On Rank-Revealing Factorisations

Article *in* SIAM Journal on Matrix Analysis and Applications · April 1994

DOI: 10.1137/S0895479891223781

Shivkumar Chandrasekaran
University of California, Santa Barbara
119 PUBLICATIONS **3,750** CITATIONS

SIAM J. MATRIX ANAL. APPL.
Vol. 15, No. 2, pp. 592–622, April 1994

ON RANK-REVEALING FACTORISATIONS*

SHIVKUMAR CHANDRASEKARAN† AND ILSE C. F. IPSEN†

Abstract. The problem of finding a rank-revealing QR (RRQR) factorisation of a matrix A consists of permuting the columns of A such that the resulting QR factorisation contains an upper triangular matrix whose linearly dependent columns are separated from the linearly independent ones. In this paper a systematic treatment of algorithms for determining RRQR factorisations is presented.

In particular, the authors start by presenting precise mathematical formulations for the problem of determining a RRQR factorisation, all of them optimisation problems. Then a hierarchy of "greedy" algorithms is derived to solve these optimisation problems, and it is shown that the existing RRQR algorithms correspond to particular greedy algorithms in this hierarchy. Matrices on which the greedy algorithms, and therefore the existing RRQR algorithms, can fail arbitrarily badly are presented.

Finally, motivated by an insight from the behaviour of the greedy algorithms, the authors present "hybrid" algorithms that solve the optimisation problems almost exactly (up to a factor proportional to the size of the matrix). Applying the hybrid algorithms as a follow-up to the conventional greedy algorithms may prove to be useful in practice.

Key words. condition estimation, pivoting, orthogonal factorisation, numerical rank, singular values

AMS subject classifications. 65F20, 65F25, 65F35, 15A42

1. Introduction.

The problem of finding a rank-revealing QR (RRQR) factorisation of a matrix A consists of permuting the columns of A such that the resulting QR factorisation contains an upper triangular matrix whose linearly dependent columns are separated from the linearly independent ones. RRQR factorisations are useful in problems such as subset selection and linear dependence analysis [21], [29], [37], [39], subspace tracking [6], [14], and rank determination [9]. Further applications are given in [12] and [17].

To determine a RRQR factorisation one could just adopt the brute force approach and inspect all possible column permutations until one has found a factorisation to one's liking. The operation count, of course, is guaranteed to be combinatorial. Consequently, much effort has gone in designing RRQR algorithms whose operation count is polynomial in the size of the matrix.

The first such algorithm, the QR factorisation with column pivoting [7], [16], [19], was developed by Golub in 1965 and by Faddeev, Kublanovskaya, and Faddeeva in 1966. It makes use of column permutations and orthogonal rotations to maintain the triangular structure of the matrix. About ten years later a second algorithm was published by Golub, Klema, and Stewart [20], based on applying the first algorithm to certain singular vectors of the matrix. At about the same time, a third algorithm appeared in a paper by Gragg and Stewart [22] that works on the inverse of the matrix. These three algorithms constitute the basis for all known RRQR algorithms.

Yet, it took another ten years for the next batch of algorithms by Stewart [32], Foster [17], and Chan [9] to appear. By this time it was known that there are matrices

* Received by the editors December 23, 1991; accepted for publication (in revised form) October 26, 1992. The work presented in this paper was supported by National Science Foundation grant CCR-9102853.

† Department of Computer Science, Yale University, New Haven, Connecticut 06520 (ipsen@math.ncsu.edu, chandras@math.ncsu.edu).

for which Golub's RRQR algorithm [7], [16], [19] can fail arbitrarily badly; Kahan's matrix [28] is such an example.

Again the field lay fallow for several years. Recently Hong and Pan [26] proved that an optimal RRQR factorisation is able to produce an estimate of a singular value that is accurate up to a factor proportional to the matrix size. This result implies that, in exact arithmetic and with a combinatorial operation count, RRQR factorisations have the potential of being accurate and reliable. (Much more than that, though, the result represents a statement about the relation between matrix columns and singular values: it says that there are k columns in the matrix that can reproduce, up to a factor in the matrix size, the kth singular value of the matrix.)

These days, the potential of RRQR factorisations is investigated for use in truncated singular value decompositions [10], [23], Lanczos methods [14], total least squares [37], and sparse matrix computations [3]–[5], [30]. Stewart has extended the RRQR factorisation by allowing orthogonal rotations from the right, resulting in the so-called URV decomposition [1], [31], [35], [36].

The state of affairs regarding RRQR factorisations can be summed up as follows. Despite the variety of algorithms, the problem of what it means to find a RRQR decomposition has never been clearly defined. Most definitions of a RRQR factorisation are about as fuzzy as the one we gave in the first sentence of this paper. Relationships or connections among the different RRQR algorithms are not known. All algorithms have the potential of failing badly. For some, we know the matrices where they fail badly. No criteria, other than a few test matrices, are known for comparing algorithms and judging their quality. Surprisingly, in numerical experiments, most RRQR algorithms turn out to be accurate and fast.

In this paper we present a systematic treatment of algorithms for determining RRQR factorisations. We start by presenting three precise mathematical formulations for the problem of determining a RRQR factorisation: one is a maximisation problem, one is a minimisation problem, and a third one is a combination of the two. We derive a hierarchy of "greedy" algorithms to solve the maximisation problem. It turns out that algorithms for solving the minimisation problem can be obtained by running algorithms for the maximisation problem on the inverse of the matrix and vice versa. This gives two parallel hierarchies of greedy algorithms for determining RRQR factorisations. We show that the existing RRQR algorithms correspond to particular greedy algorithms in this hierarchy. Moreover, we present matrices on which the greedy algorithms, and therefore the existing RRQR algorithms, fail arbitrarily badly.

Finally, motivated by our insight from the behaviour of the greedy algorithms, we present three "hybrid" algorithms that solve the optimisation problems with an accuracy given by the bounds of Hong and Pan [26]. Although the worst-case operation count of the hybrid algorithms may be combinatorial, we have not been able to find a matrix where this occurs. We present a few numerical experiments to demonstrate that applying the hybrid algorithms as a follow-up to the conventional RRQR algorithms may prove to be useful in practice.

2. The problem.

In this section we give mathematical formulations of the problem of determining a rank-revealing QR (RRQR) factorisation of a matrix M.

Let M be a real $m \times n$ matrix and $m \geq n$. We assume that the singular values $\sigma_i(M)$ of M are arranged in decreasing order

$$\sigma_1(M) \geq \cdots \geq \sigma_n(M).$$

We also assume that k is a given integer such that $1 \leq k < n$ and $\sigma_k(M) > 0$. In the applications where rank-revealing factorisations are of relevance, $\sigma_k(M)$ and $\sigma_{k+1}(M)$ are usually "well-separated," and $\sigma_{k+1}(M)$ is "small," of the order of the error in the computation, which means that the matrix has numerical rank k. Although our algorithms do not use this, it is useful to keep it in mind.

Denote by

$$M\Pi = QR$$

the QR factorisation of M with its columns permuted according to the $n \times n$ permutation matrix Π. The real $m \times n$ matrix Q has orthonormal columns, and the real $n \times n$ matrix R is upper triangular with positive diagonal elements. We block-partition R as

$$\begin{array}{c c} & \begin{array}{cc} k & n-k \end{array} \\ \begin{array}{c} k \\ n-k \end{array} & \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = R, \end{array}$$

where R_{11} is a $k \times k$ matrix.

The RRQR problem. The problems to be discussed in this paper are how to choose permutations Π such that

$$\sigma_{\min}(R_{11}) \approx \sigma_k(M)$$

or

$$\sigma_{\max}(R_{22}) \approx \sigma_{k+1}(M)$$

or both hold simultaneously. So there are three objectives leading to three different problems, all of which we refer to as "rank-revealing problems." It is an open question whether these are really three *different* objectives. That is, if we find a permutation such that $\sigma_{\min}(R_{11}) \approx \sigma_k(M)$, does it imply that $\sigma_{\max}(R_{22}) \approx \sigma_{k+1}(M)$? Our attempts at answering this question have not yielded sufficiently good answers, and in this paper we will consider them as three independent objectives.

Satisfaction of the third objective, where both bounds are satisfied simultaneously, implies that the leading k columns of $M\Pi$ have condition number $\sigma_1(M)/\sigma_k(M)$ and approximate the range space of M to an "accuracy" of $\sigma_{k+1}(M)$.

Before proceeding any further we should be more specific about those $\approx$ signs. According to the interlacing properties of singular values (Corollary 8.3.3 in [21] applied to R^T) the bounds

(I1) $\quad \sigma_{\min}(R_{11}) \leq \sigma_k(M)$,

(I2) $\quad \sigma_{\max}(R_{22}) \geq \sigma_{k+1}(M)$

hold for *any* permutation Π. So the RRQR problems can be precisely formulated as

Problem-I: $\quad \max_\Pi \sigma_{\min}(R_{11})$;

Problem-II: $\quad \min_\Pi \sigma_{\max}(R_{22})$

or that both can be solved simultaneously, though that may not be possible all the time.

Because we believe that the time complexity of these problems is combinatorial, we are content to find permutations Π that guarantee

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{p(n)}$$

or

$$\sigma_{\max}(R_{22}) \leq q(n)\sigma_{k+1}(M)$$

or that both bounds hold simultaneously. Here $p(n)$ and $q(n)$ are low degree polynomials in n. We say that a permutation Π that achieves one or both of these inequalities gives rise to a RRQR factorisation $M\Pi = QR$. An algorithm that attempts to solve Problem-I is called a Type-I algorithm and has the suffix I in its name. An algorithm that attempts to solve Problem-II is called a Type-II algorithm and has the suffix II in its name.

3. Overview of RRQR algorithms. We accomplish two tasks in this paper: first, we demonstrate that all existing RRQR algorithms form a hierarchy of greedy algorithms; and second, we present a set of new algorithms that are more accurate than the existing RRQR algorithms.

The existing algorithms in the literature guarantee that

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{n2^k} \qquad \text{or} \qquad \sigma_{\max}(R_{22}) \leq \sigma_{k+1}(M)n2^{n-k},$$

where the bounds are worst-case bounds. In practice, however, the existing algorithms perform quite well and the worst-case bounds are rarely obtained. There also exists an algorithm [20] with simultaneous worst-case bounds

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{n2^{\min(k,n-k)}}, \qquad \sigma_{\max}(R_{22}) \leq \sigma_{k+1}(M)n2^{\min(k,n-k)}.$$

In contrast, our new algorithms guarantee

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{\sqrt{k(n-k+1)}}$$

or

$$\sigma_{\max}(R_{22}) \leq \sigma_{k+1}(M)\sqrt{(k+1)(n-k)},$$

or both. The *existence* of such RRQR factorisations was established in [26]. Although we believe that the operation count of our new algorithms is combinatorial in the worst case, preliminary numerical experiments indicate that they may be fast in practice.

We ignore brute force algorithms for finding permutations Π because they do not exploit any properties of the matrix. Their operation count is therefore *always* combinatorial.

Now we start with the presentation of a unified approach to the existing RRQR algorithms. Our approach simplifies the presentation and analysis of these algorithms, and it also directly motivates our new algorithms. To this end, we make the following simplification. If $M\Pi = QR$ is a QR factorisation of M for some permutation Π, and if $R\bar{\Pi} = \bar{Q}\bar{R}$ is a RRQR factorisation of R, then

$$M\Pi\bar{\Pi} = Q\bar{Q}\bar{R}$$

is a RRQR factorisation of M. Hence one can ignore the original matrix M and work with the triangular matrix R instead.

4. Type-I greedy algorithms.

It is our goal to find algorithms to solve Problem-I

$$\max_{\Pi} \sigma_{\min}(R_{11})$$

that guarantee

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{p(n)},$$

where $p(n)$ is a low degree polynomial in n. Problem-I is likely to represent a combinatorial optimisation problem, and this suggests that a greedy algorithm might do well.

The basic idea for our greedy algorithm, which we call Greedy-I, is very simple. The objective of Problem-I is to find k well-conditioned columns of M. So suppose that we already have $l < k$ well-conditioned columns of M. Then Greedy-I picks a column from the remaining $n - l$ columns of M such that the smallest singular value of the given l columns plus the new column is as large as possible. Starting with $l = 0$ this is done k times to pick k well-conditioned columns of M. Note that Greedy-I does not discard a column once it has been chosen.

```
ALGORITHM GREEDY-I
$R^{(0)} = R$
For $l = 0$ to $k - 1$ do
Set
```

$$\begin{array}{c|cc} & l & n-l \\ \hline l & A & B \\ n-l & & C \end{array} = R^{(l)}.$$

```
Denote the columns of $B$ and $C$ by $b_i = Be_i$ and $c_i = Ce_i$.
  1. Find the next column $l + j$ of $R^{(l)}$ such that
```

$$\max_{1\leq i\leq n-l} \sigma_{\min} \begin{pmatrix} A & b_i \\ & c_i \end{pmatrix} = \sigma_{\min} \begin{pmatrix} A & b_j \\ & c_j \end{pmatrix}.$$

```
  2. Exchange columns $l + 1$ and $l + j$ of $R^{(l)}$, and retriangularise it from the left with orthogonal transformations to get $R^{(l+1)}$.
```

In iteration $l = 0$, Greedy-I selects the column of R with largest norm. If everything goes right, then $R^{(k)}$ should be a rank-revealed upper triangular matrix. It is important to keep in mind that the dimensions of A, B, and C change with every iteration of Greedy-I.

Step 1 of Greedy-I, which selects the next column to be added to A, is very expensive. We make it cheaper, while at the same time retaining the greedy strategy, by performing step 1 only approximately. Thus the algorithm becomes less greedy and more efficient. Since Greedy-I can only find an approximate solution at best, further approximations will hopefully not make matters much worse.

Before continuing we make a small simplification. If $\gamma_i = \|c_i\|$, where $\|\cdot\|$ represents the two-norm, then

$$\sigma_{\min} \begin{pmatrix} A & b_i \\ 0 & c_i \end{pmatrix} = \sigma_{\min} \begin{pmatrix} A & b_i \\ 0 & \gamma_i \end{pmatrix}.$$

This means, only the two-norm of the columns of C matters rather than individual elements in a column. Therefore the problem amounts to determining the smallest singular values of an upper triangular matrix of order $l+1$.

Now we present a sequence of successively less greedy approximations to step 1 of Greedy-I that give rise to most of the existing RRQR algorithms. In other words, we show that most existing RRQR algorithms can be viewed as approximations to algorithm Greedy-I.

In the first approximation, the determination of the smallest singular values $\sigma_{\min}(\cdot)$ is replaced by directly computable quantities. We choose to approximate the smallest singular value of a matrix by the reciprocal of the largest two-norm of the rows of its inverse: if D is a nonsingular matrix of order n and

$$D^{-1} = \begin{pmatrix} r_1^T \\ r_2^T \\ \vdots \\ r_n^T \end{pmatrix},$$

then

$$\sigma_{\min}(D) \le \min_{1\le i\le n} \frac{1}{\|r_i\|} \le \sigma_{\min}(D)\sqrt{n}.$$

Consequently, the smallest singular value of a nonsingular matrix of order n can be estimated up to a factor of $\sqrt{n}$.

ALGORITHM GREEDY-I.1
Replace step 1 in algorithm Greedy-I by:
Find the next column $l+j$ of $R^{(l)}$ such that

$$\max_{1\le i\le n-l} \min_h \left\| e_h^T \begin{pmatrix} A & b_i \\ 0 & \gamma_i \end{pmatrix}^{-1} \right\|^{-1} = \min_h \left\| e_h^T \begin{pmatrix} A & b_j \\ 0 & \gamma_j \end{pmatrix}^{-1} \right\|^{-1},$$

where e_h^T is the hth row of the identity matrix of order $l+1$.

An algorithm similar to Greedy-I.1 was proposed by Stewart [34] where, for reasons of efficiency, the Frobenius norm rather than the two-norm is used.

Although we say that Greedy-I.1 is an approximation to Greedy-I, this does not necessarily imply that Greedy-I reveals the rank better than Greedy-I.1. It only means that Greedy-I.1 is less greedy than Greedy-I. In particular, if in iteration l Greedy-I and Greedy-I.1 have the same submatrix A, then the $\sigma_{\min}$ of the leading $l+1$ columns from Greedy-I is larger than or equal to the $\sigma_{\min}$ of the corresponding columns from Greedy-I.1. But there is no guarantee that in the subsequent iteration $l+1$ the $\sigma_{\min}$ of the leading $l+2$ columns of Greedy-I will be larger than or equal to the $\sigma_{\min}$ of the corresponding columns of Greedy-I.1. This is because the greedy algorithms are not allowed to change their minds and to throw out a column selected in a previous iteration, and the best local choice in one step does not necessarily lead to the global optimum.

Because

$$\begin{pmatrix} A & b_i \\ 0 & \gamma_i \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}b_i\gamma_i^{-1} \\ 0 & \gamma_i^{-1} \end{pmatrix},$$

the upper left $l \times l$ block A^{-1} is already available from the previous step, and only the last column of the inverse needs to be computed for each i, which requires $n-l$ matrix vector multiplications. But carrying the inverse along with us at every stage is costly in terms of space and we first get rid of that.

If the greedy algorithms have not failed at the lth stage, the l leading columns must be "well conditioned." Hence A must be a well-conditioned matrix. Therefore $\sigma_{\min}(A)$ cannot be "small," which in turn implies that no row of A^{-1} can have a large two-norm. But if the addition of a new column, say the ith, produces a small singular value, then the two-norm of some row of the inverse of the corresponding matrix must be large. But since we assumed that no row of A^{-1} is large, this must mean that some component of the last column of the inverse

$$\begin{pmatrix} -A^{-1}b_i\gamma_i^{-1} \\ \gamma_i^{-1} \end{pmatrix}$$

must be large in magnitude. Thus the second approximation to step 1 of Greedy-I,

$$\left\| e_h^T \begin{pmatrix} A & b_i \\ 0 & \gamma_i \end{pmatrix}^{-1} \right\| \approx \left| \begin{pmatrix} -A^{-1}b_i\gamma_i^{-1} \\ \gamma_i^{-1} \end{pmatrix}_h \right|,$$

still avoids the selection of a very bad column.

ALGORITHM GREEDY-I.2
Replace step 1 in algorithm Greedy-I by:
Find the next column $l+j$ of $R^{(l)}$ such that

$$\max_{1\le i\le n-l} \min_h \left| \begin{pmatrix} -A^{-1}b_i\gamma_i^{-1} \\ \gamma_i^{-1} \end{pmatrix}_h \right|^{-1} = \min_h \left| \begin{pmatrix} -A^{-1}b_j\gamma_j^{-1} \\ \gamma_j^{-1} \end{pmatrix}_h \right|^{-1}.$$

To eliminate the $n-l$ backsolves $A^{-1}b_i$ in Greedy-I.2, we make further use of the observation that A is probably well conditioned, so $\|A^{-1}b_i\| \approx 1$, and any large value must come from γ_i. Thus the third approximation to step 1 of Greedy-I,

$$\min_h \left| \begin{pmatrix} -A^{-1}b_i\gamma_i^{-1} \\ \gamma_i^{-1} \end{pmatrix}_h \right|^{-1} \approx \gamma_i,$$

still tries to avoid selecting a very bad column. This is nothing but the standard QR algorithm with column pivoting [7], [19], which is also described in [16].

ALGORITHM GREEDY-I.3 (GOLUB-I)
Replace step 1 in algorithm Greedy-I by:
Find the next column $l+j$ of $R^{(l)}$ such that $\max_{1\le i\le n-l}\gamma_i = \gamma_j$.

This algorithm can be implemented efficiently because the column norms γ_i need only be updated during each iteration, rather than recomputed from scratch [7].

The approximations still to be discussed do not result in algorithms that are faster than Golub-I; in fact, they may be slower, but they are necessary to derive the remaining existing RRQR algorithms.

The goal is to make a further approximation to

$$\max_{1\le i\le n-l} \gamma_i \equiv \alpha_{l+1}$$

in iteration l, where α_l is the lth diagonal element in the final upper triangular matrix in Golub-I. To this end, compute the right singular vector of the submatrix C of $R^{(l)}$ corresponding to its largest singular value $\|C\|$. Therefore the next approximation consists of finding the $(n-l) \times 1$ vector v such that

$$Cv = \|C\|u, \qquad \|v\| = \|u\| = 1,$$

and choosing as the next column the column j that corresponds to the largest component in magnitude of v,

$$|v_j| = \max_{1 \le i \le n-l} |v_i|.$$

ALGORITHM GREEDY-I.4 (CHAN-I)
Replace step 1 in algorithm Greedy-I by:
Find the next column $l+j$ of $R^{(l)}$ for which $|v_j| = \max_{1 \le i \le n-l} |v_i|$.

This algorithm was discovered independently by Chan and Hansen [11] and is related to the algorithm in [9]. Its choice of column j can be justified as follows. The Cauchy–Schwartz inequality gives

$$\gamma_j = \|Ce_j\| = \|u\| \, \|Ce_j\| \ge |u^T Ce_j| = \|C\| \, |v_j|.$$

As v has $n-l$ elements and satisfies $\|v\| = 1$, it must have a component v_j for which $|v_j| \ge 1/\sqrt{n-l}$. This is true in particular for the largest component in magnitude of v. Using this in $\gamma_j \ge \|C\| \, |v_j|$ gives

$$\frac{\alpha_{l+1}}{\sqrt{n-l}} \le \gamma_j \le \alpha_{l+1}, \qquad \alpha_{l+1} = \max_{1 \le i \le n-l} \gamma_i.$$

That is, the γ_j from algorithm Chan-I will be almost as large as that from algorithm Golub-I, if both algorithms were given the same l columns in A.

5. Threshold pivoting algorithms. We can make even further approximations to Chan-I. Algorithms Golub-I and Chan-I can be viewed as selecting large diagonal elements (*pivots*) at each stage to keep the smallest singular value as large as possible. According to the interlacing property ($I2$) of singular values, $\|C\| \ge \sigma_{l+1}(M)$, so

$$\alpha_{l+1} \ge \frac{\sigma_{l+1}(M)}{\sqrt{n-l}}, \quad 0 \le l \le k-1, \quad \text{where } \alpha_{l+1} = \max_{1 \le i \le n-l} \|Ce_i\|.$$

But all that is really needed is

$$\sigma_{\min}(R_{11}) \approx \sigma_k(M),$$

which means one may be able to get away with choosing pivots that are only as large as $\sigma_k(M)$. That is, instead of trying to achieve

$$|(R_{11})_{ll}| \approx \sigma_l(M), \qquad 1 \le l \le k,$$

we only try to ensure that

$$|(R_{11})_{ll}| \approx \sigma_k(M), \qquad 1 \le l \le k.$$

Golub-I and Chan-I try to keep all the pivots as large as possible at each stage. But since $\sigma_{\min}(R_{11})$ will be smaller than the smallest pivot, we are hoping that only the size of the smallest pivot is important, so that the conditions on the larger pivots can be relaxed.

Versions of Golub-I based on this approximation also go by the name of "threshold pivoting," and we now present two such algorithms. The first algorithm represents one of the first RRQR algorithms [20], [21] and, as we will show later, has the distinction of being able to solve both Problem-I and Problem-II simultaneously. Our name for the algorithm derives from the last names of its authors, Golub, Klema, and Stewart.

ALGORITHM GKS-I
Let $R = U\Sigma V^T$ be the singular value decomposition of R with

$$V = \begin{array}{cc} k & n-k \\ \big(V_1 & V_2\,\big). \end{array}$$

1. Compute V_1.
2. Apply algorithm Golub-I to the rows of V_1, $V_1^T\Pi = Q_v\bar{V}_1^T$.
3. Compute the QR decomposition $R\Pi = \bar{Q}\bar{R}$, which is the required rank-revealing factorisation.

To see that this is indeed a threshold pivoting algorithm, partition the singular value decomposition (SVD) of R as follows

$$R = U\begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix}(V_1 \quad V_2)^T.$$

Substituting the result of step 3, $\bar{Q}^T R\Pi = \bar{R}$, in step 2 gives

$$\bar{V}_1^T\bar{R}^{-1} = Q_v^T\Sigma_1^{-1}U^T\bar{Q}.$$

Since $\bar{R}^{-1}$ and the leading k columns of $\bar{V}_1^T$ represent upper triangular matrices,

$$\frac{|(\bar{V}_1)_{ii}|}{|(\bar{R})_{ii}|} \le \|\bar{V}_1^T\bar{R}^{-1}\| = \|\Sigma_1^{-1}\| \le \frac{1}{\sigma_k(M)}.$$

Because $\bar{V}_1^T$ is the result of QR with column pivoting on a matrix with orthonormal rows, the largest element in magnitude in the ith row of $\bar{V}_1^T$ is $(\bar{V}_1)_{ii}$ and $|(\bar{V}_1)_{ii}| \ge 1/\sqrt{n}$. Combining the inequalities gives a lower bound on the pivots,

$$|(\bar{R})_{ii}| \ge \frac{\sigma_k(M)}{\sqrt{n}}.$$

So algorithm GKS-I behaves like a threshold pivoting algorithm.

We now describe a threshold pivoting algorithm that we call Foster-I because it is related to an algorithm proposed by Foster (see Algorithm 2 in [17]). For a given δ, where δ is presumably about as big as $\sigma_k(M)$, Foster-I tries to achieve $\sigma_{\min}(R_{11}) \approx \delta$ by choosing pivots greater than or equal to δ. To this end it searches the rows of R, bottom up, for an element of magnitude greater than δ. When it finds such an element it adds the corresponding column to R_{11} and continues the search. As in all greedy algorithms for Problem-I, once a column has been added to R_{11} it is never discarded again. The algorithm halts when it has finished searching n rows. If it succeeds in finding k elements larger than δ, then the first k pivots are at least as large as δ.

```
ALGORITHM FOSTER-I
$i = n$, $count = n$, $l = 0$
While ($count \geq 1$) do
    Find the maximal element in row $i$: $|R_{ij}| = \max\{|R_{ii}|, \ldots, |R_{in}|\}$
    If ($|R_{ij}| \geq \delta$) then
        Insert column $j$ between the $l$th and $(l+1)$st columns
        Retriangularise $R$
        $l = l + 1$
    else
        $i = i - 1$
    $count = count - 1$
```

Here we have come to the end of our approximations to Greedy-I, which was a greedy algorithm for solving the Type-I problem

$$\max_{\Pi} \sigma_{\min}(R_{11}).$$

6. (Pessimistic) analysis of the greedy algorithms. In the previous sections we presented a succession of approximations to algorithm Greedy-I with little formal justification. Now we need to investigate how big $\sigma_k(M)/\sigma_{\min}(R_{11})$ from these algorithms can be. Algorithm Greedy-I represents the "best" method in the greedy sense, so we expect its worst-case behaviour to be indicative of that of the other greedy algorithms.

Suppose Greedy-I has already set aside l columns

$$R^{(l)} = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix},$$

where A is a $l \times l$ matrix. It then chooses as the $(l+1)$st column that column j which when added to A maximises the smallest singular value, so

$$\bar{\sigma}_{l+1} = \max_{1 \leq i \leq n-l} \sigma_{\min} \begin{pmatrix} A & b_i \\ 0 & c_i \end{pmatrix} = \sigma_{\min} \begin{pmatrix} A & b_j \\ 0 & c_j \end{pmatrix}.$$

To estimate how small $\bar{\sigma}_{l+1}$ can be we need to compute a lower bound on the smallest singular value. To this end we compute a lower bound instead for the column Golub-I would select, given the same A, because this also serves as a lower bound for the column Greedy-I picks. So assume that Golub-I picks column q. This column has the largest norm among all columns of C.

Just as in algorithm Greedy-I.1, we estimate $\sigma_{\min}$ by the reciprocal of the largest two-norm of the rows of the inverse

$$\begin{pmatrix} A & b_q \\ 0 & \gamma_q \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & -A^{-1} b_q \gamma_q^{-1} \\ 0 & \gamma_q^{-1} \end{pmatrix}.$$

The norm of the row with the largest norm among the leading l rows of the inverse is bounded from above by

$$\max_i \|e_i^T A^{-1}\| + \|A^{-1} b_q\| \gamma_q^{-1} \leq \frac{1}{\bar{\sigma}_l} + \frac{\|b_q\|}{\bar{\sigma}_l} \gamma_q^{-1} \leq \sqrt{2} \frac{\sqrt{\gamma_q^2 + \|b_q\|^2}}{\bar{\sigma}_l} \gamma_q^{-1} \leq \sqrt{2} \frac{\|M\|}{\bar{\sigma}_l} \gamma_q^{-1},$$

where the penultimate inequality is a result of the Cauchy–Schwartz inequality. The norm of the $(l+1)$st row of the inverse clearly cannot exceed the upper bound on the maximal norm of the leading l rows. Since the maximal row norm of the inverse approximates the smallest singular value of the matrix, we have

$$\bar{\sigma}_{l+1} \geq \frac{1}{\sqrt{2(l+1)}} \frac{\bar{\sigma}_l}{\sigma_1(M)} \gamma_q.$$

Using the interlacing property $(I2)$

$$\gamma_q = \max_{1 \leq i \leq n-l} \gamma_i \geq \frac{\sigma_{l+1}(M)}{\sqrt{n-l}}$$

with the above inequality gives a lower bound for the smallest singular value

$$\bar{\sigma}_{l+1} \geq \sigma_{l+1}(M) \frac{\bar{\sigma}_l}{\sigma_1(M)} \frac{1}{\sqrt{2(l+1)(n-l)}}.$$

This goes to show that even if the leading l columns had been selected so that $\bar{\sigma}_l$ was as accurate as possible, there could be a potentially serious deterioration in the quality of estimation from lth to $(l+1)$st singular value if the $(l+1)$st column is chosen according to a greedy strategy. This is because a greedy algorithm, once it has decided on a column, can never get rid of it. And a column that participates in an *accurate* estimation of $\bar{\sigma}_l$ may not be a column to be included in an accurate estimation of $\bar{\sigma}_{l+1}$. In particular, the estimate $\bar{\sigma}_{l+1}$ worsens with the ill conditioning of the leading l columns in $R^{(l)}$.

In fact, there exist matrices that almost achieve the above bound. One such example is the Kahan matrix [28]

$$K_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & s & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & s^{n-1} \end{pmatrix} \begin{pmatrix} 1 & -c & \cdots & -c \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -c \\ 0 & \cdots & 0 & 1 \end{pmatrix},$$

where $c^2 + s^2 = 1$. Greedy-I, Greedy-I.1, Greedy-I.2, and Golub-I do not cause any permutation of the columns of K_n. We prove this for Greedy-I by induction. Since all columns of K_n have unit norm, no column permutations are necessary in the first iteration of Greedy-I. Suppose no permutations are necessary during the first l iterations, so

$$K_n = \begin{pmatrix} K_l & b_1 & \dots & b_{n-l} \\ & c_1 & \dots & c_{n-l} \end{pmatrix}.$$

In the $(l+1)$st iteration Greedy-I selects the $(l+1)$st column by examining

$$\sigma_{\min} \begin{pmatrix} K_l & b_i \\ & c_i \end{pmatrix} = \sigma_{\min} \begin{pmatrix} K_l & b_i \\ & \gamma_i \end{pmatrix}, \quad \gamma_i = \|c_i\|, \quad 1 \leq i \leq n-l.$$

But all b_i are identical for the Kahan matrix, as are all γ_i; hence no permutations are necessary in iteration $l+1$.

Yet K_n is not in rank-revealed form. For $n = 100$, $k = 99$, and $c = 0.2$, the singular values are

$$\begin{aligned} \sigma_{100}(K_{100}) &\approx 3 \times 10^{-9}, \\ \sigma_{99}(K_{100}) &\approx 0.1482, \\ \bar{\sigma}_{99} &\approx 4 \times 10^{-9}. \end{aligned}$$

Although the 99th and 100th singular values are well separated, the smallest singular value of the first 99 columns chosen by the greedy algorithms is exponentially smaller than $\sigma_{99}(K_{100})$.

Traditionally, the Kahan matrix has served as an example to demonstrate the failure of algorithm Golub-I to make the *last* diagonal element of the same order of magnitude as $\sigma_{100}(K_{100})$. But from our discussion it is clear that Golub-I pursues a different mission: it wants to make $\bar{\sigma}_{99} \approx \sigma_{99}(K_{100})$. And it fails in *that*.

7. (Optimistic) analysis of the greedy algorithms. Now that we have seen how badly the greedy algorithms do, we wonder why they do so well in practice? This question seems to be related to other rare matrix events like pivot growth in Gaussian elimination with partial pivoting. Foster [18] considers this question for QR without column pivoting. The case of QR with column pivoting seems to be much harder to analyse, and we can only give informal reasons why the greedy algorithms Golub-I, Chan-I, and GKS-I are so effective.

The basic idea is to derive a lower bound for $\sigma_{\min}(R_{11})$ of the form

$$\frac{\sigma_k(M)}{n\|W^{-1}\|} \le \sigma_{\min}(R_{11}) \le \sigma_k(M),$$

where W is a $k \times k$ triangular matrix with

$$|W| \le 1, \qquad |W_{ii}| = 1,$$

and the inequality is componentwise. The lower triangular matrix in Gaussian elimination with partial pivoting satisfies these same two properties as the W matrices and is usually well conditioned. (Or as Kahan [28] would say, "intolerable pivot growth is a phenomenon that happens only to numerical analysts who are looking for that phenomenon.") Of course, this does not prove anything and more work is needed in this regard.

We start with the derivation of the above bound for algorithm Golub-I. Here we define the matrix W by

$$R_{11} = DW, \qquad D = \operatorname{diag}(R_{11}),$$

where $\operatorname{diag}(R_{11})$ is a diagonal matrix whose diagonal elements are the same as those of R_{11}. The diagonal elements of R_{11} in Golub-I satisfy $|(R_{11})_{ii}| \ge |(R_{11})_{ij}|$; hence W fulfills the required conditions

$$|W| \le 1, \qquad |W_{ii}| = 1.$$

The interlacing properties ($I2$) of singular values and the first few inequalities in § 5 imply that

$$|(R_{11})_{ii}| = \alpha_i \ge \frac{\sigma_i(M)}{\sqrt{n-i+1}}, \qquad 1 \le i \le k.$$

Since D is a diagonal matrix, its singular values equal its diagonal elements, so

$$\frac{1}{\sigma_{\min}(D)} = \|D^{-1}\| \leq \frac{\sqrt{n}}{\sigma_k(M)}.$$

From $\sigma_{\min}(R_{11}) \geq \sigma_{\min}(D)\sigma_{\min}(W)$ the desired bound for Golub-I follows

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{\sqrt{n}\|W^{-1}\|}.$$

Next we derive the bound for algorithm Chan-I. The proof is similar to that of Theorem 3.1 in [9]. We first define the $n \times k$ auxiliary matrix Z. Its columns are composed of the right singular vectors $v^{(l)}$ associated with the largest singular values of the lower right block of order $n-l+1$, $R_{22}^{(l)}$, of the final triangular matrix R. That is, Z is a lower trapezoidal matrix with columns

$$Ze_l = Z_l = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ v^{(l)} \end{pmatrix}, \qquad 1 \leq l \leq k,$$

where $R_{22}^{(l)} v^{(l)} = \|R_{22}^{(l)}\| u^{(l)}$ and $\|v^{(l)}\| = \|u^{(l)}\| = 1$. Then the lower triangular matrix W for Chan-I is given by

$$Z = \begin{pmatrix} W \\ * \end{pmatrix} D, \qquad D = \operatorname{diag}\left(Z_{11} \quad \ldots \quad Z_{kk} \right),$$

where Z_{ii} are the diagonal elements of Z.

According to algorithm Chan-I, the first component of $v^{(l)}$ is the largest in magnitude, hence

$$|W| \leq 1, \qquad |W_{ii}| = 1.$$

Moreover, $\|v^{(l)}\| = 1$ implies

$$|Z_{ll}| = |v_1^{(l)}| \geq \frac{1}{\sqrt{n}}, \qquad 1 \leq l \leq k,$$

and

$$\frac{1}{\|D^{-1}\|} = \sigma_{\min}(D) \geq \frac{1}{\sqrt{n}}.$$

From the interlacing property ($I2$) of singular values, $\|R_{22}^{(l)}\| \geq \sigma_l(M)$, and the fact that $v^{(l)}$ is a right singular vector with

$$[v^{(l)}]^T \, [R_{22}^{(l)}]^{-1} = \frac{1}{\|R_{22}^{(l)}\|} [u^{(l)}]^T,$$

we get

$$\|Z_l^T R^{-1}\| = \|[v^{(l)}]^T \, [R_{22}^{(l)}]^{-1}\| \leq \frac{1}{\sigma_l(M)}.$$

Since Z^T and R are upper triangular this implies

$$\|DW^TR_{11}^{-1}\| \leq \|Z^TR^{-1}\| \leq \sqrt{k}\max_{1\leq l\leq k}\|Z_l^TR^{-1}\| \leq \frac{\sqrt{k}}{\sigma_k(M)}.$$

Hence

$$\frac{\|R_{11}^{-1}\|}{\|D^{-1}\|\,\|W^{-1}\|} \leq \frac{\sqrt{k}}{\sigma_k(M)}$$

gives the desired bound for algorithm Chan-I

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{n\|W^{-1}\|}.$$

At last we derive the bound for algorithm GKS-I, which is also given in [20] and in Theorem 12.2.1 in [21]. Let $R = U\Sigma V^T$ be the SVD of R and partition

$$V = \begin{array}{cc} k & n-k \\ \begin{pmatrix} V_1 & V_2 \end{pmatrix} \end{array}.$$

Algorithm GKS applies algorithm Golub-I to V_1^T, so

$$V_1^T\Pi = Q_v\bar{V}_1^T,$$

where $\bar{V}_1$ is a lower trapezoidal matrix. The matrix W for GKS-I is defined by

$$\bar{V}_1 = \begin{pmatrix} W \\ * \end{pmatrix} D, \qquad D = \operatorname{diag}(\bar{V}_{11} \quad \dots \quad \bar{V}_{kk}),$$

where $\bar{V}_{ii}$ are diagonal elements of $\bar{V}_1$ and W is a lower triangular matrix. Because $\bar{V}_1$ comes from algorithm Golub-I, its diagonal elements are the largest elements in magnitude in each column, so $|\bar{V}_{ii}| \geq |\bar{V}_{ji}|$ and the matrix W satisfies the required properties

$$|W| \leq 1, \qquad |W_{ii}| = 1.$$

Since each column of $\bar{V}_1$ has unit norm, $|\bar{V}_{ii}| \geq 1/\sqrt{n}$, and since W^T and the final matrix R are upper triangular, one gets

$$\frac{\|R_{11}^{-1}\|}{\sqrt{n}\|W^{-1}\|} \leq \frac{\|R_{11}^{-1}\|}{\|W^{-T}\|\,\|D^{-1}\|} \leq \|DW^TR_{11}^{-1}\| \leq \|\bar{V}_1^TR^{-1}\|.$$

Moreover, from § 5 we know that (with $\bar{R}$ now renamed R)

$$\|\bar{V}_1^TR^{-1}\| = \|\Sigma_1^{-1}\| = \frac{1}{\sigma_k(M)}.$$

Combining the last two inequalities yields

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{\sqrt{n}\|W^{-1}\|}.$$

To summarise, we have demonstrated in this section that the failure of algorithms Golub-I, Chan-I, and GKS-I depends on $\|W^{-1}\|$, where W is a triangular matrix satisfying

$$|W| \leq 1, \qquad |W_{ii}| = 1.$$

The lower triangular matrix L in Gaussian elimination with partial pivoting satisfies the same properties as W, and it generally turns out that $\|L^{-1}\|$ is small, say, like $O(n)$. Although this does not prove anything, it does show that all these rare matrix events are closely related. The probability of pivot growth in Gaussian elimination with partial pivoting is closely related to the probabilities of Golub-I, Chan-I, and GKS-I failing.

For the above matrices W of order k a tight upper bound on $\|W^{-1}\|$ is [16], [28]

$$\|W^{-1}\| \leq \frac{1}{3}\sqrt{4^k + 6k - 1} \leq \sqrt{k}2^k, \qquad k > 1,$$

and, as illustrated in § 6, the Kahan matrix essentially achieves this bound.

8. Unification. After having discussed greedy algorithms for the solution of Problem-I

$$\max_{\Pi} \sigma_{\min}(R_{11}),$$

we now turn to greedy algorithms for Problem-II

$$\min_{\Pi} \sigma_{\max}(R_{22}).$$

Fortunately, a simple observation greatly reduces this task.

Section 3 explains why it suffices to solve Problem-I for triangular matrices $\bar{R}$ and to consider

$$\bar{R}\Pi = QR, \qquad R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}.$$

Suppose that $\bar{R}$ is nonsingular, invert both sides of the above equation,

$$\Pi^T \bar{R}^{-1} = \begin{pmatrix} R_{11}^{-1} & -R_{11}^{-1} R_{12} R_{22}^{-1} \\ 0 & R_{22}^{-1} \end{pmatrix} Q^T,$$

and take transposes on both sides

$$\bar{R}^{-T}\Pi = Q \begin{pmatrix} R_{11}^{-T} & 0 \\ -R_{22}^{-T} R_{12}^T R_{11}^{-T} & R_{22}^{-T} \end{pmatrix}.$$

Now Problem-II can be formulated as

$$\begin{aligned} \min_{\Pi} \sigma_{\max}(R_{22}) &= \min_{\Pi} \frac{1}{\sigma_{\min}(R_{22}^{-1})} \\ &= \frac{1}{\max_{\Pi} \sigma_{\min}(R_{22}^{-1})} \\ &= \frac{1}{\max_{\Pi} \sigma_{\min}(R_{22}^{-T})}. \end{aligned}$$

Hence solving Problem-II is equivalent to solving Problem-I for the inverse. We call this *the unification principle* as it lets us unify the algorithms and analyses of Problem-I and Problem-II.

Applying a Type-I algorithm to the inverse gives

$$\bar{R}^{-T}\bar{\Pi} = \bar{Q}\begin{pmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{pmatrix},$$

where P_{11} is an upper triangular matrix of order $n-k$, P_{22} is an upper triangular matrix of order k, and (hopefully) $\sigma_{\min}(P_{11}) \approx \sigma_{n-k}(\bar{R}^{-T})$. Hence we need to make some adjustments as P_{11} should correspond to R_{22}^{-T}, which is lower triangular. Moreover, P_{11} should really have been the lower right block.

The necessary adjustments are achieved by a sequence of permutations, which can be accumulated in Q and Π. First permute the two block columns and the two block rows,

$$\begin{pmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{pmatrix} \to \begin{pmatrix} P_{12} & P_{11} \\ P_{22} & 0 \end{pmatrix} \to \begin{pmatrix} P_{22} & 0 \\ P_{12} & P_{11} \end{pmatrix}.$$

Then reverse the ordering of the columns and of the rows in P_{11} and P_{22} separately. This is accomplished by means of permutation matrices J_p of order p that have ones on the antidiagonal,

$$\begin{pmatrix} P_{22} & 0 \\ P_{12} & P_{11} \end{pmatrix} \to \begin{pmatrix} J_k P_{22} J_k & 0 \\ J_{n-k} P_{12} J_k & J_{n-k} P_{11} J_{n-k} \end{pmatrix}.$$

Now the resulting matrix has the desired form; it is lower triangular with P_{11} in the lower right corner.

Therefore, the postprocessing step consisting of the above permutations proves that applying a Type-I algorithm to the rows of the inverse amounts to executing a Type-II algorithm. In fact, we call such an algorithm the *Type-II version* of the *Type-I algorithm*. This notion is completely symmetric with respect to the two types, as one can equally well construct a *Type-I version* of a *Type-II algorithm* to solve Problem-I.

Unification principle. Running a Type-I algorithm on the rows of the inverse of the matrix yields a Type-II algorithm.

9. Type-II greedy algorithms.

In this section we illustrate the unification principle by exhibiting the Type-II version of algorithm Golub-I, and by proving that algorithm GKS-I also solves Problem-II.

We use the name Stewart-II for the Type-II version of algorithm Golub-I, as it was first proposed in [33], though not quite in the form in which we are presenting it.

```
ALGORITHM STEWART-II
$R^{(0)} = R$
For $l = 0$ to $n - k - 1$ do
    Set
        $$\begin{array}{c|cc} & n-l & l \\ \hline n-l & A & B \\ l & & C \end{array} = R^{(l)},$$
    1. Find the next column $j$ of $R^{(l)}$ such that
        $$\max_{1\le i\le n-l} \|e_i^T A^{-1}\| = \|e_j^T A^{-1}\|.$$
    2. Exchange columns $n-l$ and $j$ of $R^{(l)}$, and retriangularise it
       from the left with orthogonal transformations to get $R^{(l+1)}$.
```

Clearly, algorithm Stewart-II obtains the right ordering of the columns by sending the selected columns to the right end of the matrix. In all other matters it is completely equivalent to running Golub-I on the rows of the inverse.

A few clarifying remarks may be in order. Just because a Type-II version of an algorithm can be constructed by applying a Type-I algorithm to the rows of the inverse of the matrix, this does not mean that is also how it should be implemented. There may very well be a way to reformulate the Type-II version so that it avoids explicit dealings with inverses.

Furthermore, it is important to realise that a Type-I algorithm and its Type-II version, in general, come up with different column permutations; and that solving Problem-I does not entail solving Problem-II. All the unification principle says is that if there is an algorithm for solving Problem-I, then a simple modification will give an algorithm for solving Problem-II and vice versa.

There is another advantage of the unification principle. It allows us to carry over the analyses and worst-case examples for a Type-I algorithm, with suitable modifications, to its Type-II version and vice versa. A few examples follow.

In § 6 we explained that the lower bounds for the singular value estimates from algorithms Golub-I, Chan-I, and GKS-I can be cast in the form

$$\sigma_{\min}(R_{11}) \ge \frac{\sigma_k(M)}{n\|W^{-1}\|},$$

where W are triangular matrices satisfying

$$|W| \le 1, \qquad |W_{ii}| = 1.$$

The unification principle therefore admits upper bounds for the singular value estimates from the Type-II versions of Golub-I, Chan-I, and GKS-I of the form

$$\sigma_{\max}(R_{22}) \le \sigma_{k+1}(M) n \|W^{-1}\|,$$

where, again, W are triangular matrices satisfying

$$|W| \le 1, \qquad |W_{ii}| = 1.$$

As for other existing Type-II algorithms, the Type-II version of Chan-I, which we call Chan-II, was published apparently independently in [22], [9], [17]. The Type-II version of GKS-I was first published in [20] and will be called GKS-II. The Type-II version of Foster-I, which we refer to as Foster-II, was first published in [17]. The detailed exposition of Foster-II in [17] also serves to illuminate our algorithm Foster-I.

We still owe a justification of our claim that GKS-I also solves Problem-II [20], [21]. Let $R = U\Sigma V^T$ be the SVD of the final triangular matrix R, where

$$\begin{array}{cc} & \begin{array}{cc} k & n-k \end{array} \\ \begin{array}{c} k \\ n-k \end{array} & \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix} \end{array} = V, \qquad \Sigma = \begin{pmatrix} \Sigma_1 & \\ & \Sigma_2 \end{pmatrix}.$$

This implies

$$\frac{1}{\sigma_{\min}(R_{11})\|V_{11}^{-1}\|} = \frac{\|R_{11}^{-1}\|}{\|V_{11}^{-1}\|} \leq \|V_{11}^T R_{11}^{-1}\| \leq \|\Sigma_1^{-1}\| = \frac{1}{\sigma_k(M)},$$

and

$$\frac{1}{\|R_{22}\|} = \sigma_{\min}(R_{22}^{-1}) \geq \sigma_{\min}(\Sigma_2^{-1}V_{22}) \geq \sigma_{\min}(\Sigma_2^{-1})\sigma_{\min}(V_{22}) = \frac{1}{\sigma_{k+1}(M)\|V_{22}^{-1}\|},$$

so

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{\|V_{11}^{-1}\|}, \qquad \|R_{22}\| \leq \sigma_{k+1}(M)\|V_{22}^{-1}\|.$$

According to the CS decomposition, §2.6 in [21],

$$\|V_{11}^{-1}\| = \|V_{22}^{-1}\|.$$

Since GKS-I attempts to keep $\|V_{11}^{-1}\|$ small, it therefore automatically also tries to keep $\|V_{22}^{-1}\|$ small. Therefore GKS-I solves both, Problem-I and Problem-II.

At last we demonstrate how the worst-case example of a Type-I algorithm can be converted to a worst-case example for its Type-II version. Section 6 illustrates that the Kahan matrix

$$K_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & s & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & s^{n-1} \end{pmatrix} \begin{pmatrix} 1 & -c & \cdots & -c \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -c \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

represents a worst case for algorithms Greedy-I, Greedy-I.1, Greedy-I.2, and Golub-I. It follows from the unification principle that the modified Kahan matrix whose inverse is given by

$$\bar{K}_n^{-1} = \begin{pmatrix} 1 & -c & \cdots & -c \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -c \\ 0 & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} s^{n-1} & 0 & \cdots & 0 \\ 0 & s^{n-2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix},$$

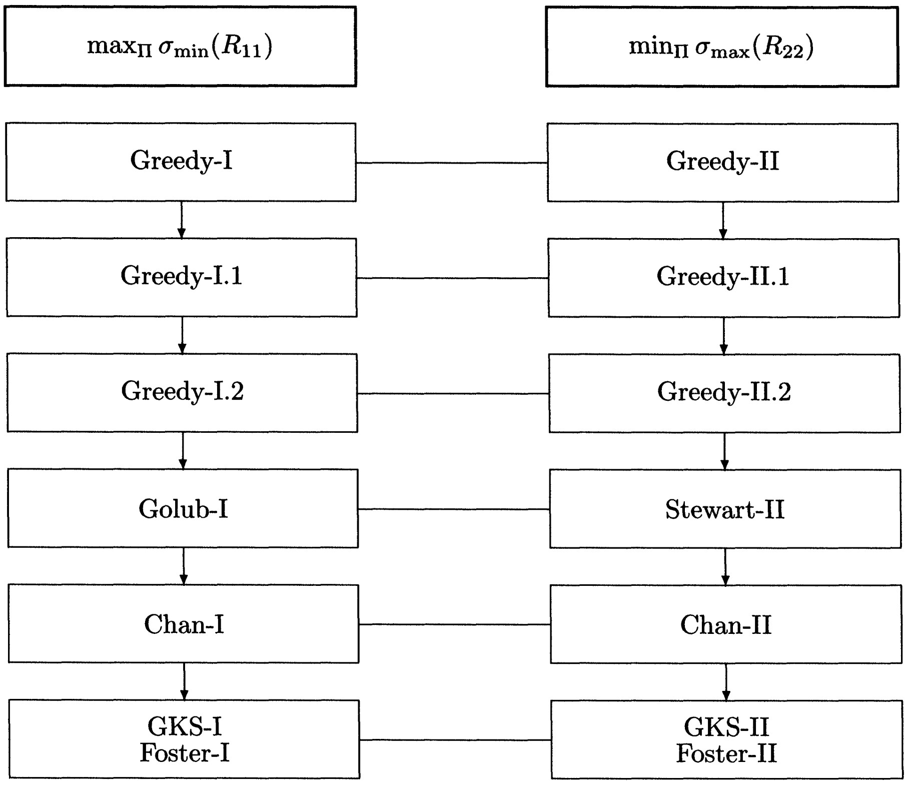

FIG. 10.1. *The greedy algorithms.*

where $c^2 + s^2 = 1$, represents a worst case for algorithms Greedy-II, Greedy-II.1, Greedy-II.2, and Stewart-II, the Type-II versions of the respective Type-I algorithms.

10. Summary.

This ends our presentation of the existing RRQR algorithms. We gave three mathematical problems that we called rank-revealing problems,

Problem-I: $\max_\Pi \sigma_{\min}(R_{11})$,
Problem-II: $\min_\Pi \sigma_{\max}(R_{22})$,

and the third was to solve Problem-I and Problem-II simultaneously. We then exhibited a sequence of successively less greedy algorithms to solve Problem-I. By means of the unification principle, we demonstrated the existence of Type-II versions of these algorithms, which are also greedy but solve Problem-II instead. Figure 10.1 illustrates the two parallel hierarchies made up from the Type-I and Type-II algorithms, where the corresponding Type-I and Type-II algorithms are next to each other, and each algorithm is less greedy than the one above it. Each of the existing RRQR algorithms has a place in this hierarchy. Examples of exponential failure of these greedy algorithms are provided by the Kahan and modified Kahan matrices.

We have ignored the greedy algorithms based on condition number estimators for triangular matrices, e.g., [2]–[5], [25], [34], because their behaviour depends very much on the particular condition number estimator.

The worst-case bounds

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{n2^k}, \qquad \|R_{22}\| \leq \sigma_{k+1}(M)n2^{n-k}$$

reveal that Type-I greedy algorithms work pretty well for small k, while Type-II greedy algorithms work well when k is close to n. This prompts the question whether a Type-I and a Type-II greedy algorithm can be combined into a single algorithm that works all the time. The answer is given in the next section.

11. Overview of the hybrid algorithms. In this section we present algorithms Hybrid-I and Hybrid-II. They are guaranteed to solve Problem-I and Problem-II, respectively. We also present algorithm Hybrid-III. It is guaranteed to solve both Problem-I and Problem-II simultaneously.

In particular, Algorithm Hybrid-I guarantees that

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{\sqrt{k(n-k+1)}},$$
$$\sigma_{\max}(R_{22}) \leq \sigma_{\min}(R_{11})\sqrt{k(n-k+1)}.$$

Note that Hybrid-I does *not* solve Problem-II. According to the unification principle, the Type-II version of Hybrid-I, which we call Hybrid-II, must guarantee that

$$\sigma_{\max}(R_{22}) \leq \sigma_{k+1}(M)\sqrt{(k+1)(n-k)},$$
$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_{\max}(R_{22})}{\sqrt{(k+1)(n-k)}}.$$

Note again that Hybrid-II does not solve Problem-I. Hybrid-III does solve both Problem-I and Problem-II simultaneously, and it guarantees that

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{\sqrt{k(n-k+1)}},$$
$$\sigma_{\max}(R_{22}) \leq \sigma_{k+1}(M)\sqrt{(k+1)(n-k)}.$$

Of course, the brute force algorithm, which tries every combination of columns, also solves these problems, but its operation count is combinatorial. What about the hybrid algorithms? Unfortunately, we lack a complete analysis of the worst-case operation count of the hybrid algorithms, although we believe that it may be combinatorial as well. However, preliminary experimental results in §15 demonstrate that the hybrid algorithms are rather efficient in practice.

As in the previous sections we assume that k is given. Although this may not be a realistic assumption, a proper choice of k depends very much on the problem to be solved, and we refer to [20], [33] for the discussion of this issue.

12. Algorithm Hybrid-I. The algorithm Hybrid-I is a combination of Golub-I and Stewart-II, though in a practical implementation one may want to replace Stewart-II by Chan-II.

The obvious strategy of running Stewart-II after Golub-I is not guaranteed to solve Problem-I because Golub-I and Stewart-II almost always produce a unique ordering of columns, so the result of this strategy would merely equal the result of Stewart-II.

Instead, our idea is to alternate between Golub-I and Stewart-II and to let each work on a different part of the matrix: Stewart-II works on the (1,1) block of order k, and Golub-I works on the (2,2) block of order $n-k+1$ of the matrix. Suppose Golub-I has picked the best column from the (2,2) block and put it in position k. Stewart-II then determines whether the kth column is indeed a good column. If not, it puts the worst column from the (1,1) block into position k. Now it is again Golub-I's turn to put the best column from the (2,2) block in position k. This process continues until Golub-I and Stewart-II agree on the kth column. To understand the resulting algorithm Hybrid-I, we briefly review Golub-I and Stewart-II.

Golub-I is good at approximating the largest singular value of $M\Pi = QR$. In its first iteration it finds the "most linearly independent" column of R, i.e., the column with largest norm. Suppose we permute this column to the first position and retriangularise the matrix. Then the first column $r_{11}e_1$ of the resulting triangular matrix approximates the largest singular value of M,

$$|r_{11}| \leq \sigma_{\max}(M) \leq \sqrt{n}|r_{11}|.$$

Since Stewart-II is the Type-II version of Golub-I, it is good at approximating the largest singular value of M^{-1} by finding the most linearly independent row of R^{-1}. Suppose we permute this row of R^{-1} to the last position and retriangularise the inverse to get the triangular matrix $\bar{R}^{-1} = \bar{\Pi}R^{-1}\bar{Q}$. Then the last column $r_{nn}^{-1}e_n$ of $\bar{R}^{-1}$ approximates the largest singular value of M^{-1},

$$|r_{nn}^{-1}| \leq \sigma_{\max}(M^{-1}) \leq \sqrt{n}|r_{nn}^{-1}|.$$

But since $\bar{R}^{-1}$ is triangular, r_{nn} is the trailing diagonal element of $\bar{R}$ and it approximates the smallest singular value of M,

$$\sigma_{\min}(M) \leq |r_{nn}| \leq \sqrt{n}\sigma_{\min}(M).$$

We illustrate Hybrid-I on a 5×5 example, where $k = 3$ and the symbol "x" represents nonzero matrix elements. First we run Golub-I on the $(2,2)$ block of order $n-k+1$ so that diagonal element r_{kk} has largest norm among all columns of the $(2,2)$ block

$$\begin{array}{cc|ccc}
 & k-1 & k & & \\
 & & & & \\
x & x & x & x & x \\
 & x & x & x & x \\
\hline
 & & r_{kk} & x & x \\
 & & & x & x \\
 & & & & x
\end{array}$$

Now we enlarge the $(1,1)$ block from order $k-1$ to order k so that the kth diagonal element can transfer information between the two algorithms. Then we run Stewart-II on the $(1,1)$ block of order k so that the (modified) diagonal element $\bar{r}_{kk}$ has smallest norm.

$$\begin{array}{ccc|cc} & & k & k+1 & \\ x & x & x & x & x \\ & x & x & x & x \\ & & \bar{r}_{kk} & \otimes & \otimes \\ \hline & & & x & x \\ & & & & x \end{array}$$

A run of Golub-I followed by Stewart-II constitutes one iteration. The circled elements in the $(1,2)$ block are modified by orthogonal rotations from the left due to retriangularisation in Stewart-II. They are part of the $(2,2)$ block for the subsequent run of Golub-I and illustrate how one algorithm changes the part of the matrix associated with the other algorithm. The $(1,1)$ block input to Stewart-II undergoes similar changes in column k due to column permutations during Golub-I.

```
ALGORITHM HYBRID-I(k)
$\bar{R}^{(0)} = R$, $l = 0$
**Repeat**
  $l = l + 1$, *permuted* $= 0$
  Set
      $$\bar{R}^{(l)} = \begin{pmatrix} \bar{A} & \bar{B} \\ & \bar{C} \end{pmatrix},$$
  where $\bar{A}$ is of order $k-1$ and $\bar{C}$ is of order $n-k+1$.
  Golub-I:
  1. Find the column $k+j-1$ of $\bar{R}^{(l)}$ such that $\|\bar{C}e_j\| = \max_{1\le i\le n-k+1} \|\bar{C}e_i\|$
  2. **If** $\|\bar{C}e_1\| < \|\bar{C}e_j\|$ **then**
       *permuted* $= 1$
       Exchange columns $k$ and $k+j-1$ of $\bar{R}^{(l)}$
       Retriangularise it from the left with orthogonal transformations to
       get
           $$R^{(l)} = \begin{pmatrix} A & B \\ & C \end{pmatrix},$$
     where $A$ is of order $k$ and $C$ is of order $n-k$.
  Stewart-II:
  3. Find the column $j$ of $R^{(l)}$ such that $\|e_j^T A^{-1}\| = \max_{1\le i\le k} \|e_i^T A^{-1}\|$
  4. **If** $\|e_k^T A^{-1}\| < \|e_j^T A^{-1}\|$ **then**
       *permuted* $= 1$
       Exchange columns $j$ and $k$ of $R^{(l)}$
       Retriangularise it from the left with orthogonal transformations to
       get $\bar{R}^{(l+1)}$.
**until** not *permuted*
```

The final matrix is

$$R = \begin{pmatrix} \bar{R}_{11} & \bar{R}_{12} \\ & \bar{R}_{22} \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix},$$

where $\bar{R}_{11}$ is of order $k-1$ and R_{11} is of order k.

The two if statements assure that permutations are performed only in case of a strict inequality but not in case of a tie.

We proceed with an analysis of Hybrid-I because it is not clear that Hybrid-I eventually halts, and that it indeed increases $\sigma_{\min}(R_{11})$. We first show that *if* Hybrid-I halts then

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{\sqrt{k(n-k+1)}},$$
$$\sigma_{\max}(R_{22}) \leq \sigma_{\min}(R_{11})\sqrt{k(n-k+1)}.$$

Suppose Hybrid-I halts. Then Golub-I applied to $\bar{R}_{22}$ does not change the first column $r_{kk}e_1$ of $\bar{R}_{22}$, where r_{kk} is the kth diagonal element of R. Hence

$$|r_{kk}| \geq \frac{\sigma_{\max}(\bar{R}_{22})}{\sqrt{n-k+1}} \geq \frac{\sigma_{\max}(R_{22})}{\sqrt{n-k+1}}$$

since R_{22} is a submatrix of $\bar{R}_{22}$. Moreover, Stewart-II applied to R_{11} does not change the last row $r_{kk}e_k^T$ of R_{11}, and

$$|r_{kk}| \leq \sigma_{\min}(R_{11})\sqrt{k}.$$

Combining the two inequalities for r_{kk} gives the first desired bound

$$\sigma_{\max}(R_{22}) \leq \sigma_{\min}(R_{11})\sqrt{k(n-k+1)}.$$

Applying the interlacing property ($I2$) to $\bar{R}_{22}$,

$$|r_{kk}| \geq \frac{\sigma_{\max}(\bar{R}_{22})}{\sqrt{n-k+1}} \geq \frac{\sigma_k(M)}{\sqrt{n-k+1}},$$

and combining the previous two inequalities yields the second desired bound

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_k(M)}{\sqrt{k(n-k+1)}}.$$

Thus, if Hybrid-I halts, it solves Problem-I.

To prove that Hybrid-I indeed halts, we make use of the fact that columns are permuted only in case of strict inequalities. The basic idea is to show that $|\det(A)|$ is a strictly increasing function during the algorithm. Remember that A is the leading principal submatrix of order k. Since $|\det(A)|$ is unique for any given column ordering, no column ordering repeats if $|\det(A)|$ is strictly increasing. As there are only a finite number of column orderings, Hybrid-I must eventually halt.

It remains to show that $|\det(A)|$ is strictly increasing during Hybrid-I. By assumption from § 2 we have that $\sigma_k(M) > 0$. So we can assume that our initial ordering of columns is such that $|\det(A)| > 0$. Stewart-II does not change $\det(A)$ because $|\det(A)|$ is invariant under application of orthogonal transformations from the left to $(A \quad B)$ and to C; and under permutation of the columns of A and of the columns of $\binom{B}{C}$. To see how Golub-I affects $\det(A)$, we divide Golub-I into two phases: the first phase keeps $|\det(A)|$ invariant, while the second one may change $|\det(A)|$. Accordingly, we identify and separate the first column $(b^T \quad \gamma e_1^T)^T$ of the matrix affected by Golub-I,

$$\begin{pmatrix} \bar{B} \\ \bar{C} \end{pmatrix} = \begin{pmatrix} b & \tilde{B} \\ \gamma e_1 & \tilde{C} \end{pmatrix}.$$

In the first phase the columns of $\binom{\tilde{B}}{\tilde{C}}$ are permuted, so that the first column of the permuted $\tilde{C}$ has largest norm among all columns of $\tilde{C}$, and then the permuted $\tilde{C}$ is retriangularised to give $\dot{C}$. In the second phase, the relevant matrix elements are γ and the nonzero elements α and β of $\dot{C}e_1$,

$$\begin{array}{c} \\ \\ k \\ k+1 \\ \\ \end{array}\begin{array}{c} \begin{array}{cccc} & k & k+1 & \end{array} \\ \begin{pmatrix} * & * & * & * \\ & \gamma & \alpha & * \\ & & \beta & * \\ & & & * \end{pmatrix} \end{array}.$$

Golub-I permutes columns k and $k+1$ if $\gamma^2 < \alpha^2 + \beta^2$, in which case the matrix becomes

$$\begin{array}{c} \\ \\ k \\ k+1 \\ \\ \end{array}\begin{array}{c} \begin{array}{cccc} & k & k+1 & \end{array} \\ \begin{pmatrix} * & * & * & * \\ & \alpha & \gamma & * \\ & \beta & & * \\ & & & * \end{pmatrix} \end{array}.$$

The matrix is retriangularised by eliminating β via a Givens rotations from the left, which affects only rows k and $k+1$ and results in

$$\begin{array}{c} \\ \\ k \\ k+1 \\ \\ \end{array}\begin{array}{c} \begin{array}{cccc} & k & k+1 & \end{array} \\ \begin{pmatrix} * & * & * & * \\ & \sqrt{\alpha^2+\beta^2} & x & * \\ & & x & * \\ & & & * \end{pmatrix} \end{array},$$

where the two x represent new numbers. Other than the kth diagonal element, which changed from γ to $\sqrt{\alpha^2+\beta^2}$, no diagonal element of A changed. But the kth diagonal element underwent a strict increase in magnitude since $|\gamma| < \sqrt{\alpha^2+\beta^2}$, and therefore $|\det(A)|$ is a strictly increasing function during Hybrid-I. Consequently, algorithm Hybrid-I must halt.

Section 15 presents some numerical experiments on the running time of Hybrid-I.

13. Algorithm Hybrid-II. In this section we present algorithm Hybrid-II, the Type-II version of Hybrid-I. According to the unification principle, Hybrid-II guarantees that

$$\sigma_{\max}(R_{22}) \le \sigma_{k+1}(M)\sqrt{(k+1)(n-k)}$$
$$\sigma_{\min}(R_{11}) \ge \frac{\sigma_{\max}(R_{22})}{\sqrt{(k+1)(n-k)}}$$

From the interlacing properties $(I1)$ and $(I2)$ it follows that Hybrid-I(k+1) guarantees the same bounds as Hybrid-II(k). Thus, one way to implement Hybrid-II(k) is via Hybrid-I(k+1).

```
ALGORITHM HYBRID-II(k)
Hybrid-I(k+1)
```

Although nonsingularity is needed for the application of the unification principle, this implementation of Hybrid-II(k) has the advantage of doing without the requirement that the matrix be nonsingular. However, to reduce the proof that Hybrid-I(k+1) halts to the proof for Hybrid-I(k) requires $\sigma_{k+1}(M) > 0$, which may not be true. Our proof that Hybrid-II halts does so without this assumption, and it also enables us to design the more accurate algorithm Hybrid-III by providing additional insight into the nature of the problem.

The basic idea of the proof is again to demonstrate the strict increase of the determinant of the leading $k \times k$ principal submatrix during Hybrid-II. Unfortunately, we cannot prove that the absolute value of the determinant of the leading $(k+1) \times (k+1)$ block is strictly increasing because that would necessitate the assumption $\sigma_{k+1}(M) > 0$. To facilitate understanding of the proof, we first describe in more detail the implementation of Hybrid-II(k) based on Hybrid-I(k+1).

```
ALGORITHM HYBRID-II(k)
$R^{(0)} = R$, $l = 0$
**Repeat**
  $l = l + 1$, *permuted* $= 0$
  Set
      $R^{(l)} = \begin{pmatrix} A & B \\ & C \end{pmatrix}$,
  where $A$ is of order $k$ and $C$ is of order $n - k$.
    Golub-I:
   1. Find the column $k + j$ of $R^{(l)}$ such that $\|Ce_j\| = \max_{1 \le i \le n-k} \|Ce_i\|$
   2. **If** $\|Ce_1\| < \|Ce_j\|$ **then**
          *permuted* $= 1$
          Exchange columns $k + 1$ and $k + j$ of $R^{(l)}$
          Retriangularise it from the left with orthogonal transformations to
          get
              $\hat{R}^{(l)} = \begin{pmatrix} \hat{A} & \hat{B} \\ & \hat{C} \end{pmatrix}$,
      where $\hat{A}$ is of order $k + 1$ and $\hat{C}$ is of order $n - k - 1$.
    Stewart-II:
```

```
3. Find the column $j$ of $\hat{R}^{(l)}$ such that $\|e_j^T\hat{A}^{-1}\| = \max_{1\le i\le k+1}\|e_i^T\hat{A}^{-1}\|$
4. **If** $\|e_{k+1}^T\hat{A}^{-1}\| < \|e_j^T\hat{A}^{-1}\|$ **then**
       *permuted* $= 1$
       Exchange columns $j$ and $k+1$ of $\hat{R}^{(l)}$
       Retriangularise it from the left with orthogonal transformations to
       get $R^{(l+1)}$.
**until** not *permuted*
The final matrix is
```

$$R = \begin{pmatrix} \hat{R}_{11} & \hat{R}_{12} \\ & \hat{R}_{22} \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix},$$

where $\hat{R}_{11}$ is of order $k+1$ and R_{11} is of order k.

As in Hybrid-I, the two if statements assure that permutations are performed only in case of a strict inequality but not in case of a tie.

Again, as we had assumed that $\sigma_k(M) > 0$, we can assume that our initial ordering of columns is such that $|\det(A)| > 0$. Although this proof is based on Hybrid-I(k+1) it is slightly different from the proof we gave for Hybrid-I(k) because now we are focusing on column k instead of column $k+1$. Clearly, Golub-I does not affect $|\det(A)|$ but Stewart-II does. We divide Stewart-II into two phases. The first phase keeps $|\det(A)|$ invariant while the second phase may change $|\det(A)|$. Accordingly, we identify and separate the last column $(a^T \quad \alpha e_1^T)^T$ of the matrix affected by Stewart-II,

$$\hat{A} = \begin{pmatrix} \tilde{A} & a \\ & \alpha \end{pmatrix}.$$

In the first phase the columns of $\tilde{A}$ are permuted, so that the last row of $\tilde{A}^{-1}$ has largest norm among all rows of $\tilde{A}^{-1}$, and then the permuted $\tilde{A}^{-1}$ is retriangularised from the right to give $\dot{A}^{-1}$. In the second phase the relevant matrix elements are α, the element β above it, and the trailing nonzero γ of $e_k^T\dot{A}$,

$$\begin{array}{c} \\ k \\ k+1 \end{array}\begin{array}{c} \begin{array}{ccc} & k & k+1 \end{array} \\ \begin{pmatrix} * & * & * \\ & \gamma & \beta \\ & & \alpha \end{pmatrix} \end{array}.$$

Since Stewart-II is the Type-II version of Golub-I, it permutes to the last position the column corresponding to the row with largest norm in the inverse, whose relevant elements are

$$\begin{array}{c} \\ k \\ k+1 \end{array}\begin{array}{c} \begin{array}{ccc} & k & k+1 \end{array} \\ \begin{pmatrix} * & * & * \\ & \frac{1}{\gamma} & -\frac{\beta}{\gamma\alpha} \\ & & \frac{1}{\alpha} \end{pmatrix} \end{array}.$$

Stewart-II permutes columns k and $k+1$ if

$$\frac{1}{\gamma^2} + \frac{\beta^2}{\gamma^2\alpha^2} > \frac{1}{\alpha^2} \qquad \text{or} \qquad \gamma^2 < \alpha^2 + \beta^2.$$

But this is the same situation as in Hybrid-I(k), and it follows that the kth diagonal element of R_{11} changes from γ to $\sqrt{\alpha^2+\beta^2}$ while all other diagonal elements of R_{11} remain unchanged. As we just proved that $|\gamma| < \sqrt{\alpha^2+\beta^2}$, $|\det(A)|$ is strictly increasing during Hybrid-II.

Because we were able to prove that Hybrid-II halts, requiring only that $\sigma_k(M) > 0$, we can show directly that Hybrid-II(k) satisfies

$$\sigma_{\max}(R_{22}) \le \sigma_{k+1}(M)\sqrt{(k+1)(n-k)},$$
$$\sigma_{\min}(R_{11}) \ge \frac{\sigma_{\max}(R_{22})}{\sqrt{(k+1)(n-k)}}.$$

The proof is similar to the one that establishes the bounds for Hybrid-I.

14. Algorithm Hybrid-III.

Our last new algorithm is Hybrid-III, which satisfies

$$\sigma_{\min}(R_{11}) \ge \frac{\sigma_k(M)}{\sqrt{k(n-k+1)}},$$
$$\sigma_{\max}(R_{22}) \le \sigma_{k+1}(M)\sqrt{(k+1)(n-k)}.$$

There are several implementations of Hybrid-III. We present the one that is simplest to describe. This implementation, motivated by the fact that the determinant of the leading principal submatrix of order k is a strictly increasing function in both Hybrid-I and Hybrid-II, consists of running Hybrid-I and Hybrid-II in alternation until no more permutations take place.

```
ALGORITHM HYBRID-III(k)
Repeat
    Hybrid-I(k)
    Hybrid-II(k)
Until no permutations occur
```

The halting argument for Hybrid-III follows easily from the halting of Hybrid-I and Hybrid-II. We had shown earlier that during Hybrid-I and Hybrid-II, the determinant of the leading $k \times k$ principal submatrix is a strictly increasing function. So it must be true during Hybrid-III also. Hence Hybrid-III halts.

When Hybrid-III has halted, both Hybrid-I and Hybrid-II do not cause any further permutations in the matrix. Therefore the bounds guaranteed by Hybrid-I and Hybrid-II must hold simultaneously now. That is

$$\sigma_{\min}(R_{11}) \ge \frac{\sigma_k(M)}{\sqrt{k(n-k+1)}},$$
$$\sigma_{\max}(R_{22}) \le \sigma_{k+1}(M)\sqrt{(k+1)(n-k)}$$

must be true.

Because we do not know whether the solution of Problem-I also implies the solution of Problem-II or vice versa, it is not clear whether the output of algorithm Hybrid-I also satisfies the bounds that govern the output from Hybrid-II. In particular we are therefore not able to compare the operation counts of Hybrid-III with those of Hybrid-I or Hybrid-II.

REMARK 14.1. As we mentioned earlier, it is more practical to replace algorithm Stewart-II in the hybrid algorithms with algorithm Chan-II.

Moreover, there are other ways of implementing algorithm Hybrid-III. For example, replacing the Stewart-II part of Hybrid-I with a simpler algorithm results in an Hybrid-III algorithm: Instead of moving the most linearly dependent column to the kth position, in turn permute *every* one of the leading k columns to the kth position. Obviously, this algorithm solves Problem-I and Problem-II simultaneously. The corresponding Type-II version, which involves replacing the Golub-I part of Hybrid-II, also solves Problem-I and Problem-II simultaneously according to the unification principle. This idea has been taken up in [32].

However, we believe that the original version of Hybrid-III described at the beginning of this section is more efficient in practice than the latter two (provided it is properly implemented with good condition number estimators in place of Golub-I and Stewart-II, and run as a postprocessor to either Golub-I or Chan-II).

15. Some numerical experiments.

Although we have demonstrated that the three hybrid algorithms halt in exact arithmetic, we know very little about their worst-case running times. In this section we present some preliminary numerical results for Hybrid-I, which also apply to Hybrid-II and Hybrid-III as the implementations for the latter two algorithms can be based on Hybrid-I. In practice, the hybrid algorithms are best run as postprocessors to the more efficient greedy algorithms, like Golub-I or Chan-II.

In the experiments to follow, we counted the number of iterations in Hybrid-I when it is run after Golub-I. To prevent cycling in the algorithm due to roundoff errors, we carried out permutations only if the pivot increased by more than $n^2\epsilon$, where ϵ is the machine precision. To estimate the dependence of the running time of Hybrid-I on the matrix size n and the separation of the singular values $\sigma_k(M)/\sigma_{k+1}(M)$, we generated fifty random matrices of size fifty, to which we applied Hybrid-I with $k = 37$. Then we multiplied the last $n - k$ singular values of these fifty matrices by 0.1 to increase the separation between the singular values but did not change the singular vectors. Hybrid-I was applied to these fifty new matrices. The same process was repeated on one hundred random matrices of size one hundred with $k = 75$. Table 15.1 shows how many times Hybrid-I required a certain number of iterations. Hybrid-I seems to require fewer iterations when the gap between $\sigma_k(M)$ and $\sigma_{k+1}(M)$ is larger, and—in these experiments, at least—the number of iterations does not deteriorate too much with increase in matrix size.

16. Conclusion.

In this paper we proposed three optimisation problems which we called rank-revealing QR (RRQR) problems. We presented a unifying treatment of the existing algorithms by placing them in a hierarchy of greedy algorithms. Finally, we presented three new hybrid algorithms for solving the three rank-revealing problems. Unfortunately, we were not able to estimate the worst-case running time of the hybrid algorithms.

Most of the discussion for the RRQR factorisations can be extended in a simple manner to rank-revealing LU (RRLU) factorisations [8], [27] by replacing orthogonal transformations with elementary Gauss transformations and row interchanges for partial pivoting. Partial pivoting prevents the ill conditioning of the Gauss transformations. Compared to RRQR factorisations, the bounds for RRLU factorisations

TABLE 15.1
*Hybrid-*I *run-time estimate.*

Matrix size →	50	50	100	100
k →	37	37	75	75
Avg($\frac{\sigma_k(M)}{\sigma_{k+1}(M)}$) →	1.0804	10.804	1.0406	10.406
No. of iter. ↓	↓ no. of occurrences ↓			
1	21	25	16	45
2	7	8	3	8
3	5	3	9	9
4	5	4	15	15
5	4	4	11	6
6	1	5	13	6
7	0	0	6	1
8	1	0	7	5
9	4	1	4	1
10	1	0	7	1
11	1	0	1	1
12	0	0	0	2
13	0	0	4	0
14	0	0	0	0
15	0	0	0	0
16	0	0	1	0
17	0	0	1	0
18	0	0	1	0
19	0	0	1	0
Total	50	50	100	100

are generally worse and, due to pivoting and the resulting fill-in, their operation are counts higher. It is not clear to us which applications would benefit from RRLU factorisations.

In a subsequent paper [13] we show that very naturally the hybrid algorithms give rise to new algorithms for computing the URV decomposition [34]–[36] and also to a new divide-and-conquer algorithm for the SVD. In fact, using a preceding RRQR algorithm to accelerate the computation of eigenvalues or singular values is not new, see for instance [15], [24], [38] where a Jacobi method is preceded by QR with column pivoting.

In this paper, we present only one algorithm for each of the three optimisation problems, but one can easily design other kinds of approximate and exact algorithms. Our motivation for the three hybrid algorithms was to perform column interchanges based on what we believed would result in a high rate of convergence. But sometimes one may want to trade off number of column exchanges for maintainance of sparsity [3], [4], [30] or minimisation of communication costs.

The ideas presented in this paper may aid in the design of special-purpose algorithms. Instead of choosing the best two columns to exchange, one could compromise and choose a column exchange that maintains sparsity or keeps communication costs low, while still ensuring that the determinant of the leading $k \times k$ principal submatrix increases strictly so that the algorithm halts. We hope that the ideas presented in this paper prove helpful in developing algorithms for such problems.

Acknowledgments. We thank Françoise Chatelin and Axel Ruhe for helpful discussions and for bringing reference [16] to our attention.

REFERENCES

[1] G. ADAMS, M. GRIFFIN, AND G. STEWART, *Direction-of-arrival estimation using the rank-revealing URV decomposition*, International Conference on Acoustics, Speech and Signal Processing 91, 1991, pp. 1385–1388.

[2] C. BISCHOF, *Incremental condition estimation*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 312–322.

[3] C. BISCHOF AND P. HANSEN, *A block algorithm for computing rank-revealing QR-factorizations*, Numerical Algorithms, 2 (1992), pp. 371–392.

[4] ———, *Structure-preserving and rank-revealing QR-factorizations*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1332–1350.

[5] C. BISCHOF, D. PIERCE, AND J. LEWIS, *Incremental condition estimation for sparse matrices*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 644–659.

[6] C. BISCHOF AND G. SHROFF, *On updating signal subspaces*, IEEE Transactions on Signal Processing, 40 (1992), pp. 96–105.

[7] P. BUSINGER AND G. GOLUB, *Linear least squares solutions by Householder transformations*, Numer. Math., 7 (1965), pp. 269–276.

[8] T. CHAN, *On the existence and computation of LU-factorizations with small pivots*, Math. Comp, 42 (1984), pp. 535–547.

[9] ———, *Rank revealing QR factorizations*, Linear Algebra and its Applications, 88/89 (1987), pp. 67–82.

[10] T. CHAN AND P. HANSEN, *Computing truncated singular value decomposition least squares solutions by rank revealing QR factorizations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 519–530.

[11] ———, *Low-rank revealing QR factorizations*, Tech. Report 91-08, UCLA Comp. and App. Math., Los Angeles, 1991; Numer. Linear Algebra Appl., to appear.

[12] ———, *Some applications of the rank revealing QR factorization*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 727–741.

[13] S. CHANDRASEKARAN AND I. IPSEN, *Analysis of a QR algorithm for computing singular values*, Research Report 917, Department of Computer Science, Yale University, New Haven, CT, 1992.

[14] P. COMON AND G. GOLUB, *Tracking a few extreme singular values and vectors in signal processing*, Proc. IEEEE, 78 (1990), pp. 1327–1343.

[15] J. DEMMEL AND K. VESELIČ, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245.

[16] D. FADDEEV, V. KUBLANOVSKAYA, AND V. FADDEEVA, *Sur les systèmes linéaires algébraiques de matrices rectangulaires et mal-conditionnées*, Colloq. Internat. du C.N.R.S. Besançon 1966, No. 165 (1968), pp. 161–170.

[17] L. FOSTER, *Rank and null space calculations using matrix decomposition without column interchanges*, Linear Algebra Appl., 74 (1986), pp. 47–71.

[18] ———, *The probability of large diagonal elements in the QR factorization*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 531–544.

[19] G. GOLUB, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.

[20] G. GOLUB, V. KLEMA, AND G. STEWART, *Rank degeneracy and least squares problems*, Tech. Report STAN-CS-76-559, Computer Science Department, Stanford University, Stanford, CA, 1976.

[21] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, The Johns Hopkins Press, Baltimore, MD, 1989.

[22] W. GRAGG AND G. STEWART, *A stable variant of the secant method for solving nonlinear equations*, SIAM J. Numer. Anal., 13 (1976), pp. 889–903.

[23] P. HANSEN, *Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 503–518.

[24] V. HARI AND K. VESELIČ, *On Jacobi methods for singular value decompositions*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 741–754.

[25] N. HIGHAM, *A survey of condition number estimation for triangular matrices*, SIAM Rev., 29 (1987), pp. 575–596.

[26] H. HONG AND C. PAN, *The rank-revealing QR decomposition and SVD*, Math. Comp., 58 (1992), pp. 213–232.

[27] T. HWANG, W. LIN, AND E. YANG, *Rank-revealing LU-factorizations*, Linear Algebra Appl., 175 (1992), pp. 115–141.

[28] W. KAHAN, *Numerical linear algebra*, Canadian Math. Bull., 9 (1966), pp. 757–801.

[29] V. Kane, R. Ward, and G. Davis, *Assessment of linear dependencies in multivariate data*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 1022–1032.

[30] J. Lewis and D. Pierce, *Sparse rank revealing QR factorization*, Tech. Report MEA-TR-193, Boeing Computer Services, Seattle, WA, 1992.

[31] R. Mathias and G. W. Stewart, *A Block QR Algorithm and the Singular Value Decomposition*, Linear Algebra Appl., 182 (1993), pp. 91–100.

[32] C. Pan and P. Tang, *Bounds on singular values revealed by QR factorizations*, Preprint MCS-P332-1032, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1992.

[33] G. Stewart, *Rank degeneracy*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 403–413.

[34] ———, *Incremental condition calculation and column selection*, Tech. Report UMIACS-TR 90-87, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, 1990.

[35] ———, *An updating algorithm for subspace tracking*, IEEE Transactions on Signal Processing, SP-40 (1992), pp. 1535–1541.

[36] ———, *Updating a rank-revealing ULV decomposition*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 494–499.

[37] S. van Huffel and J. Vandewalle, *Subset selection using the total least squares approach in collinearity problems with errors in the variables*, Linear Algebra Appl., 88/89 (1987), pp. 695–714.

[38] K. Veseliĉ and V. Hari, *A note on a one-sided jacobi algorithm*, Numer. Math, 56 (1989), pp. 627–633.

[39] S. Wold, A. Ruhe, H. Wold, and W. Dunn, III, *The collinearity problem in linear regression. The partial least squares* (PLS) *approach to generalized inverses*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 735–743.