.

[9] ———, *The use of conjugate gradients for systems of linear equations possessing "property A"*, this Journal, 9 (1972), pp. 325–332.