Rank and Null Space Calculations Using Matrix Decomposition without Column Interchanges

Leslie V. Foster
Department of Mathematics and Computer Science
San Jose State University
San Jose, California 95192

Submitted by James H. Wilkinson

ABSTRACT

The most widely used stable methods for numerical determination of the rank of a matrix A are the singular value decomposition and the QR algorithm with column interchanges. Here two algorithms are presented which determine rank and nullity in a numerically stable manner without using column interchanges. One algorithm makes use of the condition estimator of Cline, Moler, Stewart, and Wilkinson and relative to alternative stable algorithms is particularly efficient for sparse matrices. The second algorithm is important in the case that one wishes to test for rank and nullity while sequentially adding columns to a matrix.

1. INTRODUCTION

The most widely used stable methods for numerical determination of the rank or nullity of an $M \times N$ matrix A are the singular value decomposition (SVD) and the QR algorithm with column interchanges (see [5], [17]). The singular value decomposition is considered more precise whereas the QR algorithm with column interchanges is more efficient and in practice usually is sufficiently precise [5]. However in certain applications both algorithms are inefficient. For example with a sparse matrix A the SVD does not take sufficient advantage of the sparsity of A and the QR algorithm with column interchanges forces an ordering of the columns based on numerical not sparsity considerations.

We will present two algorithms which determine rank and nullity in a numerically stable manner without using column interchanges in the sense that although columns are dropped the columns are not reordered. One

52 Vanderbilt Ave., New York, NY 10017 0024-3795/86/$3.50

THEOREM 4.2. *If s^k is the smallest singular value of A_k, and if e_k and $\hat{w}_k$ as in step 2 in the above algorithm are calculated on a computer with relative machine precision η, then* (3.8) *is true with $\gamma = \sqrt{N}$. Furthermore when $e_k \leqslant \varepsilon$ the calculated $\hat{w}_k$ satisfies* (3.9).

As shown in Section 3 and discussed earlier, this theorem implies that if ε is chosen to lie in a sufficiently large, usually moderate sized gap in the singular values, then the correct nullity is calculated in exact or finite precision calculations.

Algorithm 2 operates on matrices A_k of column dimension usually $N - p$ or less, whereas Algorithm 1 operates on matrices A_k of up to N columns. Thus for p large enough it can be shown that Algorithm 2 will be somewhat more efficient than 1. However, we feel that the most valuable use of Algorithm 2 will be for problems where we wish to sequentially add column vectors in a specified order until a matrix A is formed with a specified numerical nullity or rank. In this case Algorithm 1 is not appropriate but Algorithm 2 is quite natural. Suppose p is the desired specified ε nullity, and suppose that the calculated ε nullity is p when A has N columns. If we let $c = p/N$, then we may consider cases where the largest component in magnitude of each $\hat{w}_k$ is the first component (worst case) or a middle component (typical case). When using algorithm 2 to select N, the operation (multiplication) counts are less than:

$$MN^2 - \frac{N^3}{6} + \frac{N^3(15c - 21c^2 + 7c^3)}{6} \qquad \text{(worst case)}$$

$$MN^2 - \frac{N^3}{6} + \frac{N^3(6c - 9c^2 + 4c^3)}{6} \qquad \text{(typical case)} \qquad (4.3)$$

for large M and N, $M \geqslant N$. For sequential rank tests an alternative to Algorithm 2 would be to perform rank calculations via the SVD and successively update the SVD as columns are added. However when column n ($1 \leqslant n \leqslant N$) is added, an update of complete SVD would require $Mn^2 + n^3$ (for *each* n) multiplications [3], and so successive construction of complete SVDs would require an order of magnitude more operations than indicated in (4.3). Potential algorithms which update partial SVDs also appear relatively inefficient. When adding a column, to update the QR decomposition that results from column interchanges (SQRDC [5] or HFTI [17]) also appears inefficient. Such updating can require a complete reordering of the existing column order, and if so, apparently will be expensive.

We should mention that Algorithm 2 requires some storage in addition to the storage required for A. In particular, the Householder transformation, the

accepted null vectors, and the current R_k can be stored in the storage area for A, but the Givens transformations in general cannot be. These may require up to $N^2/2$ extra storage locations.

Finally, in this section we would like to briefly discuss an application—the minimal basis problem [8]—of the sequential rank tests which Algorithm 2 facilitates. This problem has many important applications in engineering control theory [8,9,16,26]. Furthermore, in the approaches described in [9,16,26] for solving the minimal basis problem, the rank and nullity of certain matrices are tested while sequentially adding columns to the matrices. For such tests our algorithm has proven the numerical stability and precision of rank determination, whereas the algorithms used in [9,16,26] do not. The interested reader is referred to the references [9,16,26] for more details on the minimal basis problem.

5. EXAMPLES, DISCUSSION AND CONCLUSIONS

The following simple example compares various methods of determining rank and nullity and illustrates some of our results.

EXAMPLE 5.1. Let A be

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & -a & 1 & 2 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & a \end{pmatrix},$$

and suppose we wish to determine the ε nullity and an ε null space of A for $\varepsilon = a^2$, where a is small. The singular values of A are $\sqrt{5}$, $\sqrt{2}$, a, and $a^2/\sqrt{10}$, where (as throughout the example) for simplicity we list our results accurate to the lowest order power of a only. Thus the a^2 nullity of A is one. An ε null vector for A is the singular vector $\mathbf{w} = (1, -1, -0.2a, -0.4a)^t$, and $\|A\mathbf{w}\|/\|\mathbf{w}\| = a^2/\sqrt{10}$.

Applying our implementation of algorithm 1, step 1 has $A_1 = R_1$ already triangular. In step 2 STRCO produces a trial null vector $\hat{\mathbf{w}}_1 = (1, -1, -0.2a, -0.4a)$ with $e_1 = \|A\hat{\mathbf{w}}_k\|/\|\hat{\mathbf{w}}_k\| = a^2/\sqrt{10} < a^2$. Therefore we accept $\mathbf{w}_1 = \hat{\mathbf{w}}_1$ as a null vector as in step 4, and we drop column 1 (or column 2) of R_1 (and A_1), retriangularizing using Givens transformations to obtain R_2. STRCO now produces the trial null vector $(0, 2, -1)^T$ for R_2 (and A_2) with $e_2 = a > \varepsilon$, and so we stop. Thus $N_\varepsilon(A) = 1$ is calculated, and $\mathbf{w}_1$ is in the calculated null space with $\|A\mathbf{w}_1\|/\|\mathbf{w}_1\| = a^2/\sqrt{10}$.

If we apply our implementation of Algorithm 2, for $k = 1$ we obtain $R_1 =$ (column 1 of A), $\hat{\mathbf{w}}_1 = (1)$, $e_1 = 1 > a^2$; for $k = 2$, $R_2 =$ (columns 1 and 2 of A), $\hat{\mathbf{w}}_k = (-1, 1)^T$, $e_2 = a/\sqrt{2} > a^2$; and for $k = 3$, $R_3 =$ (the first 3 columns of A), $\hat{\mathbf{w}}_3 = (-1, 1, a)^T/a$, $e_3 = a^2/\sqrt{2} < a^2$. Therefore we accept $\mathbf{w}_1 = (-1, 1, a, 0)^T$ as a null vector. For $k = 3$ in step 3 of the algorithm we drop column 1 (or 2) and retriangularize to obtain R_4 as a 4×2 triangular matrix. For $k = 4$, $\hat{\mathbf{w}}_4 = (a, 1)^T$ and $|t|/\|\hat{\mathbf{w}}_k\| = 1/\sqrt{1 + a^2} > \epsilon$, and finally, for $k = 5$, $\hat{\mathbf{w}}_5 = (0, -2, 1)$, $|t|/\|\hat{w}_k\| = a > a^2$. Thus $N_\epsilon(A) = 1$ is calculated, and $\mathbf{w}_1 = (-1, 1, a, 0)^T$ is in the calculated null space with $\|A\mathbf{w}_1\|/\|\mathbf{w}_1\| = a^2/\sqrt{2}$.

We can now illustrate some properties of other potential methods for null space determination. One scheme would be to accept a column of A as linearly dependent if the elements on a diagonal of the triangular portion of some QR factorization of A are smaller than some specified tolerance TOL. For this example such a scheme would select a null space of dimension 0 or 3, both incorrect. An alternative to this idea has been suggested by Heath [15]: if R_1 in $A = QR_1$ has a small diagonal entry, one drops the column of A corresponding to the first small entry. The result is retriangularized to form R_2, which is examined for small diagonal entries, repeating until some R_k has no small entries. If this algorithm is performed on our A with a tolerance TOL $= a^2$, no dependences are detected. If TOL is increased the calculated nullity will be 1 only for TOL in the narrow interval $a \leq \text{TOL} < \sqrt{5}\,a$ and then the null vector calculated by assuming the small diagonal entry is zero is $\mathbf{w} = (1, -1, 0, 0)^T$, satisfies $\|A\mathbf{w}\|/\|\mathbf{w}\| = a/\sqrt{2} \gg a^2$, and is not an a^2 null vector. A third potential scheme would be to use our Algorithm 2 except that if an entering column provokes $e_k <$ TOL, then consider the last entered column the culprit and drop it rather than some prior column. Applying this scheme to our example, we obtain $e_1 = 1$, $e_2 = a/\sqrt{2}$, $e_3 = a^2/\sqrt{2} < a^2$, $e_4 = a/\sqrt{2}$ and $e_5 = a^2/(2\sqrt{2}) < a^2$, and we incorrectly select an a^2 null space with 2 (almost linearly dependent) basis vectors. Finally we mention that the QR algorithm with column interchanges (SQRDC [5] or HFTI [17]) produces an R with diagonal entries $\cong 2$, 1, $a\sqrt{5}/2$, and $a^2/\sqrt{5}$, which do correctly reflect the nullity of A, and the column interchanges of [18] can be successfully used.

We should mention that Heath [14] notes a potential difficulty of his proposed scheme. He mentions, for example, the (in)famous example of Wilkinson [25], which has no small diagonal entries and is ill conditioned for moderate or larger N. The above example, or more simply

$$R = \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix},$$

illustrates that the size of diagonal entries of a matrix need not correspond to

the degree of ill-conditioning even for N very small. This, we presume, is the reason that Golub and Wilkinson [12] have reported "all algorithms based on search for negligible r_{ii} (diagonal entries of R) failed disastrously" in rank determinations when using QR decompositions without interchanges for certain Jordan canonical form calculations. Our algorithms are successful for the Wilkinson and other examples mentioned.

As additional tests of the accuracy of our Algorithm 1 we have run numerous numerical experiments on dense random matrices of size up to 100×100 formed by multiplying diagonal matrices by many (20 to 50) Householder transformations. We display in Figure 1 a typical graph which pictures the complete singular value spectrum as calculated by SSVDC in Linpack (open rectangles) and the set of approximate singular values (asterisks) calculated by our Algorithm 1. To save space we have presented only one graph with a particular spectrum shape. However we have run hundreds of matrices from the class described above, with a large variety of spectrum shapes, and our Algorithm 1 was *always* similar in precision to the enclosed graph. Across the entire spectrum, for hundreds of matrices and thousands of singular values,

$$\frac{1}{6} \leqslant \frac{(\text{approx. singular value } k)}{(\text{singular value } k \text{ from SSVDC})} \leqslant 9.$$

For smaller singular values, which are of more interest in many applications, the above ratio was typically bounded by $\frac{1}{2}$ and 2, not $\frac{1}{6}$ and 9. Our experiments were run on a CDC Cyber computer.

To discuss some of the related literature we might note that our algorithms can be used to construct basic solutions for rank deficient least squares problems with nullity p, that is, to construct a solution with p zero components. This problem is discussed in [23] (without any error analysis) and in [11]. In [11] the results are based on the SVD and the QR algorithm with interchanges and therefore apparently are not best suited to sparse problems or sequential testing of rank, as discussed earlier. We should also note that there are similarities between our algorithms and the algorithms of [18]—for example, the use of ideas related to the condition estimator of Cline, Moler, Stewart, and Wilkinson [4]. However, the approaches differ in that in [18] column interchanges are explicitly required, whereas in our algorithms they are not.

An important potential use of our results is to aid in solving rank deficient sparse least squares problems. Our implementation of Algorithm 2 described in Section 4 could be used in least squares problems in a manner similar to the use of the rank determination procedure of Heath mentioned above (see [15]). Work on numerical testing of such an implementation of our algorithm is

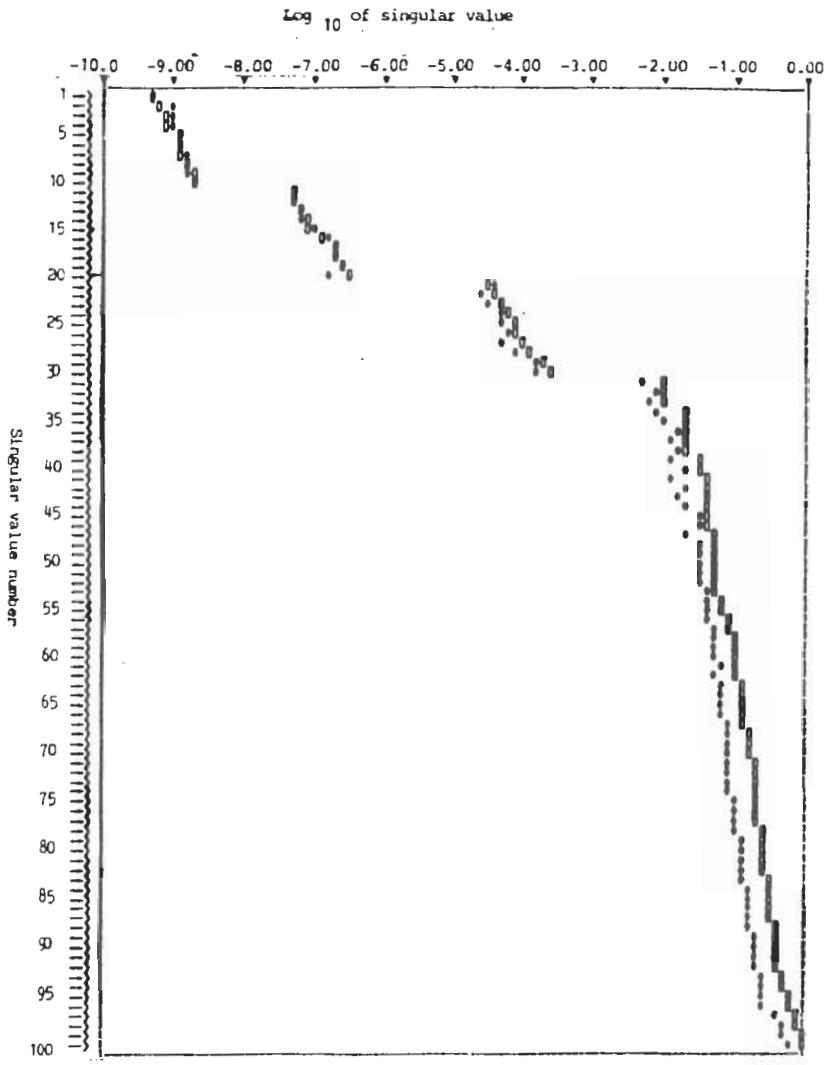


FIG. 1. True (open rectangles) and approximate (asterisks) singular values for a 100×100 matrix.

being planned. This implementation will be based on the least squares algorithm of George and Heath [10], which apparently is a very good sparse least squares algorithm. However, it is important to note that the ideas discussed in Section 3 are largely implementation independent. It is therefore likely that Algorithm 1 of Section 3 can be successfully used in conjunction with, for example, the method of Peters and Wilkinson [22] (see Bjork and Duff [1]), the normal equations approach or with as yet undeveloped methods. Also there is potential for use with iterative methods such as the method in [21]. What is required is the ability to identify the largest components of approximate null vectors and to drop columns from the problem—nothing more.

It is interesting to note that our Algorithm 1 and Algorithm 2, respectively, appear to involve concepts similar to the methods of backwards [6] and stepwise [6, 20] regression in statistics. However the criteria for selecting columns to drop are different in our and the statistical approaches.

Finally, we repeat that our algorithms require a usually moderate sized gap in the singular values in order to correctly determine the rank of A. However, we do not feel that this is a serious limitation, because if there is no such gap then the physical interpretation of numerical rank is unclear.

To summarize our results, we have presented two stable algorithms that do not require column interchanges for determining the rank and nullity of a matrix. Potential applications to sparse matrix rank determination and rank determination when sequentially adding columns have been discussed.

APPENDIX. PROOF OF THEOREM 4.2

First we should note that our notation and subscripts will probably be clearer if the reader considers them in relation to a simple example—perhaps Example 5.1. For notational convenience let us examine this theorem for $k = k^*$ so that we can let k ($k = 1, 2, \ldots, k^*$) be a variable as in the algorithm description. Also we will assume that M, the number of rows of A, is greater than or equal to N, the number of columns of A. This may be done without loss of generality for our proof, simply by adding zero rows to A when $M < N$.

We begin by assuming that the algorithm is run in inexact arithmetic. To distinguish the inexact arithmetic values from certain exact arithmetic values to be defined later, we will place dots over inexact arithmetic values that would be floating point numbers in a computer implementation. In this notation we wish to prove that if s^{k^*} is the smallest singular value of A_{k^*}

(where A_{k^*} is the submatrix of A defined below), then

$$\dot{e}_{k^*} \leqslant \sqrt{N}\, s^{k^*} + \Delta_A \tag{A.1}$$

and that if $\dot{e}_{k^*} \leqslant \varepsilon$ then

$$\frac{\|A_{k^*}\dot{\hat{\mathbf{w}}}_{k^*}\|}{\|\dot{\hat{\mathbf{w}}}_{k^*}\|} \leqslant \dot{e}_{k^*} + \Delta_A, \tag{A.2}$$

where $\Delta_A = \alpha\eta\|A\|$. Equation (A.2) is only the right half of the condition (3.9). We will not prove the left half of (3.9) here, since it is not needed in order to reach any of the conclusions of Section 3 or 4. For simplicity of notation we will drop the superscript k^* when we refer to s^{k^*} so that $s = s^{k^*}$ below.

Now for the algorithm run in inexact arithmetic let n_k and ρ_k, $k = 1,2,\ldots,k^*$, be the values of n and ρ at the kth entry in step 2, and furthermore, for convenience, let $n^* \equiv n_{k^*}$ and $\rho^* \equiv \rho_{k^*}$. Thus when $k = k^*$, step 3 will have been entered ρ^* different times. Let k_ρ, $\rho = 0,\ldots,\rho^* - 1$, be the values of k at the $(\rho+1)$st entry in step 3, and also for convenience define $k_{\rho^*} = k^* + 1$. Quantities of importance below will be $j_{\rho^*} = n^* - \rho^*$; for $\rho = 0,\ldots,\rho^* - 1$, $j_\rho = n_{k_\rho} - \rho - 1$ = the number of columns in $\dot{R}_{k_\rho}$; and for $\rho = 0,\ldots,\rho^* - 1$, i_ρ = the number of the column of $\dot{R}_{k_\rho}$ dropped in step 3 of the algorithm. Note that $i_\rho \leqslant j_\rho + 1$, $j_\rho - 1 \leqslant j_{\rho+1}$ and that $n_{k_\rho} = k_\rho - \rho$, $\rho = 0,\ldots,\rho^* - 1$. For Example 5.1 we would have for $k^* = 5$: $n^* = 4$; $\rho^* = 1$; $n_1 = 1$, $n_2 = 2$, $n_3 = 3$, $n_4 = 3$, $n_5 = 4$; $\rho_1 = \rho_2 = \rho_3 = 0$, $\rho_4 = \rho_5 = 1$; $k_0 = 3$, $k_1 = 6$; $j_0 = 2$, $j_1 = 3$; $i_0 = 1$.

Now consider the matrix A, and note that by an arbitrarily small perturbation of A we can form a matrix A' such that A' has no exact column dependences. To be specific assume that $\|A' - A\| \leqslant \eta\|A\|$. In the following for $1 \leqslant k \leqslant k^*$ we will let A'_k and A_k, respectively, be formed by removing the same columns of A' and A as were removed in forming $\dot{A}_k$ from A. Otherwise, unless explicitly indicated otherwise, all our notation without dots for real number quantities will refer to exact arithmetic application of our algorithm to A' in the following manner: enter step 3 only when $k = k_\rho$, $\rho = 0,\ldots,\rho^* - 1$, and then in step 3 drop column i_ρ of R_{k_ρ}, where k_ρ and i_ρ are the integer quantities determined by the inexact arithmetic algorithm. Therefore all the integer parameters (n_k, ρ_k, n^*, ρ^*, k_ρ, k^*, j_ρ, and i_ρ) defined above will be unchanged. Furthermore, for notational simplicity, for $\rho = 0,\ldots,\rho^* - 1$ let $T_\rho = \hat{R}_{k_\rho}$ = (the principal $j_\rho \times j_\rho$ submatrix of R_{k_ρ}), and let $T_{\rho^*} = R_{k^*}$.

Now define $c_0 = \infty$, and for $k = 1, 2, \ldots, k^*$ define $c_k = |t|/\|\hat{w}_k\|$, where t and $\hat{w}_k$ are as defined in step 2 of the algorithm. Note that $\hat{w}_k$ necessarily exists and that $c_k \neq 0$ by our assumption on the column independence of A'. Furthermore, by the logic of our algorithm it follows easily that $e_k = \min_{h \in S_k} c_h$ and $\hat{e}_k = \min_{h \in \hat{S}_k} c_h$, where $S_k = \{h : 1 \leqslant h \leqslant k,\ h \neq k_\rho,\ \rho = 0, \ldots, \rho_k - 1,\ h \text{ integer}\}$ and $\hat{S}_k = \{h : 0 \leqslant h \leqslant k-1,\ h \neq k_\rho,\ \rho = 0, \ldots, \rho_k - 1,\ h \text{ integer}\}$. Note that S_k and $\hat{S}_k$ contain n_k members, since a member of $\hat{S}_k$ is added every time step 4 is entered.

It is of use to relate the above c_k's to the columns of T_ρ^{-1}, $\rho = 0, 1, \ldots, \rho^*$. However, by our assumption on A', T_ρ^{-1} will exist, and furthermore it follows easily from the constructions of step 2 that for $k \in S_{k^*}$, $\hat{w}_k/t$ is the first $n_k - \rho_k$ entries in column $n_k - \rho_k$ of $T_{\rho_k}^{-1}$. That is, if for $1 \leqslant k \leqslant k^*$ we have $d_k \equiv 1/c_k = \|\hat{w}_k/t\|$, then for $k \in S_{k^*}$, d_k is the norm of the $(n_k - \rho_k)$th column of $T_{\rho_k}^{-1}$, and for $1 \leqslant k \leqslant k^*$, $\hat{e}_k = 1/\max_{h \in \hat{S}_k} d_h$ and $e_k = 1/\max_{h \in S_k} d_h$. For later convenience we also define $\hat{e}_{k^*+1} \equiv e_{k^*}$.

To show that $\hat{e}_k$ and e_k are related to the singular values of A_k, we define P_ρ^T, for $\rho = 0, \ldots, \rho^* - 1$, to be the $j_\rho \times j_\rho$ column permutation matrix that moves the i_ρth column of a matrix to the last column and moves columns $i_\rho + 1$ to j_ρ one column to the left. (If $i_\rho \geqslant j_\rho$ let $P_\rho^T = I$.) Also, for $\rho = 0, \ldots, \rho^* - 1$ let U_ρ^T be a $j_\rho \times j_\rho$ unitary matrix such that $U_\rho^T T_\rho P_\rho^T$ is triangular. Then by the constructions of the algorithm it follows that the principal $(j_\rho - 1) \times (j_\rho - 1)$ submatrices of $U_\rho^T T_\rho P_\rho^T$ and $T_{\rho+1}$ are the same, and furthermore, since $T_{\rho+1}$ is triangular, for $\rho = 0, \ldots, \rho^* - 1$,

$$\begin{aligned} &\text{the principal } (j_\rho - 1) \times (j_\rho - 1) \text{ submatrices} \\ &\text{of } P_\rho T_\rho^{-1} U_\rho \text{ and } T_{\rho+1}^{-1} \text{ are the same.} \end{aligned} \tag{A.3}$$

We now, for $\rho \in \{1, \ldots, \rho^*\}$, define $\hat{W}_\rho$, to be a $j_\rho \times j_\rho$ diagonal matrix with 1's on the diagonal from positions $j_{\rho-1}$ to j_ρ ($\hat{W}_\rho$ is a zero matrix if $j_\rho < j_{\rho-1}$), and W_ρ to be a $j_\rho \times j_{\rho-1}$ matrix with 1's on its diagonal. Next, for some $\rho \in \{1, \ldots, \rho^*\}$ let $\mathbf{x}_\rho$ be a vector with j_ρ components. Since we are using subscripts currently to indicate different vectors and matrices, we will use the notation $(\mathbf{x}_\rho)_h$ to indicate component h of $\mathbf{x}_\rho$ and $(A)_h$ to indicate column h of matrix A. Then by (A.3) and the triangularity of T_ρ,

$$\begin{aligned} T_\rho^{-1}\mathbf{x}_\rho &= \sum_{h=1}^{j_\rho} \left(T_\rho^{-1}\right)_h (\mathbf{x}_\rho)_h \\ &= \sum_{h=1}^{j_{\rho-1}-1} W_\rho \left(P_{\rho-1} T_{\rho-1}^{-1}\right)(U_{\rho-1})_h (\mathbf{x}_\rho)_h + \sum_{h=j_{\rho-1}}^{j_\rho} \left(T_\rho^{-1}\right)_h (\mathbf{x}_\rho)_h. \end{aligned}$$

If we define $\mathbf{x}_{\rho-1} = \sum_{h=1}^{j_{\rho-1}-1}(U_{\rho-1})_h(\mathbf{x}_\rho)_h$ and $\hat{\mathbf{x}}_\rho = \hat{W}_\rho \mathbf{x}_\rho$, then it follows that

$$T_\rho^{-1}\mathbf{x}_\rho = W_\rho P_{\rho-1} T_{\rho-1}^{-1}\mathbf{x}_{\rho-1} + T_\rho^{-1}\hat{W}_\rho \hat{\mathbf{x}}_\rho. \tag{A.4}$$

Furthermore, since $U_{\rho-1}$ is unitary, it follows that $\|\mathbf{x}_{\rho-1}\|^2 = (\mathbf{x})_1^2 + \cdots + (\mathbf{x})_{j_{\rho-1}-1}^2$ and therefore

$$\|\mathbf{x}_\rho\|^2 = \|\mathbf{x}_{\rho-1}\|^2 + \|\hat{\mathbf{x}}_\rho\|^2. \tag{A.5}$$

Now for $\rho \in \{0,\ldots,\rho^*\}$ let $\mathbf{x}_\rho$ be any j_ρ-vector with $\|\mathbf{x}_\rho\| = 1$. Also, for $\gamma = 0,\ldots,\rho-1$, define $\hat{P}_\gamma = W_\rho P_{\rho-1} W_{\rho-1} P_{\rho-2} \cdots W_{\gamma+1} P_\gamma$, and define $\hat{P}_\rho = I$. Then we may apply (A.4) inductively to conclude that

$$T_\rho^{-1}\mathbf{x}_\rho = \hat{P}_0 T_0^{-1}\mathbf{x}_0 + \sum_{\gamma=1}^{\rho} \hat{P}_\gamma T_\gamma^{-1}\hat{W}_\gamma \hat{\mathbf{x}}_\gamma. \tag{A.6}$$

where by (A.5)

$$\|\mathbf{x}_0\|^2 + \|\hat{\mathbf{x}}_1\|^2 + \cdots + \|\hat{\mathbf{x}}_\rho\|^2 = \|\mathbf{x}_\rho\|^2 = 1. \tag{A.7}$$

Furthermore, for $\gamma = 1,\ldots,\rho$, since $T_\gamma^{-1}\hat{W}_\gamma\hat{\mathbf{x}}_\gamma$ will only involve columns $j_{\gamma-1}$ to j_γ of T_γ, and since for any vector $\mathbf{x}$ with j_γ components $\|\hat{P}_\gamma \mathbf{x}\| = \|\mathbf{x}\|$, due to the definitions of k_ρ and j_ρ and due to (A.6), we have

$$\begin{aligned}
\|T_\rho^{-1}\mathbf{x}_\rho\| &\leq \sqrt{j_0}\left(\max_{h=1,\ldots,k_0-1} d_h\right)\|\mathbf{x}_0\| \\
&\quad + \sum_{\gamma=1}^{\rho} \sqrt{j_\gamma - j_{\gamma-1}+1}\left(\max_{h=k_{\gamma-1}+1,\ldots,k_\gamma-1} d_h\right)\|\hat{\mathbf{x}}_\gamma\| \\
&\leq \frac{1}{\hat{e}_{k_\rho}}\left(\sqrt{j_0}\|\mathbf{x}_0\| + \sum_{\gamma=1}^{\rho}\sqrt{j_\gamma - j_{\gamma-1}+1}\,\|\hat{\mathbf{x}}_\gamma\|\right).
\end{aligned} \tag{A.8}$$

However, since $j_0 + \sum_{\gamma=1}^{\rho}(j_\gamma - j_{\gamma-1}+1) = n_\rho$, due to (A.7) it follows easily from (A.8) that for $\rho = 0,\ldots,\rho^*-1$ we have $\|T_\rho^{-1}\mathbf{x}_\rho\| \leq \sqrt{n_\rho}/\hat{e}_{k_\rho}$ and for $\rho = \rho^*$ we have $\|T_\rho^{-1}\mathbf{x}_\rho\| \leq \sqrt{n^*}/\hat{e}_{k^*+1} \leq \sqrt{n^*}/e_{k^*}$. Since $n^* \leq N$ and for

$\rho = 0, \ldots, \rho^* - 1$ we have $e_{k^*} \leqslant \hat{e}_{k_\rho}$ we may conclude that for $\rho = 0, \ldots, \rho^*$,

$$\|T_\rho^{-1}\| \leqslant \frac{\sqrt{N}}{e_{k^*}}. \tag{A.9}$$

Finally if we let s' be the smallest singular value of A'_{k^*}, then since A'_{k^*} and R_{k^*} are unitarily equivalent and since T_{ρ^*} is just the first $n^* - \rho^*$ rows of the matrix R_{k^*} with $n^* - \rho^*$ columns, we may conclude from (A.9) by standard arguments that $e_{k^*} \leqslant \sqrt{N}s'$. In addition, since $\|A' - A\| \leqslant \eta\|A\|$, it follows by standard results that

$$e_{k^*} \leqslant \sqrt{N}s + \sqrt{N}\eta\|A\|, \tag{A.10}$$

where s is the smallest singular value of A_{k^*}.

Now to show (A.1) let us assume that the minimum c_h, $h \in S_{k^*}$, occurs at $h = k$, so that $e_{k^*} = c_k = 1/d_k$. Also recall the convention used in the proof that in step 2 of the algorithm $\hat{\dot{R}}_k$, $\dot{R}_k$, $\dot{\mathbf{r}}$, $\dot{t}$, $\dot{\mathbf{x}}$, and $\hat{\dot{\mathbf{w}}}_k$ are inexact arithmetic values when the algorithm is applied to A, and $\hat{R}_k$, R_k, $\mathbf{r}$, t, $\mathbf{x}$, and $\mathbf{w}_k$ are corresponding values resulting from using exact arithmetic and A'. Since the construction of $\hat{\dot{R}}_k$ involves the application of a sequence of Householder and Given transformations to certain columns of A, since $\|A' - A\| \leqslant \eta\|A\|$, and due to properties of triangular matrices, it follows from standard results [24] that for the calculated vector $\dot{\mathbf{x}}$ a matrix E_1 exists with $(\hat{R}_k + E_1)\dot{\mathbf{x}} = \mathbf{r}$ in exact arithmetic, where $\|E_1\| \leqslant \alpha_1\eta\|A\|$. Here α_1 and below each of the α's used represent parameters that can be chosen to depend only on the size of A and are not large. Also $|t - \dot{t}| \leqslant \alpha_2\eta\|A\|$ follows easily. Since $\hat{R}_k\mathbf{x} = \mathbf{r}$, it follows that $\mathbf{x} - \dot{\mathbf{x}} = \hat{R}_k^{-1}E_1\dot{\mathbf{x}}$. Furthermore, due to this equation, since $\hat{R}_k$ is a submatrix of T_{ρ_k}, since $\|\hat{\mathbf{w}}_k - \hat{\dot{\mathbf{w}}}_k\| = \|\mathbf{x} - \dot{\mathbf{x}}\|$, since $\|\dot{\mathbf{x}}\| \leqslant \|\hat{\dot{\mathbf{w}}}_k\|$, since by assumption $1/e_{k^*} = d_k = \|\mathbf{w}_k\|/|t|$, and by (A.9), it then follows that $\|\hat{\mathbf{w}}_k - \hat{\dot{\mathbf{w}}}_k\| \leqslant \|T_{\rho_k}^{-1}\|\alpha_1\eta\|A\|\,\|\hat{\dot{\mathbf{w}}}_k\| \leqslant \sqrt{N}(1/e_{k^*})\alpha_1\eta\|A\|\,\|\hat{\dot{\mathbf{w}}}_k\| = \sqrt{N}(\|\hat{\mathbf{w}}_k\|/|t|)\alpha_1\eta\|A\|\,\|\hat{\dot{\mathbf{w}}}_k\|$. These equations imply that $|t|/\|\hat{\dot{\mathbf{w}}}_k\| \leqslant |t|/\|\hat{\mathbf{w}}_k\| + \alpha_1\sqrt{N}\eta\|A\|$. Then since $|\dot{t}| \leqslant |t| + \alpha_2\eta\|A\|$ and since $\|\hat{\dot{\mathbf{w}}}_k\| \geqslant 1$, it follows that $\dot{t}/\|\hat{\dot{\mathbf{w}}}_k\| \leqslant t/\|\hat{\mathbf{w}}_k\| + \alpha_3\eta\|A\| = e_{k^*} + \alpha_3\eta\|A\|$, where $\alpha_3 = \alpha_2 + \alpha_1\sqrt{N}$. Finally, due to this inequality, by (A.10), and since by construction $\dot{e}_{k^*} \leqslant \dot{t}/\|\hat{\dot{\mathbf{w}}}_k\|$, we conclude that (A.1) is true with $\alpha = \sqrt{N} + \alpha_3$.

Now to prove (A.2) let us assume that $e_{k^*} \leqslant \varepsilon$. Then since by construction $\hat{\dot{e}}_{k^*} > \varepsilon$, it follows that in step 2 of the algorithm $\dot{e}_{k^*} = |\dot{t}|/\|\hat{\dot{\mathbf{w}}}_k\|$ for $k = k^*$. For this k it easily follows that $|t - \dot{t}| \leqslant \alpha_5\eta\|A\|$ and $\mathbf{r} - \hat{R}_k\dot{\mathbf{x}} = \hat{R}_k(\mathbf{x} - \dot{\mathbf{x}}) = E_2\dot{\mathbf{x}}$ with $\|E_2\| \leqslant \alpha_4\eta\|A\|$, as in our arguments to prove (A.1). But then $\|R_k\hat{\dot{\mathbf{w}}}_k\| = \|[(\mathbf{r} - \hat{R}_k\dot{\mathbf{x}})^T, t]\| = \|[(E_2\dot{\mathbf{x}})^T, t]\|$, so that $\|A'_k\hat{\dot{\mathbf{w}}}_k\| = \|R_k\hat{\dot{\mathbf{w}}}_k\| \leqslant \|E_2\|\,\|\dot{\mathbf{x}}\| + |t| \leqslant \|E_2\|\,\|\dot{\mathbf{x}}\| + |t - \dot{t}| + |\dot{t}|$. However, since $\|\hat{\dot{\mathbf{w}}}_k\| \geqslant \|\dot{\mathbf{x}}\|$, since $\|\hat{\dot{\mathbf{w}}}_k\| \geqslant 1$, and by our

bound on $|t - \hat{t}|$, then $\|A'_k\hat{\mathbf{w}}_k\|/\|\hat{\mathbf{w}}_k\| \leq \|E_2\| + \alpha_5\eta\|A\| + |\hat{t}|/\|\hat{\mathbf{w}}_k\| \leq \dot{e}_{k*} + (\alpha_4 + \alpha_5)\eta\|A\|$. Finally, since $\|A - A'\| \leq \eta\|A\|$, we conclude easily that (A.2) is true with $\alpha = \alpha_4 + \alpha_5 + 1$. The difference in the values of α used in proving (A.1) and (A.2) can be resolved by choosing a new α which is the maximum of the two.

REFERENCES

1 A. Bjork and I. S. Duff, A direct method for the solution of sparse linear least squares problems, *Linear Algebra Appl.* 34:43–67 (1980).

2 A. Bjork, Methods for sparse linear least squares problems, in *Sparse Matrix Computations* (J. Bunch and D. Rose, Eds.), Academic, 1976, pp. 177–199.

3 J. Bunch and C. Nielsen, Updating the singular value decomposition, *Numer. Math.* 31:111–129 (1978).

4 A. Cline, C. Moler, G. Stewart, and J. Wilkinson, An estimate for the condition number of a matrix, *SIAM J. Numer. Anal.* 16:368–375 (1979).

5 J. Dongarra, C. Moler, J. Bunch, and G. Stewart, LINPACK *User's Guide*, SIAM, Philadelphia, 1979.

6 N. Draper and H. Smith, *Applied Regression Analysis*, Wiley, 1966.

7 I. Duff and J. Reid, A comparison of some methods for the solution of sparse overdetermined systems of linear equations, *Inst. Math. Appl.* 17:267–280 (1976).

8 G. Forney, Jr., Minimal basis of rational vector spaces, with applications to multivariable systems, *SIAM J. Control* 13:493–520 (1975).

9 L. Foster, A practical solution to the minimal design problem, *IEEE Trans. Automat. Control* AC-24:449–454 (1979).

10 A. George and M. Heath, Solution of sparse linear least squares problems using Givens rotations, *Linear Algebra Appl.* 34:69–83 (1980).

11 G. Golub, V. Klema, and G. Stewart, Rank degeneracy and least squares problems, Stanford Univ. Technical Report STAN-CS-76-559, 1976.

12 G. Golub and J. Wilkinson, Ill-conditioned eigensystems and the computation of the Jordan canonical form, *SIAM J. Numer. Anal.* 13:578–619 (1976).

13 G. Golub, Some modified matrix eigenvalue problems, *SIAM Rev.* 15:318–334 (1973).

14 R. Grimes and J. Lewis, Condition number estimators for sparse matrices, *SIAM J. Sci. Statist. Comput.* 2:384–388 (1981).

15 M. Heath, Some extensions of an algorithm for sparse linear least squares problems, *SIAM J. Sci. Statist. Comput.* 3:223–237 (1982).

16 S. Kung and T. Kailath, Fast projection methods for minimal design problems in linear systems theory, *Automatica* 66:399–403 (1980).

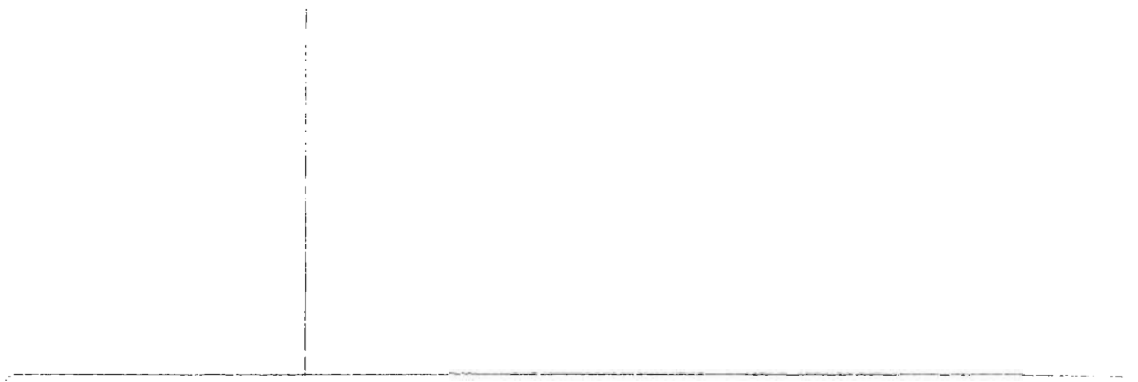
17 C. Lawson and R. Hanson, *Solving Least Squares Problems*, Prentice-Hall, 1974.

18 T. Manteuffel, An interval analysis approach to rank determination in linear least squares problems, *SIAM J. Sci. Statist. Comput.* 2:335–348 (1981).

19 D. O'leary, Estimating matrix condition numbers, *SIAM J. Sci. Statist. Comput.* 1:205–209 (1980).

20 M. Osborne, On the computation of stepwise regressions, *Austral. Comput. J.* 8:61–68 (1976).
21 C. Paige and M. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares, *ACM Trans. Math. Software* 8:43–71 (1982).
22 G. Peters and J. Wilkinson, The least squares problem and psuedo-inverses, *Comput. J.* 13:309–316 (1970).
23 J. Rosen, Minimum and basic solutions to singular linear systems, *SIAM J. Appl. Math.* 12:156–162 (1964).
24 G. Stewart, *Introduction to Matrix Computations*, Academic, 1974.
25 J. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford U.P., 1965.
26 L. Foster, A comparison of solutions to the minimal design problem, *IEEE Trans. Automat. Control*, Feb. 1984.

Received 25 March 1983; revised 25 October 1984



algorithm makes use of the condition estimator of Cline, Moler, Stewart and Wilkinson [4] and is well suited for application to large sparse matrices whose rank defects are not large. The second algorithm presented uses certain matrix inverses for condition estimates and is of importance when one wishes to test for rank and nullity while sequentially adding columns to a matrix. As we will describe later this last problem arises in certain approaches to the minimal basis problem [8, 9, 16] of engineering control theory.

In this paper in Section 2 we define and discuss numerical rank and nullity as well as present essential background material. In Section 3 we present a description of our algorithms as well as their analysis in relation to singular values. In Section 4 we recommend specific implementations and prove numerical stability. Finally in Section 5 we present an example, discussion, and conclusions. Section 5 will further discuss related literature but here we might mention that Heath [15] considers sparse matrices with rank deficiencies using an approach somewhat similar to ours. As we will see our approach can be substantially more accurate in the determination of rank determination.

Three norms will be used in developing our results. For $\mathbf{x} \in R^N$ these are defined by

$$\|\mathbf{x}\|_2 = \left(\sum x_i^2\right)^{1/2}, \qquad \|\mathbf{x}\|_1 = \sum |x_i|, \qquad \|\mathbf{x}\|_\infty = \max(|x_i|).$$

The first of these, the usual Euclidean vector norm, will be so basic that the suffix 2 will be omitted. Each of the vector norms provides a corresponding induced matrix norm defined by

$$\|A\|_p = \max \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p} \qquad (p = 2, 1, \infty),$$

and again the suffix 2 will be omitted.

2. NUMERICAL RANK AND NULLITY

As is well known, on a finite precision computer if $M > N$ it is highly probable that no column of an $M \times N$ matrix A will be exactly linearly dependent on the other columns of A and no vector $\mathbf{x}$ will exactly satisfy $A\mathbf{x} = 0$. Therefore for computer calculations we need to define the numerical rank and nullity of A. We will do so with respect to a tolerance ε and the Euclidean matrix norm $\| \ \|$. Although no single definition of numerical rank is best for all applications, rank determination based on singular values is very widely used and provides a point of comparison for other methods of rank

determination (see [11]). As we see, the following definition will lead to singular values:

DEFINITION 2.1. *The numerical ε rank of an $M \times N$ matrix A is defined by*

$$R_\varepsilon(A) = \min_B \{\operatorname{rank}(B) : \|A - B\| \leqslant \varepsilon\},$$

and the numerical ε nullity $N_\varepsilon(A)$ is defined by

$$N_\varepsilon(A) = \max\{\operatorname{nullity}(B) : \|A - B\| \leqslant \varepsilon\}.$$

In order to help clarify as well as use Definition 2.1, several equivalent formulations of numerical rank and nullity are important. To describe these we first note that any $M \times N$ matrix A has a singular value decomposition $A = UDV^T$, where ^{T} indicates transpose, D is an $M \times N$ diagonal matrix, and U and V are, respectively, $M \times M$ and $N \times N$ orthogonal matrices. When $M \geqslant N$ we will call the diagonal entries $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_N$ the singular values of A. For later convenience we choose to order the singular values so that the first singular value is the smallest. When $M < N$ we will say that A has N singular values: $s_1 = s_2 = \cdots = s_{N-M} = 0$ and the diagonal entries of A, which we order $s_{N-M+1} \leqslant \cdots \leqslant s_{N-1} \leqslant s_N$. We have chosen to define N singular values even when $M < N$. This convention, although nonstandard, will substantially simplify our later description. The columns of U and of V, respectively, are termed left and right singular vectors of A. Given a vector y in Euclidean M space R^M, and a subspace S of R^M, we also define the distance between y and S by $\operatorname{dist}(y, S) = \min\{\|y - z\| : z \in S\}$. We then have:

THEOREM 2.2. *For an $M \times N$ matrix A*

$$N_\varepsilon(A) = \max_S \{\operatorname{dimension}(S) : S \text{ is a subspace of } R^N \text{ such that } x \in S, \quad x \neq 0 \Rightarrow \frac{\|Ax\|}{\|x\|} \leqslant \varepsilon\}, \tag{2.3}$$

$$N_\varepsilon(A) = \text{number of singular values of } A \text{ that are } \leqslant \varepsilon, \tag{2.4}$$

$$R_\varepsilon(A) = \min_S \{\operatorname{dimension}(S) : S \text{ is a subspace of } R^M \text{ such that } x \in R^N,\ x \neq 0 \Rightarrow \frac{\operatorname{dist}(Ax, S)}{\|x\|} \leqslant \varepsilon\} \tag{2.5}$$

$$R_\varepsilon(A) = \text{number of singular values of } A \text{ that are } > \varepsilon, \tag{2.6}$$

$$R_\varepsilon(A) = N - N_\varepsilon(A). \tag{2.7}$$

Proof. The results (2.4) and (2.6) are well known [11]. Although the author has not seen (2.3) or (2.5) elsewhere in the forms presented here, these results and (2.7) follow easily from standard results in linear algebra. In the interest of space, we will not present the straightforward proofs here. ■

The results (2.3) and (2.5) are geometrical—(2.3) stating that ε nullity is the largest dimension of a space that is approximately annihilated by A, and (2.5) stating that ε rank is the smallest dimension of a space that approximates well vectors in the range of A. A subspace S of dimension $N_\varepsilon(A)$ that satisfies (2.3) will be called an ε null space of A. The results (2.4) and (2.6) suggest one method of calculating the numerical rank of A—via the SVD. For many of our results we will find (2.3) and (2.7) more useful.

Some other well-known results that will be of use to us follow. If $\mathbf{x} \in R^N$, then

$$\|\mathbf{x}\|_\infty \leqslant \|\mathbf{x}\| \leqslant \sqrt{N}\|\mathbf{x}\|_\infty \quad \text{and} \quad \frac{\|x\|_1}{\sqrt{N}} \leqslant \|\mathbf{x}\| \leqslant \|\mathbf{x}\|_1. \tag{2.8}$$

Furthermore, if $\mathbf{a}_1, \ldots, \mathbf{a}_N$ are the columns of an $M \times N$ matrix A and $K = \max(\|\mathbf{a}_1\|, \ldots, \|\mathbf{a}_N\|)$ then

$$K \leqslant \|A\| \leqslant K\sqrt{N}. \tag{2.9}$$

An additional characterization of the singular values of an $M \times N$ matrix A is the Courant-Fischer characterization: $s_k = \min_S(\max_{\mathbf{x} \in S}\|A\mathbf{x}\|/\|\mathbf{x}\|)$, where the minimum is over subspaces S of dimension k. Furthermore if an $M \times N$ matrix A has singular values $s_1, \ldots, s_N$ and if B is a matrix formed by deleting a column of A, then the singular values $\hat{s}_1, \ldots, \hat{s}_{N-1}$ of B interleave those of A: $s_1 \leqslant \hat{s}_1 \leqslant s_2 \leqslant \hat{s}_2 \leqslant \cdots \leqslant \hat{s}_{N-1} \leqslant s_N$. It is also useful to note that $\|A\| = s_N$, and if $s_1 \neq 0$ and if A^+ is the generalized inverse of A [$AA^+A = A$, $A^+AA^+ = A^+$, $(AA^+)^T = AA^+$, $(A^+A)^T = A^+A$], then $\|A^+\| = 1/s_1$. In addition the definition of the condition of A is $\text{cond}(A) = \|A\|\|A^+\|$ for the two norm, and $\text{cond}_1(A)$ and $\text{cond}_\infty(A)$ are defined similarly. Finally we should note that the QR factorization $A = QR$, Q unitary and R right trapezoidal, is of use in the sequel.

3. ALGORITHMS AND COMPARISON WITH SINGULAR VALUES

In this section we wish to describe our algorithms and show that they calculate rank with a precision comparable to that of singular values without

calculating singular values. The principle result of this section and we feel of the paper is a demonstration that this may be done by an algorithm based on dropping columns of a matrix A corresponding to largest magnitude components of approximate null vectors of A. As we will describe, these approximate null vectors can be determined by using the condition estimator of Cline, Moler, Stewart, and Wilkinson, by using certain matrix inverses or by other methods.

We first outline objectives to be met in an algorithm which seeks to determine the ε nullity of an $M \times N$ matrix A with a reliability comparable with that achieved by the use of the singular value decomposition. The algorithm should compute successive approximate singular vectors $\mathbf{w}_k$ $(k = 1,2,\ldots)$ and corresponding approximate singular values e_k defined by

$$e_k = \frac{\|A\mathbf{w}_k\|}{\|\mathbf{w}_k\|} \qquad (\text{with} \quad e_1 \leqslant e_2 \leqslant \cdots \text{ holding approximately}).$$

If the first e_k to exceed ε is e_{p+1}, so that

$$e_k \leqslant \varepsilon, \quad k = 1,\ldots,p \quad \text{and} \quad e_{p+1} > \varepsilon, \tag{3.1}$$

then p is to be regarded as the ε nullity as determined by the algorithm. For the approximation to be satisfactory from this point of view two conditions must be met. First it must be true that

$$e_k \leqslant \gamma s_k \qquad (k = 1,\ldots,p+1) \tag{3.2}$$

with some fairly modest value of γ (i.e., the e_k must not be substantial overestimates of the s_k, the true singular values). Secondly, since the true singular vectors are orthogonal (i.e., maximally linearly independent), the approximate singular vectors should also be far removed from linear dependence; otherwise the e_k would have no real significance. To insure this we require that

$$\text{cond}_1(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_p) = \text{cond}_1(W) \leqslant C, \tag{3.3}$$

where again C is a modest constant. The conditions (3.1) to (3.3) lead to:

THEOREM 3.4. *Let L_1 be the number of singular values of A less than or equal to ε/γ, and let L_2 be the number of singular values less than or equal to $C\varepsilon$. If the conditions* (3.1) *and* (3.2) *are satisfied, then $p \geqslant L_1$. If*

conditions (3.1) *and* (3.3) *are satisfied, then* $p \leq L_2$. *If* (3.1), (3.2), (3.3) *are true and* A *has a gap in its singular values with no singular values in*

$$\frac{\varepsilon}{\gamma} < s \leq C\varepsilon, \tag{3.5}$$

then $p = N_\varepsilon(A)$.

Proof. To prove (3.1) and (3.2) imply $p \geq L_1$, note that $\varepsilon < e_{p+1}$ and $e_{p+1} \leq \gamma s_{p+1}$ imply $\varepsilon/\gamma < s_{p+1}$. Then $p \geq L_1$ follows immediately.

To show (3.1) and (3.3) imply $p \leq L_2$, let S be the subspace spanned by $\mathbf{w}_1, \ldots, \mathbf{w}_p$. Then for $\mathbf{w} = W\mathbf{y} \in S$

$$\|A\mathbf{w}\| \leq \sum_{i=1}^{p} \|A\mathbf{w}_i\| |y_i| \leq \max_{i=1,\ldots,p} \|A\mathbf{w}_i\| \|\mathbf{y}\|_1$$

$$\leq \varepsilon \Big(\max_{i=1,\ldots,p} \|\mathbf{w}_i\| \Big) \|\mathbf{y}\|_1,$$

using (3.1) and properties of matrix norms. Furthermore $\operatorname{cond}_1(W) = \|W\|_1 \min_{\mathbf{z} \neq 0}(\|\mathbf{z}\|_1 / \|W\mathbf{z}\|_1)$ is a property of condition numbers and so $C \geq \|W\|_1 \|\mathbf{y}\|_1 / \|\mathbf{w}\|_1$. Therefore for any $\mathbf{w} \in S$, $\|A\mathbf{w}\|/\|\mathbf{w}\| \leq \varepsilon(\max_{i=1,\ldots,p} \|\mathbf{w}_i\|)\|\mathbf{y}\|_1 C/(\|W\|_1 \|\mathbf{y}\|_1) \leq \varepsilon C$. Since S is a subspace of dimension p, by the Courant-Fischer theorem the singular value s_p of A satisfies $s_p \leq C\varepsilon$. Therefore A has at least p singular values less than or equal to $C\varepsilon$, and $p \leq L_2$ follows.

If (3.1)–(3.3) and (3.5) are true, then $L_1 = L_2$, and $p = L_1 = L_2 = N_\varepsilon(A)$ follows immediately. ∎

The importance of the result is the following: If our rank determination is based on approximate singular values and if (3.1) is true, then it is natural to let the estimated ε nullity determined by the algorithm be p. If (3.1)–(3.3) are true and ε has been chosen to lie in a gap in the singular value of A as in (3.5), then this calculated ε nullity equals the ε nullity based on singular values. When ε is close to a singular value, so that (3.5) is not true, then the calculated ε nullity may not agree with the ε nullity determined by singular values. However, we do not feel that this is a serious limitation in our algorithm, since if ε is not contained in a reasonable sized gap in the singular values of A, then even when singular values are used the physical interpretation of numerical ε rank is unclear [11]. Also, if (3.5) is not true, then Theorem 3.4 still provides bounds on the calculated nullity.

We would like to develop algorithms that are simple and that insure that (3.2) and (3.3) are true with γ and C not large. In order to fully develop these algorithms we will describe several modifications related to the condition (3.2). The first is introduced to allow inclusion of error analysis in our development. We will assume that e_k, our kth approximate singular value, satisfies

$$e_k \leq \gamma s_k + \Delta_A, \qquad k = 1, \ldots, p+1 \tag{3.6}$$

rather than (3.2). Here $\Delta_A = \alpha\eta\|A\|$, where η is the relative machine precision and α is assumed to depend only on the dimension of A and is not large. Furthermore, for $e_k \leq \varepsilon$ we assume that

$$e_k - \Delta_A \leq \frac{\|A\mathbf{w}_k\|}{\|\mathbf{w}_k\|} \leq e_k + \Delta_A \tag{3.7}$$

instead of the stronger condition $e_k = \|A\mathbf{w}_k\|/\|\mathbf{w}_k\|$. Of course, if $\Delta_A = 0$ then (3.6) and (3.7) are equivalent to our earlier conditions, and if $\Delta_A \neq 0$ then (3.6) and (3.7) lead to a theorem similar to Theorem 3.4, as we will see.

Another modification related to (3.2) is now introduced, since we will find it easiest to estimate an interior singular value s_k of A by first estimating the smallest singular value s^k of certain submatrices A_k of A. Note the use of the superscript to distinguish s^k from s_k. In our algorithm we will assume that e_k our estimate of s_k is related to s^k by

$$e_k \leq \gamma s^k + \Delta_A, \qquad k = 1, \ldots, p+1 \tag{3.8}$$

with Δ_A as earlier. In addition let us assume that when $e_k \leq \varepsilon$ our algorithm calculates an approximate null vector $\hat{\mathbf{w}}_k$ of the submatrix A_k such that

$$e_k - \Delta_A \leq \frac{\|A_k\hat{\mathbf{w}}_k\|}{\|\hat{\mathbf{w}}_k\|} \leq e_k + \Delta_A. \tag{3.9}$$

We will see shortly that (3.8) and (3.9) lead directly to (3.6), (3.7), and a theorem like Theorem 3.4. In the algorithm below we will assume that e_k and $\hat{\mathbf{w}}_k$ satisfying (3.8) and (3.9) are available. The details of calculating e_k and $\hat{\mathbf{w}}_k$ will be discussed in the next section.

To describe our algorithms we assume that a parameter ε is given and that we wish to determine the ε nullity p of an $M \times N$ matrix A and an ε nullspace S of A. If desired, (2.7) can then be used to obtain a computed

rank. We will refer to the columns of A by $\mathbf{a}_n$, $n = 1, \ldots, N$. Our algorithms are:

ALGORITHM 1.

1. Let $A_1 = A$, $k = 1$, $\rho = 0$.
2. Calculate e_k.
3. If $e_k \leq \varepsilon$, let $\rho \leftarrow \rho + 1$, determine $\hat{\mathbf{w}}_k$ satisfying (3.9), and let $\mathbf{w}_\rho$ be an N-vector formed by expanding $\hat{\mathbf{w}}_k$ putting zeros in elements corresponding to columns of A dropped in forming A_k. Let A_{k+1} be A_k with a column dropped corresponding to a largest element in absolute value of $\hat{\mathbf{w}}_k$ (ties may be broken arbitrarily). Let $k \leftarrow k + 1$ and go to 2.
4. If $e_k > \varepsilon$, stop. Let the current ρ be p, the calculated nullity of A, and $S = \text{span}\{\mathbf{w}_1, \ldots, \mathbf{w}_p\}$.

ALGORITHM 2.

1. Let $A_1 = \mathbf{a}_1$, $k = 1$, $n = 1$, $\rho = 0$.
2. Calculate e_k.
3. If $e_k \leq \varepsilon$, let $\rho \leftarrow \rho + 1$, determine $\hat{\mathbf{w}}_k$ satisfying (3.3), and let $\mathbf{w}_\rho$ be an N-vector formed by expanding $\hat{\mathbf{w}}_k$ putting zeros in elements corresponding to columns of A dropped in forming A_k. Let A_{k+1} be A_k with a column dropped corresponding to a largest element in absolute value of $\hat{\mathbf{w}}_k$ (ties may be broken arbitrarily). Let $k \leftarrow k + 1$ and go to 2.
4. If $e_k > \varepsilon$ and $n < N$, then let $A_{k+1} = (A_k, \mathbf{a}_{n+1})$, $k \leftarrow k + 1$, $n \leftarrow n + 1$, and go to 2.
5. If $e_k > \varepsilon$ and $n = N$, stop. Let the current ρ be p, the calculated nullity, and $S = \text{span}\{\mathbf{w}_1, \ldots, \mathbf{w}_p\}$.

As we will discuss in more detail later, Algorithm 1 has the advantage that the quantity e_k must be determined only $p + 1$ times. On the other hand, the matrices A_k of Algorithm 2 will generally be smaller than those of Algorithm 1. We now have

THEOREM 3.10. *Let L_1 be the number of singular values of A less than or equal to $(\varepsilon - \Delta_A)/\gamma$, and let L_2 be the number of singular values of A less than or equal to $C(\varepsilon + \Delta_A)$. If the conditions* (3.1), (3.8), *and* (3.9) *are true, then $p \geq L_1$. If the conditions* (3.1) *and* (3.3) *are true, then $p \leq L_2$. If* (3.1), (3.3), (3.8), *and* (3.9) *are true and A has no singular values in*

$$\frac{\varepsilon - \Delta_A}{\gamma} < s \leq C(\varepsilon + \Delta_A), \tag{3.11}$$

then $p = N_\varepsilon(A)$.

Proof. By our constructions $\|\mathbf{w}_k\| = \|\hat{\mathbf{w}}_k\|$ and $\|A_k\hat{\mathbf{w}}_k\| = \|A\mathbf{w}_k\|$ are trivially true, so that (3.9) implies (3.7). By successive application of the interlace theorem of Section 2, $s^k \leqslant s_k$, so that (3.6) follows immediately from (3.8). Now the proof of Theorem 3.4 is valid with minor modifications in some of the inequalities to reflect introduction of Δ_A. ∎

It is important for the success of our algorithms that neither C nor γ be large. We will discuss the size of γ in the next section. We will now discuss C; we assume without loss of generality that the calculated null vectors $\mathbf{w}_i$, $i = 1, \ldots, p$, are normalized so that $\|\mathbf{w}_i\|_\infty = 1$.

THEOREM 3.11. *If the null vectors are normalized so that* $\|\mathbf{w}_i\|_\infty = 1$, *then C satisfies*

$$C \leqslant N2^p. \tag{3.12}$$

Proof. Since in each algorithm we dropped columns of A corresponding to a largest component in absolute value of $\hat{\mathbf{w}}_k$, it follows easily that there exists a row permutation matrix P_r and a column permutation matrix P_L such that $W' = P_r W P_L$ is lower trapezoid with ones on the diagonal and whose elements have magnitude one or less. Let Z be the first p rows of W'. Then for $\mathbf{x} \in R^p$ clearly $\|W\mathbf{x}\|_1 \geqslant \|Z\mathbf{x}\|_1$. To obtain a lower bound for $\|Z\mathbf{x}\|_1$ we will let $\mathbf{y} = Z\mathbf{x}$ and assume without loss of generality that $\|\mathbf{y}\|_\infty = 1$. Then if Z has components z_{ij}, for $i > 1$ we have $x_i = y_i - \sum_{j=1}^{i-1} z_{ij}x_j$ and so $|x_i| \leqslant 1 + |x_1| + \cdots + |x_{i-1}|$. Since $|x_1| = |y_1| \leqslant 1$, by induction it is easily proven that $|x_i| \leqslant 2^{i-1}$ and consequently that $\|\mathbf{x}\|_1 \leqslant 2^p - 1 \leqslant 2^p$. Now for $\mathbf{y} = Z\mathbf{x}$, $\|W\mathbf{x}\|_1/\|\mathbf{x}\|_1 \geqslant \|Z\mathbf{x}\|_\infty/\|\mathbf{x}\|_1 = \|\mathbf{y}_\infty\|/\|\mathbf{x}\|_1 \geqslant 1/2^p$. Furthermore, since $\|\mathbf{w}_i\|_\infty = 1$, it follows by properties of matrix norms that $\|W\|_1 = \max_{i=1,\ldots,p}\|\mathbf{w}_i\|_1 \leqslant N$. Finally a property of condition numbers is that $\text{cond}_1(W) = \|W\|_1 \min_{\mathbf{x} \in R^p}\|\mathbf{x}\|_1/\|W\mathbf{x}\|_1 \leqslant N2^p$, by the above inequalities. ∎

If the calculated rank deficiency p is small and we assume that γ is not large, then Theorem 3.10 and Theorem 3.11 imply that our algorithms correctly determine an ε rank for ε contained in a moderate sized gap in the singular values A. However, if p is not small, then the bounds in (3.5), (3.11), and (3.12) are disappointingly large. In practice, though, the bound (3.12) is not realistic. In fact the bound (3.12) appears to be similar in nature to certain other known exponential bounds in linear algebra problems—for example, the bound in the growth of elements in Gaussian elimination with partial pivoting [25], or the bound on the error of the LINPAK condition estimator when using the QR algorithm with column interchanges [5, 17]—that in practice are never achieved.

To examine the point further, for a given $N \times p$ matrix B let $P_r B P_L = LU$ be the LU decomposition of the permutation of B that is obtained by using

Gaussian elimination with partial ($P_L = I$) or with complete pivoting. Our matrix W' in the proof of Theorem 3.11 is of the same form as L in the above LU decomposition, that is, lower trapezoidal with unit diagonal elements and all elements of magnitude one or less. Such matrices L are known to usually be well conditioned [2, 7], that is, $C \ll 2^p N$. To examine C experimentally we generated $N \times p$ matrices of the form W' with ones on the diagonal and elements below the diagonal chosen uniformly randomly between $+1$ and -1, and we calculated an upper bound for C. Our experiments indicated that although C may be large for larger values of N when p/N is near one, for p/N not near one C was not large for a wide range of N values. For example, for $p = N/4$, which would correspond to a calculated nullity equal to 25% of the dimension of the domain of A, our calculated bound on C was always 65 or less for several hundred matrices W' ranging in size up to 200×50. The numerical experiments in the last section further our conclusion that in practice C is not large. We might add that exceptional examples exist: for example, for $p \geq 3$ the author has constructed matrices A with $p = N/2$, $C \geq 2^p$, and no singular values in $0 < s < 2^{p-2}\varepsilon/\sqrt{p}$, and yet with $N_\varepsilon(A) \neq p = N_\varepsilon(A)+1$. Such matrices appear to be exceedingly rare in practice.

4. IMPLEMENTATIONS AND ERROR ANALYSIS

The results of Section 3 were to some extent general, and they are applicable to a variety of implementations of either algorithm. In this section we wish to provide specific natural implementations, complete the partial error analysis of Section 3, and discuss the size of γ.

Implementations of Algorithm 1 can be based on subroutines in LINPACK or, in the case of sparse matrices, other published approaches. To describe such implementations the following additions are required to our previous description of Algorithm 1:

IMPLEMENTATION OF ALGORITHM 1.

1. Let $A_1 = A$, $k = 1$, $\rho = 0$. Use the LINPACK subroutine SQRDC (with or without the pivoting option)—or, in the case of a sparse matrix A, the algorithm of [10]—to determine R in a QR factorization of A_1. Let $R_1 = R$.
2. As outlined below, use the LINPACK subroutine STRCO—or, in the case of sparse matrices, its sparse matrix modification [14]—to determine an approximate null vector $\hat{w}_k$ to R_k (and A_k). Determine e_k as outlined below.

3. If $e_k \leqslant \varepsilon$, let $\rho \leftarrow \rho + 1$ and let $\mathbf{w}_\rho$ be an N-vector formed by expanding $\hat{\mathbf{w}}_k$ putting zeros in elements corresponding to columns of A dropped in forming A_k. Using the subroutine SCHEX—or, for sparse matrices, a suitable modification as described below—drop the column of R_k (and A_k) corresponding to the component of $\hat{\mathbf{w}}_k$ with largest magnitude, and retriangularize the result using Givens transformations to form R_{k+1}. Let $k \leftarrow k + 1$ and go to 2.
4. If $e_k > \varepsilon$, stop. Let the current ρ be p, the calculated nullity. Let $S = \text{span}\{\mathbf{w}_1, \ldots, \mathbf{w}_p\}$.

The details of the LINPACK subroutines are available in [5] and will not be presented here except to discuss e_k and $\mathbf{w}_k$. The LINPACK routine STRCO and its sparse variation [14] determine a vector $\mathbf{y}$ such that the vector $\mathbf{x}$ in $R_k\mathbf{x} = \mathbf{y}$ is large relative to $\mathbf{y}$. The trial null vector $\hat{\mathbf{w}}_k$ described for step 2 will be the $\mathbf{x}$ vector of STRCO. Also, our approximation e_k will be the calculated ratio $\|\mathbf{y}\|/\|\mathbf{x}\| = \|\mathbf{y}\|/\|\hat{\mathbf{w}}_k\|$. This calculation requires a simple change in STRCO, since STRCO calculates $\|\mathbf{y}\|_1/\|\mathbf{x}\|_1$ not $\|\mathbf{y}\|/\|\mathbf{x}\|$.

We should note that the sparse matrix algorithm [10] of step 1 uses Givens transformations, takes strong advantage of the sparsity of A, allows use of convenient fixed data structures, and is numerically stable. For sparse (or dense) matrices in step 3 a Hessenberg matrix results after a column of R_k is dropped, and as is standard [5], this matrix can be retriangularized by applying a sequence of Givens transformation making use of any sparsity in R_k [15]. For sparse matrices this updating can be done within the fixed data structure of [10].

To analyze the conditions (3.8) and (3.9) we note that although the author knows of no published theoretical bounds for the condition estimators of Cline, Moler, Stewart, and Wilkinson used in the LINPACK subroutine STRCO, extensive numerical tests have been performed [4, 19]. If $e_k = \|\mathbf{y}\|/\|\mathbf{x}\|$ as above and if $s(A_k + E)$ represents the smallest singular value of a matrix $A_k + E$, then the tests and analysis of [4] and [19] indicate that a ratio similar to $e_k/s(A_k + E)$, but involving the one norm, is typically 2 and almost always 10 or less, and E is a matrix such that $\|E\|/\|A_k\|$ is a modest multiple of machine precision. Furthermore, since these LINPACK condition estimators, when modified appropriately, appear to work well for norms other than $\|\ \|_1$ [19], we expect that $e_k/s(A_k + e) \leqslant \gamma$ with $\gamma = 10$ or less. Since $s(A_k + E) \leqslant s(A_k) + \|E\|$, (3.2) follows with $\Delta_A = \alpha\eta\|A_k\| \leqslant \alpha\eta\|A\|$, where α a modest sized factor.

To examine (3.9) we note that the algorithms SCHEX, STRCO, and SQRDC and the use of the Givens transformations described above are all stable procedures [5, 24]. It follows that for any k, a unitary matrix Q_k and error matrices E_k and E_k' exist with $Q_k(A_k + E_k) = R_k$, $(R_k + E_k')\hat{\mathbf{w}}_k = \mathbf{y}$, $\|E_k\| \leqslant$

$\alpha_k\|A_k\|\eta$, and $\|E_k'\| \leq \alpha_k'\|A_k\|\eta$, where $\hat{w}_k$ is the calculated vector defined above, and where α_k and α_k' are of modest size and depend only on the size of A. These results imply that

$$\left| \frac{\|A_k\hat{w}\|}{\|\hat{w}_k\|} - \frac{\|y\|}{\|\hat{w}_k\|} \right| \leq \|Q_kE_k + E_k'\| \leq (\alpha_k + \alpha_k')\eta\|A_k\|.$$

However, since $\|A_k\| \leq \|A\|$, since there are at most $p \leq N$ different k's, and since $\|y\|/\|\hat{w}_k\| = e_k$, then (3.9) is true with $\alpha = \max(\alpha_k + \alpha_k')$, $k = 1, \ldots, p$. To summarize these comments:

THEOREM 4.1. *If the condition estimator of Cline, Moler, Stewart, and Wilkinson as used above has the properties described, then* (3.8) *and* (3.9) *are true.* ■

This result, along with the results of Section 3, implies a type of stability for Algorithm 1. More specifically, if ε lies in a sufficiently large gap in the singular values of A as described in Theorem 3.10, then the correct nullity is calculated in finite precision arithmetic as well as in exact arithmetic.

To complete the discussion of Algorithm 1 we note that the primary value of the algorithm becomes apparent when comparing for sparse matrices the efficiency of Algorithm 1 with other stable methods of rank and nullity determination. The SVD is apparently inefficient when applied to sparse matrices, since excessive fill-in occurs, and furthermore the QR algorithms with the column interchanges described in LINPACK, by Lawson and Hanson [16], or more recently by Manteuffel [18] do not order the columns of A on the basis of sparsity considerations, so that excessive fill-in will often result. For our algorithm the columns of A can be preordered with preservation of sparsity as the sole basis. Subsequently columns are dropped but not otherwise reordered. Therefore, if the effort expended in steps 2 and 3 is moderate, as will usually be the case (see [14] for timings of the sparse matrix version of STRCO), and if p is not large, our Algorithm 1 will be substantially more efficient than the abovementioned alternatives. In the case of dense matrices we note without presenting the details that Algorithm 1 will be more efficient than the SVD and comparable to (p small) or somewhat less efficient than (p not small) SQRDC [5] with column interchanges.

Finally we should note that for dense matrices Algorithm 1 can be implemented using essentially only the computer memory required for the matrix A. For sparse matrices space is needed for fill-in in forming the triangular factors R_k.

Our second algorithm could be implemented similarly to the first algorithm, using the LINPACK condition estimator to select trial null vectors.

However, for Algorithm 2 an alternate method is somewhat more efficient and leads to a rigorous bound for γ. In our description of this implementation we will assume that the reader is familiar with Givens and Householder orthogonal transformations [25]. Algorithm 2 can be implemented with the following additions to the steps in Section 3. We restrict our attention to dense matrices, since algorithm 2 is not best suited to sparse matrices.

IMPLEMENTATION OF ALGORITHM 2.

1. Let $A_1 = \mathbf{a}_1$ (the first column of A), $k = 1$, $n = 1$, $\rho = 0$, and $\hat{e}_1 = \infty$. Let Q_1 represent the Householder transformation such that $Q_1 A_1 = R_1$ is upper triangular.
2. Note that R_k will have $n - \rho$ columns. If R_k has no columns, let $e_k = \hat{e}_k$ and go to step 4. Otherwise let $\hat{R}_k$ be the principle $(n - \rho - 1) \times (n - \rho - 1)$ submatrix of R_k, let $\mathbf{r}$ be the first $n - \rho - 1$ elements of the last column of R_k, and let t be the $(n - \rho)$th diagonal entry in R_k. Then let $\mathbf{x}$ be the calculated solution to $\hat{R}_k \mathbf{x} = \mathbf{r}$, $\hat{\mathbf{w}}_k = (-\mathbf{x}, 1)^T$ [if $n - \rho = 1$, let $\hat{\mathbf{w}}_k = (1)$] and $e_k = \min(\hat{e}_k, |t| / \|\hat{\mathbf{w}}_k\|)$.
3. If $e_k \leqslant \varepsilon$, let $\rho = \rho + 1$ and let $\mathbf{w}_\rho$ be an N-vector formed by expanding $\hat{\mathbf{w}}_k$ putting zeros in elements corresponding to columns of A not included in forming R_k, and normalizing so that $\|\mathbf{w}_\rho\|_\infty = 1$. Let $R_k^\dagger$ be R_k with the column dropped corresponding to a largest magnitude component of $\hat{\mathbf{w}}_k$, and let Q_{k+1} represent the Givens transformations required so that $Q_{k+1} R_k^\dagger = R_{k+1}$ is triangular (let $Q_{k+1} = I$ if $R_k^\dagger$ has no columns). Also let $\hat{e}_{k+1} = \hat{e}_k$, $k \leftarrow k + 1$, and go to 2.
4. If $e_k > \varepsilon$ and $n < N$, then let $\hat{\mathbf{a}}_{n+1} = Q_k \cdots Q_1 \mathbf{a}_{n+1}$ and let Q_{k+1} represent the Householder transformation so that $Q_{k+1}(R_k, \hat{a}_{k+1}) = R_{k+1}$ is triangular. Let $\hat{e}_{k+1} = \min(\hat{e}_k, e_k)$, $k \leftarrow k + 1$, $n \leftarrow n + 1$, and go to 2.
5. If $e_k > \varepsilon$ and $n = N$, stop. Let the current ρ be p, the calculated nullity, and $S = \operatorname{span}\{\mathbf{w}_1, \ldots, \mathbf{w}_p\}$.

Perhaps the motivation of this implementation is not immediately clear. However, if $A = QR$ and if no columns of A are dropped (so $e_k > \varepsilon$, $k = 1, \ldots, N$), then it follows easily that in exact arithmetic $\|\hat{\mathbf{w}}_k\| / |t|$ is the two norm of column k of R^+, the psuedoinverse of R. Therefore, by the constructions of the algorithm, e_N would be the reciprocal of the maximum two norm of a column of R^+. Since here the smallest singular value of A is $1 / \|R^+\|_2$, we conclude by (2.9) that in the case that no columns are dropped the tests in steps 3 and 4 involving e_k would be a precise way to examine rank deficiency. That this property persists when columns are dropped and in inexact arithmetic is the content of the following theorem, whose proof is in the appendix.