A Block Algorithm for Computing Rank-Revealing QR Factorizations[*]

Christian H. Bischof[†] Per Christian Hansen[‡]

Argonne Preprint MCS-P251-0791
Published in Numerical Algorithms 2(3-4):371-392, 1992.

Abstract

We present a block algorithm for computing rank-revealing QR factorizations (RRQR factorizations) of rank-deficient matrices. The algorithm is a block generalization of the RRQR-algorithm of Foster and Chan. While the unblocked algorithm reveals the rank by peeling off small singular values one by one, our algorithm identifies groups of small singular values. In our block algorithm, we use incremental condition estimation to compute approximations to the nullvectors of the matrix. By applying another (in essence also rank-revealing) orthogonal factorization to the nullspace matrix such created, we can then generate triangular blocks with small norm in the lower right part of R. This scheme is applied in an iterative fashion until the rank has been revealed in the (updated) QR factorization. We show that the algorithm produces the correct solution, under very weak assumptions for the orthogonal factorization used for the nullspace matrix. We then discuss issues concerning an efficient implementation of the algorithm and present some numerical experiments. Our experiments show that the block algorithm is reliable and successfully captures several small singular values, in particular in the initial block steps. Our experiments confirm the reliability of our algorithm and show that the block algorithm greatly reduces the number of triangular solves and increases the computational granularity of the RRQR computation.

Key words. rank-revealing QR factorization, numerical rank, block algorithm.

AMS (MOS) subject classification. 65F25, 65F20.

[*]This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U. S. Department of Energy, under Contract W-31-109-Eng-38. The second author was also sponsored by a travel grant from the Knud Højgaards Fond.

[†]Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL 60439, `bischof@mcs.anl.gov`. This work was partially completed while the author was visiting the IBM Scientific Center in Heidelberg, Germany.

[‡]UNI•C (Danish Computing Center for Research and Education), Building 305, Technical University of Denmark, DK-2800 Lyngby, Denmark, `unipch@wuli.uni-c.dk`.

1 Introduction

The rank-revealing QR factorization (RRQR Factorization) is a valuable tool in numerical linear algebra, because it detects the numerical rank of a matrix and because it provides the necessary information to solve many rank-deficient least-squares problems, namely, accurate information about rank and numerical nullspace. The RRQR factorization takes advantage of the efficiency and simplicity of the QR factorization, yet it produces information that is almost as reliable as that computed by means of the more expensive singular value decomposition.

Its main use arises in the solution of rank-deficient least-squares problems, for example, in geodesy [15], computer-aided design [19], nonlinear least-squares problems [24], the solution of integral equations [12], and in the calculation of splines [18]. Other applications arise in beamforming [5], spectral estimation [23], and regularization [20,21,28].

We briefly summarize the properties of a *rank-revealing QR factorization.* Let A be an $m \times n$ matrix A (w.l.o.g. $m \geq n$) and with singular values

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0, \tag{1}$$

and define the numerical rank r of A with respect to the threshold τ as the number of singular values strictly greater than τ: $\sigma_r > \tau \geq \sigma_{r+1}$. Also, let A have a QR factorization of the form

$$A\,P = QR = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \tag{2}$$

where P is a permutation matrix and Q has orthonormal columns. We then say that (2) is a RRQR factorization of A if the following three properties are satisfied:

1. R_{11} is an $r \times r$ matrix with condition number of the order σ_1/σ_r,
2. $||R_{22}||_2$ is of the order σ_{r+1}, and
3. r is the numerical rank of A.

Whenever there is a well-determined gap in the singular value spectrum between σ_r and σ_{r+1}, and hence the numerical rank r is well defined, then the RRQR factorization (2) reveals the numerical rank of A by having a submatrix R_{22} with elements of the order σ_{r+1} and a well-conditioned leading submatrix R_{11}.

We note that recently Stewart suggested a rank-revealing complete orthogonal factorization, the so-called URV decomposition [27]. This factorization decomposes

$$A = U \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} V^T,$$

where U and V are orthogonal and both $\|R_{12}\|_2$ and $\|R_{22}\|_2$ are of the order σ_{r+1}. In particular, compared with the RRQR factorization, URV decompositions have a general orthogonal matrix V instead of the permutation matrix P. URV decompositions are more expensive to compute, but they are well suited for nullspace updating. RRQR factorizations, on the other hand, are more suited for the least-squares setting, since one need not store the orthogonal matrix V (the other orthogonal matrix is usually applied to the right-hand side "on the fly"). Of course, RRQR factorizations can be used to compute an initial URV decomposition, where $U = Q$ and $V = P$. We also note that the matrix

$$P^T \begin{pmatrix} R_{11}^{-1} R_{12} \\ -I \end{pmatrix},$$

which can be easily computed from (2), is usually a good approximation of the nullvectors, and a few steps of subspace iteration suffice to compute nullvectors that are correct to working precision [10].

An algorithm for computing an RRQR factorization was discovered independently by Foster [13] and Chan [8]. Their idea is first to compute any QR factorization of A and then to post-process the triangular factor R to "peel off" the small singular values of A one at a time. Their algorithm has the additional property that

$$\sigma_{\min}(R(1:k, 1:k)) \approx \sigma_k(A), k = r, \ldots, n, \tag{3}$$

where $\sigma_{\min}(A)$ is the smallest singular value of a matrix A.

This property, however, is not important for most applications of RRQR factorizations [10], where one is interested only in capturing the break between "large" and "small" singular values. In this paper, we propose a block method that gives up (3), but can identify several small singular values at a time and will capture the break between "large" and "small" singular values.

Our paper is organized as follows. In Section 2 we describe how we compute an initial estimate of the numerical rank using incremental condition estimation (ICE) and how we can use ICE to arrive at cheap estimates for the nullvectors of leading triangular submatrices. In Section 3 we introduce our block algorithm, which will identify a rank-revealing permutation by essentially computing a rank-revealing orthogonal factorization of the nullvector matrices generated by ICE. In Section 4 we show that this scheme is guaranteed to capture the break between "large" and "small" singular values, under very weak assumptions for the orthogonal factorization applied to the nullvectors. In Section 5 we discuss implementation aspects of the algorithm and elaborate on design tradeoffs. In particular, we motivate the computational savings that can result from a "good" initial QR factorization. In Section 6 we present some numerical experiments and compare the numerical results of our algorithm with the unblocked Chan/Foster scheme. In Section 7, we summarize our results.

2 Deriving a Rank Estimate from the Initial QR Factorization

In this section we describe how we compute an initial guess of the numerical rank r of A using some initial QR factorization of A. To this end, we use *incremental condition estimation* (ICE) [2, 6], a condition estimation scheme designed to give approximations to the smallest or largest singular value and the corresponding left singular vector of a sequence of increasingly larger upper triangular matrices.

Using the formulation of [6], we can describe the key step in ICE for computing approximations to the smallest singular value as follows. We are given an upper triangular matrix $R_k \in \Re^{k\times k}$ and a vector $x_k \in \Re^k$ with $||x_k||_2 = 1$ so that $x_k^T R_k = d_k^T$ and $\delta_k \equiv ||d_k||_2 \approx \sigma_{\min}(R_k)$. That is, x_k is a good approximation of the left singular vector of R_k associated with its smallest singular value. We then find $s_k = \sin\phi_k$ and $c_k = \cos\phi_k$ such that $||d_{k+1}||_2$ is minimized, where

$$(d_{k+1})^T = (x_{k+1})^T R_{k+1} = \begin{pmatrix} s_k x_k \\ c_k \end{pmatrix}^T \begin{pmatrix} R_k & v \\ 0 & \gamma \end{pmatrix}, \tag{4}$$

and then $\delta_{k+1} = ||d_{k+1}||_2$ is taken to be an estimate of $\sigma_{\min}(R_{k+1})$. As shown in [2,6], (s_k, c_k) and δ_{k+1} can be computed in $O(k)$ flops. ICE also provides for an efficient and reliable method for computing approximations to right singular vectors of the triangular matrix R: If x_k is an approximate left singular vector of R_k, then $y_k \equiv R_k^{-1}x_k/||R_k^{-1}x_k||_2$ is a corresponding approximate right singular vector of R_k, because the computation of $R_k^{-1}x_k$ corresponds to "one half" step of inverse iterations with $R_k R_k^T$. Moreover, if δ_k is small, then $\left\| R \begin{pmatrix} y_k \\ 0 \end{pmatrix} \right\|_2 = ||R_k y_k||_2$ implies that the vector $\begin{pmatrix} y_k \\ 0 \end{pmatrix}$ is an approximate nullvector of R. The numerical experiments from [2,26] suggest that this scheme is a reliable means for computing good approximations to $\sigma_{\min}(R_{k+1})$. In particular, the estimates for δ_k derived from y_k (i.e., after applying one backsolve to the initial ICE vector) were always found to be correct to one digit [6].

ICE is well suited for monitoring the smallest singular value and condition number of the leading triangular factor during the computation. Given an initial QR factorization

$$AP = QR,$$

and a threshold τ separating "small" and "large" singular values, we can use ICE to cheaply compute an initial estimate of the numerical rank of A with respect to τ. To this end, we monitor the smallest singular values δ_k of the leading triangular submatrices $R(1\!:\!k, 1\!:\!k)$ for $k = 1, \ldots, n$, using the ICE scheme just described. We note that in general δ_k will be estimated as the QR factorization is computed, because of the incremental nature of ICE. This property of ICE is crucial for the implementation of restricted pivoting schemes, where column exchanges in the initial QR factorization are restricted to "computationally convenient" ones [1,3,4,5].

We recall that the δ_k's form a nonincreasing sequence of lower bounds for the corresponding singular values $\sigma_k(A)$; that is,

$$\delta_k \geq \delta_{k+1}.$$

Moreover, if the δ_k's are the exact singular values, then

$$\delta_k \leq \sigma_k(A).$$

The number $\tilde{r}$ of lower bounds δ_k strictly greater than the threshold τ,

$$\delta_1 \geq \ldots \geq \delta_{\tilde{r}} > \tau \geq \delta_{\tilde{r}+1}, \tag{5}$$

is therefore guaranteed to be a lower bound for the numerical rank r. Further, we are guaranteed that $R(1:\tilde{r}, 1:\tilde{r})$ is well conditioned because its smallest singular value is greater than τ. In practice, the δ_k's that we compute by means of ICE are only estimates of the exact singular values; but one the other hand these estimates are so good that we need not distinguish between exact singular values and the estimates.

Rank-revealing QR factorizations are motivated by the following lemma, which is proved in [10].

Lemma 1 *For any $R \in \Re^{n\times n}$ and any $W = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} \in \Re^{n\times p}$ with a nonsingular $W_2 \in \Re^{p\times p}$, we have*

$$\|R(n-p+1:n, n-p+1:n)\|_2 \leq \|RW\|_2\, \|W_2^{-1}\|_2. \tag{6}$$

This means that if we can determine a matrix W with p linearly independent columns, all of which lie approximately in the nullspace of R (i.e., $\|RW\|_2$ is small), and if W_2 is well conditioned such that $(\sigma_{\min}(W_2))^{-1} = \|W_2^{-1}\|_2$ is not large, then we are guaranteed that the elements of the bottom right $p \times p$ block of R will be small.

By applying ICE to the initially computed QR factorization, we compute a lower bound $\tilde{r}$ of the numerical rank of A and an upper trapezoidal matrix

$$X = \left(\begin{pmatrix} x_{\tilde{r}+1} \\ 0 \end{pmatrix}, \ldots, x_n \right) \in \Re^{n\times(n-\tilde{r})} \tag{7}$$

(that is, each x_k has as many zeros appended as are needed to arrive at an n-vector). By solving $RY = X$ and normalizing the columns of the upper trapezoidal matrix Y, we now obtain a set of approximate right nullvectors Y that satisfy

$$\|RY\|_2 \leq \|RY\|_F = \left(\sum_{i=\tilde{r}+1}^{n} \delta_i^2 \right)^{1/2} \leq \tau\sqrt{n-\tilde{r}}.$$

Thus, if the bottom triangular part Y_2 of Y is well conditioned, then (6) guarantees that we have found an RRQR factorization.

3 A Block Algorithm for Finding Rank-Revealing Permutations

Lemma 1 implies that the computation of a rank-revealing QR factorization hinges on finding a permutation P such that the smallest singular value of the bottom block W_2 of the nullspace matrix W of AP is reasonably large. This point is also crucial if we wish to be sure that the δ_k are close to the σ_k, due to the following Theorem:

Theorem 2 ([8]) *Let $W_2^{(i)}$ denote the bottom $(n-i+1) \times (n-i+1)$ block of W. Also, let $\delta_i \equiv \sigma_{\min}(R(1:i,1:i))$ and $\theta_i \equiv \|R(i:n,i:n)\|_2$. Then the i-th singular value σ_i of A satisfies*

$$\frac{\sigma_i}{\sqrt{n-i+1}\,\|(W_2^{(i)})^{-1}\|_2} \le \delta_i \le \sigma_i \le \theta_i \le \sigma_i\sqrt{n-i+1}\,\|(W_2^{(i)})^{-1}\|_2\,. \tag{8}$$

To obtain an RRQR factorization for the special case $p = 1$ (i.e., W in (6) is a vector, and i in Theorem 2 is $n-1$), one can take W to be the singular vector v_n corresponding to the smallest singular value of R and P the permutation that brings the largest element (in modulus) of v_n to the last position [14]. This choice then guarantees that $\mid r_{nn} \mid \le \sqrt{n}\,\sigma_{\min}(A)$.

This idea was extended by Foster [13] and Chan [8] to higher dimensions, namely the case $p > 1$. The idea is to "peel off" the small singular values of R one by one: for $i = n, n-1, \ldots, r$ one computes the right null-vectors $y^{(i)}$ of increasingly smaller $i \times i$ triangular submatrices. One then applies a left cyclic shift of columns j through i of R, where j is determined from

$$|y_j^{(i)}| = \|y^{(i)}\|_\infty\,.$$

This changes the triangular structure of R into upper Hessenberg form, in the sense that a lower bidiagonal appears from column j through column i. The triangular form of R is then restored by means of Givens rotations. It is shown in [9] that in this fashion we produce a matrix of approximate nullvectors W that satisfies $\|(W_2^{(i)})^{-1}\|_2 < \sqrt{n}\,2^{n-i}$, and hence the bounds in (8) are guaranteed to be tight as long as $p = n - r$ is not too large. Experimental results show that this a-priori bound is rather pessimistic, and this conclusion goes along with recent results by Hong and Pan [22] which show that there *exists* a permutation matrix P such that for the triangular factor R partitioned as in (2) we have

$$\|R_{22}\|_2 \le \sigma_{r+1}(A)\,\sqrt{r(n-r) + \min(r, n-r)}\,, \tag{9}$$

and

$$\sigma_{\min}(R_{11}) \ge \sigma_r(A)\,\frac{1}{\sqrt{r(n-r) + \min(r, n-r)}}. \tag{10}$$

No practical algorithm is known, however, for computing this particular permutation P.

The extra work in the Chan/Foster-algorithm, beyond the initial QR factorization, is $O((n-r)n^2)$. In essence, for each of the $n-r$ small singular values of A, it requires some triangular solves for the computation of $y^{(i)}$ and some Givens rotations to reduce the Hessenberg matrix back to triangular form.

In our approach, we construct a well-conditioned W_2 several columns at a time. Remember that through a triangular solve with the ICE vectors we obtained an upper trapezoidal matrix $Y \in \Re^{n\times(n-\tilde{r})}$ whose columns are approximations to the nullvectors of $R(1:i, 1:i), i = \tilde{r}+1, \ldots, n$ and hence approximate nullvectors of AP. If we can now find a row permutation Π and a well-conditioned $p \times (n-\tilde{r})$ bottom block of ΠY,

$$(\Pi Y)(n-p+1:n, 1:n-\tilde{r}), \qquad p \le n-\tilde{r},$$

then a QR factorization of $R\Pi^T$ is guaranteed to have a small trailing $p\times p$ block. If we set $p=1$, we obtain exactly the unblocked strategy. The main advantage of using $p>1$ is that we capture all the small singular values that are well approximated by the current QR factorization, whereas the unblocked algorithm would require several (compute nullvectors + update QR factorization) iterations to achieve the same effect.

As a result, the block approach is likely to require much fewer triangular solves (which are comparatively expensive for sparse matrices and parallel machines). We also have many more choices as to what permutations we employ. As will be seen in Section 5, this freedom allows one to reduce the work required for updates of the triangular factor and to increase the computational granularity of the orthogonal updates.

Before we present the outline of the block algorithm, we give some thought as to how we can generate a row permutation Π that generates a well-conditioned bottom block in ΠY. Consider the orthogonal factorization

$$\Pi Y = ZH, \tag{11}$$

where Y and Z are upper trapezoidal, that is, they have the form

$$\begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \\ 0 & * & * \\ 0 & 0 & * \end{pmatrix}.$$

Π is a permutation matrix, and H is orthogonal. Now define $E(l)$ to be an $l\times l$ "exchange matrix", for example,

$$E(4) = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

and let E_r be shorthand for E(number of rows of Y), and E_c be short for E(number of columns of Y). Then the factorization (11) is equivalent to

$$(E_c Y^T E_r)(E_r \Pi^T E_r) = (E_c H^T E_c)(E_c Z^T E_r), \tag{12}$$

and now $(E_c Y^T E_r)$ and $(E_c Z^T E_r)$ have the form

$$\begin{pmatrix} * & * & * & * & * & * & * & * \\ 0 & * & * & * & * & * & * & * \\ 0 & 0 & * & * & * & * & * & * \end{pmatrix};$$

that is, (12) is a QR factorization, whose objective is to find a column permutation that isolates a leading well-conditioned submatrix in $(E_c Y^T E_r)$. In essence, this is another (though somewhat weaker) rank-revealing QR factorization. Hence, through "flipping Y around", we are in a position to use the rank-revealing strategies we are already familiar with.

To generate a permutation Π we assemble Y^T and then flip its rows and columns to arrive at an upper triangular matrix

$$\tilde{Y} = (E_c Y^T E_r). \tag{13}$$

We then compute the QR factorization

$$\tilde{Y}\tilde{\Pi} = \tilde{H}\tilde{Z}, \tag{14}$$

such that $\tilde{Z}$ has a well-conditioned leading triangular submatrix. Comparing (11), (12), and (14), we immediately see that

$$\Pi \equiv E_r \tilde{\Pi}^T E_r. \tag{15}$$

As our block size p we choose the size of the largest leading triangular block of $\tilde{Z}$, whose smallest singular value is larger than a threshold ρ_Z. A leading triangular block in $\tilde{Z}$ corresponds to a trailing triangular block in

$$Z \equiv (E_r \tilde{Z}^T E_c). \tag{16}$$

As our p nullvectors we now choose the last p columns of Z, normalized to unit norm. This scaling could change the smallest singular value of Z, so one might add the additional condition that the norm of the first p columns be close to 1, but we found this to be naturally the case in our experiments; the smallest singular values of the last p columns of Z before and after scaling were essentially identical.

We also mention in passing that one could look at this problem as a "subset selection" problem [14, 17], where one strives to identify the p most linearly dependent columns in Y, but subset selection algorithms are more expensive than the approach suggested here.

We are now ready to present an outline of our block algorithm in Figure 1. After computing an initial QR factorization, we repeatedly compute

```
1.  Compute an initial QR Factorization: $AP = QR$.
    Use ICE to compute $\delta_k \approx \sigma_{\min}(R(1:k, 1:k))$
    and approx. left singular vectors $x_k$, $k = 1, \ldots, n$.
2.  Initial rank estimate: $\tilde{r} \leftarrow \mathrm{argmax}_{1 \le k \le n}\{\delta_k > \tau\}$
3   $n_{last} \leftarrow n$; $W \leftarrow [\,]$.
4.  repeat
5.      $Y \leftarrow R(1\!:\!n_{last}, 1\!:\!n_{last})^{-1} \left( \begin{pmatrix} x_{\tilde{r}+1} \\ \mathbf{0} \end{pmatrix} \cdots \; x_{n_{last}} \right)$.
        Form $\tilde{Y}$ as defined in (13).
6.      Compute permutation $\tilde{\Pi}$ and QR factorization $\tilde{Y}\,\tilde{\Pi} = \tilde{H}\tilde{Z}$
        such that $\sigma_{\min}(\tilde{Z}(1:p, 1:p)) \ge \rho_Z$ for some $1 \le p \le n_{last} - \tilde{r}$.
7.      Form $Z$ as defined in (16).
        Let $Z_p$ be the last $p$ columns of $Z$, normalized to have unit norm.
8.      Form $\Pi$ as defined in (15), and update the matrix of approximate nullvectors:
           $W \leftarrow \left( \begin{pmatrix} Z_p \\ 0 \end{pmatrix}, \begin{pmatrix} \Pi & 0 \\ 0 & I \end{pmatrix} W \right)$.
9.      Update QR factorization:
        compute $\tilde{Q}$ and $\tilde{R}$ such that $R(1\!:\!n_{last}, 1\!:\!n_{last})\Pi^T = \tilde{Q}\tilde{R}$
10.     $R \leftarrow \begin{pmatrix} \tilde{R} & \tilde{Q}^T R(1:n_{last}, n_{last}+1:n) \\ 0 & R(n_{last}+1:n, n_{last}+1:n) \end{pmatrix}$
        $Q \leftarrow Q \begin{pmatrix} \tilde{Q} & 0 \\ 0 & I \end{pmatrix}$; $P := P \begin{pmatrix} \Pi^T & 0 \\ 0 & I \end{pmatrix}$.
11.     $n_{last} \leftarrow n_{last} - p$;
12.     Use ICE to compute $\delta_k \approx \sigma_{\min}(R(1:k, 1:k))$ and approx. left
        singular vectors $x_k$, $k = 1, \ldots, n_{last}$ for the updated $R$.
13.     New rank estimate $\tilde{r} \leftarrow \mathrm{argmax}_{1 \le k \le n_{last}}\{\delta_k > \tau\}$
14. until $(\tilde{r} = n_{last})$
```

Figure 1: An outline of the block RRQR algorithm.

lower bounds on the rank of A and peel off blocks of small trailing submatrices by identifying well-conditioned submatrices in our current nullspace matrix. Since the smallest singular value of the leading $p \times p$ block of $\tilde{Z}$ is the same as the smallest singular value of the trailing $p \times p$ block in Z, our algorithm ensures that each bottom $p \times p$ block of Z_p (computed in step 7) has a smallest singular value that is not smaller than some threshold ρ_Z. The matrix W formed in step 9 will be upper trapezoidal, and the smallest singular value of each diagonal block will be at least ρ_Z. For example, when $n = 10$, $r = 4$, and we require three block steps, identifying $p = 1, 3, 2$, respectively, W will have the form

$$\begin{pmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ + & + & * & * & * & * \\ 0 & + & * & * & * & * \\ 0 & 0 & + & + & + & * \\ 0 & 0 & 0 & + & + & * \\ 0 & 0 & 0 & 0 & + & * \\ 0 & 0 & 0 & 0 & 0 & + \end{pmatrix},$$

where for emphasis the three diagonal blocks have been indicated by plusses.

So far we have not detailed which strategy we shall use to generate a well-conditioned leading triangular block in the QR factorization of $\tilde{Y}$. This is primarily an implementation issue, although an important one. We shall come back to this issue in Section 5.

4 Theoretical Aspects of the Block Algorithm

In this section we show that the algorithm described in the previous section does indeed compute the desired RRQR factorization. Specifically, we shall show that

1. at every block step we are always able to find at least a one-by-one well-conditioned subblock in $\tilde{Y}$; and

2. if each of the diagonal blocks of W_2 has a reasonably large smallest singular value, the same is true for W_2 itself.

Hence our block algorithm will terminate in at most $(n - \tilde{r}^{(0)})$ steps, where $\tilde{r}^{(0)}$ is the rank estimate derived from the initial QR factorization, and the bounds in (8) will be tight.

The first issue is easily resolved by the following lemma.

Lemma 3 *Let $\tilde{Y}$ be an $k \times l$ $(k \leq l)$ matrix with rows of unit norm. Then there exists a permutation $\tilde{\Pi}$ such that in the QR factorization $\tilde{Y}\tilde{\Pi} = \tilde{H}\tilde{Z}$, where $\tilde{H}$ is orthogonal and $\tilde{Z}$ is upper triangular, we have*

$$|\ z_{11}\ | \geq \max_{1 \leq j \leq l} \|\tilde{Y}(:, j)\|_2 \geq 1/\sqrt{l}.$$

Proof. If $\tilde{\Pi}$ is a permutation that exchanges rows 1 and j, the absolute value of z_{11} will be $\|\tilde{Y}(:,j)\|_2$, and the first inequality of the lemma follows immediately. For the second inequality, note that since the rows of $\tilde{Y}$ are normalized, any row of $\tilde{Y}$ contains an element whose absolute value is not smaller than $1/\sqrt{l}$, and as a result the maximum column norm of $\tilde{Y}$ will be greater than $1/\sqrt{l}$. □.

The bound in the lemma is pessimistic, in that $\max_{1\le j\le l} \|\tilde{Y}(:,j)\|_2$ is usually significantly greater than $1/\sqrt{l}$, but it shows that we shall make progress in our algorithm as long as we permute a column of $\tilde{Y}$ with a reasonable large norm into the first position, independent of whether the rest of the factorization of $\tilde{Y}$ captures the large singular values of $\tilde{Y}$ or not. This also shows that in the worst case we shall extract only one small singular value of R at every step, just as in the unblocked algorithm.

In our algorithm, we base the determination of $\tilde{r}$ on the lower bounds δ_k for σ_k. This is justified, as long as $\delta_{\tilde{r}} \approx \sigma_{\tilde{r}}$, that is, as long as the bounds in (8) are tight. To show this, we establish a bound on the smallest singular value of W_2, the bottom $(n-\tilde{r})\times(n-\tilde{r})$ submatrix of W constructed in the algorithm of Figure 1.

Theorem 4 *Let W_2 be a block upper triangular matrix whose columns have norm at most one, and with diagonal blocks $Z^{(J)},\ldots,Z^{(1)}$, where $Z^{(j)}$ is of size p_j and $\sigma_{min}(Z^{(j)}) \ge \rho_Z, j=1,\ldots,J$. Then*

$$\|W_2^{-1}\|_2 \le (\sqrt{p_1}+2)\cdots(\sqrt{p_{J-1}}+2)\,\|(Z^{(1)})^{-1}\|_2 \cdots \|(Z^{(J)})^{-1}\|_2. \qquad (17)$$

Proof. Consider the square block matrix $D = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$ all whose columns have norm at most one. The inverse of D is given by $D^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}BC^{-1} \\ 0 & C^{-1} \end{pmatrix}$, and therefore

$$\begin{aligned} \|D^{-1}\|_2 &\le \|A^{-1}\|_2 + \|C^{-1}\|_2 + \|A^{-1}\|_2\,\|B\|_2\,\|C^{-1}\|_2 \\ &\le (\|A^{-1}\|_2{}^{-1} + \|C^{-1}\|_2{}^{-1} + \|B\|_2)\,\|A^{-1}\|_2\,\|C^{-1}\|_2. \qquad (18) \end{aligned}$$

If p_B denotes the number of columns of B, we readily obtain $\|B\|_2 \le \|B\|_F \le \sqrt{p_B}$. Consider now $\|A^{-1}\|_2{}^{-1}$, which is identical to the smallest singular value of A. As a result of the interlacing inequalities for singular values, it follows that $\|A^{-1}\|_2{}^{-1}$ is less than or equal to the norm of any column of A, and thus $\|A^{-1}\|_2{}^{-1} \le 1$. The same holds for $\|C^{-1}\|_2{}^{-1}$. Hence, we obtain

$$\|D^{-1}\|_2 \le (\sqrt{p_B}+2)\|A^{-1}\|_2{}^{-1}\|C^{-1}\|_2{}^{-1}. \qquad (19)$$

Applying this bound recursively, we immediately arrive at (17). □

Using this result, we can now show that the bounds for $\sigma_{\tilde{r}}$ and $\sigma_{\tilde{r}+1}$ at the termination of our block algorithm are tight bounds.

Theorem 5 *Let W_2 denote the bottom $(n-\tilde{r})\times(n-\tilde{r})$ submatrix of W obtained at the termination of our block algorithm. Then*

$$\frac{\sigma_{\tilde{r}}}{3\,(n-\tilde{r}+1)\,\|W_2^{-1}\|_2} \leq \delta_{\tilde{r}} \leq \sigma_{\tilde{r}} \tag{20}$$

and

$$\sigma_{\tilde{r}+1} \leq \|R(\tilde{r}+1:n,\tilde{r}+1:n)\|_2 \leq \sigma_{\tilde{r}+1}\sqrt{n-\tilde{r}}\,\|W_2^{-1}\|_2. \tag{21}$$

Proof. Equation (21) was proved in [8, Theorem 3.1]. To prove (20), imagine that we perform one more step of the block post-processing scheme, with a fixed block-size of one (i.e., one step of the Chan/Foster algorithm). The bottom element of the right nullvector produced in this imaginary step is numerically greater than or equal to $\sqrt{n-\tilde{r}}$. If we left-append W with this nullvector to form $\bar{W} = \begin{pmatrix} \bar{W}_1 \\ \bar{W}_2 \end{pmatrix}$ with $\bar{W}_2 \in \Re^{(n-\tilde{r}+1)\times(n-\tilde{r}+1)}$, then we have from [8, Theorem 3.1]:

$$\frac{\sigma_{\tilde{r}}}{\sqrt{n-\tilde{r}+1}\,\|\bar{W}_2^{-1}\|_2} \leq \delta_{\tilde{r}} \leq \sigma_{\tilde{r}}.$$

If we then apply Eq. (18) to $\bar{W}_2$, we immediately obtain (20). □

Theorems 4 and 5 ensure that the bounds (20) and (21) will be tight as long as the diagonal blocks Z^j of W_2 are well conditioned, and as a result we shall correctly capture the numerical rank.

5 Implementation Considerations

In this section we discuss implementation issues of the block RRQR algorithm. In particular, we show the following:

1. Some (if only very restricted) column exchanges in the initial QR factorization are useful in preventing pathological underestimates of the initial rank.

2. Applying column pivoting to the columns that were rejected by ICE in the initial QR factorization helps in obtaining an initial Y that captures many of the small singular values, and hence increases the likelyhood of a large block size p in the first block iteration.

3. The structure of the permutations employed in the QR factorization (14) of $\tilde{Y}$ has a significant impact on the cost of the update of the QR factorization of $R(1:n_{last}, 1:n_{last})\Pi^T$ (step 9 of the algorithm).

If we do not allow any pivoting at all in the initial QR factorization of A, then our algorithm will not be effective in the (rare) cases where the very first rows of A are linearly dependent. The following example illustrates this fact.

Let A have numerical rank $r = n-1$, and let the very first column of A consist of elements of size σ_n. Then ICE correctly determines that the smallest singular value of *all* the leading principal submatrices of R are small, so the rank estimate from ICE is $\tilde{r} = 0$, and this leads to a first block step of block-size $p = n$. Even though we can hope to get a only well-conditioned 1×1 block in $\tilde{Y}$, the QR factorization (14) would be that of an $n \times n$ matrix.

In order to avoid such a pathological underestimate of the initial rank, we should use some column pivoting strategy that tries to maintain leading triangular matrices in R reasonably well conditioned and permutes columns that would make R too ill-conditioned to the back. The issue of computing an initial QR factorization with approximate RRQR structure has been considered in detail by Bischof and Hansen [4], and it has been shown there how ICE can be advantageously employed in this context. Schemes tailored to factorizations of dense matrices on various architectures are described in [1,3].

Let us now consider some issues related to the construction of the first Y matrix in the block algorithm. In computing the initial QR factorization of A, after $\tilde{r}$ steps we arrive at a situation where

$$AP = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & \tilde{A}_{22} \end{pmatrix} . \tag{22}$$

R_{11} is upper triangular of size $\tilde{r}$, and ICE tells us that the smallest singular value of any larger leading submatrix would be less than τ. That is, columns $\tilde{r}+1, \ldots, n$ of AP are considered linearly dependent with respect to the threshold τ. We now have the choice of completing the QR factorization (22) either without any further column exchanges, or by applying some column pivoting based on $\tilde{A}_{22}$. This decision does not change our choice of $\tilde{r}$, but we can greatly improve our chances of obtaining a well-conditioned $(n-\tilde{r}) \times (n-\tilde{r})$ submatrix $Y_2 \equiv Y(\tilde{r}+1:n, :)$.

To illustrate this aspect, consider the following example. Assume that in the partially completed QR factorization (22), after reducing the first column of $\tilde{A}_{22}$, the next diagonal element $(\tilde{r}+1, \tilde{r}+1)$ of R becomes zero. Then the smallest singular value of all leading submatrices of R is zero, and the vector

$$\begin{pmatrix} R_{11}^{-1} R_{12}(:,1) \\ -1 \end{pmatrix} , \tag{23}$$

appended with an appropriate number of zeros, is the corresponding nullvector. As a result, all the $n - \tilde{r}$ columns of the nullspace matrix Y assembled by ICE will consist of duplicates of the vector (23) appended with zeros. Hence, the rank of this nullspace matrix is one, leading to a block size of one in the first iteration. If, on the other hand, we apply a pivoting strategy to $\tilde{A}_{22}$ that tries to maintain leading well conditioned triangular matrices of the corresponding triangular factor R_{22}, then the smallest singular values of leading submatrices of R will not drop so quickly. Hence, the assembled ICE vectors are much more likely to be linearly independent, and the first block size is therefore likely to be close to $n - \tilde{r}$.

Another issue that warrants attention is the column exchanges employed during the orthogonal factorization of Y (step 6 of the block algorithm). The structure of the column exchanges employed in the factorization of $\tilde{Y}$ determines the structure of Π^T, and hence the cost of the QR factorization update of $R\Pi^T$.

As is the case in the unblocked Chan/Foster algorithm, we prefer permutations Π^T consisting of a sequence of *left cyclic shifts.* If Π^T is the product of p such shift permutations, then the matrix $R(1 : n_{last}, 1 : n_{last})\Pi^T$ in step 9 has exactly p nonzero lower diagonals, and so the cost of the update is at most $O(pn^2)$ flops. Note that by generating p lower bands, our block algorithm allows for the use of Householder transformations or block orthogonal transformations [7,25], in reducing $R\Pi^T$ back to triangular form, whereas the unblocked algorithm is limited to Givens rotations.

The following 10×10 example shows the structure of $R(1 : n_{last}, 1 : n_{last})\Pi^T$ for a block size of $p = 2$ and Π^T consisting of a left cyclic shift of columns 7 through 10, followed by a left cyclic shift of columns 5 through 9 (the corresponding column permutations $\tilde{\Pi}$ of $\tilde{Y}$ are a right cyclic shift of columns 1 through 4 followed by a right cyclic shift of columns 2 through 6):

$$\begin{pmatrix}
\ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast \\
 & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast \\
 & & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast \\
 & & & \ast & \ast & \ast & \ast & \ast & \ast & \ast \\
 & & & & \ast & \ast & \ast & \ast & \ast & \ast \\
 & & & & \ast & \ast & \ast & \ast & & \ast \\
 & & & & & \ast & \ast & \ast & & \ast \\
 & & & & & \ast & \ast & \ast & & \\
 & & & & & & \ast & \ast & & \\
 & & & & & & & \ast & &
\end{pmatrix}.$$

$R\Pi^T$ has two lower diagonals, starting from column 5. Obviously, the more of the leading triangle of R is preserved in $R\Pi$, the less the work involved in reducing $R\Pi$ back to triangular form. Thus, it is advantageous to restrict the permutations applied in the QR factorization of $\tilde{Y}$ to those that involve columns that are close to the beginning of $\tilde{Y}$.

A strategy that can easily be implemented is *threshold pivoting.* In the k-th step of the traditional column pivoting strategy [16]—cf. Eq. (22)—one chooses as pivot column j the column whose "residual" $||\tilde{A}_{22}(:, j)||_2$ is maximum. In threshold pivoting, the pivot column j is the closest column whose "residual" is within a factor ρ_Y of $\max_{1 \leq i \leq n-k} ||\tilde{A}_{22}(:, i)||_2$.

Another pivoting strategy one could employ to limit the extent of column exchanges in $\tilde{Y}$ is *restricted column pivoting.* Here, at every step, only a selected number of columns can be chosen as pivot candidates. In [4] it has been shown that such a strategy is useful in computing approximate RRQR factorizations

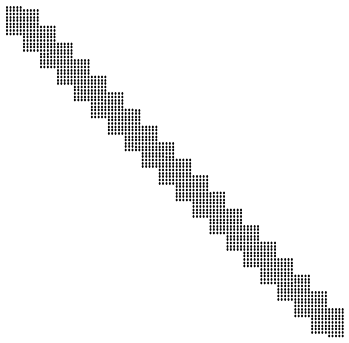

Figure 2: 100×100 matrix with staircase structure

while preserving the sparsity structure of the initial matrix, and this strategy could be applied here as well.

6 Numerical Results

This section illustrates some of the implementation issues, demonstrates the reliability of our algorithm, and compares it with the unblocked algorithm. Our test matrices were square matrices of order 100 and with a staircase structure as shown in Figure 2. Using PRO-MATLAB, we generated matrices of rank $r = 80$ and rank $r = 95$ with a geometric distribution of singular values between $\sigma_1 = 1$ and $\sigma_r = 10^{-2}$, and also between $\sigma_{r+1} = 10^{-5}$ and $\sigma_n = 10^{-7}$. There is a well-defined gap between "large" and "small" singular values, and our threshold $\tau = 5 \cdot 10^{-4}$ is chosen to lie within this gap. We generated five random matrices of rank 80 and 95 each. Since the orthogonal transformations we apply in obtaining a banded structure usually grade a matrix in that numerically small elements tend to appear in the lower right-hand corner, we generated five more test matrices by "flipping" those matrices, that is, we applied the exchange matrix $E(100)$ to rows and columns. As a result, if A has a small trailing submatrix, the flipped version of A has a small *leading* submatrix. In the latter case it is much harder to compute the rank-revealing structure, since the small submatrix has to be transferred to the lower right hand side during the factorization process in order to reveal the rank.

To avoid pathetic underestimates of the initial numerical rank, we employed a very restricted column pivoting strategy in the initial QR factorization: We considered only the next five columns as candidate pivot columns, and in that restricted window, we applied threshold pivoting with a threshold of 10. Even with this very restricted pivoting strategy, the initial rank estimate is usually close to the true rank.

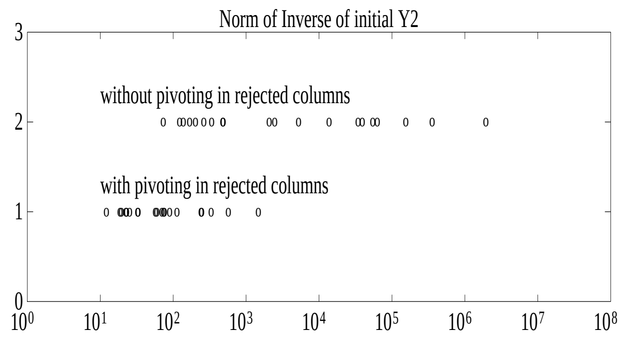

Figure 3: Histograms of $||Y_2^{-1}||_2$ for a set of 20 test-matrices, treated without (top) and with (bottom) column pivoting among the "rejected columns"

To illustrate the beneficial effect that column pivoting in $\tilde{A}_{22}$ (see (22)) has on the norm of Y_2^{-1}, we considered two cases. In one case, we did not perform any column exchanges in $\tilde{A}_{22}$; in the other we applied the traditional column pivoting strategy. Since $\tilde{A}_{22}$ corresponds to the bottom block of the columns that the initial QR factorization considered "dependent", and hence permuted to the back, $\tilde{A}_{22}$ is usually dense, and column pivoting introduces only negligible extra overhead. Nonetheless, the effect on $||Y_2^{-1}||_2$ is considerable, as is shown in Figure 3 which displays histograms of the distribution of $\log ||Y_2^{-1}||_2$ for the two cases. The top histogram corresponds to the case where we did not employ any column exchanges in the factorization of $\tilde{A}_{22}$, and as a result we obtain some fairly illconditioned Y_2. In contrast, column pivoting in $\tilde{A}_{22}$ leads to much better conditioned Y_2. Therefore, in our experiments, we always apply the traditional column pivoting strategy to the "dependent" columns in the initial QR factorization.

The measures of the quality of the RRQR factorization are

- the computed relative gap $\delta_{\tilde{r}}/\theta_{\tilde{r}+1}$ (where $\delta_{\tilde{r}}$ is the ICE-estimate of $\sigma_{\tilde{r}}$ and $\theta_{\tilde{r}+1} = ||R(\tilde{r}+1:n, \tilde{r}+1:n)||_2$ is an estimate of $\sigma_{\tilde{r}+1}$), and

- the norm $||W_2^{-1}||_2$ of the inverse of the bottom triangular block of the computed nullspace matrix W.

The relative gap should be close to σ_r/σ_{r+1} to ensure that we capture the correct numerical rank, and the gap between "large" and "small" singular values. $||W_2^{-1}||_2$ should be as small as possible to ensure tight guaranteed bounds for the estimates $\delta_{\tilde{r}}$ and $\theta_{\tilde{r}+1}$, and a good approximation to the nullspace.

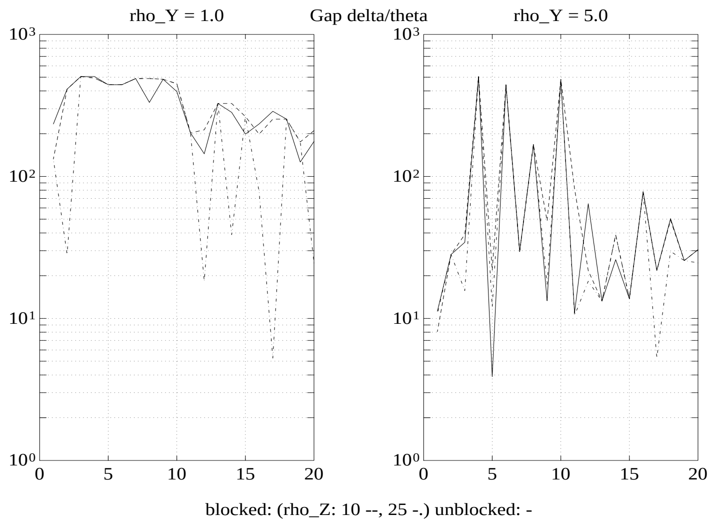

Figure 4: The computed relative gap $\delta_{\tilde{r}}/\theta_{\tilde{r}+1}$ using different thresholds ρ_Y and ρ_Z in the blocked and unblocked algorithms

In the block RRQR algorithm, we employ a QR factorization of Y with threshold column pivoting, with thresholds $\rho_Y = 1$ (i.e., the traditional strategy is performed), and $\rho_Y = 5$. The same thresholds are also employed in Chan's unblocked algorithm for deciding on the pivot column, i.e. one permutes column k of R to the back if

$$k = \arg\max_j \{ \frac{||y||_\infty}{\mid y_j \mid} \leq \rho_Y \},$$

where y is the singular vector corresponding to the smallest singular value of the currently considered leading triangular submatrix. For the threshold ρ_Z in step 6 of the algorithm, we experimented with

$$\rho_Z = 10.0, \quad 25.0, \quad 100.0.$$

The parameter ρ_Z has no equivalent in the unblocked algorithm.

In Figures 4 and 5 we show the gap $\delta_{\tilde{r}}/\theta_{\tilde{r}+1}$, and the norm $||W_2^{-1}||_2$ for the unblocked algorithm, and our block algorithm for $\rho_Z = 10$ and $\rho_Z = 25$. The left plot corresponds to choosing $\rho_Y = 1$, the right one to $\rho_Y = 5$. We see that the unblocked algorithm and the blocked one with $\rho_Z = 10$ behave just about the same, whereas $\rho_Z = 25$ can lead to worse results. For $\rho_Z = 25$ in some cases we were content with the initial QR factorization, and the resulting peaks in $||W_2^{-1}||_2$ are quite noticeable for $\rho_Y = 1$. For $\rho_Z = 100.0$ we always accepted the initial QR factorization, and in most cases we did not consider it sufficiently rank revealing to be generally acceptable.

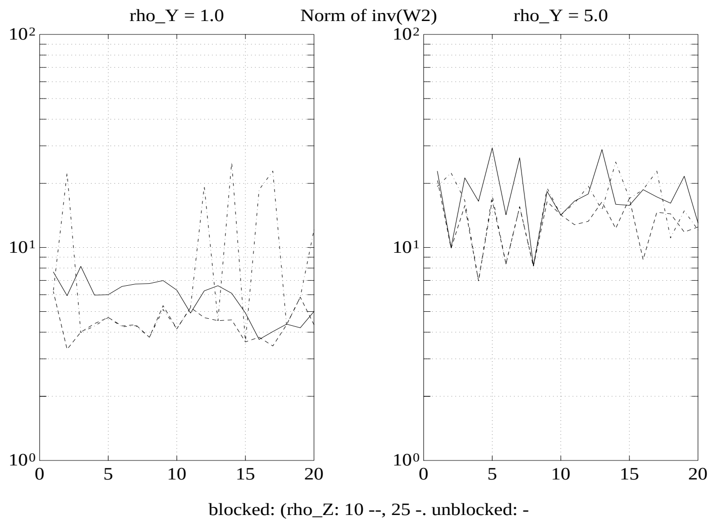

Figure 5: The norm $||W_2^{-1}||_2$, computed by using different thresholds ρ_Y and ρ_Z in the blocked and unblocked algorithms

The threshold ρ_Y has some influence on the quality of our RRQR factorization, in that $\rho_Y = 1$ leads to a relative gap $\delta_{\tilde{r}}/\theta_{\tilde{r}+1}$ consistently very close to its optimal value 10^3 and to very well conditioned W_2's. The cost of these good bounds is a significant amount of pivoting and updating of the QR factorization (steps 9 and 10 of the algorithm). Typically, the pivot column for these test matrices was one of the very first columns of R. We also see that when we change the threshold ρ_Y to 5.0 in order to reduce the extent of column exchanges and the cost of updating $R\Pi^T$, the quality measures deteriorate somewhat: $||W_2^{-1}||_2$ by a factor of about 10, with a more pronounced effect for the relative gaps. Nonetheless, we find they are still good enough to ensure the computation of a reliable RRQR factorization. The deterioration of $||W_2^{-1}||_2$ is due to the fact that more block steps are performed when $\rho_Y = 5$. For example, for $\rho_Y = 5$ the average number of block steps was 2.65, whereas it was 2.00 for $\rho_Y = 1$. As Theorem 4 suggests, $||W_2^{-1}||_2$ will deteriorate when more block steps are performed. For the block algorithm, there was no significant difference in the number of block steps performed for the matrices of rank 80 or 95, while the unblocked algorithm requires 20 and 5 steps, respectively. For all those algorithms, and essentially independent of the thresholds ρ_Y and ρ_Z, the norm $||A\,W||_2$ was of the order 10^{-5}.

In all cases, the block algorithm identified a large block in the first step, and the average values for the size of the first block is shown in Table 1. Recall

Table 1: Average Size of First Block

	$\rho_Y = 1$	$\rho_Y = 5$
$r = 80$	16.6	16.2
$r = 95$	4.2	2.9

that the maximum possible block sizes are 20 and 5, respectively. It is not surprising that the size of the first block is smaller when ρ_Y is larger, since then we work less to identify a well-conditioned leading triangle. This result confirms the computational benefits that we predicted for our block algorithm: While the unblocked algorithm has to identify each small singular value one by one, hence requiring $n - r$ iterations where we compute a nullvector and update a QR factorization, the block algorithm identifies all those small singular values that are well captured with the current QR factorization. Otherwise, we noticed much the same behavior, in that the same columns of A were permuted to the back, and this is borne out by the similar behavior of blocked and unblocked algorithms in Figures 4 and 5. We also noticed that for both the unblocked and the blocked algorithm, $\rho_Y = 5$ led to a significant decrease of the work.

From these experimental results we draw the following conclusions:

- The quality of the singular value estimates $\delta_{\tilde{r}}$ and $\theta_{\tilde{r}+1}$, and thus the reliability of the rank estimate $\tilde{r}$, is quite sensitive to the size of $||W_2^{-1}||_2$, that is, to the condition of the bottom triangular block of the null space matrix W.

- The fact that $||AW||_2$ is small does not guarantee that one has obtained an RRQR factorization.

- A pivoting threshold $\rho_Y > 1$ can significantly reduce the amount of work for updating $R\Pi^T$, at the expense of somewhat weaker bounds. A choice of $\rho_Y = 5$ seems to be adequate.

- The threshold ρ_Z has a large influence on the quality of the RRQR factorization. A threshold $\rho_Z = 10$ seems to be adequate to guarantee a reliable RRQR factorization.

- The first block in our block algorithm is usually by far the largest; thus, in this step, the computational savings of the block algorithm are the most pronounced.

7 Conclusions

We have presented a block algorithm for computing a rank-revealing QR factorization. The algorithm is based on incremental condition estimation as a means for computing blocks of approximate nullvectors, which are then used to isolate blocks of the triangular factor with small norm. While the unblocked algorithm peels off the small singular values one at a time, thereby neglecting implicit information about other small singular values already captured in the current QR factorization, the block algorithm captures all small singular values that are well approximated in the current QR factorization.

We showed how we can generate well-conditioned triangular blocks in the nullspace matrix (and hence small trailing blocks in the triangular factor) by computing what is essentially a rank-revealing QR factorization of a suitably permuted nullspace matrix. We also proved the correctness of this approach.

We discussed implementation issues of this algorithm and compared one such implementation with the unblocked algorithm. As expected, the blocked algorithm behaves numerically like the unblocked one, yet it requires many fewer passes. As a result, the block algorithm significantly decreases the number of triangular solves employed and allows for the use of Householder transformations. This feature makes the block algorithm attractive for sparse matrices and for high-performance (in particular parallel) architectures, where triangular solves and sequential Givens updates are expensive to implement.

Acknowledgments

We thank Wolfgang Rönsch for his comments which greatly improved the initial manuscript.

References

[1] Christian H. Bischof. A block QR factorization algorithm using restricted pivoting. In *Proceedings SUPERCOMPUTING '89*, pages 248–256, Baltimore, Md., 1989. ACM Press.

[2] Christian H. Bischof. Incremental condition estimation. *SIAM Journal on Matrix Analysis and Applications*, 11(2):312–322, 1990.

[3] Christian H. Bischof. A parallel QR factorization algorithm with controlled local pivoting. *SIAM Journal on Scientific and Statistical Computing*, 12(1):36–57, 1991.

[4] Christian H. Bischof and Per Christian Hansen. Structure-preserving and rank-revealing QR factorizations. *SIAM Journal on Scientific and Statistical Computing*, 12(6):1332–1350, 1991.

[5] Christian H. Bischof and Gautam M. Shroff. On updating signal subspaces. *IEEE Transactions on Signal Processing*, 40(1):96–105, 1992.

[6] Christian H. Bischof and Ping Tak Peter Tang. Robust incremental condition estimation. Preprint MCS-P225-0391, Argonne National Laboratory, Mathematics and Computer Science Division, 1991.

[7] Christian H. Bischof and Charles F. Van Loan. The WY representation for products of Householder matrices. *SIAM Journal on Scientific and Statistical Computing*, 8:s2–s13, 1987.

[8] Tony F. Chan. Rank revealing QR factorizations. *Linear Algebra and Its Applications*, 88/89:67–82, 1987.

[9] Tony F. Chan and Per Christian Hansen. Computing truncated SVD least squares solutions by rank revealing QR factorizations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):519–530, 1990.

[10] Tony F. Chan and Per Christian Hansen. Some applications of the rank revealing QR factorization. Technical Report 90-09, University of California at Los Angeles, Department of Mathematics, 1990.

[11] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford Press, London, 1987.

[12] Lars Eldén and Robert Schreiber. An application of systolic arrays to linear discrete ill-posed problems. *SIAM Journal on Scientific and Statistical Computing*, 7:892–903, 1986.

[13] L. V. Foster. Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra and Its Applications*, 74:47–71, 1986.

[14] G. H. Golub, V. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Technical Report TR–456, University of Maryland, Dept. of Computer Science, 1976.

[15] G. H. Golub, P. Manneback, and P. L. Toint. A comparison between some direct and iterative methods for certain large scale geodetic least-squares problem. *SIAM Journal on Scientific and Statistical Computing*, 7:799–816, 1986.

[16] Gene H. Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7:206–216, 1965.

[17] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1983.

[18] Thomas A. Grandine. An iterative method for computing multivariate C^1 piecewise polynomial interpolants. *Computer Aided Geometric Design*, 4:307–319, 1987.

[19] Thomas A. Grandine. Rank deficient interpolation and optimal design: An example. Technical Report SCA–TR–113, Boeing Computer Services, Engineering and Scientific Services Division, February 1989.

[20] Per Christian Hansen. Truncated SVD solutions to discrete ill-posed problems with ill-determined numerical rank. *SIAM Journal of Scientific and Statistical Computing* 11(3):503–518, 1990.

[21] Per Christian Hansen, Takishi Sekii, and Hiromoto Shibahashi. The modified truncated SVD-method for regularization in general form. submitted for publication in SIAM J. Sci. Stat. Computing, 1991.

[22] Yoo Pyo Hong and Chiang-Tsuan Pan. The rank revealing QR decomposition and SVD. Preprint MCS-P188-1090, Argonne National Laboratory, Mathematics and Computer Science Division, 1990.

[23] S. F. Hsieh, K. J. R. Liu, and K. Yao. *Comparisons of Truncated QR and SVD Methods for AR Spectral Estimations*, pages 403–418. Elsevier Science Publishers, 1991.

[24] Jorge Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Proceedings of the Dundee Conference on Numerical Analysis*, Berlin, 1978. Springer Verlag.

[25] Robert Schreiber and Charles Van Loan. A storage efficient WY representation for products of Householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 10(1):53–57, 1989.

[26] Gautam M. Shroff and Christian H. Bischof. Adaptive condition estimation for rank-one updates of QR factorizations. Preprint MCS-P166-0790, Argonne National Laboratory, Mathematics and Computer Science Division, 1990.

[27] G. W. Stewart. An updating algorithm for subspace tracking. Technical report, University of Maryland, Department of Computer Science, 1990.

[28] Bertil Waldén. *Using a Fast Signal Processor to Solve the Inverse Kinematic Problem with Special Emphasis on the Singularity Problem.* PhD thesis, Linköping University, Dept. of Mathematics, 1991.