Rank Revealing *QR* Factorizations *

Tony F. Chan
Department of Mathematics
University of California at Los Angeles
Los Angeles, California 90024

In memory of James H. Wilkinson

ABSTRACT

An algorithm is presented for computing a column permutation Π and a QR factorization $A\Pi = QR$ of an m by n $(m \geqslant n)$ matrix A such that a possible rank deficiency of A will be revealed in the triangular factor R having a small lower right block. For matrices of low rank deficiency, the algorithm is guaranteed to reveal the rank of A, and the cost is only slightly more than the cost of one regular QR factorization. A *posteriori* upper and lower bounds on the singular values of A are derived and can be used to infer the numerical rank of A.

1. INTRODUCTION

A very useful factorization of an m by n $(m \geqslant n)$ matrix A is the QR factorization, given by $A\Pi = QR$, where $\Pi \in R^{n \times n}$ is a permutation matrix, $Q \in R^{m \times n}$ has orthogonal columns and satisfies $Q^T Q = I_n$, and $R \in R^{n \times n}$ is upper triangular.

If A has full rank, then R is nonsingular. In many applications in which A is nearly rank deficient, it is desirable to select the permutation Π so that the rank deficiency is exhibited in R having a small lower right block [13]. For if R is partitioned as

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

where R_{22} is r by r, then it is easy to show that $\sigma_{n-r+1}(A) \leqslant \|R_{22}\|_2$, where

we have used the notation σ_i to denote the ith singular value of A, with $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n$. Therefore, if $\|R_{22}\|_2$ is small, then A has at least r small singular values and thus is close to being rank r deficient. The converse, unfortunately, is not true. In other words, if A has r small singular values, then it is not guaranteed that a given QR factorization of A has a small $\|R_{22}\|_2$, as the following example shows.

EXAMPLE 1.   Let $A_n \in R^{n \times n}$ be the matrix of order n illustrated below:

$$A_n = \text{diag}(1, s, s^2, \ldots, s^{n-1}) \begin{pmatrix} 1 & -c & -c & \cdots & -c \\ 0 & 1 & -c & \cdots & -c \\ \vdots & & \ddots & \ddots & \cdots & -c \\ \vdots & & & \ddots & \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix},$$

where s and c satisfy $s^2 + c^2 = 1$. For $n = 50$, $c = 0.2$, we have $\sigma_n(A_n) \approx 10^{-4}$. On the other hand, A_n is its own QR factorization and obviously has no small R_{22} block for any value of r.

Besides being able to reveal rank deficiency of A, a QR factorization with a small R_{22} block is very useful in many applications, such as in rank deficient least squares computation [8, 11] and in the subset selection problem [7, 8]. Therefore a variety of techniques have been proposed to compute it. Since the QR factorization is essentially unique once the permutation Π is fixed, these techniques all amount to finding an appropriate column permutation of A. Perhaps the best known of these is the *column pivoting strategy* [2]. Although this strategy is usually very effective in producing a triangular factor R with a small $\|R_{22}\|$, very little is known in theory about its behavior, and it can fail on some matrices [5, 9]. For example, column pivoting leaves the matrices A_n in Example 1 unchanged and fails to produce a small $\|R_{22}\|$. While such examples are rare in practice, it is still of fundamental interest to understand what distinguishes column permutations of A that produce rank revealing QR factorizations from permutations that don't, and to construct algorithms that can identify them. In this paper, we present an algorithm that is guaranteed to work for matrices of low rank deficiency and costs only slightly more than one regular QR factorization of A. In addition, empirical evidence shows that it works for most higher rank deficient matrices as well. It is important to note that even if the singular value decomposition (SVD) of A is known, it is still not obvious how to compute such a rank revealing QR factorization. In [7], an algorithm is presented for solving the subset selection problem that is based on first computing the SVD of A. Our algorithm does not require computing

the SVD of A. In the rank one deficient case, the two algorithms produce the same results, but for higher dimensional rank deficient matrices, they are different. Our algorithm is most closely related to one recently developed by Foster [6], and we shall comment on this in more detail in Section 4.

In Section 2, we shall first study the rank one deficiency case. In Section 3, we present an algorithm for the general rank deficiency case and derive *a priori* bounds on the size of the computed $\|R_{22}\|$. In Section 4, we derive *a posteriori* upper and lower bounds on the singular values of A, which can be used to infer its rank. In Section 5, we discuss how this algorithm can be implemented efficiently. Some numerical examples will be given in Section 6.

We shall use the notation x_i or $(x)_i$ to denote the ith component of a vector x, and a_{ij} to denote the (i, j)th element of matrix A.

2.   REVEALING RANK ONE DEFICIENCY

Assume that A is nearly rank one deficient. We would like to find a column permutation of A such that the resulting QR factorization has a small pivotal element r_{nn}. It turns out that this permutation can be found by inspecting the size of the elements of the singular vector of A corresponding to the smallest singular value σ_n. This procedure was first pointed out in [7].

THEOREM 2.1.   *Suppose that we have a vector $x \in R^n$ with $\|x\|_2 = 1$ such that $\|Ax\|_2 = \epsilon$, and let Π be a permutation such that if $\Pi^T x = y$, then $|y_n| = \|y\|_\infty$. Then if $A\Pi = QR$ is the QR factorization of $A\Pi$, then*

$$|r_{nn}| \leqslant \sqrt{n}\, \epsilon.$$

Proof.   The proof of this theorem was stated as Exercise P6.4-4 in [8]. We shall prove it here. First we note that since $|y_n| = \|y\|_\infty$ and $\|y\|_2 = \|x\|_2 = 1$, we have $|y_n| \geqslant \sqrt{1/n}$. Next we have

$$Q^T A x = Q^T A \Pi \Pi^T x = Ry = \begin{pmatrix} \vdots \\ \vdots \\ r_{nn} y_n \end{pmatrix}.$$

Therefore,

$$\epsilon = \|Ax\|_2 = \|Q^T A x\|_2 = \|Ry\|_2 > |r_{nn} y_n|,$$

from which the result follows.   ∎

Let $v \in R^n$ with $\|v_n\| = 1$ be the right singular vector of A corresponding to the smallest singular value σ_n. Then we have

$$\|Av\|_2 = \sigma_n.$$

Therefore, by Theorem 2.1, if we define the permutation Π by

$$\left|\left(\Pi^T v\right)\right|_n = \|v\|_\infty,$$

then $A\Pi$ has a QR factorization with a pivot r_{nn} at least as small as $\sqrt{n}\,\sigma_n$ in absolute value.

Since only the permutation Π is needed, it is not necessary to compute the SVD of A in order to find v exactly. In practice, one can use a few steps of inverse iteration [3, 12] to compute an approximation to v from which the permutation Π can be determined. In the more interesting case where $\sigma_n \ll \sigma_{n-1}$, the inverse iteration should converge rapidly. This suggests a two pass algorithm in which one first computes any QR factorization of A, then performs inverse iteration with R to find an approximate v, then determines Π, and then computes the QR factorization of $A\Pi$. The above procedure is similar to a two pass algorithm derived in [4] for computing an LU factorization of a square matrix B with a small nth pivot. In fact, the two algorithms are very much related, but we will not pursue that here.

3. REVEALING HIGHER DIMENSIONAL RANK DEFICIENCY

In this section, we consider the case where A is nearly rank r deficient, with $r > 1$. Our goal is to find a permutation Π such that if

$$A\Pi = QR \equiv Q\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

is the QR factorization of $A\Pi$, with $R_{22} \in R^{r \times r}$, then $\|R_{22}\|$ is small in some norm.

A natural way to extend the one dimensional result of Section 2 is to repeatedly apply the one dimensional algorithm, for $r = 1, 2, \ldots$, to R_{11}, the leading principal triangular part of R. Suppose that we have already isolated a small r by r block R_{22}. To isolate a small $r + 1$ by $r + 1$ block, we compute, using the one dimensional algorithm given in Section 2, a permutation P such that $R_{11}P = Q_1\tilde{R}_{11}$ is the QR factorization of $R_{11}P$ and where the $(n - r,$

$n - r$)th element of $\tilde{R}_{11}$ is small. Then with

$$\tilde{\Pi} \equiv \Pi \begin{pmatrix} P & 0 \\ 0 & I \end{pmatrix},$$

$$\tilde{Q} \equiv Q \begin{pmatrix} Q_1 & 0 \\ 0 & I \end{pmatrix},$$

it can be easily verified that

$$A\tilde{\Pi} = \tilde{Q} \begin{pmatrix} \tilde{R}_{11} & Q_1^T R_{12} \\ 0 & R_{22} \end{pmatrix}$$

is the QR factorization of $A\tilde{\Pi}$.

The above procedure can be summarized in the following algorithm:

ALGORITHM RRQR(r)

Purpose: Computes a permutation Π and a QR-factorization

$$A\Pi = QR \equiv Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

with a small $R_{22} \in R^{r \times r}$ to reveal a possible rank r deficiency in A.

Compute any QR-factorization of A: $A\Pi = QR$.
Initialize $W \in R^{n \times r}$ to zero.
For $i = n, n - 1, \ldots, n - r + 1$, **do**

1. $R_{11} \leftarrow$ leading $i \times i$ block of R.
2. Compute the singular vector $v \in R^i$ of R_{11} corresponding to $\sigma_{\min}(R_{11})$ with $\|v\|_2 = 1$, and set $\delta_i = \sigma_{\min}(R_{11})$.
3. Compute a permutation $P \in R^{i \times i}$ such that $|(P^T v)_i| = \|P^T v\|_\infty$.
4. Assign $\tilde{v} \equiv \begin{pmatrix} v \\ 0 \end{pmatrix} \in R^n$ to the ith column of W.
5. Compute $W \leftarrow \tilde{P}^T W$, where $\tilde{P} \equiv \begin{pmatrix} P & 0 \\ 0 & I \end{pmatrix}$.
6. Compute the QR factorization: $R_{11} P = Q_1 \tilde{R}_{11}$.
7. $\Pi \leftarrow \Pi \tilde{P}$.
8. $Q \leftarrow Q \begin{pmatrix} Q_1 & 0 \\ 0 & I \end{pmatrix}$.
9. $R \leftarrow \begin{pmatrix} \tilde{R}_{11} & Q_1^T R_{12} \\ 0 & R_{22} \end{pmatrix}$.

End For

The matrix W is used to store the singular vectors corresponding to the smallest singular values of the successive leading triangular blocks R_{11}. It is updated in the above algorithm for theoretical reasons only (see Lemma 3.2) and need not be explicitly incorporated in an actual implementation.

For the above algorithm to produce the desired QR factorization, we must prove the following two assertions:

(1) At step 2, R_{11} has a small singular value so that the (i, i)th element of $\tilde{R}_{11}$ is guaranteed to be small.

(2) At step 9, the last [i.e., the $(n - i)$th] row of $Q_1^T R_{12}$ is small.

If these two assertions are true, then the lower $n - i + 1$ by $n - i + 1$ block of R in step 9 is small and we have the desired QR factorization.

We shall next prove more precise versions of these two assertions. With regard to the first assertion, we have the following lemma.

LEMMA 3.1.   Let $B \in R^{n \times k}$ be a matrix containing any subset of k columns of A. Then

$$\sigma_{\min}(B) \equiv \sigma_k(B) \leqslant \sigma_k(A).$$

Proof.   Follows from the variational characterization of singular values. See [8].   ∎

Since $(R_{11}^T, 0)^T$ is the first i columns of $Q^T A \Pi$, by Lemma 3.1 we have

$$\sigma_{\min}(R_{11}) < \sigma_i(Q^T A \Pi) = \sigma_i(A).$$

This guarantees that R_{11} has a small singular value if $\sigma_i(A)$ is small, i.e., if the algorithm has not yet "captured" the complete approximate null space of A. Moreover, it also follows that $\delta_i \geqslant \delta_j$ if $i < j$.

Proving the second assertion is more subtle. We shall first need to establish some properties of the matrix W employed in the algorithm.

LEMMA 3.2.   The matrix $W \equiv [w_{n-r+1}, \ldots, w_n] \in R^{n \times r}$ computed by Algorithm RRQR(r) satisfies the following properties: For $i = n, n - 1, \ldots, n - r + 1$,

(1) $\|w_i\|_2 = 1$,

(2) $(w_i)_j = 0$ for $j > i$,

(3) $|(w_i)_i| = \|w_i\|_\infty \geqslant 1/\sqrt{i}$,

(4) $\|A \Pi w_i\|_2 = \delta_i \leqslant \sigma_i(A)$.

Proof. We shall prove the lemma by induction on r. The lemma is obviously true for the case $r = 1$ because of the one dimensional result in Theorem 2.1. Now assume that the lemma is true for $r = k$. That is, after k steps of Algorithm RRQR(k), the vectors $\tilde{w}_j$, for $j = n, n - 1, \ldots, n - k + 1$, satisfy the four properties stated in the lemma. We shall show that the vectors w_j, for $j = n, \ldots, n - k$, computed by Algorithm RRQR($k + 1$) also satisfy these properties. First observe that Algorithm RRQR($k + 1$) is simply Algorithm RRQR(k) followed by one more pass through the loop with index $i = n - k$. At step 5 of this last pass, the matrix W is updated by

$$W \leftarrow \tilde{P}^T(\tilde{v}, \tilde{w}_{n-k+1}, \ldots, \tilde{w}_n)$$

$$= (\tilde{P}^T\tilde{v}, \tilde{P}^T\tilde{w}_{n-k+1}, \ldots, \tilde{P}^T\tilde{w}_n),$$

where by the induction hypothesis, the vectors $\tilde{w}_j$, $j = n, \ldots, n - k + 1$, satisfy the properties stated in the lemma. Note that since $\tilde{P}^T$ is a permutation affecting only the first $n - k$ components of each of the $\tilde{w}_j$ vectors, the updated vectors w_j for $j = n, \ldots, n - k + 1$, automatically satisfy the first three properties. Moreover, $\tilde{v}$ is chosen so that $w_{n-k} = \tilde{P}^T\tilde{v}$ also satisfy these three properties. To verify the fourth property for $j = n, \ldots, n - k + 1$, note that the permutation for Algorithm RRQR($k + 1$) is $\Pi\tilde{P}$ and

$$\|A\Pi\tilde{P}w_j\|_2 = \|A\Pi\tilde{P}\tilde{P}^T\tilde{w}_j\|_2 = \|A\Pi\tilde{w}_j\|_2 \leqslant \sigma_j$$

by the induction hypothesis. Finally, for $j = n - k$, we have

$$\|A\Pi\tilde{P}w_{n-k}\|_2 = \|A\Pi\tilde{P}\tilde{P}^T\tilde{v}\|_2 = \|A\Pi\tilde{v}\|_2$$

$$= \left\|Q\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}\begin{pmatrix} v \\ 0 \end{pmatrix}\right\|_2$$

$$= \|R_{11}v\|_2$$

$$= \sigma_{\min}(R_{11})$$

$$= \delta_{n-k}$$

$$\leqslant \sigma_{n-k}(A)$$

where the last bound is arrived at by using Lemma 3.1 and the fact that R has the same singular values as A. This completes the inductive proof. ∎

We are now ready to state our main results.

THEOREM 3.1. *Let the matrix $W \in R^{n \times r}$ computed by Algorithm* RRQR(r) *be partitioned as $W^T \equiv (W_1^T, W_2^T)$, where $W_2 \in R^{r \times r}$ is upper triangular and nonsingular. Then the QR-factorization $A\Pi = QR$ as computed by Algorithm* RRQR(r) *satisfies*

$$\|R_{22}\|_2 \leqslant \sigma_{n-r+1} \|W_2^{-1}\|_2 \sqrt{r} .$$

Proof. Denote the columns of W by $[w_{n-r+1}, \ldots, w_n]$. Define $Y \equiv Q^T A \Pi W \in R^{n \times r}$. From property (4) of Lemma 3.2, we have

$$\|Y\|_2 \leqslant \|Y\|_F = \sqrt{\sum_{i=n}^{n-r+1} \delta_i^2} \leqslant \sqrt{r} \, \delta_{n-r+1} \leqslant \sqrt{r} \, \sigma_{n-r+1}.$$

On the other hand, we have

$$Y = Q^T A \Pi W = RW = \begin{pmatrix} R_{11}W_1 + R_{12}W_2 \\ R_{22}W_2 \end{pmatrix},$$

from which follows that

$$\|Y\|_2 \geqslant \|R_{22}W_2\|_2 \geqslant \frac{\|R_{22}\|_2}{\|W_2^{-1}\|_2}.$$

Combining the above, we get

$$\sigma_{n-r+1} \geqslant \delta_{n-r+1} \geqslant \frac{\|R_{22}\|_2}{\|W_2^{-1}\|_2 \sqrt{r}},$$

from which the desired result follows.   ∎

This theorem states that $\|R_{22}\|_2$ is bounded by the rth smallest singular values of A, provided that $\|W_2^{-1}\|_2$ is not too large. From Lemma 3.2, we have that W_2 is upper triangular with nonzero diagonal elements and is obviously nonsingular. Moreover, Algorithm RRQR can be interpreted as trying to produce a well-conditioned W_2 by putting the largest element in absolute value of each column of W on the diagonal of W_2. In this sense, it is analogous to the strategy proposed in [7], in which the permutation of Π is

obtained via a *QR* factorization with column pivoting applied to isolate a well-conditioned subset of the rows of the matrix formed by the r singular vectors corresponding to the r smallest singular values of A. The present algorithm has the advantage that the SVD of A need not be computed and that an *a priori* bound for $\|R_{22}\|$ can be derived from an *a priori* upper bound for $\|W_2^{-1}\|_2$. In fact, an *a priori* bound for individual elements of R_{22} is possible.

THEOREM 3.2. *Algorithm* RRQR(r) *computes a permutation* Π *and a QR factorization of A given by* $A\Pi = QR$ *where the elements of the lower r by r upper triangular block of R satisfy*

$$|r_{ij}| \leqslant \sigma_j\sqrt{j} + \sum_{k=i}^{j-1} 2^{j-1-k}\sigma_k\sqrt{k} \qquad (3.1)$$

$$\leqslant 2^{j-i}\sigma_i\sqrt{n} \qquad for \quad n - r < i \leqslant j \leqslant n. \qquad (3.2)$$

Proof. By employing Lemma 3.2, we have, for $n - r < i \leqslant j \leqslant n$,

$$\sigma_j \geqslant \|A\Pi w_j\|_2 = \|Rw_j\|_2 \geqslant \left|\left(Rw_j\right)_i\right| = \left|\sum_{k=i}^{j} r_{ik}(w_j)_k\right|.$$

Isolating the $k = j$ term in the sum and rearranging terms, we get

$$\left|r_{ij}(w_j)_j\right| \leqslant \sigma_j + \sum_{k=i}^{j-1} \left|r_{ik}(w_j)_k\right|.$$

Since by Lemma 3.2, $|(w_j)_j| = \|(w_j)\|_\infty \geqslant 1/\sqrt{j}$, we have

$$|r_{ij}| \leqslant \sigma_j\sqrt{j} + \sum_{k=i}^{j-1} |r_{ik}|.$$

Solving this recurrence in the index j, we get the bound given in (3.1). Using the bound $\sigma_k\sqrt{k} \leqslant \sigma_i\sqrt{n}$ in each term in the sum in (3.1), we get the bound in (3.2). ∎

The factor 2^{j-i} in the bound (3.2) in Theorem 3.2 indicates that the bound for an element of R_{22} increases with its distance from the main

diagonal. In fact, the bound for the element $|r_{n-r+1,n}|$ grows like 2^r. For large values of r, this bound can be quite large and overconservative for most problems encountered in practice. On the other hand, many problems in application have small values of r, and for these problems, the bounds in Theorem 3.2 should be quite reasonable. For example, if $r = 4$, then we have, for $n - 4 < i \leqslant j \leqslant n$,

$$\max_{ij} |r_{ij}| \leqslant 16\sqrt{n}\,\sigma_i.$$

Therefore, any small singular values of A will be revealed in the triangular factor R. In other words, for matrices of low rank deficiency, Algorithm RRQR(r) is *guaranteed* to produce rank revealing QR factorizations.

4. BOUNDING THE SINGULAR VALUES

Very often, it is desirable to be able to infer the rank of A from its QR factorization by estimating the small singular values of A from the triangular factor R [1, 10]. From Theorem 3.1, we can easily derive the following *a posteriori* bounds on the singular values of A in terms of the matrices R and W computed by Algorithm RRQR(r).

COROLLARY 4.1.   *Let* $W \in R^{n \times r}$ *and* $R \in R^{n \times n}$ *be as computed by Algorithm* RRQR(r). *Let* R_{22}^j *and* W_2^j *denote the lower right j by j upper triangular blocks of R and W respectively. Then, for* $1 \leqslant j \leqslant r$,

$$\frac{\sigma_{n-j+1}}{\sqrt{j}\,\left\|\left(W_2^j\right)^{-1}\right\|_2} \leqslant \delta_{n-j+1} \leqslant \sigma_{n-j+1} \leqslant \|R_{22}^j\|_2 \leqslant \sigma_{n-j+1}\sqrt{j}\,\left\|\left(W_2^j\right)^{-1}\right\|_2.$$

The quantities δ_{n-j+1} and $\|R_{22}^j\|_2$ are computable *a posteriori* lower and upper bounds respectively for the singular value σ_{n-j+1}. If $\|(W_2^j)^{-1}\|_2$ is not much larger than the optimal value of one, then the above corollary states that these bounds should be very tight and one can therefore infer the numerical rank of A. In practice, the value of $\|R_{22}^j\|_2$ can be estimated by more easily computable norms of R_{22}^j, such as the 1-norm, the ∞-norm, and the Frobenius norm. These *a posteriori* bounds also enable Algorithm RRQR to be implemented in an *adaptive* manner: the algorithm can be terminated if the lower bounds indicate that all the small singular values of A have been revealed.

Algorithm RRQR is closely related to two algorithms recently given by Foster [6]. His algorithms are designed to reveal the rank and nullity of a given matrix and to produce a subset of columns spanning the approximate range space of the matrix. In fact, the strategy that he used in deciding which column to discard at each stage of his algorithm is exactly the same as the one used in RRQR to select which column is to be permuted to the last position, namely, by inspecting the largest element in absolute value of the singular vector of the leading block of a triangular factor. The main difference between the two algorithms is that RRQR produces a *QR* factorization of the original matrix whereas Foster's algorithm only produces a factorization for a subset of the columns of the original matrix. More importantly, the availability of the full *QR* factorization in Algorithm RRQR allows the computation of *a posteriori* upper and lower bounds on the singular values of *A* as presented in this section, which makes it possible to estimate the rank of *A* more reliably that the usually pessimistic *a priori* bounds can.

5. IMPLEMENTATION NOTE

The main work of Algorithm RRQR consists of three parts: the computation of the initial *QR* factorization, the computation of the singular vector v of R_{11} by inverse iteration at each iteration, and the *QR*-factoring of $R_{11}P$ at each iteration. Ignoring lower order terms, the first two take $n^2(m - n/3)$ (assuming that *Q* need not be accumulated) and In^2r flops respectively, where *I* is the number of inverse iterations used at each iteration. Usually, $I = 2$ is sufficient in practice, especially if an efficient condition number estimator is used to select the initial vector.

The refactoring of $R_{11}P$, if done naively, say by using Householder transformations, would be extremely inefficient. Due to the special structure of the problem, a much more efficient procedure can be derived by using Givens rotation instead. The column of R_{11} to be permuted to the last column can be swapped with adjacent columns one at a time until the last column is reached. For example, if column 1 is to be permuted to column k, we can swap the pair $(1,2),(2,3),\ldots,(k-1,k)$ successively. With each swap, exactly one nonzero is introduced in the lower triangular part of R_{11}, which can then be annihilated by applying an appropriate Givens rotation from the left. In most applications, these extra Givens rotations can be stored in factored form and do not have to be applied to the orthogonal factor *Q* of the initial *QR* factorization. Assuming the worst case in each iteration, i.e., that the first column of R_{11} is to be permuted to the last every time, this takes

$2n^2r$ flops. Therefore, the total work for Algorithm RRQR(r) is given by

$$W(r) = n^2\left(m - \frac{n}{3}\right) + In^2r + 2n^2r$$

$$= n^2\left(m - \frac{n}{3}\right) + 4n^2r, \qquad \text{assuming that} \quad I = 2.$$

Therefore, if $r \ll n$, the extra work performed after the initial QR factorization is of a lower order and negligible, especially if $m \gg n$. Even in the extreme case of $r = n$, we have $W(n) = mn^2 + \frac{11}{3}n^3$, which is still smaller than the cost of $2mn^2 + 4n^3$ for computing the SVD [8].

It is interesting to note that the cost of implementing the column pivoting strategy is $O(mn)$ while that for RRQR is $O(n^2r)$. Therefore, if r is small and $m \gg n$, the overhead of RRQR is less than that for column pivoting.

6. NUMERICAL EXAMPLES

In this section, we present a few numerical examples to illustrate the theory developed in the earlier sections. All computations are done on a VAX/780 in single precision, with a relative machine precision of about 10^{-7}. The initial QR factorization is performed with no column permutation, and the inverse iterations are performed with $I = 5$ with the initial vector chosen to be $(1, 0, 1, 0, \ldots, 1, 0)^T$. For each of the examples, we compute the QR factorization with Algorithm RRQR, compute the lower and upper a posteriori bounds δ_{n-j+1} and $\|R_{22}^j\|_2$ given in Corollary 4.1, and compare these with the true singular values σ_{n-j+1} and with the upper bounds obtained by the QR-factorization with and without column pivoting.

The first example is the matrix A_{50} given earlier in the introduction. It turns out that, due to roundoff errors, QR with column pivoting actually pivots with this matrix. We therefore use the following modified matrix instead:

$$\hat{A}_n = A_n + \text{diag}(n\epsilon, (n-1)\epsilon, \ldots, \epsilon),$$

where $\epsilon = 10^{-6}$. This small perturbation only changes the singular values of the matrix by a small relative amount and forces no column pivoting. The results are shown in Table 1, in which the column permutations produced by RRQR and QR with column pivoting are also presented. We see that in this case, Algorithm RRQR succeeds in revealing the one dimensional null space of

TABLE 1

RESULTS FOR A_{50}

i	RRQR lower	True σ_i	RRQR upper	Pivoted *QR*		Unpivoted *QR*	
				K	Upper	K	Upper
50	9.29E − 05	0.0001	0.0002	1	0.3678	50	0.3678
49	0.4349	0.4112	0.4538	48	0.4112	49	0.4112
48	0.4262	0.4222	0.4937	49	0.4227	48	0.4227
47	0.4531	0.4327	0.4967	46	0.4351	47	0.4351
46	0.4440	0.4431	0.5143	47	0.4495	46	0.4495
45	0.4722	0.4536	0.5179	44	0.4665	45	0.4665
44	0.4627	0.4642	0.5357	45	0.4868	44	0.4868
43	0.4924	0.4749	0.5399	42	0.5106	43	0.5106
42	0.4824	0.4858	0.5581	43	0.5382	42	0.5382
41	0.5137	0.4968	0.5627	40	0.5698	41	0.5698

K = corresponding column of A permuted to ith column of $A\Pi$.

the matrix whereas *QR* with column pivoting fails to do so. Since the values of $\|(W_2^j)^{-1}\|_2$ are very close to one, the *a posteriori* bounds are very tight and there is no trouble in identifying the numerical rank. In fact, since the lower bound for σ_{49} is clearly not small, one can confidently truncate the algorithm after only two iterations. Note that since the lower bounds δ_i are computed only approximately by inverse iteration, they can actually be slightly larger than the true singular values. But this slight inaccuracy does not affect the ability to infer the rank.

The second example is the class of matrices defined as follows. Let $H_n = I - (2/n)ee^T$, where $e = (1, 1, \ldots, 1)^T$. Define

$$C = H_{50}\begin{pmatrix} D \\ 0 \end{pmatrix}H_{10},$$

where D is a diagonal matrix. Since H_n is orthogonal, the singular values of C are the diagonal entries of D. The results for two different D's with well-defined numerical rank are presented in Tables 2–3. In Table 4, we present an example where the numerical rank is not as well-defined. All the results show that the *a posteriori* bounds produced by RRQR are tight and that they reveal the numerical rank as predicted. *QR* with column pivoting also succeeds on these examples, although it is not possible to infer the rank, since a lower bound on the singular values is not available. The column permutations produced are also different than those of RRQR. *QR* factorization with no pivoting fails to reveal any near rank deficiency in all cases.

TABLE 2

RESULTS FOR C WITH $D = \mathrm{diag}(1,1,1,1,1,10^{-4},10^{-4},10^{-4},10^{-4},10^{-4})$

i	RRQR lower	True σ_i	RRQR upper	Pivoted QR		Unpivoted QR	
				K	Upper	K	Upper
10	0.0001	0.0001	0.0002	3	0.0001	10	0.0001
9	0.0001	0.0001	0.0002	6	0.0001	9	0.0001
8	0.0001	0.0001	0.0002	7	0.0002	8	0.0002
7	0.0001	0.0001	0.0002	10	0.0002	7	0.0002
6	0.0001	0.0001	0.0002	9	0.0002	5	1.0000
5	0.4472	1.0000	1.0000	8	1.0000	6	1.0000
4	0.4472	1.0000	1.0000	4	1.0000	4	1.0000

TABLE 3

RESULTS FOR C WITH $D = \mathrm{diag}(1,10^{-4},1,10^{-4},1,10^{-4},1,10^{-4},1,10^{-4})$

i	RRQR lower	True σ_i	RRQR upper	Pivoted QR		Unpivoted QR	
				K	Upper	K	Upper
10	0.0001	0.0001	0.0002	5	0.0001	10	0.0001
9	0.0001	0.0001	0.0002	2	0.0001	8	0.0002
8	0.0001	0.0001	0.0002	8	0.0002	6	0.0002
7	0.0001	0.0001	0.0002	4	0.0002	4	1.0000
6	0.0001	0.0001	0.0002	6	0.0002	9	1.0000
5	0.4472	1.0000	1.0000	10	1.0000	2	1.0000
4	0.6325	1.0000	1.0000	7	1.0000	7	1.0000

TABLE 4

RESULTS FOR C WITH $D = \mathrm{diag}(10^{-5},10^{-4},10^{-3},10^{-2},10^{-1},1,1,1,1,1)$

i	RRQR lower	True σ_i	RRQR upper	Pivoted QR		Unpivoted QR	
				K	Upper	K	Upper
10	9.8E − 06	9.8E − 06	1.2E − 05	1	1.3E − 05	1	4.9E − 05
9	0.0001	0.0001	0.0001	2	0.0001	2	1.0000
8	0.0009	0.0009	0.0017	3	0.0017	3	1.0000
7	0.0076	0.0100	0.0263	6	0.0263	10	1.0000
6	0.0706	0.1000	0.2193	4	0.2193	4	1.0000
5	0.4479	1.0000	1.0000	9	1.0000	5	1.0000
4	0.4490	1.0000	1.0000	5	1.0000	9	1.0000
3	0.6334	1.0000	1.0000	8	1.0000	8	1.0000
2	0.7767	1.0000	1.0000	7	1.0000	7	1.0000
1	0.8947	1.0000	1.0000	10	1.0000	6	1.0000

In all the above examples, the actual values of $\|(W_2^j)^{-1}\|_2$ are not much larger than the optimal value of one, and are much smaller than that suggested by the exponential *a priori* bounds given in Theorem 3.2. This is probably typical for most problems encountered in practice. In fact, we have not succeeded in finding an example for which the *a priori* bounds are achieved. Thus, not only is Algorithm RRQR guaranteed to work for matrices of low rank deficiency, it should almost always work for high rank deficient ones as well.

7. CONCLUSION

In this paper, we have presented a simple and efficient algorithm for computing a QR factorization that is designed to reveal the numerical rank of a given matrix. Our theory shows that the algorithm is guaranteed to work for matrices of low rank deficiency, and numerical experiments indicate that it is also very likely to work even in the high rank deficient cases. The *a posteriori* bounds for the singular values, which are by-products of the algorithm, can be used to infer the numerical rank of the matrix, *without explicitly computing the SVD of the matrix*. Thus, this special QR factorization could be used reliably and efficiently in many matrix computations for which a rank-revealing QR factorization is needed, such as in rank deficient least squares computations and the subset selection problem.

REFERENCES

1 N. Anderson and I. Karasalo, On computing bounds for the least singular value of a triangular matrix, *BIT* 15:1–4 (1975).
2 P. Businger and G. H. Golub, Linear least squares solutions by Householder transformations, *Numer. Math.* 1:269–276 (1965).
3 T. F. Chan, Deflated decomposition of solutions of nearly singular systems, *SIAM J. Numer. Anal.* 21(4):738–754 (Aug. 1984).
4 _____, On the existence and computation of LU-factorizations with small pivots, *Math. Comp.* 42(166):535–547 (Apr. 1984).
5 D. K. Faddeev, V. N. Kublanovskaja, and W. N. Faddeeva, Sur les systèmes linéares algébriques de matrices rectangulaires et mal-conditionées, in *Programmation en Mathématiques Numériques*, Editions Centre Nat. Recherche Sci., Paris, VII, 1968, pp. 161–170.

6   Leslie V. Foster, Rank and null space calculations using matrix decomposition without column interchanges, *Linear Algebra Appl.*, 74:47–71 (1986).

7   G. H. Golub, V. Klema, and G. W. Stewart, Rank degeneracy and least squares problems, Technical Report STAN-CS-76-559, Computer Science Dept., Stanford Univ., 1976.

8   G. H. Golub and C. F. Van Loan, *Matrix Computation*, Johns Hopkins U.P., Baltimore, 1983.

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

10   I. Karasalo, A criterion for truncation of the *QR*-decomposition algorithm for the singular linear least squares problem, *BIT* 14:156–166 (1974).

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

12   G. W. Stewart, On the implicit deflation of nearly singular systems of linear equations, *SIAM J. Sci. Statist. Comput.* 2(2):136–140 (1981).

13   ____, Rank degeneracy, *SIAM J. Sci. Statist. Comput.* 5(2):403–413 (1984).