Abstract: An algorithm is presented for computing a column permutation Π and a QR-factorization $A\Pi = QR$ of an m by n ($m \geq n$) 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 low rank deficient matrices, 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.

Rank Revealing QR-Factorizations

Tony F. Chan

Research Report YALEU/DCS/RR-398
June 1985

The author was supported in part by the Department of Energy under contract DE-AC02-81ER10996 and by the Army Research Office under contract DAAG-83-0177.

Keywords: *QR-Factorization, rank deficient matrices, least squares computation, subset selection, rank, singular values.*

1. Introduction

A very useful factorization of an m by n ($m \geq n$) 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 $Q^TQ = I_n$ 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 [10]. 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 $\sigma_{n-r+1}(A) \leq ||R_{22}||_2$, where we have used the notation σ_i to denote the i–th singular value of A, with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$. 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 for $n = 4$:

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

where s and c satisfy $s^2 + c^2 = 1$. For $n = 50, c = .2$, $\sigma_n(A_n) \approx 10^{-4}$. 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 [7] and in the subset selection problem [6, 7]. 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 most well–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 behaviour and it can fail on some matrices [5]. 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 low rank deficient matrices 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 [6], an algorithm is presented for solving the subset selection problem and 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.

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 $||R_{22}||$ and *a posteriori* upper and lower bounds on the singular values of A. In Section 4, we discuss how this algorithm can be implemented efficiently. Some numerical examples will be given in Section 5.

We shall use the notation x_i or $(x)_i$ to denote the i–th 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 [6].

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

$$|r_{nn}| \leq \sqrt{n}\epsilon.$$

Proof.

The proof of this theorem was stated as Exercise P6.4–4 in [7]. We shall prove it here. First we note that since $|y_n| = \|y\|_\infty$ and $\|y\|_2 = \|x\|_2 = 1$, we have $|y_n| \geq \sqrt{\frac{1}{n}}$. Next we have

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

Therefore,

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

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

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

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, 9] 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, 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 a LU factorization of a square matrix B with a small n–th 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 i–th 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 $(n-i+1)$ lower 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*

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

Proof. Follows from the variational characterization of singular values. See [7]. ∎

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

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

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.

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. $|(w_i)_i| = ||w_i||_\infty \geq \frac{1}{\sqrt{i}}$.
4. $||A\Pi w_i||_2 = \delta_i \leq \sigma_i(A)$.

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 $\widetilde{w}_j$, for $j = n, n-1, \ldots, n-k+1$, satisfies 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

$$\begin{aligned} W \leftarrow & \widetilde{P}^T(\widetilde{v}, \widetilde{w}_{n-k+1}, \ldots, \widetilde{w}_n) \\ = & (\widetilde{P}^T\widetilde{v}, \widetilde{P}^T\widetilde{w}_{n-k+1}, \ldots, \widetilde{P}^T\widetilde{w}_n), \end{aligned}$$

where by the induction hypothesis, the vectors $\widetilde{w}_j$, $j = n, \ldots, n-k+1$ satisfies the properties stated in the lemma. Note that since $\widetilde{P}^T$ is a permutation affecting only the first $n-k$ components of each of the $\widetilde{w}_j$ vectors, the updated vectors w_j, for $j = n, \ldots, n-k+1$, automatically satisfy the first three properties. Moreover, $\widetilde{v}$ is chosen so that $w_{n-k} \equiv \widetilde{P}^T\widetilde{v}$ 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 $\Pi\widetilde{P}$ and

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

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

$$\begin{aligned} \|A\Pi\widetilde{P}w_{n-k}\|_2 &= \|A\Pi\widetilde{P}\widetilde{P}^T\widetilde{v}\|_2 = \|A\Pi\widetilde{v}\|_2 \\ &= \left\| Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \begin{pmatrix} v \\ 0 \end{pmatrix} \right\|_2 \\ &= \|R_{11}v\|_2 \\ &= \sigma_{\min}(R_{11}) \\ &= \delta_{n-k} \\ &\leq \sigma_{n-k}(A) \end{aligned}$$

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*

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

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

$$\|y\|_2 = \delta_{n-r+1} \leq \sigma_{n-r+1}.$$

On the other hand, with $e_1 \equiv (1, 0, \cdots, 0)^T \in R^r$, we have

$$y = Q^TA\Pi We_1 = RWe_1 = \begin{pmatrix} R_{11}W_1 + R_{12}W_2 \\ R_{22}W_2 \end{pmatrix} e_1,$$

from which follows that

$$\|y\|_2 \geq \|R_{22}W_2e_1\|_2 \geq \frac{\|R_{22}\|_2}{\|W_2^{-1}\|_2}.$$

Combining the above, we get

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

from which the desired result follows.

∎

This theorem states that $\|R_{22}\|_2$ is bounded by the r–th 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, each one of its columns has unit Euclidean length 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 on the diagonal. In this sense, it is analogous to the strategy proposed in [6], 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 :*

$$|r_{ij}| \leq \sigma_j + \sum_{k=i}^{j-1} 2^{j-1-k}\sigma_k\sqrt{k} \tag{3.1}$$

$$\leq 2^{j-i}\sigma_i\sqrt{n} \qquad \text{for} \quad n-r < i \leq j \leq n. \tag{3.2}$$

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

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

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

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

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

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

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

∎

The term 2^{j-i} in the bound (3.2) in Theorem 3.2 indicates that the bound for an element of R_{22} is larger the further the element is 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 overly conservative 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 $n - 4 < i \leq j \leq n$,

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

Therefore, any small singular values of A will be revealed in the triangular factor R. In other words, for low rank deficient matrices, 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, 8]. 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 $1 \leq j \leq r$,*

$$\frac{\sigma_{n-j+1}}{\|(W_2^j)^{-1}\|_2} \leq \delta_{n-j+1} \leq \sigma_{n-j+1} \leq \|R_{22}^j\|_2 \leq \sigma_{n-j+1}\|(W_2^j)^{-1}\|_2.$$

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 $\infty-$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.

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 takes $n^2(m - \frac{n}{3})$ (assuming that Q needs not be accumulated) and In^2r flops respectively, where I is the number of inverse iterations used at each each iteration. Usually, $I = 2$ is sufficient in practice, especially if an efficient condition number estimator is used to select the initial vector.

The re–factoring 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 Given's 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 pairs: $(1,2),(2,3),\ldots,(k-1,k)$ 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 Given's rotation from the left. In most applications, these extra Given's rotation 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 :

$$\begin{aligned} W(r) &= n^2(m - \frac{n}{3}) + In^2r + 2n^2r \\ &= n^2(m - \frac{n}{3}) + 4n^2r, \qquad \text{assuming that } I = 2. \end{aligned}$$

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 $r = n$, $W(n) = mn^2 + \frac{11}{3}n^3$, which is still smaller than the cost of $2mn^2 + 4n^3$ for computing the SVD.

It is interesting to note that the overhead for the column pivoting strategy is $O(mn)$ while the overhead for RRQR is $O(n^2r)$. Therefore, if r is small and $m >> n$, 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 $(1, 0, 1, 0, \cdots, 1, 0)^T$. 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 to the true singular values σ_{n-j+1} and to the upper bounds obtained by the regular 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 round-off errors, QR with column pivoting actually pivots with this matrix. We therefore use the following modified matrix instead

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

where $\epsilon = 10^{-6}$. This small perturbation only change 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 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's are computed only approximately by inverse iteration, they can actually be slightly larger than the true singular values. But this slight inacurracy does not affect the ability to infer the rank.

The second example is the class of matrices defined as follows. Let $H_n = I - \frac{2}{n}ee^T$, 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

i	$RRQR$ lower	true σ_i	$RRQR$ upper	K	pivoted QR upper	K	regular QR 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

Table 1: Results for A_{50} (K = corresponding column of A permuted to i-th column of $A\Pi$)

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 producd 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 are not available. The column permutations produced are also different than those of RRQR. Regular QR-factorization with no pivoting fails in all cases.

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 *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 low rank deficient matrices, it will 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 low rank deficient matrices 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.

Acknowledgement : The author would like to thank Dr. Robert Schreiber for helpful discussions on the development of Algorithm RRQR and Professor Charles Van Loan for pointing out Theorem 2.1 in the book [7].

i	$RRQR$ lower	true σ_i	$RRQR$ upper	K	pivoted QR upper	K	regular QR 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 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	K	pivoted QR upper	K	regular QR 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 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	K	pivoted QR upper	K	regular QR 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

Table 4: Results for C with $D = \mathrm{diag}(10^{-5},10^{-4},10^{-3},10^{-2},10^{-1},1,1,1,1,1)$

References

[1] N. Anderson and I. Karasalo, *On Computing Bounds for the Least Singular Value of a Triangular Matrix,* BIT, 15(1975), pp. 1–4.

[2] G.H. Golub and P. Businger, *Linear Least Squares Solutions by Householder Transformations,* Numer. Math., 1 (1965), pp. 269–276.

[3] T.F. Chan, *Deflated Decomposition of Solutions of Nearly Singular Systems,* Siam J. Numer. Anal., 1984, 21/4 August (1984), pp. 738–754.

[4] ————, *On the Existence and Computation of LU-factorizations with Small Pivots,* Math. Comp., 42/166 April (1984).

[5] D.K. Faddeev, V.N. Kublanovskaja and W.N. Faddeeva, *Sur les Systemes Lineares Algebriques de Matrices Rectangularies et Mal-Conditionees,* Programmation en Mathematiques Numeriques, Editions Centre Nat. Recherche Sci., Paris, VII (1968), pp. 161–170.

[6] G.H. Golub, G.W. Stewart and V. Klema, *Rank Degeneracy and Least Squares Problems,* Technical Report STAN-CS-76-559, Computer Science Dept., Stanford University, 1976.

[7] G.H. Golub and C.F. Van Loan, *Matrix Computations,* The Johns Hopkins University Press, Baltimore, 1983.

[8] I. Karasalo, *A Criterion for Truncation of the QR-Decomposition Algorithm for the Singular Linear Least Squares Problem,* BIT, 14 (1974), pp. 156–166.

[9] G.W. Stewart, *On the Implicit Deflation of Nearly Singular Systems of Linear Equations,* SIAM J. Sci. Stat. Comp., 2/2 (1981), pp. 136–140.

[10] ————, *Rank Degeneracy,* SIAM J. Sci. Stat. Comp., 5/2 (1984), pp. 403–413.