1997

Rank revealing QR factorizations

Lily L. Dalton
San Jose State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

RANK REVEALING QR FACTORIZATIONS

A Thesis

Presented to

The Faculty of the Department of Mathematics

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Lily L. Dalton

August 1997

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

APPROVED FOR THE DEPARTMENT OF MATHEMATICS

Leslie V. Foster

Dr. Leslie V. Foster

Maurice C. Stanley

Dr. Maurice C. Stanley

Howard Swann

Dr. Howard S. G. Swann

APPROVED FOR THE UNIVERSITY

Serena H. Stanford

ABSTRACT

RANK REVEALING QR FACTORIZATIONS

by Lily L. Dalton

In the face of computer arithmetic errors or other errors in the entries in a matrix it is difficult or impossible to accurately determine the mathematical rank of a matrix A. Therefore, in practice, one must calculate the numerical rank not the mathematical rank of a matrix. The numerical rank depends on a tolerance ϵ and it is best defined using the singular values of A.

The singular value decomposition and QR with column pivoting are the two most widely used algorithms for finding the numerical rank of an m $\times$ n matrix A where $m \geq n$. Many mathematical programs like MATLAB have these algorithms built in already. We will show that the singular value decomposition is fairly inefficient and QR with column pivoting sometime fails.

As we will see, the Foster/Chan RRQR algorithm works well when used to find the numerical rank of the matrices where QR with column pivoting fails, and the algorithm is much more efficient than the singular value decomposition. We will also present some applications of the singular value decomposition and QR with column pivoting, and show that those applications can also be done using Foster/Chan RRQR. Last, we will discuss the strong RRQR.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction and Basic Definitions

1.1 Introduction

In the face of computer arithmetic errors or other errors in the entries in a matrix it is difficult or impossible to accurately determine the mathematical rank of a matrix A. Therefore, in practice, one must calculate the numerical rank not the mathematical rank of a matrix. The numerical rank depends on a tolerance ϵ and it is best defined using the singular values of A.

Let A be an m $\times$ n real matrix and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ be the singular values of A. We say A has numerical ϵ rank r if given $\epsilon > 0$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \epsilon \geq \sigma_{r+1} \geq \cdots \geq \sigma_n \geq 0$. A has rank r if $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$. For simplicity we will just say A has numerical rank r instead of numerical ϵ rank r.

Let $A\Pi = QR = Q\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$ be a rank revealing QR factorization of a real m $\times$ n matrix A where R_{11} is an r $\times$ r invertible upper triangular matrix, Π is a permutation matrix, and Q is an orthogonal matrix. Then the numerical rank of A is r. A rank revealing QR factorization separates the linearly independent columns from the linearly dependent ones.

The first rank revealing QR factorization algorithm was developed by Golub in 1965 [11]. This is the algorithm known as QR with column pivoting. It is an efficient algorithm but, as we will see, it has the potential to incorrectly identify the numerical rank for matrices with small rank deficiencies. About ten years later a second algorithm was proposed by Golub, Klema, and Stewart [10]. At almost the same time a third algorithm was introduced by Gragg and Stewart [12]. In the middle 80's Chan [4] and Foster [8] independently proposed another algorithm, which we call Foster/Chan RRQR. RRQR is an abbreviation for rank revealing QR factorization. The algorithm is efficient and, under certain mild assumptions, guarantees the accurate

determinations of numerical rank for small rank deficiencies. In the practice it seems also to work well for matrices with large rank deficiencies, but the theory does not guarantee this.

In 1992, Hong and Pan [16] presented another rank revealing QR factorization which is able to compute an estimated singular value that is accurate up to a factor proportional to a polynomial function in the matrix size. Therefore it correctly determines the numerical rank for matrices with large rank deficiencies. However, the algorithm is not so efficient. In 1994, Chandrasedkaran and Ipsen [2] introduced three rank revealing QR factorization that potentially all compute an estimated singular value that has about the same accuracy as Hong and Pan's algorithm in practice but they could not show it in theory. Recently an algorithm was proposed by Gu and Eisenstat [14] in 1996. This is the algorithm we call strong RRQR. As we will see it can be proven to be accurate and relatively efficient.

As we know the singular value decomposition and QR with column pivoting are the two most widely used algorithms for finding the numerical rank of an m $\times$ n matrix A where $m \geq n$. Many mathematical programs like MATLAB have these algorithms built in already. We will show that the singular value decomposition is fairly inefficient and QR with column pivoting sometime fails.

As we will see, the Foster/Chan RRQR algorithm works well when used to find the numerical rank of the matrices where QR with column pivoting fails, and the algorithm is much more efficient than the singular value decomposition. We will also present some applications of the singular value decomposition and QR with column pivoting, and show that those applications can also be done using Foster/Chan RRQR. Last, we will discuss the strong RRQR. All the matrices we use in this thesis have size m $\times$ n where $m \geq n$.

In section 2, we will introduce the singular value decomposition and three of

its applications. The applications are determining the numerical rank, the subset selection problem, and the least squares problem. In section 3 we will present QR with column pivoting, and show how to use it to find the numerical rank of A and give an example where the algorithm fails. In section 4 we will introduce the Foster/Chan algorithm, give some numerical examples, show how to use the Foster/Chan algorithm to find the numerical rank, and solve the subset selection and least squares problem. In section 5 we will present the strong RRQR algorithm.

Most of the theorems presented in Section 2 and Section 3 are proved in standard numerical linear algebra textbooks, so if no proof is given please refer to any standard numerical linear algebra textbook. My personal favorite is *Numerical Linear Algebra and Applications* by Biswa Nath Datta.

1.2 Basic Definitions

- $\|x\|_2 = (\sum x_i^2)^{1/2}$.
- $\|x\|_1 = \sum |x_i|$.
- $\|x\|_\infty = \max(|x_i|)$.
- $(x)_j$ = the j-th component of x.
- $dist(y, S) = \min\{\|y - z\|_2 : z \in S\}$.
- $\|A\|_p = \max_{x \neq 0} \|Ax\|_p / \|x\|_p$, *where* $p = 1, 2, \infty$.
- The 1-norm of A can be computed by $\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{m} |a_{ij}|$.
- A *flop* is a floating-point operation $\alpha \circ \beta$, where α and β are floating-point numbers and $\circ$ is one of $+$, $-$, $\times$, and $\div$. Taking the absolute value or comparing two floating-point numbers is also counted as a flop.

- $R_\epsilon(A)$ = the numerical ϵ rank of A.
- $N_\epsilon(A)$ = the numerical ϵ nullity of A.
- A real matrix A is called *nonsingular* if $R_\epsilon(A) = n$. Otherwise it is *singular*.
- A real matrix A is said to have *full column rank* if its columns are linearly independent.
- A real matrix A is said to have *full numerical rank* if it has full column rank. If A does not have full numerical rank, it is numerical rank deficient. For simplicity we will just say A is rank deficient instead of numerical rank deficient.
- *Angle between subspaces* Let F and G be subspaces in $\Re^m$ whose dimemsions satisfy

$$p = \dim(F) \geq \dim(G) = q \geq 1$$

The *principal angles* $\theta_1, \ldots, \theta_q \in [0, \frac{\pi}{2}]$ between F and G are defined recursively by

$$\cos(\theta_k) = \max_{u \in F} \max_{v \in G} u^T v = u_k^T v_k,$$

subject to

$\|u\|_2 = \|v\|_2 = 1,$

$u^T u_i = 0 \; for \; i = 1, \ldots, k-1,$

$v^T v_i = 0 \; for \; i = 1, \ldots, k-1.$

2 Singular Value Decomposition

In this section we will define the *Singular Value Decomposition* of an m × n matrix A and three applications of *Singular Value Decomposition*. The applications are the problem of determining numerical rank, the subset selection problem, and the least squares problem. For simplicity we will denote the *Singular Value Decomposition* of A as the SVD of A. Here is some notation that we will be using:

- $\sigma_i(A)$ = the i-th largest singular value of A.
- $\sigma_{max}(A)$ = the largest singular value of A.
- $\sigma_{min}(A)$ = the smallest singular value of A.

2.1 Definition of SVD

Theorem 2.1 (SVD) *If A is a real m × n matrix then there exist orthogonal matrices U, V and a diagonal matrix Σ such that*

$$U^T A V = \Sigma \tag{1}$$

where U is m × m, V is n × n, and the diagonal entries of Σ are $\sigma_1, \ldots, \sigma_n$ ($\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$).

$A = U\Sigma V^T$ is called the SVD of A.

Let u_i be the i-th column of U and v_i be the i-th column of V. The σ_i are singular values of A and the vectors u_i and v_i are, respectively the i-th left singular vector and the i-th right singular vector. And for $i = 1, \ldots, n$

$$Av_i = \sigma_i u_i.$$

The singular values of a matrix are unique but the singular vectors are not unique. Suppose A has a multiple singular value $\sigma > 0$, then the right singular vectors of σ can

be chosen to be linear combinations of any orthonormal basis of the space spanned by the eigenvectors associated with the multiple eigenvalue $\lambda = \sigma^2$ of AA^T.

The estimated total flop count for this algorithm is $2mn^2 + 4n^3$ [4].

2.2 Applications of SVD

In this section we will first discuss how to find the numerical rank of A using SVD, then use SVD to solve the subset selection problem, and finally solve the least squares problem using SVD.

2.2.1 The Numerical Rank Problem

Calculating the SVD of the matrix is a pretty good method for finding the numerical rank of a matrix A. In particular, many software packages such as MATLAB have the SVD built in. SVD is known to be a stable and reliable algorithm when used to find the numerical rank of A. In practical applications that use singular values, we need to know when to accept a computed singular value to be "zero". Of course we should also take into the consideration the "noise"(relative error) of the data. There are many criteria to use to accept a computed singular value to be "zero". A particular one is as mentioned in [6]; a practical criterion would the following:

> *Accept a computed singular value to be zero if it is less than or equal to* $\epsilon = 10^{-t}\|A\|_\infty$, *where the entries of* A *are correct to* t *digits.*

From [11] we can say:

> $R_\epsilon(A) = r$ *if the computed singular values* $\bar{\sigma}_1, \ldots, \bar{\sigma}_n$ *satisfy* $\bar{\sigma}_1 \geq \cdots \geq \bar{\sigma}_r > \epsilon \geq \bar{\sigma}_{r+1} \geq \cdots \geq \bar{\sigma}_n \geq 0$.

Theorem 2.2 *For an* $m \times n$ *matrix* A

- $N_\epsilon(A) = \max\{\dim(S) : S \text{ is a subspace of } \Re^n \text{ such that } x \in S, x \neq 0 \Rightarrow \frac{\|Ax\|_2}{\|x\|_2} \leq \epsilon\}$.

- $N_\epsilon(A) =$ *number of singular values of* A *that are* $\leq \epsilon$.
- $R_\epsilon(A) = \min\{\dim(S) :$ S *is a subspace of* $\Re^m$ *such that* $x \in \Re^n, x \neq 0 \Rightarrow \frac{dist(Ax,S)}{\|x\|_2} \leq \epsilon\}$.
- $R_\epsilon(A) =$ *number of singular values of* A *that are* $> \epsilon$.
- $R_\epsilon(A) = n - N_\epsilon(A)$.

Proof: Please see [8] $\diamond$.

As we can see it is fairly easy to find the numerical rank of a matrix using SVD, but do keep in mind that it may not be easy to find the numerical rank of a matrix when there is no sufficiently large gap between the singular values.

2.2.2 The Subset Selection Problem

In this section we will discuss solving the subset selection problem using the Singular Value Decomposition. Subset selection is the problem of determining r linearly independent columns of a given matrix A where $R_\epsilon(A) = r$. One approach would be examine all $\binom{n}{r}$ possible combinations but there are more efficient algorithms that choose either the best set or a good set of linearly independent columns. One approach is to find a permutation matrix Π such that the submatrix consisting of the first r columns of $A\Pi$ is as well-conditioned as possible.

Theorem 2.3 *Let* $A = U\Sigma V^T$ *be the SVD of* A, *and define the matrix* $B_1 \in \Re^{m \times r}$ *by*

$$A\Pi = [B_1, B_2]$$

where the numerical rank of A *is* r *and* Π *is an* $n \times n$ *permutation matrix. If*

$$\Pi^T V = \begin{pmatrix} \hat{V}_{11} & \hat{V}_{12} \\ \hat{V}_{21} & \hat{V}_{22} \end{pmatrix} \tag{2}$$

where $\hat{V}_{11}$ *is* $r \times r$ *and nonsingular.* $\hat{V}_{12}$ *is* $r \times (n\text{-}r)$. $\hat{V}_{21}$ *is* $(n\text{-}r) \times r$. $\hat{V}_{22}$ *is* $(n\text{-}r) \times (n\text{-}r)$. *then*

$$\frac{\sigma_r(A)}{\|\hat{V}_{11}^{-1}\|_2} \leq \sigma_r(B_1) \leq \sigma_r(A).$$

The above theorem suggests that in order to obtain a sufficiently independent subset of columns of A, we choose Π such that the resulting $\hat{V}_{11}$ is as well conditioned as possible. One way to obtain this matrix Π is to use QR with column pivoting which will be discussed in a later section.

The following algorithm for solving subset selection using SVD can be found in [11]:

Algorithm 2.1 *Given a real* $m \times n$ *matrix* A *and* $R_\epsilon(A) = r$, *the following algorithm computes a permutation* Π *such that the first* r *columns of* $A\Pi$ *are independent.*

1. *Compute* $A = U\Sigma V^T$, *the SVD of* A, *and save* V.

2. *Determine* $R_\epsilon(A)$ *of* A *and partition*

$$V = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}$$

where V_{11} *is* $r \times r$. V_{12} *is* $r \times (n\text{-}r)$. V_{21} *is* $(n\text{-}r) \times r$. V_{22} *is* $(n\text{-}r) \times (n\text{-}r)$.

3. *Use QR with column pivoting to compute*

$$Q^T[V_{11}^T, V_{21}^T]\Pi = [R_{11}, R_{12}]$$

where Q *is orthogonal,* Π *is a permutation matrix, and* R_{11} *is upper triangular and nonsingular, and set* $A\Pi = [B_1, B_2]$ *where* B_1 *has* r *columns and* B_2 *has* $n - r$ *columns.*

From equation 2 we have:

$$\begin{pmatrix} \hat{V}_{11} \\ \hat{V}_{21} \end{pmatrix} = \Pi^T \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} = \begin{pmatrix} R_{11}^T Q^T \\ R_{21}^T Q^T \end{pmatrix}.$$

Since

$$\hat{V}_{11}\hat{V}_{11}^T = (R_{11}^T Q^T)(R_{11}^T Q^T)^T = R_{11}^T Q^T Q R_{11} = R_{11}^T R_{11},$$

we have

$$\sigma_r(\hat{V}_{11}) = \sigma_r(R_{11}^T Q^T) = \sigma_r(R_{11}).$$

This implies that $\hat{V}_{11}$ is nonsingular (since R_{11} is nonsingular). Since $\sigma_r(\hat{V}_{11}) = \sigma_r(R_{11})$, we have $\|\hat{V}_{11}^{-1}\|_2 = \|R_{11}^{-1}\|_2$. From [11], we know: "heuristically, column pivoting tends to produce a well-conditioned R_{11}, and so the overall process tends to produce a well-conditioned $\hat{V}_{11}$". Therefore the matrix B_1 above is just what we are looking for.

2.2.3 The Least Squares Problem

In this section we show how solving the least squares problem allows us to deal with the rank-deficient case using the Singular Value Decomposition. The general context of a least squares problem is: Given an m × n matrix A and an m-vector b, we want to find an n-vector x such that the 2-norm of the residual vector, $\|Ax - b\|_2$, is as small as possible. It is usually written as:

$$\|r(x)\|_2 = \min \|Ax - b\|_2. \tag{3}$$

Let $A = U\Sigma V^T$ be the SVD of A, then we have

$$\begin{aligned} Ax - b &= U\Sigma V^T x - b \\ &= U(\Sigma V^T x) - U(U^T b) \\ &= U(\Sigma y - c) \end{aligned}$$

where $y = V^T x$ and $c = U^T b$. Since U is an orthogonal matrix, U preserves lengths. That is, $\|U(\Sigma y - c)\|_2 = \|\Sigma y - c\|_2$. Then $\|Ax - b\|_2 = \|\Sigma y - c\|_2$. So the use of the

SVD of A has reduced the least squares problem from a full matrix A to one with a diagonal matrix Σ: Find y such that $\|\Sigma y - c\|_2$ is minimum.

The reduced problem is very easy to solve. We have:

$$\|\Sigma y - c\|_2^2 = \sum_{i=1}^{r} |\sigma_i y_i - c_i|^2 + \sum_{i=r+1}^{m} |c_i|^2 \tag{4}$$

where $R_c(A) = r$. The vector

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

is given by:

$$y_i = \begin{cases} c_i/\sigma_i & \text{if } i \leq r \\ arbitrary & \text{if } i > r \end{cases}.$$

As we can see the y_i for $i > r$ do not appear in equation 4. Therefore they have no effect on r. The y_i for $i > r$ are chosen arbitrarily: therefore in the rank-deficient case we have infinitely many solutions to the least squares problem.

The calculation above is summarized in [6] as follows:

Algorithm 2.2 *Least Squares using SVD*

1. *Find the SVD of A:*

$$A = U\Sigma V^T.$$

2. *Form* $c = U^T b = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$.

3. *Compute* $y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$ *, choosing*

$$y_i = \begin{cases} c_i/\sigma_i & \text{if } i \leq r \\ arbitrary & \text{if } i > r \end{cases}.$$

4. Compute the family of least squares solutions:

$$x = Vy.$$

From step 3 we can say that in the rank deficient case, the minimum 2-norm solution is the one by setting $y_i = 0$ for $i > r$. Therefore we can express the solution as follows:

$$x = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i.$$

As we can see, using SVD to solve the least squares problem is fairly simple and it works for matrices with full numerical rank and rank deficient matrices. We will present QR with column pivoting in the next section.

3 QR Factorization with Column Pivoting

In this section we will define the algorithm QR with column pivoting developed by Golub in 1965. We will show how to use it to find the numerical rank of an m × n matrix A and give an example where the algorithm fails. QR with column pivoting is a fairly efficient algorithm, but it does not always determine the correct numerical rank for matrices of a certain type.

3.1 The QR Factorization with Column Pivoting Algorithm

Let $u \in \Re^n$ be nonzero. An n × n matrix H of the form

$$H = I - 2\frac{u^T u}{u u^T}$$

is called a Householder matrix. H has the following properties:

1. $\|Hx\|_2 = \|x\|_2$ for every $x \in \Re^n$.
2. H is an orthogonal matrix.
3. $H^2 = I$.
4. H has a simple eigenvalue -1 and (n-1)-fold eigenvalue 1.
5. $\det(H) = -1$.

Lemma 3.1 *Given a nonzero vector $x \neq e_1$, there always exists a Householder matrix H such that Hx is a multiple of e_1.*

Theorem 3.1 *Let A be an $m \times n$ matrix with numerical rank r. Then there exists an $n \times n$ permutation matrix Π and an $m \times m$ orthogonal matrix Q such that*

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

where R_{11} is an $r \times r$ upper triangular matrix with nonzero diagonal entries. If (math) rank of A is r, then $R_{22} = 0$.

Proof: Since the numerical rank of A is r, we can find a permutation matrix Π such that

$$A\Pi = (A_1, A_2)$$

where A_1 is m $\times$ r and has linearly independent columns. Let

$$Q^T A_1 = \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}$$

be the QR factorization of A_1, where R_{11} is an r $\times$ r upper triangular matrix with nonzero diagonal entries. Then

$$Q^T A\Pi = (Q^T A_1, Q^T A_2) = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \diamond .$$

Now we are ready to present the algorithm QR factorization with column pivoting. The following algorithm was presented in [6].

Algorithm 3.1 (QR Factorization with Column Pivoting) *Let A be a real m $\times$ n matrix. The following algorithm computes the QR factorization with column pivoting of A.*

1. *Find the column of A having the maximum norm. Form a permutation matrix Π_1 such that first column of $A\Pi_1$ has the maximum norm. Form a Householder matrix H_1 such that*

$$A^{(1)} = H_1 A\Pi_1$$

has zeros in the first column below the (1,1) entry.

2. *Find the column of the maximum norm of the submatrix $\hat{A}^{(1)}$ obtained from $A^{(1)}$ by deleting the first row and the first column. Form a permutation matrix $\hat{\Pi}_2$ such that the first column of $\hat{A}^{(1)}\hat{\Pi}_2$ has the maximum norm and then construct*

the permutation matrix where

$$\Pi_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \hat{\Pi}_2 & \\ 0 & & & \end{pmatrix}$$

Now construct a Householder matrix H_2 so that

$$A^{(2)} = H_2 A^{(1)} \Pi_2 = H_2 H_1 A \Pi_1 \Pi_2$$

has zeros in the second column of A_2 below the (2,2) entry.

- *The kth step can now easily be written down.*
- *The process is continued until the entries below the diagonal of the current matrix all become zero.*

Suppose r steps are needed. Then at the end of the rth step, we have

$$A^{(r)} = H_r \cdots H_1 A \Pi_1 \cdots \Pi_r = Q^T A \Pi = R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}.$$

The preceding method requires $4mnr - 2r^2(m+n) + 4r^3/3$ flops [11], which is a lot cheaper than the SVD that requires $2mn^2 + 4n^3$ [4] flops.

3.2 Methods and Problems with Using QR with Column Pivoting to Find the Numerical Rank

In this section we discuss how to find the numerical rank of a matrix A by using QR with Column Pivoting and show an example where it fails to reveal the correct numerical rank of a certain type of matrix.

Let $A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$ be the QR factorization with column pivoting of A, where R_{11} is r × r and R_{22} is "small" in some measure. Then we can say the numerical rank of A is r. As we can see calculating the QR with column pivoting is much cheaper than calculating the singular value decomposition.

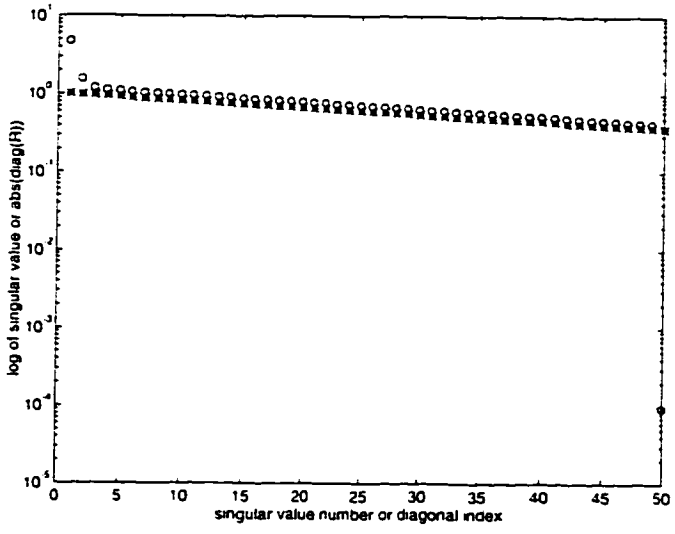

Figure 1: Kahan's example : n=50, c=0.2

$*$: the absolute values of the diagonal entries of the matrix R of QR with column pivoting, o : singular values.

Unfortunately, this algorithm does not always give the correct numerical rank. An example given by Kahan (1966) shows that a matrix can be nearly rank-deficient without having small $\|R_{22}\|_2$ at all.

Consider

$$A = diag(1, s, \cdots, s^{n-1}) \begin{pmatrix} 1 & -c & -c & \cdots & -c \\ & 1 & -c & \cdots & -c \\ & & 1 & & \vdots \\ & & & \ddots & \vdots \\ & & & & 1 \end{pmatrix}$$

with $c^2 + s^2 = 1$, $c, s > 0$. When n=50, c=0.2, we have $\sigma_n(A) \approx 10^{-4}$. Let $A\Pi = QR$ be the QR factorization with column pivoting of A. Since $r_{nn} \approx 10^{-1}$ which is not small, it has no small R_{22} block for any value of p where R_{22} is p $\times$ p. Figure 1 compares the absolute values of the diagonal entries of the matrix R of the QR with column pivoting algorithm with the singular values.

4 Foster/Chan Rank Revealing QR Factorization

The singular value decomposition and the QR factorization with column pivoting are the two most widely used methods for determining the numerical rank of an m × n matrix A. But as presented in the previous sections, QR with column pivoting does not always work and the singular value decomposition is fairly expensive. In this section we will present an algorithm that we call Foster/Chan RRQR and some of its applications. This algorithm was proposed by Chan and Foster independently and it works well with Kahan's example. In Section 4.1 we present Foster's algorithm; in Section 4.2 Chan's algorithm; in Section 4.3 some numerical examples; and in Section 4.4 some applications of the Foster/Chan RRQR and numerical examples.

The proofs of the first two lemmas will not be presented since they can be found in most numerical linear algebra textbooks.

Lemma 4.1 *Let A be an $m \times n$ real matrix with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ and B a matrix consisting of any n-1 columns of A with singular values $\hat{\sigma}_1 \geq \hat{\sigma}_2 \geq \cdots \geq \hat{\sigma}_{n-1}$, then*

$$\sigma_1 \geq \hat{\sigma}_1 \geq \sigma_2 \geq \hat{\sigma}_2 \geq \cdots \geq \hat{\sigma}_{n-1} \geq \sigma_n.$$

From the above lemma we can say that if B is a matrix consisting of any k columns of A then $\sigma_{\min}(B) = \sigma_k(B) \leq \sigma_k(A)$.

Lemma 4.2 (Courant-Fischer) *Let A be an $m \times n$ real matrix, then*

$$\sigma_{n-k+1}(A) = \min_S (\max_{x \in S} \|Ax\|_2 / \|x\|_2),$$

where the minimum is over subspaces S of dimension k.

Lemma 4.3 *Let $A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = QR$ be a QR factorization of a real $m \times n$ matrix A where $m \geq n$, Π is a permutation matrix, Q is an orthogonal matrix, and R_{11} is an upper triangular matrix. Then A and R have the same singular values.*

Proof: Since Q and Π are orthogonal matrices, $QQ^T = Q^TQ = I$ and $\Pi\Pi^T = \Pi^T\Pi = I$. Since

$$(QR)^T(QR) = R^TQ^TQR = R^TR.$$

QR and R have the same singular values. Since

$$(A\Pi)(A\Pi)^T = A\Pi\Pi^TA^T = AA^T,$$

we can say that $(A\Pi)^T$ and A^T have the same singular values. This implies that A and $A\Pi$ have the same singular values (since a real matrix and its transpose have the same singular values). Since

1. $A\Pi = QR$ and
2. R and QR have the same singular values and
3. A and $A\Pi$ have the same singular values,

we can conclude that A and R have the same singular values $\diamond$.

4.1 Foster's Algorithm

In this section we present Foster's algorithm for determining the numerical rank of a matrix A and compare the accuracy of this algorithm to the singular value decomposition.

The algorithm computes approximate singular vectors w_k for $k = n, n-1, \ldots,$ and corresponding approximate singular values e_k defined by

$$e_k = \frac{\|Aw_k\|_2}{\|w_k\|_2}, \; with \; e_n \leq e_{n-1} \leq \cdots.$$

Let $\epsilon > 0$. If the first e_k less or equal to ϵ is e_{n-p+1}, so that

$$e_k \leq \epsilon, k = n-p+1, \ldots, n \; and \; e_{n-p} > \epsilon. \tag{5}$$

then $N_\epsilon(A) = p$ is determined by the algorithm. For the approximation to be satisfactory from this point of view, the following two conditions must be satisfied. First

$$e_k \leq \gamma\sigma_k(A) \; for \; k = n-p-1, \ldots, n \tag{6}$$

with some fairly modest value γ. Secondly, since the true singular vectors are orthogonal(i.e., maximally linearly independent), the approximate singular vectors must also be far from linear dependence. To insure this, it must be true that

$$cond_1(w_{n-p+1}, \ldots, w_n) = cond_1(W) \leq C. \tag{7}$$

where C is a modest constant.

Theorem 4.1 *Let L_1 be the number of singular values of A less than or equal to $\frac{\epsilon}{\gamma}$, and let L_2 be the number of singular values less than or equal to $\sqrt{n}C\epsilon$. If the conditions (5) and (6) are satisfied, then $p \geq L_1$. If conditions (5) and (7) are satisfied, then $p \leq L_2$, where p is as defined above. If (5), (6), and (7) are true and A has a gap in its singular values with no singular values in*

$$\frac{\epsilon}{\gamma} < s \leq \sqrt{n}C\epsilon \tag{8}$$

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

Proof: (i) We want to show that (5) and (6) imply that $p \geq L_1$.

Since $\epsilon_{n-p} \leq \gamma\sigma_{n-p}$, $\epsilon_{n-p}/\gamma \leq \sigma_{n-p}$. Since $\epsilon < \epsilon_{n-p}$and $\epsilon_{n-p}/\gamma \leq \sigma_{n-p}$. $\epsilon/\gamma < \epsilon_{n-p}/\gamma \leq \sigma_{n-p}$. So $\epsilon/\gamma < \sigma_{n-p}$. Therefore $p \geq L_1$.

(ii) Now we want to show (5) and (7) imply $p \leq L_2$.

Let S be the subspace spanned by $w_{n-p+1}, \ldots, w_n$. Then for $w = Wy \in S$

$$\begin{aligned}
\|Aw\|_2 = \|AWy\|_2 &= \|A(w_{n-p+1}y_1 + \cdots + w_n y_p)\|_2 \\
&\leq \|y_1 Aw_{n-p+1}\|_2 + \cdots + \|y_p Aw_n\|_2 \\
&= \sum_{i=1}^{p} \|Aw_{n-p+i}\, y_i\|_2 \\
&\leq \sum_{i=1}^{p} \|Aw_{n-p+i}\|_2 \|y_i\|_2 \\
&\leq \max\{\|Aw_{n-p+i}\|_2 : \textit{for } i = 1, \ldots, p\} \sum_{i=1}^{p} |y_i|
\end{aligned}$$

$$
\begin{aligned}
&= \max\{\|Aw_{n-p+i}\|_2 : \text{for } i = 1,\ldots,p\}\|y\|_1 \\
&= \max\{\|w_{n-p+i}\|_2 e_{n-p+i} : \text{for } i = 1,\ldots,p\}\|y\|_1 \\
&\leq \max\{\|w_{n-p+i}\|_2 \epsilon : \text{for } i = 1,\ldots,p\}\|y\|_1 \\
&= \epsilon(\max\{\|w_{n-p+i}\|_2 : \text{for } i = 1,\ldots,p\})\|y\|_1
\end{aligned}
$$

using (5) and properties of matrix norms. Since $cond_1(W) = \|W\|_1 \max(\|z\|_1/\|Wz\|_1)$ (where $z \neq 0$) is a property of condition numbers, $C \geq (\|W\|_1\|y\|_1/\|w\|_1)$. This implies that $1/\|w\|_1 \leq (C/\|W\|_1\|y\|_1)$. We know $\frac{\|x\|_1}{\sqrt{n}} \leq \|x\|_2$ for any n-vector x. Therefore for any $w \in S$,

$$
\begin{aligned}
\|Aw\|_2/\|w\|_2 &\leq \sqrt{n}\epsilon(\max\{\|w_{n-p+i}\|_2 : \text{for } i = 1,\ldots,p\})\|y\|_1/\|w\|_1 \\
&\leq \sqrt{n}\epsilon(\max\{\|w_{n-p+i}\|_2 : \text{for } i = 1,\ldots,p\})\|y\|_1 C/(\|W\|_1\|y\|_1) \\
&= \frac{\sqrt{n}\epsilon(\max\{\|w_{n-p+i}\|_2 : \text{for } i = 1,\ldots,p\})C}{(\max\{\|w_{n-p+i}\|_1 : \text{for } i = 1,\ldots,p\})} \\
&\leq \sqrt{n}\epsilon C.
\end{aligned}
$$

Since S is a subspace of dimension p, by the Courant-Fischer theorem the singular value σ_{n-p+1} of A satisfies $\sigma_{n-p+1} \leq \sqrt{n}C\epsilon$. Therefore A has at least p singular values less than or equal to $\sqrt{n}C\epsilon$. So $p \leq l_2$.

(iii) Finally we want show that if (5), (6), (7), and (8) are true, then $p = N_\epsilon(A)$. If (5), (6), (7), and (8) are true, then $L_1 = L_2$ and $p = L_1 = L_2 = N_\epsilon(A)$ $\diamond$.

If the rank determination is based on approximate singular values and (1) is true, then the estimated ϵ nullity determined by this algorithm should be p. If (5), (6), and (7) are true and ϵ is chosen to be in a sufficiently large gap between the singular values of A as it is in (8), then the ϵ nullity calculated by this algorithm and the ϵ nullity calculated by the singular value decomposition are the same. Even if there is no sufficiently large gap between singular values, Foster felt this would not be a serious limitation in his algorithm since in this situation even using the singular value decomposition the ϵ nullity is unclear [10].

Now we are ready to present Foster's algorithm. In order to present Foster's algorithm, we must assume that a parameter ϵ is given. The following algorithm was proposed in [8].

Algorithm 4.1 (Foster's RRQR) *We want to determine the ϵ nullity p of an $m \times n$ matrix A and an ϵ null space S of A.*

1. *Let $A_n = A$, $k = n$, $\rho = 0$. Calculate $A = QR$ the QR factorization of A. Use the LINPACK subroutine SQRDC [7] (with or without the pivoting option)- or, the case of a sparse matrix A, the algorithm of [9]. Let $R_n = R$.*

2. *Compute w_k the approximate singular vector of R_k corresponding to e_k. Use the LINPACK subroutine STRCO [7] -or, in the case of sparse matrices, its sparse modification [13].*

3. *If $e_k \leq \epsilon$, let $\rho = \rho + 1$, and let $\hat{w}_k$ be an n-vector formed by expanding w_k putting zeros in elements corresponding to columns of A dropped in forming A_k. Now drop the column of R_k (and A_k) corresponding to the component of w_k with largest magnitude, and retriangularize the result using Givens transformations to form R_{k-1}. Let $k = k - 1$ and go to 2.*

4. *If $e_k > \epsilon$, stop. Let the current ρ be p, the calculated nullity of A, and let $\hat{w}_k = w_k$ for $i = n - p + 1, \ldots, n$, then $S = span\ \{w_{n-p+1}, \ldots, w_n\}$.*

Sine $R_\epsilon(A) = n - N_\epsilon(A)$ and $N_\epsilon = p$, $R_\epsilon(A) = n - p$. In a later section we show that this algorithm will work with Kahan's example.

When A is a sparse matrix, the SVD is inefficient since excessive fill-in occurs. Excessive fill-in is also often a problem with the QR with column pivoting algorithm. Foster's algorithm preserves the sparsity of the matrix. Therefore, Foster believes that in the case of sparse matrices when p is not too large, his algorithm will be more

efficient than SVD and QR with column pivoting. In the case of dense matrices, he believes his algorithm would be "more efficient than the SVD and comparable to (small p) or somewhat less efficient than (p not small) SQRDC [7] with column interchanges".

4.2 Chan's Algorithm

Chan's algorithm is basically the same as Foster's algorithm with a few minor differences. In Chan's paper, he first presented the numerical rank one deficiency case and then described an algorithm for the general rank deficiency case.

Theorem 4.2 *Let A be a real $m \times n$ matrix. Suppose that we have a vector $x \in \Re^n$ with $\|x\|_2 = 1$ such that $\|Ax\|_2 = \delta$, and let Π be a permutation matrix 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*

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

Proof: Since $\|\Pi^T x\|_2 = \|x\|_2$ and $\Pi^T x = y$, we have $\|y\|_2 = \|x\|_2$. Since $\|y\|_2 = \sqrt{y_1^2 + \cdots + y_n^2} = 1$, $y_1^2 + \cdots + y_n^2 = 1$. Since $|y_n| = \|y\|_\infty$ and $y_1^2 + \cdots + y_n^2 = 1$, we have $|y_n| \geq \sqrt{\frac{1}{n}}$. So we have

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

Therefore,

$$\delta = \|Ax\|_2 = \|Q^T Ax\|_2 = \|Ry\|_2 \geq |r_{nn}y_n| = |r_{nn}||y_n| \geq \sqrt{\frac{1}{n}}|r_{nn}|.$$

Then we have $|r_{nn}| \leq \sqrt{n}\delta$ $\diamond$.

When A is numerical rank one deficient, let $v \in \Re^n$ with $\|v\|_2 = 1$ be the right singular vector of A corresponding to the smallest singular value σ_n. Then we have

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

Therefore by the theorem above, if we define the permutation Π by

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

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 we only need the permutation matrix Π and v, we don't need to calculate the SVD of A in order to find v. There are methods to compute an approximated v from which the permutation matrix Π can be determined, but we won't discuss those methods here. See [1.15] for the methods.

Now we are ready to present Chan's algorithm. As with Foster's algorithm, in order to present Chan's algorithm we must assume that a parameter $\epsilon > 0$ is given. The following algorithm was proposed in [4].

Algorithm 4.2 (Chan's RRQR) *Let A be a real $m \times n$ matrix. We want to find the ϵ rank of A.*

1. *Compute a QR factorization of A: $A\Pi = QR$.*
2. *Let $i = n$, $p = 0$, $Q_n = Q$, $\Pi_n = \Pi$, and $R_n = R$.*
3. *Let $R_i = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$ where R_{11} is $i \times i$ and R_{12} is (m-i) $\times$ i, and R_{22} is (m-i) $\times$ (n-i).*
4. *Compute the singular vector $v_i \in \Re^i$ of R_{11} corresponding to $\sigma_{\min}(R_{11})$ with $\|v_i\|_2 = 1$, and set $\delta_i = \sigma_{\min}(R_{11})$.*
5. *If $\sigma_{\min}(R_{11}) \leq \epsilon$, let $p = p + 1$.*
6. *Compute a permutation matrix P_i such that $|(P_i^T v_i)_i| = \|P_i^T v_i\|_\infty = \|v_i\|_\infty$.*
7. *Assign $\hat{w}_i \equiv \begin{pmatrix} v_i \\ 0 \end{pmatrix} \in \Re^n$ to the ith column of W.*

8. *Compute* $W = \hat{P}_i^T W$. *where* $\hat{P}_i^T \equiv \begin{pmatrix} P_i & 0 \\ 0 & I \end{pmatrix}$ *and* $W \equiv [w_i, \cdots, w_n]$.

9. *Compute the QR factorization:* $R_{11} P_i = Q_1 \hat{R}_{11}$.

10. $\Pi_{i-1} = \Pi_i \hat{P}_i$.

11. $Q_{i-1} = Q_i \begin{pmatrix} Q_1 & 0 \\ 0 & I \end{pmatrix}$.

12. $R_{i-1} = \begin{pmatrix} \hat{R}_{11} & Q_1^T R_{12} \\ 0 & R_{22} \end{pmatrix}$. *Let i = i-1 and go to 3.*

13. *If* $\sigma_{\min}(R_{11}) > \epsilon$. *stop and let* $R = R_i$. $\Pi = \Pi_i$, *and* $Q = Q_i$.

Then $R_\epsilon(A) = n - p$.

Theorem 4.3 *The matrix* $W \equiv [w_{n-p+1}, \ldots, w_n] \in \Re^{n \times p}$ *computed by Algorithm 4.2 satisfies* $\|A\Pi w_i\|_2 = \delta_i \leq \sigma_i(A)$ *for* $i = n, \ldots, n-p+1$.

Proof: We will prove this theorem by induction on p: when p=1.

$$\begin{aligned} \|A\Pi w_n\|_2 &= \|QQ_1 R v_n\|_2 \\ &= \|R v_n\|_2 \\ &= \sigma_{\min}(R) \\ &= \delta_n \\ &= \sigma_n(A) \end{aligned}$$

where the last bound is arrived at by the fact that R has the same singular values as A. Suppose for some p=k, $\|A\Pi w_{n-k+1}\|_2 = \delta_{n-k+1} \leq \sigma_{n-k+1}(A)$. Then for p=k+1, first note that the permutation matrix $\Pi = \Pi_{n-k} \hat{P}_{n-k}$, so we have

$$\begin{aligned} \|A\Pi w_{n-k}\|_2 &= \|A\Pi_{n-k} \hat{P}_{n-k} w_{n-k}\|_2 \\ &\leq \|A\Pi_{n-k} \hat{P}_{n-k} \hat{P}_{n-k}^T \hat{w}_{n-k}\|_2 \end{aligned}$$

$$
\begin{aligned}
&= \|A\Pi_{n-k}\hat{w}_{n-k}\|_2 \\
&= \|Q_{n-k}R_{n-k}\begin{pmatrix} v_{n-k} \\ 0 \end{pmatrix}\|_2 \\
&= \|\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}\begin{pmatrix} v_{n-k} \\ 0 \end{pmatrix}\|_2 \\
&= \|R_{11}v_{n-k}\|_2 \\
&= \sigma_{\min}(R_{11}) \\
&= \delta_{n-k} \\
&\leq \sigma_{n-k}(A)
\end{aligned}
$$

where the last bound is arrived by using Lemma 4.1 and the fact that R and A have the same singular values. This completes the proof $\diamond$.

Theorem 4.4 *Let $A\Pi = QR$ be the RRQR of A computed by Algorithm 4.2 where the numerical rank is n-p. Then the elements of the lower $p \times p$ upper triangular block of R satisfy*

$$
\begin{aligned}
|r_{ij}| &\leq \sigma_j\sqrt{j} + \sum_{k=i}^{j-1} 2^{i-1-k}\sigma_k\sqrt{k} && (9) \\
&\leq 2^{j-i}\sigma_i\sqrt{n} \ \ for\ n-p < i \leq j \leq n. && (10)
\end{aligned}
$$

Proof: From the theorem above, we have, for $n - p < i \leq j \leq n$,

$$
\begin{aligned}
\sigma_j(A) \geq \|A\Pi w_j\|_2 &= \|QRw_j\|_2 \\
&= \|Rw_j\|_2 \\
&\geq |(Rw_j)_i| \\
&= |\sum_{k=i}^{j} r_{ik}(w_j)_k| \\
&= |r_{ij}(w_j)_j + \sum_{k=i}^{j-1} r_{ik}(w_j)_k| \\
&= |r_{ij}(w_j)_j - (-\sum_{k=i}^{j-1} r_{ik}(w_j)_k)|
\end{aligned}
$$

$$\begin{aligned} &\geq |r_{ij}(w_j)_j| - |-\sum_{k=i}^{j-1} r_{ik}(w_j)_k| \\ &= |r_{ij}(w_j)_j| - |\sum_{k=i}^{j-1} r_{ik}(w_j)_k|. \end{aligned}$$

This implies $|r_{ij}(w_j)_j| \leq \sigma_j + |\sum_{k=i}^{j-1} r_{ik}(w_j)_k|$. Since $|(w_j)_j| = \|w_j\|_\infty \geq \frac{1}{\sqrt{j}}$, we have

$$\begin{aligned} \frac{|r_{ij}(w_j)_j|}{|(w_j)_j|} &\leq \frac{\sigma_j}{|(w_j)_j|} + \sum_{k=i}^{j-1} \frac{|r_{ik}(w_j)_k|}{|(w_j)_j|} \\ &\leq \sigma_j\sqrt{j} + \sum_{k=i}^{j-1} |r_{ik}|. \end{aligned}$$

Now we will proof (9) by induction on j: when j=i, $|r_{ii}| \leq \sigma_i\sqrt{i} = \sigma_i\sqrt{i} + \sum_{k=i}^{i-1} 2^{i-1-k}\sigma_k\sqrt{k}$; when j=i+1,

$$\begin{aligned} |r_{i\,i+1}| &\leq \sigma_{i+1}\sqrt{i+1} + |r_{ii}| \\ &\leq \sigma_{i+1}\sqrt{i+1} + \sigma_i\sqrt{i} \\ &= \sigma_{i+1}\sqrt{i+1} + \sum_{k=i}^{i} \sigma_k\sqrt{k} \\ &= \sigma_{i+1}\sqrt{i+1} + \sum_{k=i}^{i} 2^{i-k}\sigma_k\sqrt{k} \\ &= \sigma_{i+1}\sqrt{i+1} + \sum_{k=i}^{(i+1)-1} 2^{(i+1)-1-k}\sigma_k\sqrt{k}. \end{aligned}$$

Suppose for some j=m, $|r_{im}| \leq \sigma_m\sqrt{m} + \sum_{k=i}^{m-1} 2^{m-1-k}\sigma_k\sqrt{k}$ for all i where $n - p < i \leq j$. Then for j=m+1, we have

$$\begin{aligned} |r_{i\,m+1}| &\leq \sigma_{m+1}\sqrt{m+1} + |r_{ii}| + \cdots + |r_{im}| \\ &= \sigma_{m+1}\sqrt{m+1} + \sigma_i\sqrt{i} + \sigma_{i+1}\sqrt{i+1} + \sum_{k=i}^{i} 2^{i-k}\sigma_k\sqrt{k} \\ &\quad + \cdots + \sigma_m\sqrt{m} + \sum_{k=i}^{m-1} 2^{m-1-k}\sigma_k\sqrt{k} \\ &= \sigma_{m+1}\sqrt{m+1} + \sum_{k=i}^{i+1} 2^{i+1-k}\sigma_k\sqrt{k} + \sigma_{i+2}\sqrt{i+2} + \sum_{k=i}^{i+1} 2^{i+1-k}\sigma_k\sqrt{k} \\ &\quad + \cdots + \sigma_m\sqrt{m} + \sum_{k=i}^{m-1} 2^{m-1-k}\sigma_k\sqrt{k} \end{aligned}$$

$$
\begin{aligned}
&= \sigma_{m+1}\sqrt{m+1} + \sum_{k=i}^{i+2} 2^{i+2-k}\sigma_k\sqrt{k} + \sigma_{i+3}\sqrt{i+3} + \sum_{k=i}^{i+2} 2^{i+2-k}\sigma_k\sqrt{k} \\
&\quad + \cdots + \sigma_m\sqrt{m} + \sum_{k=i}^{m-1} 2^{m-1-k}\sigma_k\sqrt{k} \\
&\vdots \\
&= \sigma_{m+1}\sqrt{m+1} + \sum_{k=i}^{m} 2^{m-k}\sigma_k\sqrt{k} \\
&= \sigma_{m+1}\sqrt{m+1} + \sum_{k=i}^{(m+1)-1} 2^{(m+1)-1-k}\sigma_k\sqrt{k}.
\end{aligned}
$$

Therefore, we have shown the bound in (9). Using the bound $\sigma_k\sqrt{k} \leq \sigma_i\sqrt{n}$ in each term in the sum in (9), we get the bound in (10) $\diamond$.

The bound for the element $|r_{n-p+1,n}|$ grows like 2^p. For large values of p, this bound can be quite large in theory but numerical examples have shown otherwise [4]. When a matrix A with small p, the bound in the theorem above should be quite reasonable. For example, if p=3, then we have, for $n-3 < i \leq j \leq n$,

$$\max|r_{ij}| \leq 2^3\sqrt{n}\sigma_i = 8\sqrt{n}\sigma_i.$$

So the small singular values of A will be revealed in the triangular factor R. Chan believes for matrices of low rank deficiency, his RRQR is guaranteed to produce rank revealing QR factorizations.

Chan's algorithm consists of three main parts: the computation of the initial QR factorization, the computation of the singular vector v_i of R_{11} corresponding to $\sigma_{\min}(R_{11})$ at each iteration, and the computation of the QR factorization of RP_i at each iteration.

In Chan's algorithm the permutation P_i is needed at each iteration. In order to find P_i we need to compute v_i. One can use a few steps of inverse iteration [3, 15] to compute an approximated v_i. Then it is not necessary to compute the SVD of R_{11} in order to find v_i. By ignoring the lower terms, the computation of the initial QR

factorization takes $n^2(m - n/3)$ flops (supposing that Q need not be accumulated) [4] and the computation of v_i at each iteration takes In^2p flops where I is the number of inverse iterations used at each iteration (usually $I = 2$ is sufficient in practice) [4].

If we use the Householder transformations to factor $R_{11}P_i$ at each iteration, then the algorithm would be extremely inefficient. Givens rotation is much more efficient. Supposing the worst case in each iteration, i.e., that the first column of R_{11} is to be permuted to the last column every time, this takes $2n^2p$ flops [4]. The total flop count for Chan's algorithm is $n^2(m - \frac{n}{3}) + 4n^2p$(supposing $I = 2$) [4]. In the worst case where p=n, the total flop count for Chan's algorithm is $mn^2 + \frac{11}{3}n^3$ [4]. It is still much smaller than the computation of SVD which takes $2mn^2 + 4n^3$ flops [4].

4.3 Numerical Examples

In this section we will show some examples to illustrate that RRQR does reveal the numerical rank of an $m \times n$ matrix where $m \geq n$. All numerical examples were done using MATLAB. The MATLAB program RRQR was written by Bischof and Hansen in 1991. We will present eight examples:

- Matrix A is a random 50 × 50 matrix.
- Matrix B is a 50 × 50 Kahan's matrix with c = 0.2.
- Matrix C is a 50 × 50 Hilbert's matrix (*).
- Matrix D is a 50 × 50 Lotkin's matrix (**).
- Matrix E is a 50 × 50 random svd matrix (***) with one large singular value.
- Matrix F is a 50 × 50 random svd matrix (***) with geometrically distributed singular values.

- Matrix G is a 50 × 50 random svd matrix (* * *) with uniformly distributed logarithmic singular values.

- Matrix H is a 50 × 50 random svd matrix (* * *) with a large gap between singular values.

* Hilbert's matrix is a square matrix with elements 1/(i+j-1). It is a famous example of a badly conditioned matrix.

** The Lotkin's matrix is the Hilbert matrix with its first row altered to all ones. This matrix is unsymmetric, ill-conditioned, and has many negative eigenvalues of small magnitude. Its inverse has integer entries and is known explicitly.

* * * A random svd matrix is a random matrix with pre-assigned singular values.

-: QR with column pivoting, * : Foster/Chan RRQR, o : singular value.

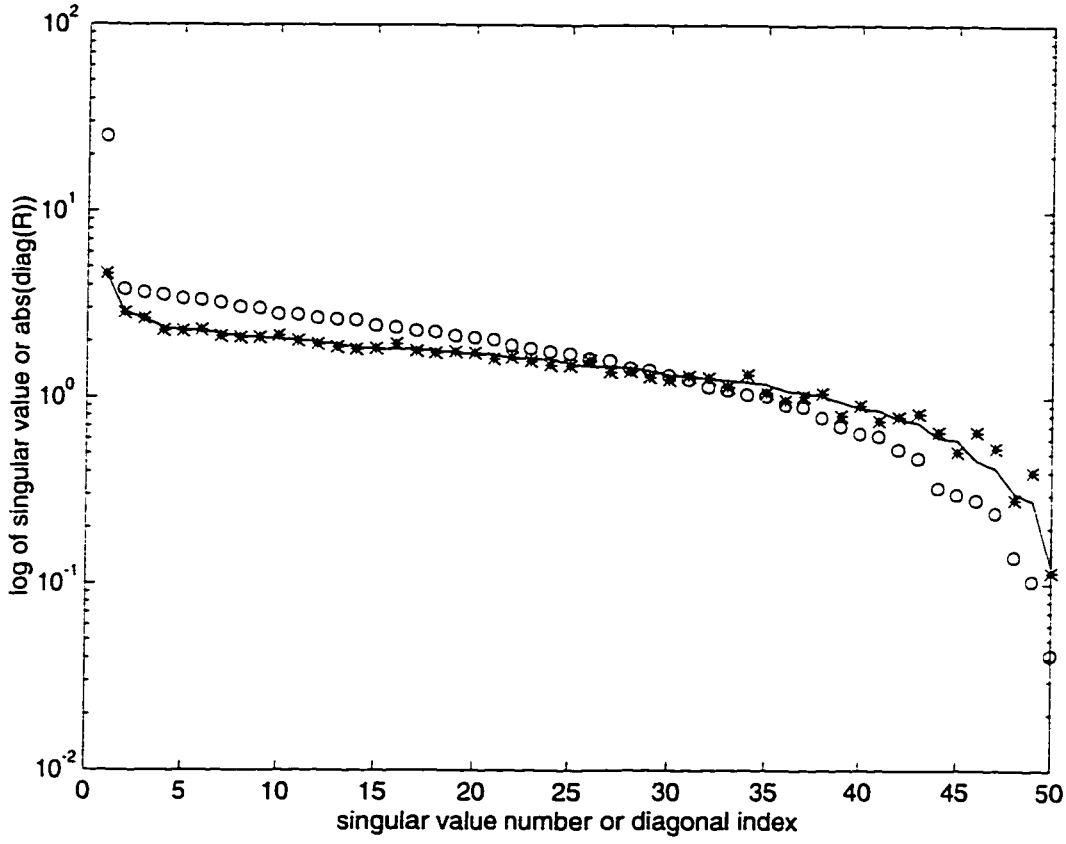

Figure 2: 50 x 50 random matrix

-: QR with column pivoting. * : Foster/Chan RRQR. o : singular value.

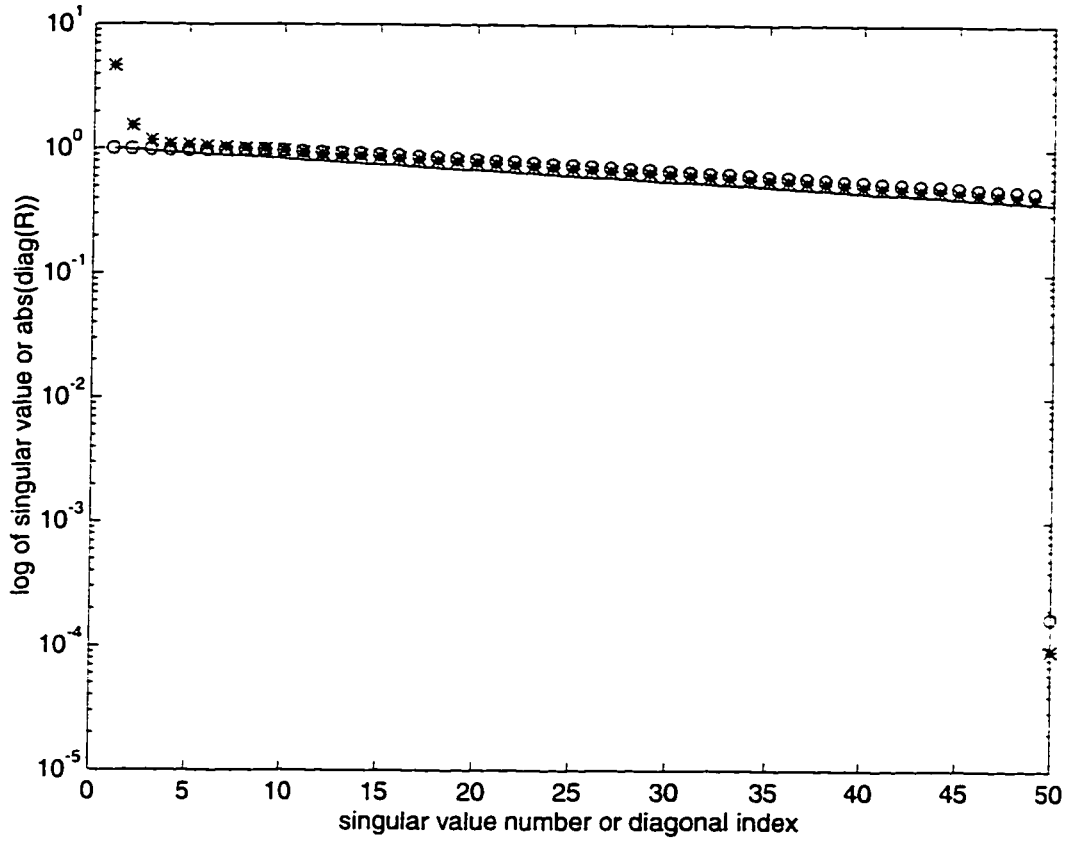

Figure 3: 50 x 50 Kahan's matrix c=0.2

Note that the absolute value of the smallest diagonal entry of R of QR with column pivoting is 0.4, and of Foster/Chan RRQR is 10^{-4}, which is approximately the smallest singular value.

-: QR with column pivoting, * : Foster/Chan RRQR, o : singular value.

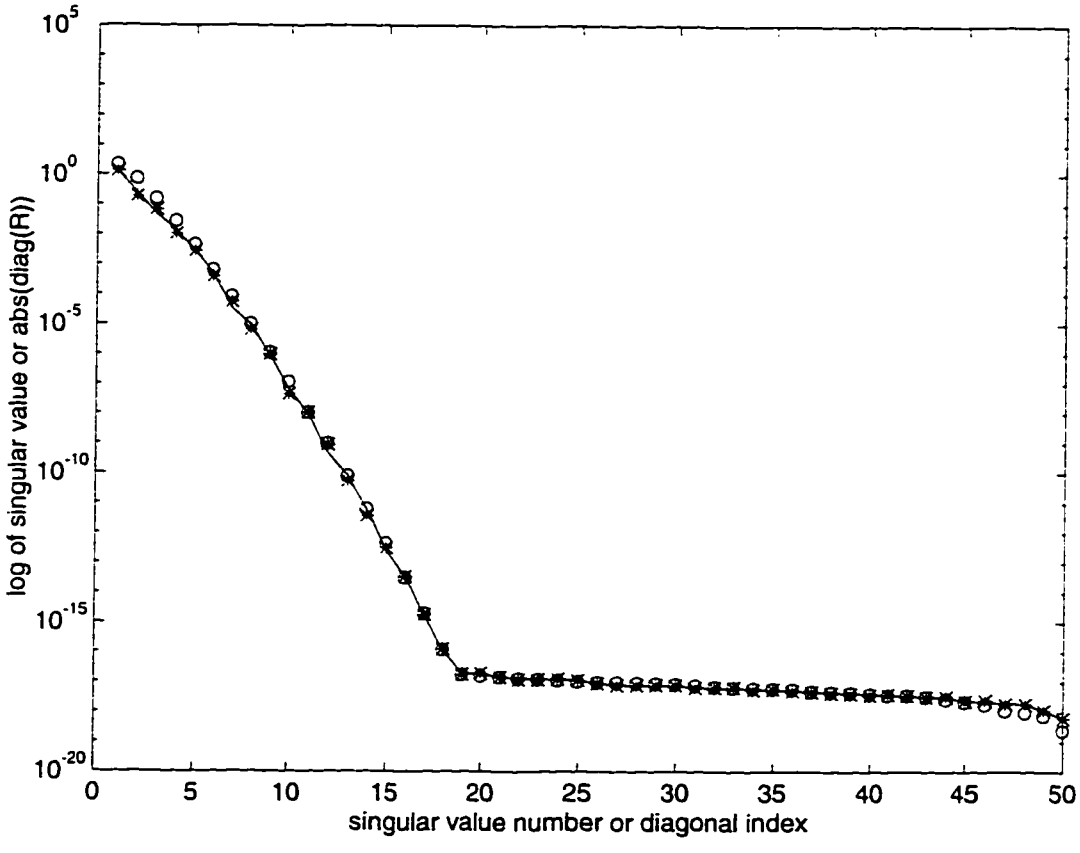

Figure 4: 50 x 50 Hilbert's matrix
For $n \geq 19$, the values are limited to 10^{-16}, the machine precision.

-: QR with column pivoting, * : Foster/Chan RRQR, o : singular value.

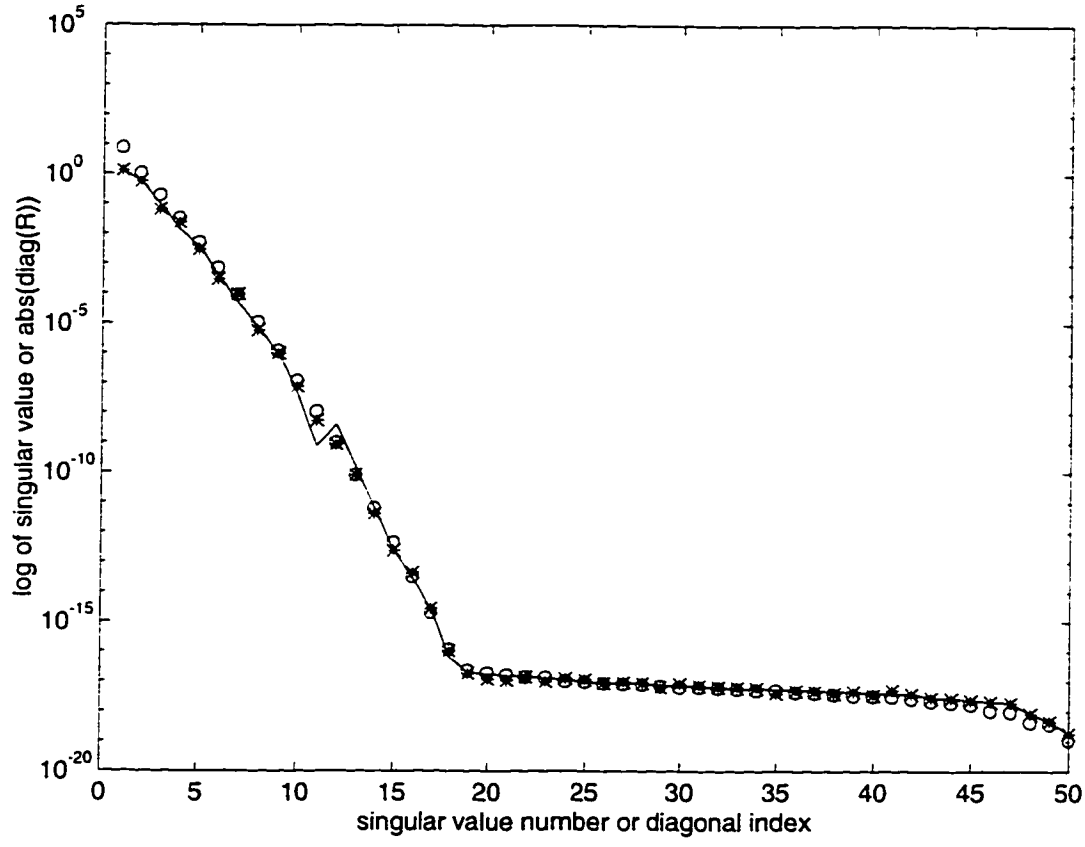

Figure 5: 50 x 50 Lotkin's matrix
For $n \geq 19$, the values are limited to 10^{-16}, the machine precision.

-: QR with column pivoting, * : Foster/Chan RRQR, o : singular value.

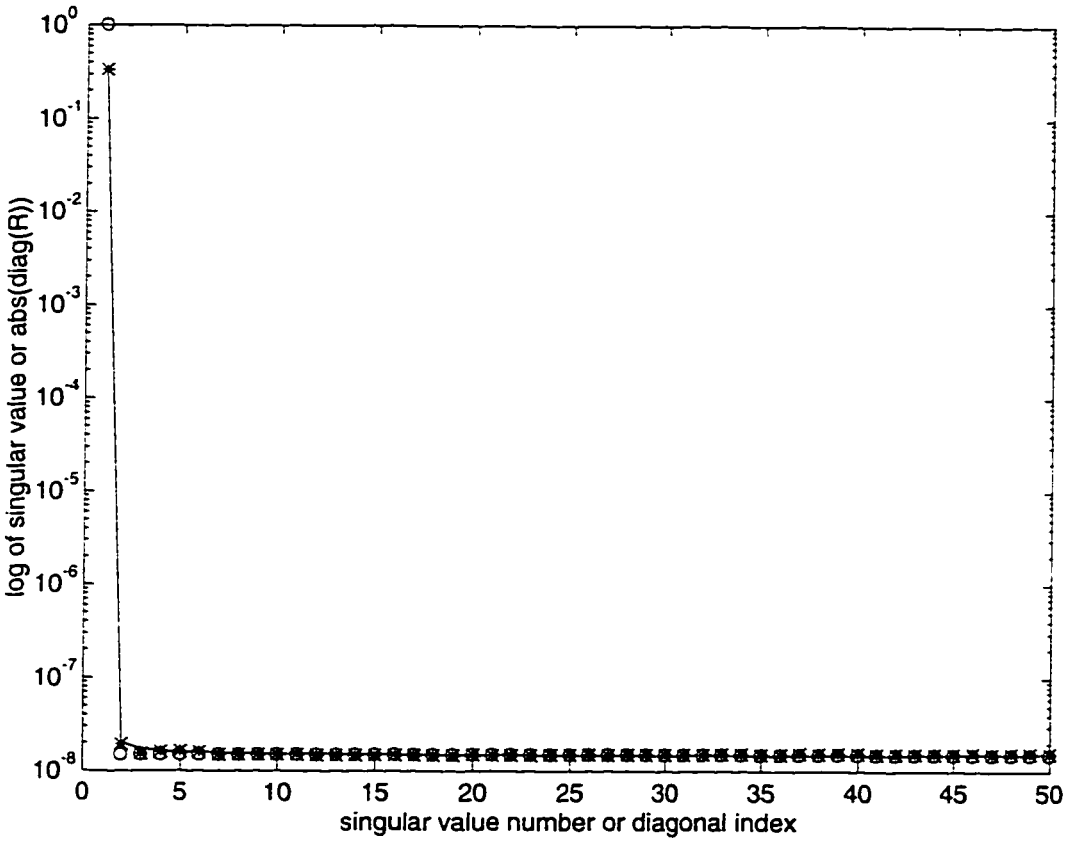

Figure 6: 50 x 50 random svd matrix with one large singular value

-: QR with column pivoting. * : Foster/Chan RRQR. o : singular value.

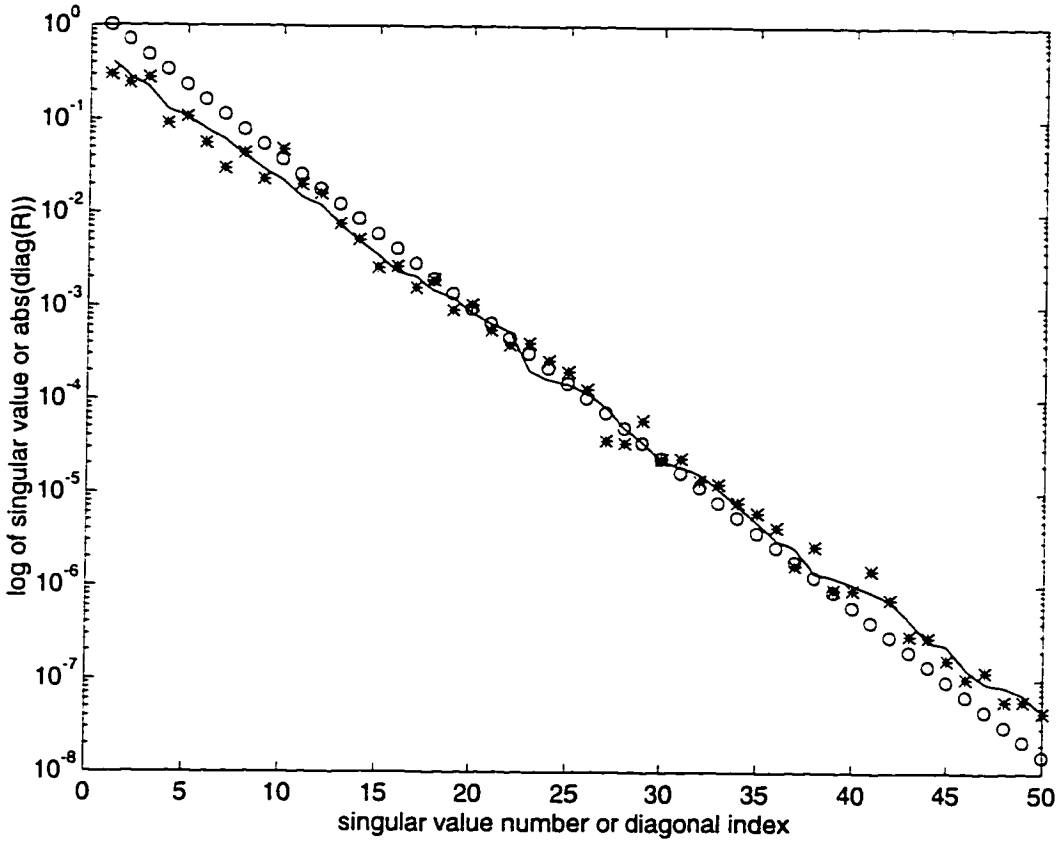

Figure 7: 50 x 50 random svd matrix with geometrically distributed singular values

-: QR with column pivoting, * : Foster/Chan RRQR, o : singular value.

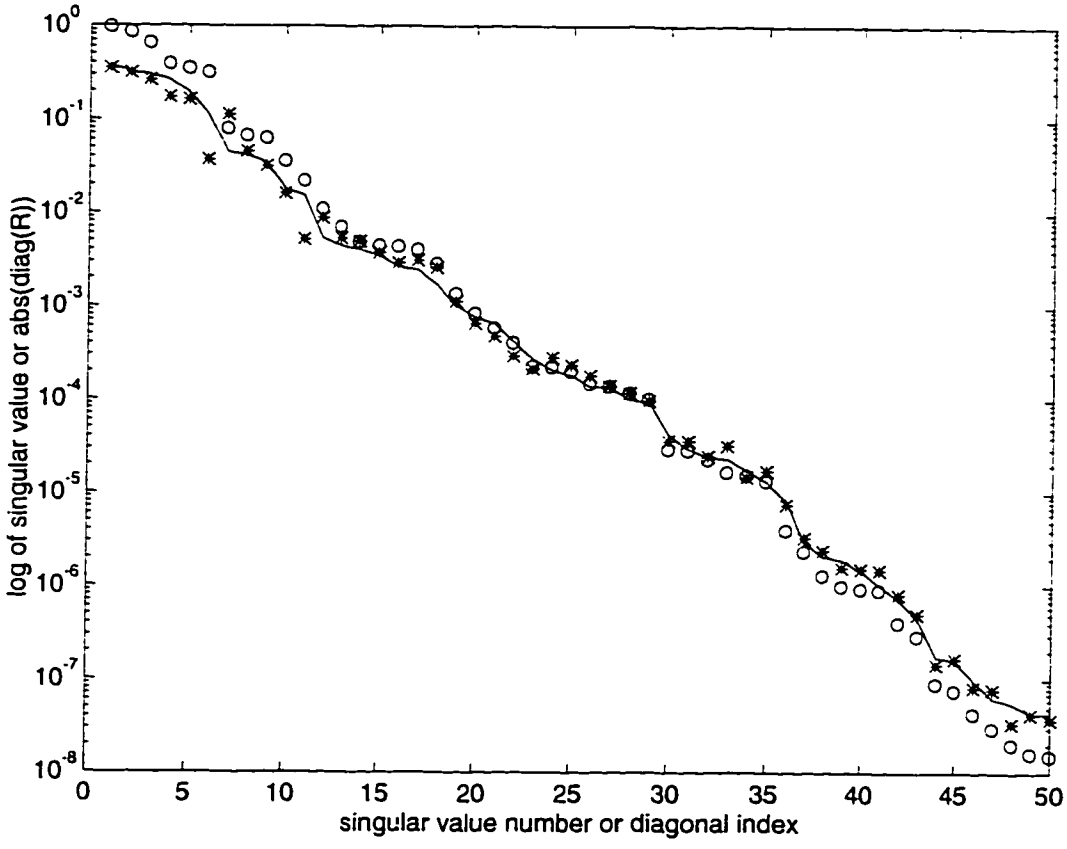

Figure 8: 50 x 50 random svd matrix with uniformly distributed logarithmic singular values

-: QR with column pivoting. * : Foster/Chan RRQR. o : singular value.

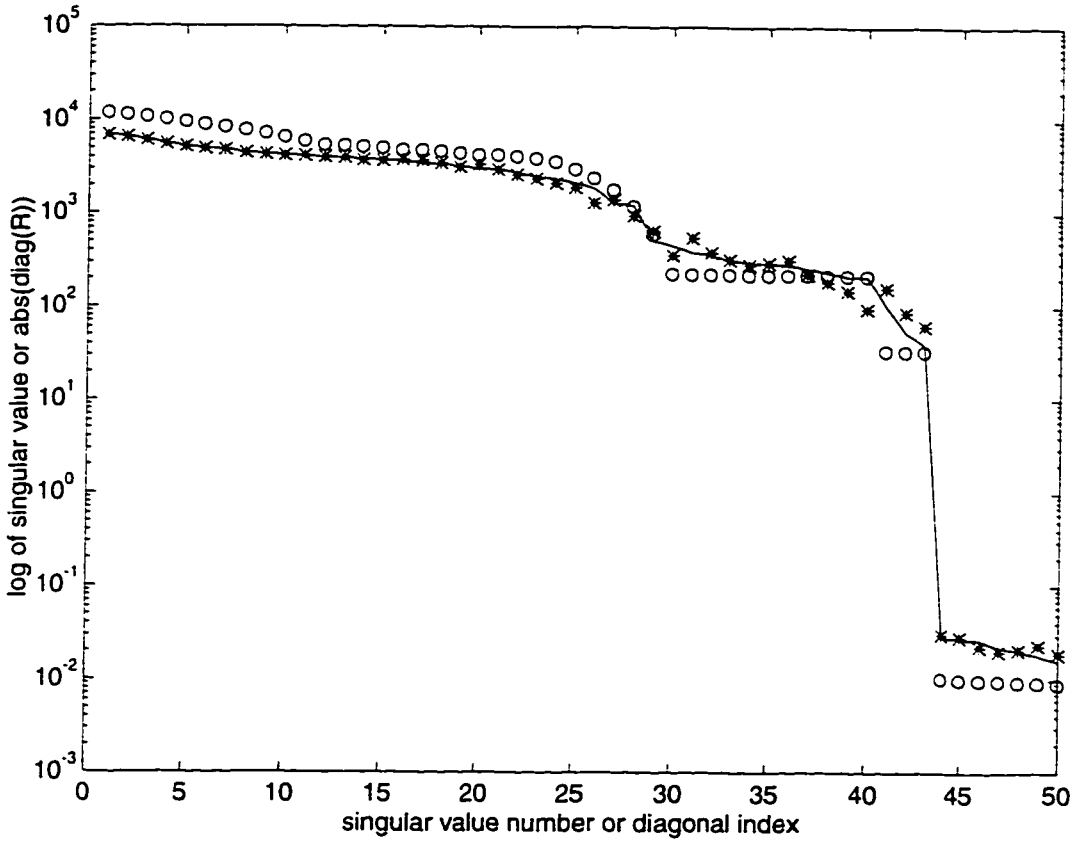

Figure 9: 50 x 50 random svd matrix with a large gap between singular values

From the graphs we can see that RRQR works for Kahan's matrix where QR with column pivoting fails, and works for other matrices as well. Therefore, RRQR is a good alternative to SVD since it is cheaper. In next section we will discuss some applications of RRQR.

4.4 Applications of Foster/Chan RRQR

As we saw in the previous sections, the Foster/Chan RRQR algorithm is a good alternative for finding the numerical rank of a matrix A since it is more efficient than the SVD and works where QR with column pivoting fails. In this section we will show that in addition to determining the numerical rank of A, Foster/Chan RRQR can be used to solve the subset selection problem and the least squares problem.

4.4.1 The Least Squares Problem

In this section we will discuss solving a least squares problem using Foster/Chan RRQR where the matrix A is ill-conditioned. When A is well-conditioned, the standard method would be a good way to proceed. Therefore, we not consider the well-conditioned matrices here. We also compare the solutions given by SVD and Foster/Chan RRQR. Since we compare the two solutions, we use the following notation:

- x_{svd} = solution - using SVD,
- x_{rrqr} = solution - using Foster/Chan RRQR,
- r_{svd} = residual corresponding to x_{svd},
- r_{rrqr} = residual corresponding to x_{rrqr}.

In a previous section, we showed that

$$x_{svd} = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i, \text{ where } R_\epsilon(A) = r.$$

The x_{svd} can always be computed from the SVD of A, but it is too expensive since only a fraction of the information provided by the SVD is used. In [5], it was shown

that x_{svd} can be computed efficiently by the Foster/Chan RRQR algorithm. We want to show that instead of using the Foster/Chan RRQR algorithm to compute x_{svd}, one may define a least squares solution in terms of the Foster/Chan RRQR algorithm.

The following algorithm was presented in [5].

Algorithm 4.3 *Least Squares using Foster/Chan RRQR*

1. *Find the Foster/Chan RRQR of A:*

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

2. *Find a right orthogonal transformation P to annihilate R_{12}, i.e.*

$$(R_{11}R_{12})P = (\hat{R}_{11}0) \text{ where } size(R_{11}) = size(\hat{R}_{11}).$$

3. $x_{qr} = \Pi P \begin{pmatrix} \hat{R}_{11}^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T b.$

Let r be the numerical rank of A and q be the number of inverse iterations used to compute accurate singular subspaces (usually q is less than 4 [5]). It takes $(2m+4n+2q(n-r)+1)n^2$ flops [5,11] to compute x_{svd} and $(2m-\frac{2}{3}n+\frac{11}{2}(n-r)+1)n^2$ flops [5] to compute x_{rrqr}. For the case of $m = n$, it takes about $(6m+8(m-r)+1)m^2$ (when q=4) flops to compute x_{svd} which is more expensive than the cost of $(\frac{4}{3}m+\frac{11}{2}(m-r)+1)m^2$ flops for computing the x_{rrqr}. In the case where $m \gg n$, the computing of x_{svd} is about $(2m+8(n-r)+1)n^2$ flops which is about the same as the computing of x_{rrqr} $((2m+\frac{11}{2}(n-r)+1)n^2)$. In the extreme case of $r = n$, it takes $(2m-\frac{2}{3}n+1)n^2$ to compute x_{rrqr} which is still cheaper than the cost of $(2m+4n+1)n^2$ flops for computing the x_{svd}.

Theorem 4.5 *The SVD and Foster/Chan RRQR solutions are related by*

$$\|x_{svd} - x_{rrqr}\|_2 \leq \|R_{22}\|_2 \|R_{11}^{-1}\|_2 (2\|x_{svd}\|_2 + \frac{\|r_{svd}\|_2}{\sigma_r}).$$

The residuals satisfy

$$\|r_{svd} - r_{rrqr}\|_2 \leq \|R_{22}\|_2(\|x_{svd}\|_2 + \frac{\|r_{svd}\|_2}{\sigma_r}).$$

Proof: Please see [5] $\diamond$.

Let $W = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} \equiv [w_{n-r}, \ldots, w_n]$ where W_2 is an n-r $\times$ n-r upper triangular matrix and w_i for $i = n-r, \ldots, n$ are as calculated in step 8 of Chan's algorithm. From [3] we know $\|R_{22}\|_2 \leq \sigma_{r+1}\sqrt{n-r}\|W_2^{-1}\|_2$ and W_2^{-1} is well-conditioned (in theory W_2^{-1} could be ill-conditioned when there is no significant gap between the singular values, but not so in practice). Since W_2^{-1} is well-conditioned, $\|W_2^{-1}\|_2$ is not too large. Then as long as σ_{r+1} is small, the theorem above guarantees that SVD and Foster/Chan RRQR will produce the solutions with the same magnitudes residual norm and solution norm. Therefore, we can conclude that both methods work equally well in many circumstances for solving the rank deficient least squares problem with well-determined ϵ rank, but using Foster/Chan RRQR is cheaper than using SVD.

4.4.2 The Subset Selection Problem

As we mentioned in a previous section in order to solve the subset selection problem the approach of finding a permutation matrix Π such that the submatrix consisting of the first r columns of $A\Pi$ is as well-conditioned as possible where the $R_\epsilon(A) = r$ is more efficient than examining all $\begin{pmatrix} n \\ r \end{pmatrix}$ possible combinations of the columns of A. The Foster/Chan RRQR algorithm produces such a permutation matrix Π. In this section we compare the subspace spanned by the basis produced by using the Foster/Chan RRQR to solve the subset selection problem and the subspace spanned by the basis produced by using the SVD to solve the same problem that we presented in a previous section.

Theorem 4.6 *Let $S(U_k)$ denote the subspace spanned by $\{u_1, \ldots, u_r\}$, the left singular vectors corresponding to singular values $\sigma_1, \ldots, \sigma_r$, and let B_{svd} and B_{rrqr} denote the submatrices consisting the first r columns of $A\Pi_{svd}$ and $A\Pi_{rrqr}$ respectively. Then*

$$\sin\theta(S(U_r), S(B_{svd})) \leq \sigma_{r+1}\|\hat{V}_{11}^{-1}\|_2\sigma_r^{-1} \tag{11}$$

$$\sin\theta(S(U_r), S(B_{rrqr})) \leq \sigma_{r+1}\|R_{11}^{-1}\|_2 \tag{12}$$

where $A = U\Sigma \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}^T$ is the SVD of A, and $\begin{pmatrix} \hat{V}_{11} & \hat{V}_{12} \\ \hat{V}_{21} & \hat{V}_{22} \end{pmatrix} = \Pi_{svd} \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}$, and $A\Pi_{rrqr} = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$.

Proof: Please see [5] $\diamond$.

From [3] we know Foster/Chan RRQR tends to produce a well-conditioned $\hat{V}_{11}$, so $\|\hat{V}_{11}^{-1}\|_2$ is not too large. Since Foster/Chan RRQR guarantees a well-conditioned R_{11} and $\|R_{11}^{-1}\|_2$ is of the order σ_r^{-1} [5] and $\|\hat{V}_{11}^{-1}\|_2$ is not too large, the theorem above ensures that the sine of both subspace angles is of the same order as σ_{r+1}/σ_r [5]. If σ_{r+1}/σ_r is small, then from the theorem above we can say both $S(B_{svd})$ and $S(B_{rrqr})$ will be close to the subspace $S(U_r)$, and the angle between $S(B_{svd})$ and $S(B_{rrqr})$ will be small. Therefore, we can conclude that even though the SVD and Foster/Chan RRQR subset selection algorithms may not produce the same set of columns, the subspaces spanned by these two sets of columns will be almost the same as long as σ_{r+1}/σ_r is small.

4.4.3 Numerical Examples

The Least Squares Problem

In a previous section we showed that the Foster/Chan RRQR can be used to compute the least squares problem, and as long as σ_{r+1} is small, SVD and Foster/Chan RRQR will produce the same solutions. In this numerical example we want to show

that the theory is accurate. Let $\epsilon = 10^{-10}$. We generated a rank deficient matrix A with numerical rank 5 and $\sigma_6 \approx 10^{-12}$. We first computed the Foster/Chan RRQR and SVD of A, and then displayed the singular values and the absolute value of the diagonals of R:

Singular Values	*Abs(diag(R))*
4.4092	2.1622
1.5086	1.1862
1.0178	9.0364×10^{-1}
7.8377×10^{-1}	7.9071×10^{-1}
7.0184×10^{-1}	6.0483×10^{-1}
1.0000×10^{-12}	1.5441×10^{-12}
1.0000×10^{-14}	2.1799×10^{-14}

Next, we computed the solution by using both SVD and Foster/Chan RRQR and computed their difference. We used

$$b = \begin{pmatrix} 4.6445 \times 10^{-1} \\ 9.4098 \times 10^{-1} \\ 5.0084 \times 10^{-2} \\ 7.6151 \times 10^{-1} \\ 7.7020 \times 10^{-1} \\ 8.2782 \times 10^{-1} \\ 1.2537 \times 10^{-1} \\ 1.5868 \times 10^{-2} \\ 6.8864 \times 10^{-2} \\ 8.6825 \times 10^{-1} \end{pmatrix}.$$

Here is the display of our results:

x_{svd}	x_{rrqr}	$x_{svd} - x_{rrqr}$
-1.6106×10^{-1}	-1.6106×10^{-1}	-6.6613×10^{-16}
3.9657×10^{-1}	3.9657×10^{-1}	3.1641×10^{-15}
4.3621×10^{-1}	4.3621×10^{-1}	3.1641×10^{-15}
-6.7157×10^{-2}	-6.7157×10^{-2}	5.8009×10^{-15}
3.8446×10^{-2}	3.8446×10^{-2}	-1.3892×10^{-14}
1.2140×10^{-1}	1.2140×10^{-1}	8.1879×10^{-15}
2.6964×10^{-1}	2.6964×10^{-1}	-1.2604×10^{-14}

As we can see, since σ_6 was small (10^{-12}) the solutions computed by SVD and Foster/Chan RRQR are approximately the same. We can say that in addition to finding

the numerical rank of A. the Foster/Chan RRQR is also a pretty good alternative to SVD for the least squares problem.

The Subset Selection

In the previous section we showed that even though the SVD and Foster/Chan RRQR subset selection algorithms may not produce the same set of columns. the subspaces spanned by these two sets of columns will be almost the same as long as σ_{r+1}/σ_r is small. In this numerical example we want to show that the theory is accurate. The matrix A used in this example was collected by a consulting firm to determine the customer satisfaction for a company which makes computer file-servers. Because of the issue of confidentiality involved in the study. we were not allow to discuses the study in detail.

The survey consisted of nine sets of problems. The sets range from service. to warranty. to cost. Each person included in the survey must be familiar with the manufacturer and was asked whether or not the manufacturer is the most familiar brand. They were divided into two groups accordingly. The size of the matrix was 63×63. It just happened that the number of people who are familiar with this manufacturer the same as the number of questions asked.

Let $\epsilon = 10^{-5}$. We first applied the Foster/Chan RRQR and found that the numerical rank of A is 53. Second. we found the set consisting of the 53 linearly independent columns of A relative to ϵ. Third, we found the mean for each person for each set of questions. Fourth, we ran correlation on the means computed in step 3. Fifth, we found the mean for each subject for each set of questions using all problem. Finally, we ran correlation on the means computed in step 4. These correlations should help us to see if there are any interactions between the problem sets. Below are the results of the correlations:

problem set	1	2	2	4	5
1	1	0.7160032	0.6150475	0.48327015	0.54976939
2	0.7160032	1	0.71499467	0.45387582	0.63878132
3	0.6150475	0.71499467	1	0.34518908	0.63272482
4	0.48327015	0.45387582	0.34518908	1	0.32479652
5	0.54976939	0.63878132	0.63272482	0.32479652	1
6	0.61997587	0.60592528	0.5235081	0.57001882	0.58173926
7	0.42843306	0.61169924	0.43606047	0.17085136	0.61047936
8	0.45911352	0.42890526	0.40485433	0.40732047	0.47192961
9	0.52013991	0.55428718	0.40097988	0.69079514	0.54811736

Problem Set	6	7	8	9
1	0.61997587	0.42843306	0.45911352	0.52013991
2	0.60592528	0.61169924	0.42890526	0.55428718
3	0.5235081	0.43606047	0.40485433	0.40097988
4	0.57001882	0.17085136	0.40732047	0.69079514
5	0.58173926	0.61047936	0.47192961	0.54811736
6	1	0.32805572	0.32727623	0.639572
7	0.32805572	1	0.37113679	0.41469389
8	0.32727623	0.37113679	1	0.48714539
9	0.639572	0.41469389	0.48714539	1

Table 1: Correlation of the linearly independent columns relative to ϵ

Since the data was first cleaned and analyzed by a professional consulting firm, we should not expect too much 'noise'. As we can see, the results of the two sets of correlations above have no significant difference. If this set of data had not been first cleaned and analyzed, then we would expect bigger differences between them.

We showed in a previous section that even though the SVD and Foster/Chan RRQR subset selection algorithms may not produce the same set of columns, the subspaces spanned by these two sets of columns will be almost the same as long as σ_{r+1}/σ_r is small, where r is the numerical rank of A. This is the case here. Since it takes too much space to list the two sets of columns produced by SVD and Foster/Chan RRQR, we will solve for the angle between the two subspaces spanned by

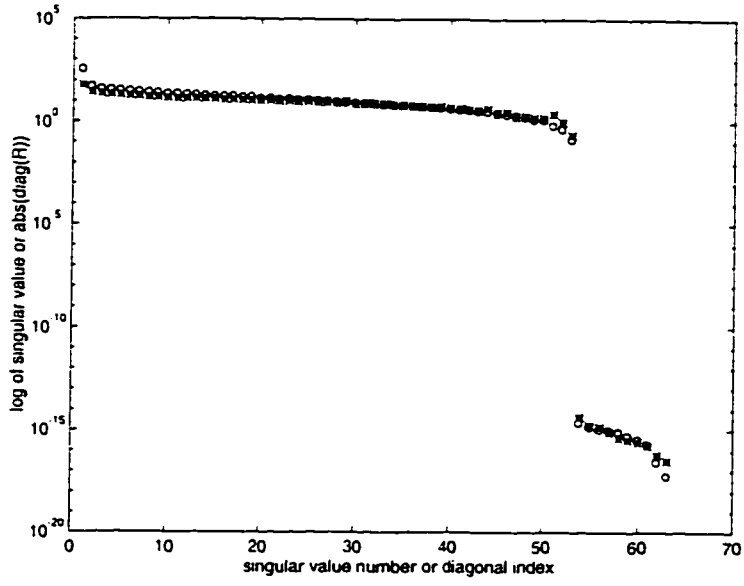

Figure 10: Data of the customer satisfaction for a company which makes computer file-servers

$*$: Foster/Chan RRQR, o : singular value.

these two sets of columns. We found that $\theta = 6.3220 \times 10^{-8}$. Since the angle between the two subspaces is small, we can say the two subspaces are approximately the same.

problem set	1	2	2	4	5
1	1	0.71638414	0.60798981	0.4907319	0.54448708
2	0.71638414	1	0.71927995	0.47020048	0.6400209
3	0.60798981	0.71927995	1	0.37647685	0.63733332
4	0.4907319	0.47020048	0.37647685	1	0.35604182
5	54448708	0.6400209	0.63733332	0.35604182	1
6	0.61433934	0.60824254	0.52703724	0.61798613	0.5800749
7	0.43482289	0.6162206	0.44465903	0.1896265	0.60998379
8	0.45935835	0.42985359	0.40861494	0.40579574	0.47376792
9	0.5167409	0.55810963	0.40444098	0.7236291	0.54497613

Problem Set	6	7	8	9
1	0.61433934	0.43482289	0.45935835	0.5167409
2	0.60824254	0.6162206	0.42985359	0.55810963
3	0.52703724	0.44465903	0.40861494	0.40444098
4	0.61798613	0.1896265	0.40579574	0.7263291
5	0.5800749	0.60998379	0.47376792	0.54497613
6	1	0.33064481	0.33449965	0.63914147
7	0.33064481	1	0.36794556	0.41171898
8	0.33449965	0.36794556	1	0.49405885
9	0.63914147	0.41171898	0.49405885	1

Table 2: Correlation of all the data

5 Strong RRQR

In 1992, Hong and Pan introduced a rank revealing QR factorization such that

$$\sigma_{\min}(R_{11})\sqrt{r(n-r)+\min(r,n-r)} \geq \sigma_r(A),$$

$$\sigma_{\max}(R_{22}) \leq \sigma_{r+1}(A)\sqrt{r(n-r)+\min(r,n-r)}$$

where $A\Pi = Q\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$ is the rank revealing QR factorization of A calculated by Hong and Pan's algorithm and r is the numerical rank of A [16]. As we can see, the bound computed by Hong and Pan's algorithm is polynomial, whereas the bound computed by the Foster/Chan RRQR is exponential. However the computer time required by Hong and Pan's algorithm could in principle grow exponentially in n [16]. In 1994, Chandrasekaran and Ipsen presented three rank revealing QR factorizations which all have approximately the same bound as Hong and Pang's algorithm and appear to be efficient in practice [2]; however Chandrasekaran and Ipsen could not prove that the algorithms require a polynomial number of operations.

In this section we will discuss an algorithm called strong RRQR. The algorithm was proposed by Gu and Eisenstat in 1996. This rank revealing QR algorithm not only reveals the numerical rank of an m × n matrix A where $m \geq n$, it also satisfies the following three conditions:

1. $\sigma_i(R_{11}) \geq \sigma_i(A)/q_1(r,n)$,
2. $\sigma_j(R_{22}) \leq \sigma_{j+1}(A)q_1(r,n)$,
3. $|\,(R_{11}^{-1}R_{12})_{i,j}\,| \leq q_2(r,n)$,

for $1 \leq i \leq r$ and $1 \leq j \leq n-r$ and some modest value f, where the numerical rank of A is r, $q_1(r,n) = \sqrt{1+f^2r(n-r)}$, $q_2(r,n) = f$, and $A\Pi = Q\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$ is the rank revealing QR factorization of A. For some rank revealing QR factorizations the elements of $R_{11}^{-1}R_{12}$ can be very large. In many of the applications, it is necessary

to find a basis for the approximate right null space of A. Since $\Pi \begin{pmatrix} -R_{11}^{-1}R_{12} \\ I_{n-r} \end{pmatrix}$ is an approximate right null space, if the elements of $R_{11}^{-1}R_{12}$ are very large, it can lead to an unstable algorithm [14].

A strong RRQR satisfies the following: every singular value of R_{11} is sufficiently large; every singular value of R_{22} is sufficiently small; and every element of $R_{11}^{-1}R_{22}$ is bounded. Also Gu and Eisenstat proved that a strong RRQR algorithm was at most 50% more expensive than QR with column pivoting when $m \approx n$.

We first present two algorithms that compute a strong RRQR factorization, assuming that the numerical rank r of A is known and $f \geq 1$. Then we present a third algorithm that computes both r and a strong RRQR factorization. Here is some notation that we will be using:

- $R_{11}, \hat{R}_{11} \in \Re^{r \times r}$ denote upper-triangular matrices with nonnegative diagonal elements.
- $R_{12}, \hat{R}_{12} \in \Re^{r \times (n-r)}$ denote general matrices.
- $R_{22}, \hat{R}_{22} \in \Re^{(m-r) \times (m-r)}$ denote upper-triangular matrices.
- In a QR factorization $X = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$, we denote $R_r(X) = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$ and $C_r(X) = R_{22}$, where $X \in \Re^{m \times n}$ and the numerical rank of X is r.
- Let B be a nonsingular $\ell \times \ell$ matrix, $1/\omega_i(B)$ denote the 2-norm of the i-th row of B^{-1} and $\omega_*(B) = (\omega_1(B), \ldots, \omega_\ell(B))^T$.
- Let C be a matrix with ℓ columns, $\gamma_j(C)$ denote the 2-norm of the j-th column of C and $\gamma_*(C) = (\gamma_1(C), \ldots, \gamma_\ell(C))$.
- Let $\Pi_{i,j}$ denote the permutation matrix that interchanges the i-th column and j-th column of a matrix.

The following three algorithms are presented in [14].

Algorithm 5.1 *Compute a strong RRQR factorization , given r.*

Let $R := R_r(A)$ *and* $\Pi := I$.

while *there exist i and j such that* $det(\hat{R}_{11})/det(R_{11}) \geq f$.

where $R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$ *and* $R_r(R\,\Pi_{i,j+r}) = \begin{pmatrix} \hat{R}_{11} & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{pmatrix}$, **do**

Find such an i and j:

Compute $R := R_r(R\,\Pi_{i,j+r})$ *and* $\Pi := \Pi\,\Pi_{i,j+r}$.

endwhile

Gu and Eisenstat argue that since the algorithm above interchanges any pair of columns that sufficiently increases $det(R_{11})$, there are only a finite number of permutations and none can be repeated. Therefore the algorithm above eventually halts. Since computation of determinants can be inefficient, the algorithm above may not be very efficient.

The following theorem is true for every pair of R and $R_k(R\Pi_{i,j+r})$ with corresponding i, j computed in the while-loop in the algorithm above.

Theorem 5.1 *Let*

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \text{ and } R_r(R\Pi_{i,j+r}) = \begin{pmatrix} \hat{R}_{11} & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{pmatrix}.$$

where R_{11} *has positive diagonal elements. Then*

$$\frac{det(\hat{R}_{11})}{det(R_{11})} = \sqrt{(R_{11}^{-1}R_{12})_{i,j}^2 + (\gamma_j(R_{22}) \,/\, \omega_i(R_{11}))^2}.$$

Proof (i) First we want to consider the special case $i = r$ and $j = 1$.

Let $R_{r-1}(R) = \begin{pmatrix} R_{11}^{(r-1)} & R_{12}^{(r-1)} \\ 0 & R_{22}^{(r-1)} \end{pmatrix}$ and $R_{r+1}(R) = \begin{pmatrix} R_{11}^{(r+1)} & R_{12}^{(r+1)} \\ 0 & R_{22}^{(r+1)} \end{pmatrix}$. Now partition

$$R_{r+1}(R) = \begin{pmatrix} R_{11}^{(r-1)} & b1 & b2 & B \\ & \gamma_1 & \beta & c_1^T \\ & & \gamma_2 & c_2^T \\ & & & R_{22}^{(r+1)} \end{pmatrix}.$$

Since $R_r = R_{r+1}$, we know that $det(R_{11}) = det(R_{11}^{(r-1)})(\gamma_1)$, $\omega_i(R_{11}) = \gamma_1$, $\gamma_j(R_{22}) = \gamma_2$, and $(R_{11}^{-1}R_{12})_{i,j} = \beta/\gamma_1$.

$$R_{r+1}(R)\Pi_{i,r+j} = \begin{pmatrix} R_{11}^{(r-1)} & b1 & b2 & B \\ & \beta & \gamma_1 & c_1^T \\ & & \gamma_2 & 0 & c_2^T \\ & & & & R_{22}^{(r+1)} \end{pmatrix}.$$

Use any QR factorization, we have $\hat{R}_{11} = \begin{pmatrix} R_{11}^{(r-1)} & * \\ 0 & \sqrt{\beta^2+\gamma_2^2} \end{pmatrix}$. Then $det(\hat{R}_{11}) = det(R_{11}^{r-1})\sqrt{\beta^2+\gamma_2^2}$. Therefore

$$\frac{det(\hat{R}_{11})}{det(R_{11})} = \sqrt{(\beta/\gamma_2)^2 + (\gamma_2/\gamma_1)^2} = \sqrt{(R_{11}^{-1}R_{12})_{i,j}^2 + (\gamma_j(R_{22}) \,/\, \omega_i(R_{11}))^2}.$$

(ii) Now suppose that $i < r$ or that $j > r$. Let $R_{11}\Pi_{i,r} = \bar{Q}\bar{R}_{11}$ be the QR factorization of $R_{11}\Pi_{i,r}$. Let $\bar{R}_{12} = \bar{Q}^T R_{12}\Pi_{1,j}$, $\bar{R}_{22} = R_{22}\Pi_{1,j}$, and $\bar{\Pi} = diag(\Pi_{i,r}\Pi_{1,j})$. Then

$$\begin{aligned} R\bar{\Pi} = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}\bar{\Pi} &= \begin{pmatrix} R_{11}\Pi_{i,r} & R_{12}\Pi_{1,j} \\ 0 & R_{22}\Pi_{1,j} \end{pmatrix} \\ &= \begin{pmatrix} \bar{Q}\bar{R}_{11} & \bar{Q}\bar{R}_{12} \\ 0 & \bar{R}_{22} \end{pmatrix} \\ &= \begin{pmatrix} \bar{Q} & \\ & I_{m-r} \end{pmatrix}\begin{pmatrix} \bar{R}_{11} & \bar{R}_{12} \\ 0 & \bar{R}_{22} \end{pmatrix} \end{aligned}$$

is a QR factorization of $R\bar{\Pi}$. Since $R_{11}\Pi_{i,r} = \bar{Q}\bar{R}_{11}$, $det(R_{11}\Pi_{i,r}) = det(\bar{Q}\bar{R}_{11})$. Since $\bar{Q}$ and $\Pi_{i,r}$ are orthogonal matrices, $det(\bar{Q}) = \pm 1$ and $det(\Pi_{i,r}) = \pm 1$. Since $\bar{R}_{11}$, R_{11} both have positive diagonal entries and $\bar{R}_{11}$, R_{11} are upper triangular matrices, $det(R_{11}) > 0$ and $det(\bar{R}_{11}) > 0$. Since

1. $det(R_{11})det(\Pi_{i,r}) = det(R_{11}\Pi_{i,r}) = det(\bar{Q}\bar{R}_{11}) = det(\bar{Q})det(\bar{R}_{11})$ and
2. $det(\bar{Q}) = \pm 1$ and $det(\Pi_{i,r}) = \pm 1$ and
3. $det(R_{11}) > 0$ and $det(\bar{R}_{11}) > 0$,

therefore we can say that $det(R_{11}) = det(\bar{R}_{11})$. Since

1. $\bar{R}_{11}^{-1} = (\bar{Q}^{-1}R_{11}\Pi_{i,r})^{-1} = \Pi_{i,r}^T R_{11}^{-1}\bar{Q}$ and
2. $\bar{R}_{11}^{-1}\bar{R}_{12} = \Pi_{i,r}^T R_{11}^{-1}\bar{Q}\bar{Q}^T R_{12}\Pi_{1,j} = \Pi_{i,r}^T R_{11}^{-1}R_{12}\Pi_{1,j}$,

we have $(R_{11}^{-1}R_{12})_{i,j} = (\bar{R}_{11}^{-1}\bar{R}_{12})_{k,1}$. Since $\bar{R}_{11}^{-1} = \Pi_{i,r}^T R_{11}^{-1}\bar{Q}$ and postmultiplication by an orthogonal matrix leaves the 2-norms of the rows unchanged, we have $\omega_i(R_{11}) = \omega_r(\bar{R}_{11})$. Finally, we have $\gamma_j(R_{22}) = \gamma_1(\bar{R}_{22})$.

$$\begin{aligned}
\begin{pmatrix} \hat{R}_{11} & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{pmatrix} = R_r(R\Pi_{i,r+j}) &= R_r[\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \Pi_{i,k+j}] \\
&= R_r[\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \begin{pmatrix} \Pi_{i,r} & \\ & \Pi_{1,j} \end{pmatrix} \Pi_{r,r+1}] \\
&= R_r[\begin{pmatrix} \bar{R}_{11} & \bar{R}_{12} \\ 0 & \bar{R}_{22} \end{pmatrix} \Pi_{r,r+1}].
\end{aligned}$$

From (i) we have

$$\frac{det(\hat{R}_{11})}{det(\bar{R}_{11})} = \sqrt{(\bar{R}_{11}^{-1}\bar{R}_{12})_{r,1}^2 + (\gamma_1(\bar{R}_{22})/\omega_r(\bar{R}_{11}))^2}.$$

Therefore

$$\begin{aligned}
\frac{det(\hat{R}_{11})}{det(R_{11})} &= \frac{det(\hat{R}_{11})}{det(\bar{R}_{11})} \\
&= \sqrt{(\bar{R}_{11}^{-1}\bar{R}_{12})_{r,1}^2 + (\gamma_1(\bar{R}_{22})/\omega_r(\bar{R}_{11}))^2} \\
&= \sqrt{(R_{11}^{-1}R_{12})_{i,j}^2 + (\gamma_j(R_{22})/\omega_i(R_{11}))^2} \diamond .
\end{aligned}$$

Let

$$\rho(R, r) = \max_{1 \le i \le r, 1 \le j \le n-r} \sqrt{(R_{11}^{-1}R_{12})_{i,j}^2 + (\gamma_j(R_{22})/\omega_i(R_{11}))^2}.$$

Then we can rewrite the algorithm above as the following:

Algorithm 5.2 *Compute a strong RRQR factorization . given r.*

Compute $R \equiv \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} := R_r(A)$ *and* $\Pi = I$.

while $\rho(R, r) > f$ **do**

Find i and j such that $\sqrt{(R_{11}^{-1}R_{12})_{i,j}^2 + (\gamma_j(R_{22})/\omega_i(R_{11}))^2} > f$;

Compute $\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} := R_r(R\,\Pi_{i,j+r})$ *and* $\Pi := \Pi\,\Pi_{i,j+r}$;

endwhile

Since the second algorithm is equivalent to the first algorithm, the second algorithm eventually halts and will produce a permutation matrix Π which satisfies $\rho(R_r(A\Pi), r) \leq f$. This satisfies the third condition for being a strong RRQR with $q_2(r, n) = f$. Since in the second algorithm we no longer compute determinants in the while-loop and there are r^2 possible pairs of i's and j's to choose from, the work required for each step of the while loop is bounded by a polynomial function of the matrix. We discuss the efficiency in more detail later in the section, but first we show that the algorithm satisfies the conditions of a strong RRQR algorithm.

Theorem 5.2 *Let*

$$R \equiv \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = R_r(A\Pi)$$

satisfy $\rho(R, r) \leq f$. *Then*

$$\sigma_i(R_{11}) \geq \frac{\sigma_i(A)}{\sqrt{1 + f^2 r(n-r)}}, \text{ for } 1 \leq i \leq r,$$

and

$$\sigma_j(R_{22}) \leq \sigma_{j+r}(A)\sqrt{1 + f^2 r(n-r)}, \text{ for } 1 \leq j \leq n-r.$$

Proof For simplicity we assume that A (and therefore R) has full column rank. Let $\alpha = \frac{\sigma_{max}(R_{22})}{\sigma_{min}(R_{11})}$.

(i) We want to show that

$$\sigma_i(R_{11}) \geq \frac{\sigma_i(A)}{\sqrt{1 + f^2 r(n-r)}}, \text{ for } 1 \leq i \leq r.$$

Let

$$R = \begin{pmatrix} R_{11} & \\ & R_{22}/\alpha \end{pmatrix} \begin{pmatrix} I_r & R_{11}^{-1} R_{12} \\ & \alpha I_{n-r} \end{pmatrix} \equiv \tilde{R}_1 W_1.$$

Then by [17, Thm. 3.3.16],

$$\sigma_i(R) \leq \sigma_i(\tilde{R}_1)\|W_1\|_2, \text{ for } 1 \leq i \leq r.$$

Since $\sigma_{\min}(R_{11}) = \frac{\sigma_{\max}(R_{22})}{\alpha} = \sigma_{\max}(\frac{R_{22}}{\alpha})$, we have $\sigma_i(\tilde{R}_1) = \sigma_i(R_{11})$ for $1 \leq i \leq r$. Moreover

$$\begin{aligned}
\|W_1\|_2^2 &= \| \begin{pmatrix} I_r & R_{11}^{-1}R_{12} \\ & \alpha I_{n-r} \end{pmatrix} \|_2^2 \\
&= \| \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & R_{11}^{-1}R_{12} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & \alpha I_{n-r} \end{pmatrix} \|_2^2 \\
&\leq \| \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix} \|_2^2 + \| \begin{pmatrix} 0 & R_{11}^{-1}R_{12} \\ 0 & 0 \end{pmatrix} \|_2^2 + \| \begin{pmatrix} 0 & 0 \\ 0 & \alpha I_{n-r} \end{pmatrix} \|_2^2 \\
&= 1 + \|R_{11}^{-1}R_{12}\|_2^2 + \alpha^2 \\
&= 1 + \|R_{11}^{-1}R_{12}\|_2^2 + (\sigma_{\max}(R_{22})/\sigma_{\min}(R_{11}))^2 \\
&= 1 + \|R_{11}^{-1}R_{12}\|_2^2 + \|R_{22}\|_2^2 \|R_{11}^{-1}\|_2^2 \\
&\leq 1 + \|R_{11}^{-1}R_{12}\|_F^2 + \|R_{22}\|_F^2 \|R_{11}^{-1}\|_F^2 \\
&= 1 + \sum_{i=1}^{r}\sum_{j=1}^{n-r}(R_{11}^{-1}R_{12})_{i,j}^2 + \sum_{i=1}^{n-r}\sum_{j=1}^{n-r}(R_{22})_{i,j}^2 + \sum_{i=1}^{r}\sum_{j=1}^{r}(R_{11}^{-1})_{i,j}^2 \\
&= 1 + \sum_{i=1}^{r}\sum_{j=1}^{n-r}(R_{11}^{-1}R_{12})_{i,j}^2 + \sum_{i=1}^{r}\sum_{j=1}^{n-r}(\gamma_j(R_{22})^2/\omega_i(R_{11})^2) \\
&= 1 + \sum_{i=1}^{r}\sum_{j=1}^{n-r}\{(R_{11}^{-1}R_{12})_{i,j}^2 + (\gamma_j(R_{22})^2/\omega_i(R_{11})^2)\} \\
&\leq 1 + f^2 r(n-r).
\end{aligned}$$

So that $\|W_1\|_2 \leq \sqrt{1 + f^2 r(n-r)}$. Since

$$\sigma_i(R) \leq \sigma_i(\tilde{R}_1)\|W_1\|_2 \leq \sigma_i(\tilde{R}_1)\sqrt{1 + f^2 r(n-r)},$$

we have

$$\sigma_i(\tilde{R}_1) \geq \frac{\sigma_i(R)}{\sqrt{1 + f^2 r(n-r)}}.$$

Since $\sigma_i(\tilde{R}_1) = \sigma_i(R_{11})$ for $1 \leq i \leq r$ and R and A have the same singular values, we have

$$\sigma_i(R_{11}) \geq \frac{\sigma_i(A)}{\sqrt{1 + f^2 r(n-r)}} \text{ for } 1 \leq i \leq r.$$

(ii) We want to show that

$$\sigma_j(R_{22}) \leq \sigma_{j+r}(A)\sqrt{1+f^2r(n-r)},\ for\ 1 \leq j \leq n-r.$$

Let

$$\tilde{R}_2 \equiv \begin{pmatrix} \alpha R_{11} & \\ & R_{22} \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix} \begin{pmatrix} \alpha I_r & -R_{11}^{-1}R_{12} \\ & I_{n-r} \end{pmatrix} \equiv RW_2.$$

Then by [17. Thm. 3.3.16],

$$\sigma_i(\tilde{R}_2) \leq \sigma_i(R)\|W_2\|_2,\ for\ 1 \leq i \leq n.$$

Since $\sigma_{\min}(\alpha R_{11}) = \alpha\,\sigma_{\min}(R_{11}) = \sigma_{\max}(R_{22})$, we have $\sigma_{i+r}(\tilde{R}_2) = \sigma_i(R_{22})$ for $1 \leq i \leq n-r$. Moreover

$$\begin{aligned}
\|W_2\|_2^2 &= \|\begin{pmatrix} \alpha I_r & -R_{11}^{-1}R_{12} \\ & I_{n-r} \end{pmatrix}\|_2^2 \\
&= \|\begin{pmatrix} \alpha I_r & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & -R_{11}^{-1}R_{12} \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & I_{n-r} \end{pmatrix}\|_2^2 \\
&\leq \|\begin{pmatrix} \alpha I_r & 0 \\ 0 & 0 \end{pmatrix}\|_2^2 + \|\begin{pmatrix} 0 & -R_{11}^{-1}R_{12} \\ 0 & 0 \end{pmatrix}\|_2^2 + \|\begin{pmatrix} 0 & 0 \\ 0 & I_{n-r} \end{pmatrix}\|_2^2 \\
&= \alpha^2 + \|-R_{11}^{-1}R_{12}\|_2^2 + 1 \\
&= \alpha^2 + \|R_{11}^{-1}R_{12}\|_2^2 + 1 \\
&\leq 1 + f^2r(n-r)\ (from\ (i)).
\end{aligned}$$

So $\|W_2\|_2 \leq \sqrt{1+f^2r(n-r)}$. Since

$$\sigma_i(\tilde{R}_2) \leq \sigma_i(R)\|W_2\|_2\ for\ 1 \leq i \leq n,$$

we have

$$\sigma_i(\tilde{R}_2) \leq \sigma_i(R)\sqrt{1+f^2r(n-r)}\ for\ 1 \leq i \leq n.$$

Therefore

$$\begin{aligned}
\sigma_i(R_{22}) &= \sigma_{i+r}(\tilde{R}_2) \\
&\leq \sigma_{i+r}(R)\sqrt{1+f^2r(n-r)} \\
&= \sigma_{i+r}(A)\sqrt{1+f^2r(n-r)}\ for\ 1 \leq i \leq n-r.
\end{aligned}$$

where the last bound follows since R and A have the same singular values $\diamond$.

With the completion of the proof of this theorem, we have shown that the two algorithms above compute strong RRQR factorizations. The purpose of presenting the first two algorithms is to prove the existence of a strong RRQR factorization. Therefore they might not be useful in practice since the numerical rank must be given.

Next we present an algorithm that when given $f \geq 1$ and a tolerance $\epsilon > 0$ compute $R_\epsilon(A)$ and a strong RRQR factorization. Gu and Eisenstat combine the ideas in the second algorithm and QR with column pivoting for the third algorithm, but use

$$\hat{\rho}(R, r) = \max_{1 \leq i \leq r, 1 \leq j \leq n-r} \max\{ | (R_{11}^{-1} R_{12})_{i,j} |, \gamma_j(R_{22}) / \omega_i(R_{11}) \}$$

instead of $\rho(R, r)$ and computes $\omega_*(R_{11})$, $\gamma_*(R_{22})$, and $R_{11}^{-1} R_{12}$ recursively. These modifications yield greater efficiency.

Algorithm 5.3 *Compute r and a strong RRQR factorization.*

$r := 0$; $R \equiv R_{22} := A$; $\Pi := I$;

Initialize $\omega_*(R_{11})$, $\gamma_*(R_{22})$, *and* $R_{11}^{-1} R_{12}$;

while $\max_{1 \leq j \leq n-r} \gamma_j(R_{22}) \geq \epsilon$ **do**

$j_{\max} := argmax_{1 \leq j \leq n-r} \gamma_j(R_{22})$;

$r := r + 1$;

Compute $R \equiv \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} := R_r(R\, \Pi_{r, r+j_{\max}-1})$ *and* $\Pi := \Pi\, \Pi_{r, r+j_{\max}-1}$;

Update $\omega_*(R_{11})$, $\gamma_*(R_{22})$, *and* $R_{11}^{-1} R_{12}$;

while $\hat{\rho}(R, r) > f$ **do**

Find i *and* j *such* $| (R_{11}^{-1} R_{12})_{i,j} | > f$ *or* $\gamma_j(R_{22}) / \omega_i(R_{11}) > f$;

Compute $R \equiv \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} := R_r(R\, \Pi_{i, j+r})$ *and* $\Pi := \Pi\, \Pi_{i, j+r}$;

Modify $\omega_*(R_{11})$, $\gamma_*(R_{22})$, *and* $R_{11}^{-1} R_{12}$;

endwhile

endwhile

Since the inner loop of the third algorithm is essentially equivalent to the second algorithm, this algorithm must eventually halt.

Example

Let $A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{pmatrix}$. $\epsilon = 0.8$ *(we choose* ϵ *relatively large to keep our calculations in this example simple), and* $f = 1$. *We want to find the numerical rank* r *of* A *and a strong RRQR factorization using algorithm 3.*

Let $R = R_{22} = A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{pmatrix}$, $r = 0$, *and* $\Pi = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

$\gamma_1(R_{22}) = \sqrt{14}$

$\gamma_2(R_{22}) = \sqrt{29}$

Since $\max_{1 \le j \le n} \gamma_j(R_{22}) = \gamma_2(R_{22}) = \sqrt{29} > \epsilon$, *we have* $r = 1 +$ *the old* $r = 1$. *Since* $j_{\max} = 2$ *and* $r = 1$, *we apply* $\Pi_{1,2}$ *to* R:

$$
\begin{aligned}
R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} &= R_1(R\,\Pi_{1,2}) \\
&= R_1\{\begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\} \\
&= R_1 \begin{pmatrix} 2 & 1 \\ 3 & 2 \\ 4 & 3 \end{pmatrix} \\
&= \begin{pmatrix} -5.3852 & -3.7139 \\ 0 & -0.4549 \\ 0 & 0 \end{pmatrix}.
\end{aligned}
$$

$\Pi = \Pi\,\Pi_{1,2}$

$\omega_1(R_{11}) = 5.3852$

$\gamma_1(R_{22}) = 0.4549$

$R_{11}^{-1}R_{12} = 0.6897 < f$ *and* $\gamma_1(R_{22})/\omega_1(R_{11}) = 0.0845 < f$

Since $\max_{1\le j\le 1}\gamma_j(R_{22}) = 0.4549 < \epsilon$, **stop**. *Therefore we have* $r = 1$ *and* $R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = \begin{pmatrix} -5.3852 & -3.7139 \\ 0 & -0.4549 \\ 0 & 0 \end{pmatrix}$.

$\sigma_1(A) = 6.5468$, $\sigma_2(A) = 0.3742$, $q_1(1,2) = \sqrt{2}$, $q_2(1,2) = 1$, $\sigma_1(A)/q_1(1,2) = 4.6293$, $\sigma_2(A)\,q_1(1,2) = 0.5292$

Since

1. $\sigma_1(R_{11}) = 6.5468 > 4.6293$ *and*
2. $\sigma_1(R_{22}) = 0.3742 < 0.5292$ *and*
3. $|(R_{11}^{-1}R_{22})_{1,1}| = 0.6897 < q_2(1,2)$,

we have computed a strong RRQR.

Since the inner loop of the third algorithm is essentially equivalent to the second algorithm, the number of flops calculated in each inner loop in the third algorithm are bounded by a polynomial function in the matrix size. In order to show the number of flops calculated in the overall algorithm for the third algorithm is bounded by a polynomial function in the size of the matrix we must exam the total number of flops required by the loops in the third algorithm. Let $f > 1$, $\tau(k)$ be the number of interchanges performed for a particular value of k within the inner while-loop and Δ_k be the determinant of R_{11} after these interchanges are complete. Since $\det(R_{11}) = \Delta_{k-1}\gamma_{j_{max}}(C_{k-1}(A))$ before interchanges and each interchange increases $\det(R_{11})$ by at least a factor of f, we have

$$\Delta_k \ge \Delta_{k-1}\gamma_{j_{max}}C_{k-1}(A)f^{\tau(k)}.$$

From [14] we know $\sigma_{l+1}(A) \le \sigma_{\max}(C_l(A))$ for $1 \le l \le n$. Since $\|C_l(A)\|_F = \sqrt{\sigma_1(C_l(A))^2 + \cdots + \sigma_{n-l}(C_l(A))^2}$, $\sigma_{\max}(C_l(A)) \le \|C_l(A)\|_F$ for $1 \le l \le n$. Since $\|C_l(A)\|_F$ is equal to the square root of the sum of the squares of all entries of $C_l(A)$

and $C_l(A)$ consists of $n-l$ columns. $\|C_l(A)\|_F \leq \sqrt{n-l}\,\gamma_{j_{\max}}(C_l(A))$ for $i \leq l \leq n$. Therefore for $1 \leq l \leq n$ we have

$$\sigma_{l+1}(A) \leq \sigma_{\max}(C_l(A)) \leq \|C_l(A)\|_F \leq \sqrt{n-l}\,\gamma_{j_{\max}}(C_l(A)).$$

We also have

$$\begin{aligned}
\Delta_k &\geq \Delta_{k-1}\,\gamma_{j_{\max}}(C_{k-1}(A))f^{\tau(k)} \\
&\geq \frac{\Delta_{k-1}}{\sqrt{n-(k-1)}}\,\sigma_k(A)f^{\tau(k)} \\
&\geq \frac{\Delta_{k-1}}{\sqrt{n}}\,\sigma_k(A)f^{\tau(k)} \\
&\geq \frac{1}{\sqrt{n}}\left(\frac{\Delta_{k-2}}{\sqrt{n}}\sigma_{k-1}(A)f^{\tau(k-1)}\right)\sigma_k(A)f^{\tau(k)} \\
&= \left(\frac{1}{\sqrt{n}}\right)^2\Delta_{k-2}\,\sigma_{k-1}(A)\,\sigma_k(A)\,f^{\tau(k-1)}f^{\tau(k)} \\
&\geq \left(\frac{1}{\sqrt{n}}\right)^2\left(\frac{\Delta_{k-3}}{\sqrt{n}}\sigma_{k-2}(A)f^{\tau(k-2)}\right)\sigma_{k-1}(A)\,\sigma_k(A)\,f^{\tau(k-1)}f^{\tau(k)} \\
&= \left(\frac{1}{\sqrt{n}}\right)^3\Delta_{k-3}\,\sigma_{k-2}(A)\,\sigma_{k-1}(A)\,\sigma_k(A)\,f^{\tau(k-2)}f^{\tau(k-1)}f^{\tau(k)} \\
&\vdots \\
&\geq \left(\frac{1}{\sqrt{n}}\right)^k\left\{\prod_{i=1}^{k}\sigma_i(A)\right\}f^{\tau(1)}\cdots f^{\tau(k)} \\
&= \left(\frac{1}{\sqrt{n}}\right)^k\left\{\prod_{i=1}^{k}\sigma_i(A)\right\}f^{t(k)}
\end{aligned}$$

where $t(k) = \sum_{i=1}^{k}\tau(i)$ is the total number of interchanges up to this point. From [11] we know $\sigma_i(R_{11}) \leq \sigma_i(A)$. Then we have

$$\Delta_k = \prod_{i=1}^{k}\sigma_i(R_{11}) \leq \prod_{i=1}^{k}\sigma_i(A).$$

Combining the inequalities above, we have $f^{t(k)} \leq (\sqrt{n})^k$, then $t(k) \leq k\log_f\sqrt{n}$. We may conclude that the third algorithm can not have exponential complexity.

Let r be the final value of k when the third algorithm halts, then $t(r) \leq r\log_f\sqrt{n}$. Gu and Eisenstat carry out a more careful analysis as to the cost of the third algorithm

and argue that the total cost is at most around $2mr(2n - r) + 4t(r)n(m + r)$ flops [14]. Gu and Eisenstat showed that $t(r)$ is normally very small [14] (in practice $t(r)$ is even smaller), therefore the cost is about $2mr(2n - r)$ [14]. In the case when $m \gg n$, the total cost of the third algorithm is about the same as QR with column pivoting [14]: when $m \approx n$ the third algorithm is about 50% more expensive [14].

6 Conclusion

Algorithm	*Flop Count*
SVD	$2mn^2 + 4n^3$
QR Column Pivoting	$4mnr - 2r^2(m+n) + 4r^3/3$
Foster/Chan RRQR	$2mn^2 - \frac{2}{3}n^3 + \frac{r}{2}n^2(n-r)$ (max)
Strong RRQR	$4mnr - 2mr^2 + 4t_r n(m+n)$

Table 3: Computational Effort (r is the numerical rank)

In this thesis we have presented four algorithms that reveal the numerical rank of an m × n matrix A where $m \geq n$. They are SVD, QR with column pivoting, Foster/Chan RRQR, and Strong RRQR. Even though SVD is a very reliable algorithm, the table above shows that it can be expensive. QR with column pivoting is an inexpensive alternative, but it can fail to determine the correct numerical rank for some matrices. As we have shown in a previous section, QR with column pivoting fails to determine to correct numerical rank for Kahan's matrix.

The table above shows that Foster/Chan RRQR is a less expensive alternative to SVD. Even though it is a little more expensive than QR with column pivoting, it works where QR with column pivoting fails. In theory the Foster/Chan RRQR algorithm can have exponential bounds in accuracy for large rank deficiencies but it is not so in practice. From the table above we can see the Foster/Chan RRQR is cheaper than the strong RRQR. Last, in the case of $m \gg n$, the strong RRQR is almost as fast as QR with column pivoting and 50% more expensive when $m \approx n$ (since t_r is usually small).

We also showed that in addition to determining the numerical rank of A, SVD and Foster/Chan RRQR can be used to solve the subset selection problem and the least squares problem. Since QR with column pivoting and strong RRQR are both rank revealing QR factorizations, we can use the same methods as Foster/Chan RRQR to

solve the subset selection problem and least squares problem. In a previous section, we showed that even though SVD and Foster/Chan RRQR subset selection algorithms do not necessary produce the same set of columns of A, the subspaces spanned by these two sets of columns should be almost the same when σ_{r+1}/σ_r is small. We have also shown that in many situations SVD and Foster/Chan RRQR are both well suited to solve the least squares problem.

The Foster/Chan RRQR is not only a reliable and less expensive algorithm for determining the numerical rank of A compared to SVD, it can also be used to solve other problems like subset selection and least squares. For MATLAB users, the algorithm can be downloaded from the internet free of charge. It can be found at [19]. Therefore, we believe it is a good alternative to SVD in many ways since it is more efficient and can be used in many applications.

REFERENCES

1. P. Businger and G. H. Golub. *Linear least squares solutions by Householder transformations.* Numer. Math., 7 (1965), pp. 269-276.

2. S. Chandrasekaran and I. Ipsen. *On rank-revealing QR factorization.* SIAM J. Matrix Anal. Appl., 16 (1994), pp. 592-622.

3. T. F. Chan. *Deflated decomposition of solutions of nearly singular systems.* SIAM J. Numer. Anal., 12 (1984), pp. 738-754.

4. T. F. Chan. *Rank revealing QR factorizations.* Linear Algebra Appl., 88/89 (1987), pp. 67-82.

5. T. F. Chan and P. C. Hansen. *Some applications of the rank revealing QR factorization.* SIAM J. Sci. Statist. Comput., 13 (1992), pp. 727-741.

6. B. N. Datta, *Numerical Linear Algebra and Applications.* Brooks/Cole Publishing Company, 1995.

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

8. L. V. Foster, *Rank and null space calculations using matrix decomposition without column interchanges.* Linear Algebra Appl., 74 (1986), pp. 47-71.

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

10. G. H. Golub, V. Klema, and G. W. Stewart, *Rank degeneracy and least squares problems.* Stanford University Technical Report STAN-CS-76-559, 1976.

11. G. H. Golub and C. F. Van Loan, *Matrix Computation.* The Johns Hopkins University Press, Baltimore, 1983.

12. W. B. Gragg and G. W. Stewart, *A stable variant of the secant method for solving nonlinear equations.* SIAM J. numer. Anal., 13(1976), pp. 889-903.

13. R. Grimes and J. Lewis, *Condition number estimation for sparse matrices.* SIAM J. Sci. Statist. Comput., 2 (1981), pp. 384-388.

14. M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization.* SIAM J. Sci. Comput., 17 (1996), pp. 848-868.

15. N. J. Higham, *A survey of condition number estimation for triangular matrices.* SIAM Review, 29 (1987), pp. 575-596.

16. Y. P. Hong and C. T. Pan. *Rank revealing QR factorizations and the singular value decomposition*. Mathematics of Computation. 58 (1992). pp. 213-232.

17. R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press.. 1991.

18. G. W. Stewart. *On the implicit deflation of nearly singular systems of linear equations*. SIAM J. Sci. Statist. Comput.. 2 (1981). pp. 136-140.

19. MATLAB Algo - ftp.mathworks.com