Communication Avoiding Rank Revealing QR Factorization with Column Pivoting

James Demmel
Laura Grigori
Ming Gu
Hua Xiang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-46
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-46.html

May 3, 2013

COMMUNICATION AVOIDING RANK REVEALING QR FACTORIZATION WITH COLUMN PIVOTING

JAMES W. DEMMEL*, LAURA GRIGORI†, MING GU ‡, AND HUA XIANG §

Abstract. In this paper we introduce CARRQR, a communication avoiding rank revealing QR factorization with tournament pivoting. We show that CARRQR reveals the numerical rank of a matrix in an analogous way to QR factorization with column pivoting (QRCP). Although the upper bound of a quantity involved in the characterization of a rank revealing factorization is worse for CARRQR than for QRCP, our numerical experiments on a set of challenging matrices show that this upper bound is very pessimistic, and CARRQR is an effective tool in revealing the rank in practical problems.

Our main motivation for introducing CARRQR is that it minimizes data transfer, modulo poly-logarithmic factors, on both sequential and parallel machines, while previous factorizations as QRCP are communication sub-optimal and require asymptotically more communication than CARRQR. Hence CARRQR is expected to have a better performance on current and future computers, where commmunication is a major bottleneck that highly impacts the performance of an algorithm.

Key words. QR factorization, rank revealing, column pivoting, minimize communication

AMS subject classifications. 65F25, 65F20

1. Introduction. Revealing the rank of a matrix is an operation that appears in many important problems as least squares problems, low rank approximations, regularization, nonsymmetric eigenproblems (see for example [8] and the references therein). In this paper we focus on the rank revealing QR factorization [8, 7, 16], which computes a decomposition of a matrix $A \in \mathbb{R}^{m \times n}$ of the form

$$A\Pi = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}, \tag{1.1}$$

where $Q \in \mathbb{R}^{m \times m}$ is orthgonal, $R_{11} \in \mathbb{R}^{k \times k}$ is upper triangular, $R_{12} \in \mathbb{R}^{k \times (n-k)}$, and $R_{22} \in \mathbb{R}^{(m-k) \times (n-k)}$. The column permutation matrix Π and the integer k are chosen such that $||R_{22}||_2$ is small and R_{11} is well-conditioned. This factorization was introduced in [16], and the first algorithm to compute it was proposed in [6] and is based on the QR factorization with column pivoting (QRCP). A BLAS-3 version of this algorithm [25] is implemented in LAPACK [1], and its parallel version in ScaLAPACK [5].

*Computer Science Division and Mathematics Department, UC Berkeley, CA 94720-1776, USA (demmel@cs.berkeley.edu). We acknowledge funding from Microsoft (award #024263) and Intel (award #024894), and matching funding by UC Discovery (award #DIG07-10227), with additional support from ParLab affiliates National Instruments, Nokia, NVIDIA, Oracle, and Samsung, and support from MathWorks. We also acknowledge the support of the US DOE (grants DE-SC0003959, DE-SC0004938, DE-SC0005136, DE-SC0008700, DE-AC02-05CH11231), and DARPA (award #HR0011-12-2-0016).

†INRIA Rocquencourt, Alpines, B.P. 105, F-78153 Le Chesnay Cedex and UPMC Univ Paris 06, CNRS UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France (laura.grigori@inria.fr). This work has been supported in part by French National Research Agency (ANR) through COSINUS program (project PETALh no ANR-10-COSI-013).

‡Mathematics Department, UC Berkeley, CA 94720-1776, USA (mgu@math.berkeley.edu). The author was supported in part by NSF Award CCF-0830764 and by the DOE Office of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231.

§School of Mathematics and Statistics, Wuhan University, Wuhan 430072, P. R. China (hxiang@whu.edu.cn). The work of this author was supported by the National Natural Science Foundation of China under grants 10901125 and 91130022.

The performance of an algorithm is highly impacted by the amount of communication performed during its execution, where communication refers to both data transferred between different levels of the memory hierarchy of a processor or between different processors of a parallel computer. Research performed in the recent years has shown that most of the classic algorithms in direct dense linear algebra transfer more data than lower bounds on communication indicate is necessary, and new, communication optimal algorithms can and should be developed. In this context, the goal of this paper is to design a pivoted QR factorization that is effective in revealing the rank of a matrix but also minimizes communication, on both sequential and parallel machines.

There are several different definitions for determining when the factorization from equation (1.1) reveals the rank (see, for example, [15]), one of them [22, 9] says that the factorization from equation (1.1) is a rank revealing QR factorization (RRQR) if

$$\sigma_{min}(R_{11}) \geq \frac{\sigma_k(A)}{p(k,n)} \text{ and } \sigma_{max}(R_{22}) \leq \sigma_{k+1}(A)p(k,n), \tag{1.2}$$

where $\sigma_{min}(A)$ and $\sigma_{max}(A)$ are the smallest and the largest singular values of A respectively, and $p(k,n)$ is a low degree polynomial in n and k. Since the $(k+1)$-th largest singular value $\sigma_{k+1} \leq \sigma_{\max}(R_{22}) = ||R_{22}||_2$ and $||R_{22}||_2$ is small, then A can be considered to have numerical rank k. The first k columns $Q(:, 1:k)$ form an approximate orthogonal basis for the range of A and $\Pi \begin{bmatrix} R_{11}^{-1}R_{12} \\ -I \end{bmatrix}$ are approximate null vectors.

Given a rank k and a parameter $f > 1$, it is shown in [20] that there exists a permutation Π such that the factorization displayed in equation (1.1) satisfies the inequality

$$\left(R_{11}^{-1}R_{12}\right)_{i,j}^2 + \omega_i^2\left(R_{11}\right)\gamma_j^2\left(R_{22}\right) \leq f^2, \tag{1.3}$$

where $R_{i,j}$ is the element in position (i,j) of R, $\omega_i(R_{11})$ denotes the 2-norm of the i-th row of R_{11}^{-1}, and $\gamma_j(R_{22})$ denotes the 2-norm of the j-th column of R_{22}. This factorization is called a *strong RRQR factorization*, and is more powerful than the classic QR factorization with column pivoting which only guarantees that $f = O(2^n)$. A strong RRQR factorization is computed by performing first a QR factorization with column pivoting followed by additional swaps of columns. In Section 2, we discuss in more detail the characterization of a strong rank revealing QR factorization, as well as more relaxed versions of the bounds from equation (1.3).

In practice the QR factorization with column pivoting often works well, and it is widely used even if it is known to fail, for example on the so called Kahan matrix that we describe in more detail in section 4. However in terms of communication, the QR factorization with column pivoting is sub-optimal with respect to lower bounds on communication identified in [3] (under certain assumptions in the case of the QR factorization). If the algorithm is performed in parallel, then typically the matrix is distributed over P processors by using a two-dimensional block cyclic partitionning. This is indeed the approach used in the `psgeqpf` routine from ScaLAPACK. At each step of the decomposition, the QR factorization with column pivoting finds the column of maximum norm and permutes it to the leading position, and this requires exchanging $O(n)$ messages, where n is the number of columns of the input matrix. For square matrices, when the memory per processor used is on the order of $O(n^2/P)$, the

lower bound on the number of messages to be exchanged is $\Omega(\sqrt{P})$. The number of messages exchanged during the QR factorization with column pivoting is larger by at least a factor of $n/\sqrt{P}$ than the lower bound. When QRCP is executed on a sequential machine with a fast memory of size M and a slow memory, then the volume of data transferred between fast and slow memory is on the order of $\Theta(n^3)$, and the number of messages is at least $\Theta(n^3/M)$. The lower bound on the volume of communication and the number of messages is $\Omega(n^3/M^{1/2})$, $\Omega(n^3/M^{3/2})$ respectively. We note that the classic QR factorization with no pivoting in which each column is annihilated by using one Householder transformation is also sub-optimal in terms of communication. A communication optimal algorithm (modulo polylogarithmic factors), referred to as communication avoiding QR (CAQR), has been introduced in [10, 11].

In this paper we introduce CARRQR, a communication optimal (modulo polylogarithmic factors) RRQR factorization based on tournament pivoting. The factorization is based on an algorithm that computes the decomposition by blocks of b columns (panels). For each panel, tournament pivoting proceeds in two steps. The first step aims at identifying a set of b candidate pivot columns that are as well-conditioned as possible. These columns are permuted to the leading positions, and they are used as pivots for the next b steps of the QR factorization. To identify the set of b candidate pivot columns, a tournament is performed based on a reduction operation, where at each node of the reduction tree b candidate columns are selected by using the strong rank revealing QR factorization. The idea of tournament pivoting has been first used to reduce communication in Gaussian elimination [18, 19], and then in the context of a newly introduced LU factorization with panel rank revealing pivoting [24]. CARRQR is optimal in terms of communication, modulo polylogarithmic factors, on both sequential machines with two levels of slow and fast memory and parallel machines with one level of parallelism, while performing three times more floating point operations than QRCP. We expect that on computers where communication is the bottleneck, CARRQR will be faster than other algorithms as QRCP which do not minimize communication. We believe that large speedups can be obtained on future computers (if not the present for sufficiently large matrices) where communication plays an increasingly important role for the performance and parallel scalability of an algorithm.

We show that CARRQR computes a permutation that satisfies

$$\left(R_{11}^{-1}R_{12}\right)_{i,j}^{2} + \left(\gamma_j\left(R_{22}\right)/\sigma_{\min}\left(R_{11}\right)\right)^2 \leq F^2, \tag{1.4}$$

where F is a constant dependent on k, f, and n. Equation (1.4) looks very similar to equation (1.3), and reveals the matrix numerical rank in a completely analogous way (see Theorems 2.2 and 2.4). While our upper bound on F is super-exponential in n (see Theorem 2.10), our extensive experiments, including those on challenging matrices, show that this upper bound is very pessimistic in general (see Section 4). These experiments demonstrate that CARRQR is as effective as QR with column pivoting in revealing the rank of a matrix. For the cases where QR with column pivoting does not fail, CARRQR also works well, and the values on the diagonal of the R factor are very close to the singular values of the input matrix computed with the highly accurate routine DGESVJ [13, 14]. The matrices in our set were also used in previous papers discussing rank revealing factorizations [4, 20, 26, 23].

The rest of this paper is organized as follows. Section 2 presents the algebra of CARRQ and shows that it is a rank revealing factorization that satisfies equation (1.4). Section 3 analyzes the parallel and sequential performance of CARRQR and

discusses its communication optimality. Section 4 discusses the numerical accuracy of CARRQR and compares it with QRCP and the singular value decomposition. Section 5 outlines how tournament pivoting can be extended to other factorizations as Cholesky with diagonal pivoting, LU with complete pivoting, or LDL^T factorization with pivoting. Finally, section 6 concludes our paper.

2. Rank revealing QR factorization with tournament pivoting. This section presents the algebra of the QR factorization algorithm based on a novel pivoting scheme referred to as tournament pivoting and analyzes its numerical stability. We refer to this algorithm as CARRQR.

2.1. The algebra. We consider a block algorithm that partitions the matrix A of size $m \times n$ into panels of size b. In classic QR factorization with column pivoting, at each step i of the factorization, the remaining unselected column of maximum norm is selected and exchanged with the i-th column, its subdiagonal elements are annihilated, using for example a Householder transformation, and then the trailing matrix is updated. A block version of this algorithm is described in [25]. The main difficulty in reducing communication in rank revealing QR factorization lies in identifying b pivot columns at each step of the block algorithm. Our communication avoiding algorithm, CARRQR, is based on tournament pivoting, and uses a reduction operation on blocks of columns to identify the next b pivot columns at each step of the block algorithm. This idea is analogous to the reduction operation used in CALU [18] to identify the next b pivot rows. The operator used at each node of the reduction tree is a RRQR factorization. Our theoretical analysis presented in section 2.3 is general enough to account for any kind of RRQR factorization, from classical column pivoting to the "strong" RRQR factorization in [20], and gives tighter bound if a strong RRQR factorization is used. The numerical experiments from section 4 will show that using CARRQR is adequate in practice, and indeed much better than the bounds derived in this section. This is somewhat similar to using traditional column pivoting for the overall factorization: the average case is much better than the worst case, although the worst case can occur with traditional column pivoting.

To illustrate tournament pivoting, we consider an m-by-n matrix A. We use a binary reduction tree and operate on blocks of b_T columns, so there are n/b_T such blocks. In our example $b_T = n/4$, so A is partitioned as $A = [A_{00}, A_{10}, A_{20}, A_{30}]$. Hence our communication avoiding algorithm has two parameters, b which determines the panel size and b_T which determines the height of the reduction tree. These two parameters will be discussed in more detail in section 3.

Tournament pivoting starts by computing a (strong) RRQR factorization of each column block A_{i0} to identify b column candidates,

$$A_{i0}\Pi_{i0} = Q_{i0} \begin{bmatrix} R_{i0} & * \\ & * \end{bmatrix}, \text{ for } i = 0 \text{ to } 3,$$

where Π_{i0} are permutation matrices of size $b_T \times b_T$, Q_{i0} are orthogonal matrices of size $m \times m$, and R_{i0} are upper triangular matrices of size $m \times b$. The first subscript i indicates the column block of the matrix, while the second subscript 0 indicates that the operation is performed at the leaves of the binary reduction tree.

At this stage we have n/b_T sets of b column candidates. The final b columns are selected by performing a binary tree (of depth $\log_2(n/b_T) = 2$) of (strong) RRQR factorizations of matrices of size $m \times 2b$. At the first level of the binary tree, we form

two matrices by putting together consecutive sets of column candidates.

$$A_{01} = [(A_{00}\Pi_{00})(:, 1:b), (A_{10}\Pi_{10})(:, 1:b)]$$
$$A_{11} = [(A_{20}\Pi_{20})(:, 1:b), (A_{30}\Pi_{30})(:, 1:b)]$$

From each matrix we select a new set of b column candidates by again computing a (strong) RRQR factorization:

$$A_{i1}\Pi_{i1} = Q_{i1} \begin{bmatrix} R_{i1} & * \\ & * \end{bmatrix}, \text{ for } i = 0 \text{ to } 1,$$

where Π_{i1} are permutation matrices of size $2b \times 2b$, Q_{i1} are orthogonal matrices of size $m \times m$, and R_{i1} are upper triangular matrices of size $b \times b$.

At the second (and last) level of the binary tree, the two sets of b column candidates from the first level are combined into a matrix A_{02},

$$A_{02} = [(A_{01}\Pi_{01})(:, 1:b), (A_{11}\Pi_{11})(:, 1:b)].$$

The final b columns are obtained by performing one last (strong) RRQR factorization of A_{02}:

$$A_{02}\Pi_{02} = Q_{02} \begin{bmatrix} R_{02} & * \\ & * \end{bmatrix},$$

where Π_{02} is a permutation matrix of size $2b \times 2b$, Q_{02} is an orthogonal matrix of size $m \times m$, and R_{02} is an upper triangular matrix of size $b \times b$: The final b columns selected are $A_{02}\Pi_{02}(:, 1:b)$.

The matrices Π_{ij}, $i = 0, 1$ and $j = 1, 2$, are partitioned into four blocks of size $b \times b$ as

$$\Pi_{ij} = \begin{bmatrix} \Pi_{ij}^{(1)} & \Pi_{ij}^{(2)} \\ \Pi_{ij}^{(3)} & \Pi_{ij}^{(4)} \end{bmatrix}.$$

Let $\tilde{\Pi}_{ij}$, $i = 0, 1$ and $j = 1, 2$, be permutation matrices obtained by extending Π_{ij} with identity matrices,

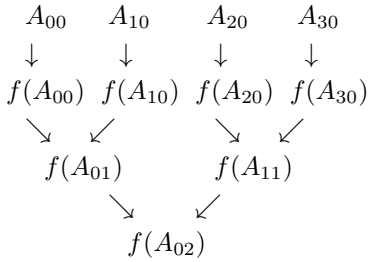
$$\tilde{\Pi}_{ij} = \begin{bmatrix} \Pi_{ij}^{(1)} & & \Pi_{ij}^{(2)} & \\ & I_r & & \\ \Pi_{ij}^{(3)} & & \Pi_{ij}^{(4)} & \\ & & & I_r \end{bmatrix},$$

where $r = n/P - b$ for $\tilde{\Pi}_{01}$, $\tilde{\Pi}_{11}$ and $r = n/2 - b$ for $\tilde{\Pi}_{02}$. The tournament pivoting process can be expressed as

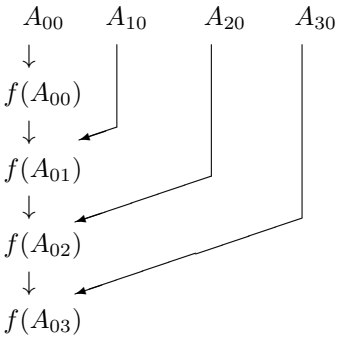
$$A \begin{bmatrix} \Pi_{00} & & & \\ & \Pi_{10} & & \\ & & \Pi_{20} & \\ & & & \Pi_{30} \end{bmatrix} \begin{bmatrix} \tilde{\Pi}_{01} & \\ & \tilde{\Pi}_{11} \end{bmatrix} \tilde{\Pi}_{02} = Q_{02} \begin{bmatrix} R_{02} & * \\ & * \end{bmatrix}.$$

In other words, the factorization performed at the root of the binary tree corresponds to the factorization of the first panel of the permuted matrix. The algorithm updates the trailing matrix using Q_{02} and then goes to the next iteration.

Different reduction trees can be used to perform tournament pivoting. The binary tree is presented in the following picture using an arrow notation. At each node of the reduction tree, $f(A_{ij})$ returns the first b columns obtained after performing (strong) RRQR of A_{ij}. The input matrix A_{ij} is obtained by adjoining the input matrices (on the other ends of the arrows pointing towards $f(A_{ij})$).



A flat tree is presented using this arrow notation in the following picture.



2.2. Selecting the first b columns from (strong) rank revealing QR factorization. The (possibly strong) rank revealing QR factorization of a matrix A_{ij} is used in tournament pivoting at each node of the reduction tree to select b candidate columns. Suppose that A_{ij} is of dimension $m_1 \times n_1$. When the matrix fits in fast memory on a sequential processor, the b column candidates may be selected by computing the first b steps of QR with column pivoting. When a strong rank revealing factorization is employed, several supplementary operations and swaps are performed, as explained in [20].

When the matrix does not fit in fast memory, or it is distributed over several processors, the b candidate columns are selected by first computing the QR factorization of A_{ij} without pivoting, and then the (strong) QR factorization of the much smaller ($2b$-by-$2b$) R factor. The first QR factorization is performed using communication avoiding QR [11] for tall and skinny matrices, referred to as TSQR, which minimizes communication.

2.3. Why QR with tournament pivoting reveals the rank. In this section, we first recall the characterization of a (strong) RRQR factorization from [20], modify it slightly, and then show (with appropriate choice of bounds) that it describes the result of tournament pivoting. The characterization depends on the particular rank k chosen. Subsection 2.3.1 analyzes the case $k = b$, i.e. the result of a single tournament. Then subsection 2.3.2 extends the analysis to any $1 \leq k \leq \min(m, n)$, i.e. the final output of the algorithm.

To set up the notation needed to explain [20], let $\omega_i(X)$ denote the 2-norm of i-th row of X^{-1}, and let $\gamma_j(X)$ denote the 2-norm of the j-th column of X.

THEOREM 2.1. *(Gu and Eisenstat [20]) Let B be an $m \times n$ matrix and let $1 \leq k \leq \min(m,n)$. For any given parameter $f > 1$, there exists a permutation Π such that*

$$B\Pi = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix},$$

where R_{11} is $k \times k$ and

$$\left(R_{11}^{-1} R_{12}\right)_{i,j}^2 + \omega_i^2\left(R_{11}\right) \gamma_j^2\left(R_{22}\right) \leq f^2. \tag{2.1}$$

The factorization in Theorem 2.1 can be computed in $O(mnk)$ flops. Inequality (2.1) is important because it implies bounds on the singular values of R_{11} and R_{22}:

THEOREM 2.2. *(Gu and Eisenstat [20]) Let the factorization in Theorem 2.1 satisfy inequality (2.1). Then*

$$1 \leq \frac{\sigma_i(B)}{\sigma_i(R_{11})}, \frac{\sigma_j(R_{22})}{\sigma_{k+j}(B)} \leq \sqrt{1 + f^2 k(n-k)},$$

for any $1 \leq i \leq k$ and $1 \leq j \leq \min(m,n) - k$.

In particular, Theorem 2.2 shows that the QR factorization in Theorem 2.1 *reveals the rank* in the sense that the singular values of R_{11} are reasonable approximations of the k largest singular values of B, and the singular values of R_{22} are reasonable approximations of the $\min(m,n) - k$ smallest singular values of B. We call this a *strong* rank revealing factorization because the bound in Theorem 2.2 grows like a low degree polynomial in n. In contrast, traditional column pivoting only guarantees that $f = O(2^n)$. Still, traditional column pivoting often works well in practice, and we will use it as a tool in our numerical experiments later.

To analyze our communication-avoiding RRQR algorithm in a more convenient way, we consider the following relaxed version of Theorem 2.1.

COROLLARY 2.3. *Let B be an $m \times n$ matrix and let $1 \leq k \leq \min(m,n)$. For any given parameter $f > 1$, there exists a permutation Π such that*

$$B\Pi = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix},$$

where R_{11} is $k \times k$ and

$$\sqrt{\gamma_j^2\left(R_{11}^{-1} R_{12}\right) + \left(\gamma_j\left(R_{22}\right) / \sigma_{\min}(R_{11})\right)^2} \leq f\sqrt{k} \quad \text{for} \quad j = 1, \cdots, n-k. \tag{2.2}$$

The proof is immediate, as the permutation Π of Theorem 2.1 automatically satisfies inequality (2.2). Below is the analogue of Theorem 2.2 based on Corollary 2.3.

THEOREM 2.4. *Assume that there exists a permutation Π for which the QR factorization*

$$B\Pi = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}$$

where R_{11} is $k \times k$, satisfies

$$\sqrt{\gamma_j^2\left(R_{11}^{-1} R_{12}\right) + \left(\gamma_j\left(R_{22}\right) / \sigma_{\min}(R_{11})\right)^2} \leq F \quad \text{for} \quad j = 1, \cdots, n-k. \tag{2.3}$$

Then

$$1 \leq \frac{\sigma_i(B)}{\sigma_i(R_{11})}, \frac{\sigma_j(R_{22})}{\sigma_{k+j}(B)} \leq \sqrt{1+F^2(n-k)},$$

for any $1 \leq i \leq k$ *and* $1 \leq j \leq \min(m,n)-k$.

Remark: By Corollary 2.3, there exists a permutation Π for which inequality (2.3) is satisfied with $F = f\sqrt{k}$. For this permutation and with this choice of F, Theorem 2.4 gives the same singular value ratio bounds as those in Theorem 2.2. However, Theorem 2.4 will prove much more convenient for our subsequent analysis and could provide a better practical bound on the singular values as we can typically expect F to be much smaller than $f\sqrt{k}$.

Proof of Theorem 2.4: Define

$$\alpha = \frac{\sigma_{\max}(R_{22})}{\sigma_{\min}(R_{11})}.$$

Then we can rewrite

$$B\Pi = Q \begin{bmatrix} R_{11} & \\ & R_{22}/\alpha \end{bmatrix} \begin{bmatrix} I & R_{11}^{-1}R_{12} \\ & \alpha I \end{bmatrix}.$$

It follows that

$$\sigma_i(B) \leq \sigma_i\left(\begin{bmatrix} R_{11} & \\ & R_{22}/\alpha \end{bmatrix}\right) \left\| \begin{bmatrix} I & R_{11}^{-1}R_{12} \\ & \alpha I \end{bmatrix} \right\|_2 \quad i = 1, \cdots, k.$$

On the other hand, the largest k singular values of $\begin{bmatrix} R_{11} & \\ & R_{22}/\alpha \end{bmatrix}$ are precisely those of the matrix R_{11}, and the 2-norm of the matrix $\begin{bmatrix} I & R_{11}^{-1}R_{12} \\ & \alpha I \end{bmatrix}$ is bounded above by

$$\sqrt{1+\|R_{11}^{-1}R_{12}\|_2^2+\alpha^2} \leq \sqrt{1+F^2(n-k)}.$$

In other words,

$$\frac{\sigma_i(B)}{\sigma_i(R_{11})} \leq \sqrt{1+F^2(n-k)}$$

for $i = 1, \cdots, k$. Conversely, observe that

$$\begin{bmatrix} \alpha R_{11} & \\ & R_{22} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix} \begin{bmatrix} \alpha I & -R_{11}^{-1}R_{12} \\ & I \end{bmatrix},$$

where the smallest $\min(m,n)-k$ singular values of $\begin{bmatrix} \alpha R_{11} & \\ & R_{22} \end{bmatrix}$ are the singular values of R_{22}, and the 2-norm of the matrix $\begin{bmatrix} \alpha I & -R_{11}^{-1}R_{12} \\ & I \end{bmatrix}$ can be similarly bounded as

$$\sqrt{1+\|R_{11}^{-1}R_{12}\|_2^2+\alpha^2} \leq \sqrt{1+F^2(n-k)}.$$

This again leads to

$$\frac{\sigma_j(R_{22})}{\sigma_{k+j}(B)} \leq \sqrt{1+F^2(n-k)}, \quad j = 1, \cdots, \min(m,n)-k.$$

This completes the proof. □

2.3.1. Analyzing one tournament step. Next we apply Theorem 2.4 to analyze one tournament step: Given a subset of b columns of matrix B that reveals its rank (for $k = b$), and another subset of b columns that reveals the rank of a different matrix $\widehat{B}$, we will show that computing a rank-revealing decomposition of just these $2b$ columns of B and $\widehat{B}$ will provide b columns that reveals the rank of the combined matrix $[B, \widehat{B}]$ (see Lemma 2.5). Then we will use this as an induction step to analyze an entire tournament, once using a binary tree (Corollary 2.6) and once with a flat tree (Corollary 2.7).

We use the following notation to denote the rank-revealing factorizations of B and $\widehat{B}$:

$$B\Pi = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix} \quad \text{and} \quad \widehat{B}\widehat{\Pi} = \widehat{Q} \begin{bmatrix} \widehat{R}_{11} & \widehat{R}_{12} \\ & \widehat{R}_{22} \end{bmatrix}, \tag{2.4}$$

where Π and $\widehat{\Pi}$ are permutation matrices; R_{11} and $\widehat{R}_{11}$ are $b \times b$ upper-triangular matrices. We assume that the factorizations in equation (2.4) satisfy

$$\gamma_j(N)^2 + \gamma_j(R_{22})^2 / \sigma_{\min}(R_{11})^2 \leq F^2, \quad \gamma_j(\widehat{N})^2 + \gamma_j(\widehat{R}_{22})^2 / \sigma_{\min}\left(\widehat{R}_{11}\right)^2 \leq \widehat{F}^2, \tag{2.5}$$

where $N = R_{11}^{-1} R_{12}$ and $\widehat{N} = \widehat{R}_{11}^{-1} \widehat{R}_{12}$. For example, by Theorem 2.1, we can choose $F = \widehat{F} = f\sqrt{b}$ when B and $\widehat{B}$ each consists of $2b$ columns; later on we will use this fact as the basis of our induction. Next we develop similar bounds for the combined matrix $\widetilde{B} = \begin{bmatrix} B & \widehat{B} \end{bmatrix}$.

Tournament pivoting continues by computing a (strong) RRQR factorization of the form

$$\bar{B}\bar{\Pi} \stackrel{def}{=} \left[Q \begin{bmatrix} R_{11} \\ \end{bmatrix}, \quad \widehat{Q} \begin{bmatrix} \widehat{R}_{11} \\ \end{bmatrix} \right] \bar{\Pi} = \widetilde{Q} \begin{bmatrix} \widetilde{R}_{11} & \bar{R}_{12} \\ & \bar{R}_{22} \end{bmatrix}, \tag{2.6}$$

where

$$\left(\widetilde{R}_{11}^{-1} \bar{R}_{12} \right)_{i,j}^2 + \omega_i^2 \left(\widetilde{R}_{11} \right) \gamma_j^2 \left(\bar{R}_{22} \right) \leq f^2. \tag{2.7}$$

Let

$$\widetilde{\Pi} = \begin{bmatrix} \Pi & \\ & \widehat{\Pi} \end{bmatrix} \begin{bmatrix} I & & & \\ & & I & \\ & I & & \\ & & & I \end{bmatrix} \begin{bmatrix} \bar{\Pi} & \\ & I \end{bmatrix}$$

be the accumulation of all permutations, then we can write $\widetilde{B}$ as

$$\begin{aligned} \widetilde{B}\widetilde{\Pi} &= \left[\widetilde{Q} \begin{bmatrix} \widetilde{R}_{11} & \bar{R}_{12} \\ & \bar{R}_{22} \end{bmatrix}, \; Q \begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix}, \; \widehat{Q} \begin{bmatrix} \widehat{R}_{12} \\ \widehat{R}_{22} \end{bmatrix} \right] \\ &= \widetilde{Q} \left[\begin{bmatrix} \widetilde{R}_{11} & \bar{R}_{12} \\ & \bar{R}_{22} \end{bmatrix}, \; \widetilde{Q}^T Q \begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix}, \; \widetilde{Q}^T \widehat{Q} \begin{bmatrix} \widehat{R}_{12} \\ \widehat{R}_{22} \end{bmatrix} \right] \\ &\stackrel{def}{=} \widetilde{Q} \begin{bmatrix} \widetilde{R}_{11} & \widetilde{R}_{12} \\ & \widetilde{R}_{22} \end{bmatrix}. \end{aligned} \tag{2.8}$$

Our goal is to derive bounds analogous to (2.5) for equation (2.8). To this end, we need to first identify the matrices $\widetilde{R}_{12}$ and $\widetilde{R}_{22}$. Note that

$$\widetilde{Q}^T Q \begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix} = \widetilde{Q}^T Q \begin{bmatrix} R_{11} N \\ R_{22} \end{bmatrix} = \widetilde{Q}^T Q \begin{bmatrix} R_{11} \\ \end{bmatrix} N + \widetilde{Q}^T Q \begin{bmatrix} \\ R_{22} \end{bmatrix}.$$

Continuing, we may write

$$Q \begin{bmatrix} R_{11} \\ \end{bmatrix} = \widetilde{Q} \begin{bmatrix} \widetilde{R}_{11} \mathcal{N} \\ \mathcal{C} \end{bmatrix}, \tag{2.9}$$

where the $b \times b$ matrices $\mathcal{N}$ and $\mathcal{C}$ are defined as follows: for each $1 \leq t \leq b$, the t-th column of the matrix on the left hand side of equation (2.9) must be some s-th column of $\widetilde{Q} \begin{bmatrix} \widetilde{R}_{11} & \bar{R}_{12} \\ & \bar{R}_{22} \end{bmatrix}$. If $s \leq b$, then we set the t-th column of $\mathcal{C}$ to be 0, and the t-th column of $\mathcal{N}$ to be all 0 except the s-th entry, which will be 1. On the other hand, if $s > b$, then we set the t-th columns of $\mathcal{C}$ and $\mathcal{N}$ to be the $(s-b)$-th columns of $\bar{R}_{22}$ and $\widetilde{R}_{11}^{-1} \bar{R}_{12}$, respectively. Since $f > 1$, we must have by inequality (2.7) that

$$\mathcal{N}_{i,j}^2 + \omega_i \left(\widetilde{R}_{11} \right)^2 \gamma_j^2 \left(\mathcal{C} \right) \leq f^2 \tag{2.10}$$

for all $1 \leq i, j \leq b$. With the additional notation

$$\widetilde{Q}^T Q \begin{bmatrix} \\ R_{22} \end{bmatrix} \stackrel{def}{=} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \tag{2.11}$$

we can further rewrite

$$\widetilde{Q}^T Q \begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix} = \begin{bmatrix} \widetilde{R}_{11} \mathcal{N} \\ \mathcal{C} \end{bmatrix} N + \widetilde{Q}^T Q \begin{bmatrix} \\ R_{22} \end{bmatrix} = \begin{bmatrix} \widetilde{R}_{11} \left(\mathcal{N} N + \widetilde{R}_{11}^{-1} C_1 \right) \\ \mathcal{C} N + C_2 \end{bmatrix}. \tag{2.12}$$

Similarly, define matrices $\widehat{\mathcal{N}}, \widehat{\mathcal{C}}, \widehat{C}_1$ and $\widehat{C}_2$ so that

$$\widetilde{Q}^T \widehat{Q} \begin{bmatrix} \widehat{R}_{12} \\ \widehat{R}_{22} \end{bmatrix} = \begin{bmatrix} \widetilde{R}_{11} \left(\widehat{\mathcal{N}} \widehat{N} + \widetilde{R}_{11}^{-1} \widehat{C}_1 \right) \\ \widehat{\mathcal{C}} \widehat{N} + \widehat{C}_2 \end{bmatrix}.$$

All this algebra has finally allowed us to identify

$$\begin{aligned} \widetilde{R}_{12} &= \begin{bmatrix} \bar{R}_{12} & \widetilde{R}_{11} \left(\mathcal{N} N + \widetilde{R}_{11}^{-1} C_1 \right) & \widetilde{R}_{11} \left(\widehat{\mathcal{N}} \widehat{N} + \widetilde{R}_{11}^{-1} \widehat{C}_1 \right) \end{bmatrix} \\ \widetilde{R}_{22} &= \begin{bmatrix} \bar{R}_{22} & \mathcal{C} N + C_2 & \widehat{\mathcal{C}} \widehat{N} + \widehat{C}_2 \end{bmatrix} \end{aligned}$$

Below, we derive bounds analogous to (2.5) for equation (2.8). We do this according to the 1×3 partition in $\widetilde{R}_{12}$ and $\widetilde{R}_{22}$. By inequality (2.7), we have

$$\gamma_j^2 \left(\widetilde{R}_{11}^{-1} \widetilde{R}_{12} \right) + \gamma_j^2 \left(\bar{R}_{22} \right)^2 / \sigma_{\min}^2 \left(\widetilde{R}_{11} \right) \leq b f^2.$$

In addition, we have

$$\begin{aligned}
&\gamma_j^2\left(\mathcal{N}N+\widetilde{R}_{11}^{-1}C_1\right)+\gamma_j^2\left(\mathcal{C}N+C_2\right)/\sigma_{\min}^2\left(\widetilde{R}_{11}\right) \qquad (2.13)\\
&=\gamma_j^2\left(\left[\begin{array}{c}\mathcal{N}N\\ \mathcal{C}N/\sigma_{\min}\left(\widetilde{R}_{11}\right)\end{array}\right]+\left[\begin{array}{c}\widetilde{R}_{11}^{-1}C_1\\ C_2/\sigma_{\min}\left(\widetilde{R}_{11}\right)\end{array}\right]\right)\\
&\leq 2\left(\gamma_j^2\left(\left[\begin{array}{c}\mathcal{N}N\\ \mathcal{C}N/\sigma_{\min}\left(\widetilde{R}_{11}\right)\end{array}\right]\right)+\gamma_j^2\left(\left[\begin{array}{c}\widetilde{R}_{11}^{-1}C_1\\ C_2/\sigma_{\min}\left(\widetilde{R}_{11}\right)\end{array}\right]\right)\right)\\
&\leq 2\left(\left\|\left[\begin{array}{c}\mathcal{N}\\ \mathcal{C}/\sigma_{\min}\left(\widetilde{R}_{11}\right)\end{array}\right]\right\|_F^2\gamma_j^2(N)+\gamma_j^2\left[\begin{array}{c}C_1\\ C_2\end{array}\right]/\sigma_{\min}\left(\widetilde{R}_{11}\right)^2\right). \qquad (2.14)
\end{aligned}$$

It follows from inequality (2.10) that

$$\left\|\left[\begin{array}{c}\mathcal{N}\\ \mathcal{C}/\sigma_{\min}\left(\widetilde{R}_{11}\right)\end{array}\right]\right\|_F^2\leq f^2b^2,$$

and it follows from definition (2.11) that $\gamma_j^2\left(\left[\begin{array}{c}C_1\\ C_2\end{array}\right]\right)=\gamma_j^2\left(R_{22}\right)$. Furthermore, equation (2.9) can be rewritten as

$$R_{11}^TR_{11}=\mathcal{N}^T\widetilde{R}_{11}^T\widetilde{R}_{11}\mathcal{N}+\mathcal{C}^T\mathcal{C},$$

which implies that

$$\begin{aligned}
\sigma_{\min}^2\left(R_{11}\right)&\leq\sigma_{\min}^2\left(\widetilde{R}_{11}\right)\|\mathcal{N}\|_2^2+\|\mathcal{C}\|_2^2\\
&\leq\sigma_{\min}^2\left(\widetilde{R}_{11}\right)\left(\left\|\left[\begin{array}{c}\mathcal{N}\\ \mathcal{C}/\sigma_{\min}\left(\widetilde{R}_{11}\right)\end{array}\right]\right\|_F^2\right)\leq f^2b^2\sigma_{\min}^2\left(\widetilde{R}_{11}\right),
\end{aligned}$$

which in turn leads to

$$1/\sigma_{\min}^2\left(\widetilde{R}_{11}\right)\leq f^2b^2/\sigma_{\min}^2\left(R_{11}\right).$$

Plugging all these results into equation (2.14), we obtain an upper bound on (2.13):

$$\begin{aligned}
\gamma_j^2\left(\mathcal{N}N+\widetilde{R}_{11}^{-1}C_1\right)+\gamma_j^2\left(\mathcal{C}N+C_2\right)/\sigma_{\min}^2\left(\widetilde{R}_{11}\right)&\leq 2f^2b^2\left(\gamma_j^2(N)+\gamma_j^2\left(R_{22}\right)/\sigma_{\min}^2\left(R_{11}\right)\right)\\
&\leq 2f^2b^2F^2.
\end{aligned}$$

Similarly, we can derive an upper bound

$$\gamma_j^2\left(\widehat{\mathcal{N}}\widehat{N}+\widetilde{R}_{11}^{-1}\widehat{C}_1\right)+\gamma_j^2\left(\widehat{\mathcal{C}}\widehat{N}+\widehat{C}_2\right)/\sigma_{\min}^2\left(\widetilde{R}_{11}\right)\leq 2f^2b^2\widehat{F}^2.$$

All these results are now summarized in the following lemma.

LEMMA 2.5. *Suppose we are given two rank-revealing QR factorizations of B and $\widehat{B}$ as in equation (2.4), that reveal their ranks as described in equation (2.5). Suppose we perform another rank-revealing QR factorization of the b selected columns of B and b selected columns of $\widehat{B}$, as described in equations (2.6) and (2.7). Then the b*

columns selected by this last QR factorization let us perform a QR factorization of $\widetilde{B} = [B, \widehat{B}]$ *as described in equation (2.8), that reveals the rank of* $\widetilde{B}$ *with the bound*

$$\sqrt{\gamma_j^2\left(\widetilde{R}_{11}^{-1}\widetilde{R}_{12}\right) + \gamma_j^2\left(\widetilde{R}_{22}\right)/\sigma_{\min}^2\left(\widetilde{R}_{11}\right)} \leq \sqrt{2}fb\max\left(F, \widehat{F}\right). \tag{2.15}$$

We may use Lemma 2.5 as the induction step to analyze the result of any tournament, with any reduction tree. We consider two cases: a binary reduction tree, and a flat reduction tree, both applied to an $m \times n$ matrix with $m \geq n$.

In the case of a complete binary tree, we can let B and $\widehat{B}$ each contain 2^h blocks of b columns each, where $1 \leq h \leq \log_2(n/b) - 1$. Assume that both B and $\widehat{B}$ satisfy relations (2.5) with $F = \widehat{F} \equiv F_h^B$ (the superscript B stands for "binary tree"). By Corollary 2.3, for the base case we can set $F_1^B = f\sqrt{b}$. Lemma 2.5 yields the following recursive relation

$$(F_{h+1}^B) \leq \sqrt{2}fbF_h^B.$$

Solving this recursion yields

$$F_{h+1}^B \leq \frac{1}{\sqrt{2b}}\left(\sqrt{2}fb\right)^{h+1}. \tag{2.16}$$

For $h = \log_2(n/b) - 1$, i.e. after the entire tournament has completed, we get

COROLLARY 2.6. *Selecting* b *columns of the* $m \times n$ *matrix* A *using QR with tournament pivoting, with a binary tree, reveals the rank of* A *in the sense of Theorem 2.4 for* $k = b$*, with bound*

$$F_{\log_2(n/b)}^B \leq \frac{1}{\sqrt{2b}}\left(\sqrt{2}fb\right)^{\log_2(n/b)} = \frac{1}{\sqrt{2b}}(n/b)^{\log_2\left(\sqrt{2}fb\right)} \tag{2.17}$$

The bound in (2.17) can be regarded as a low degree polynomial in n in general for a fixed b and f. Note that the upper bound is a decreasing function of b when $b > \sqrt{n/(\sqrt{2}f)}$.

In the case of a flat tree, we may let B and $\widehat{B}$ contain h blocks and 1 block of b columns, respectively, where $1 \leq h \leq n/b - 1$. Assume that both B and $\widehat{B}$ satisfy relations (2.5) with $F = F_h^F$ and $\widehat{F} = F_1^F = 0$, respectively (the superscript F stands for "flat tree"). Lemma 2.5 now yields the following recursive relation

$$(F_{h+1}^F) \leq \sqrt{2}fbF_h^F.$$

Solving this recursion yields

$$F_{h+1}^F \leq \frac{1}{\sqrt{2b}}\left(\sqrt{2}fb\right)^{h+1}. \tag{2.18}$$

For $h = n/b - 1$, i.e. after the entire tournament has completed, we get

COROLLARY 2.7. *Selecting* b *columns of the* $m \times n$ *matrix* A *using QR with tournament pivoting, with a flat tree, reveals the rank of* A *in the sense of Theorem 2.4 for* $k = b$*, with bound*

$$F_n^F \leq \frac{1}{\sqrt{2b}}\left(\sqrt{2}fb\right)^{n/b}. \tag{2.19}$$

Bound (2.19) is exponential in n, pointing to the possibility that the QR factorization we compute might not quite reveal the rank in extreme cases. But note that the upper bound is a rapidly decreasing function of b.

Example: $b = 1$ (traditional column pivoting). We now evaluate bounds (2.17) and (2.19) when $b = 1$. It is easy to see that the optimal single column to choose to reveal the rank is the one with the largest norm, and that this choice satisfies both Theorem 2.1 with $f = 1$ when $k = 1$, and Theorem 2.4 with $F = 1$, both of which are unimprovable. Furthermore, no matter what kind of reduction tree we use for the tournament, the column with the largest norm will be chosen by the tournament (ties may be broken in different ways), since we always pick the column of largest norm at each step. So when $b = 1$, tournament pivoting is equivalent to classical column pivoting, whose analysis is well known. Comparing our bounds from (2.17) and (2.19) to the optimal value of $F = 1$, we get $F^B_{\log_2 n} \leq \sqrt{n/2}$ and $F^F_n \leq 2^{(n-1)/2}$. So the analysis of the binary tree is reasonably close to the best bound, although the flat tree analysis gives a weaker bound.

2.3.2. Extending the analysis to a full rank-revealing QR factorization. In this section we perform a global analysis to extend the one-step analysis of Section 2.3.1 to a full rank-revealing QR factorization.

Corollaries 2.6 and 2.7 in the last section showed that applying tournament pivoting once, to pick the leading b columns of a matrix, provide a column permutation that reveals the rank in the sense of Theorem 2.4, but just for one partition of $R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$, namely when R_{11} is $b \times b$. The goal of this section is to show that using repeated tournaments to select subsequent groups of b columns can make R rank revealing for R_{11} of *any* dimension $1 \leq k \leq n$.

Since the algorithm described in the last section only chooses groups of b columns at a time, it guarantees nothing about the order of columns within a group. For example the algorithm might do nothing when given b or fewer columns. So to prove a rank revealing property for any k, not just $k = tb$, we must apply a second (lower cost) step where we apply strong RRQR to each $b \times b$ diagonal block of R.

Given this second step, we will derive an upper bound in the form of (2.3). Our approach is to reveal the "least" rank-revealed matrix given the one-step results in Section 2.3.1.

To this end, we first introduce some new notation. Let e be the vector of all 1's of various dimensions. For every matrix X, let $|X|$ be the matrix with each entry the corresponding entry of X in absolute value. Let N and H be two matrices of compatible size, by the relationship

$$N \preceq H,$$

we mean that $H - N$ is a non-negative matrix. Our global analysis will benefit from the following lemma, the proof of which we omit.

LEMMA 2.8. *Let N and H be upper triangular matrices with unit diagonal, and with non-positive entries in the upper triangular part of H. Assume that*

$$|N| \preceq |H|.$$

Then we have

$$|N^{-1}| \preceq |H^{-1}|.$$

Now let

$$N = \begin{bmatrix} I & N_{12} & \cdots & N_{1t} \\ & I & \cdots & N_{2t} \\ & & \ddots & \vdots \\ & & & I \end{bmatrix}, \tag{2.20}$$

where each N_{ij} is a $b \times b$ matrix with $1 \le i < j \le t$; and $\gamma_k(N_{ij}) \le c_i$ for $1 \le k \le b$. Our global analysis will critically depend on tight estimates of $\|N^{-1}\|_2$. As a surrogate, we will instead discuss the choices of all the N_{ij} submatrices to maximize $\|N^{-1}\|_1$. To start, define

$$H = \begin{bmatrix} I & -|N_{12}| & \cdots & -|N_{1t}| \\ & I & \cdots & -|N_{2t}| \\ & & \ddots & \vdots \\ & & & I \end{bmatrix}.$$

In other words, we construct H by flipping the signs of all positive off-diagonal entries of N. It follows that $|N| = |H|$, and from Lemma 2.8 that $|N^{-1}| \preceq |H^{-1}|$, which implies that $\|N^{-1}\|_1 \le \|H^{-1}\|_1$. Define

$$M = \begin{bmatrix} I & M_{12} & \cdots & M_{1t} \\ & I & \cdots & M_{2t} \\ & & \ddots & \vdots \\ & & & I \end{bmatrix}, \tag{2.21}$$

where $M_{ij} = -\dfrac{c_i}{\sqrt{b}} ee^T$. It is easy to verify that $\gamma_k(M_{ij}) = c_i$ for all $1 \le k \le b$ and hence M satisfies the conditions on the matrix N in equation (2.20). Straightforward algebra shows that $\|M^{-1}\|_1 = \prod_{j=1}^{t-1} \left(1 + \sqrt{b} c_j\right)$. Lemma 2.9 below identifies M as the matrix that maximizes $\|N^{-1}\|_1$.

LEMMA 2.9. *Let matrices N and M be defined by equations (2.20) and (2.21), respectively. Then*

$$\|N^{-1}\|_1 \le \|M^{-1}\|_1.$$

Proof: We only consider the matrix N for which the upper triangular entries are all non-positive, so that $|N^{-1}| = N^{-1}$. We prove Lemma 2.9 by induction on t. The lemma is obviously true for $t = 1$. For $t > 1$, we will show the sum of entries in every column of N^{-1} is bounded above by $\|M^{-1}\|_1$. Let $\widehat{N}$ and $\widehat{M}$ be the first $(t-1) \times (t-1)$ block submatrices of N and M, respectively. By induction, for any $1 \le k \le (t-1)b$, the sum of entries in the k-th column of N^{-1} is bounded above by $\|\widehat{M}^{-1}\|_1 \le \|M^{-1}\|_1$. For any $(t-1)b + 1 \le k \le tb$, define

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_{t-1} \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_{t-1} \end{bmatrix} = -\widehat{N}^{-1} y,$$

where y_i is the $(k-(t-1)b)$-th column of N_{it}, with $\|y_i\|_2 \leq c_i$. By definition, y and x are the k-th columns of N and N^{-1} without the bottom b components, hence the sum of entries in the k-th column of N^{-1} is simply

$$1 + e^T x = 1 + \sum_{j=1}^{t-1} e^T x_j.$$

Since $x_1 = -y_1 + \sum_{j=2}^{t-1} (-N_{1,j})\, x_j$, it follows from the Cauchy-Schwartz inequality that

$$e^T x_1 = -e^T y_1 + \sum_{j=2}^{t-1} e^T (-N_{1,j})\, x_j \leq \sqrt{b} c_1 + \sum_{j=2}^{t-1} \sqrt{b} c_1 e^T x_j = \sqrt{b} c_1 \left(1 + \sum_{j=2}^{t-1} e^T x_j \right),$$

and that

$$1 + e^T x \leq 1 + \sqrt{b} c_1 + \left(1 + \sqrt{b} c_1\right) \sum_{j=2}^{t-1} e^T x_j = \left(1 + \sqrt{b} c_1\right) \left(1 + \sum_{j=2}^{t-1} e^T x_j \right).$$

To continue with the same argument to the end, we have

$$1 + e^T x \leq \prod_{j=1}^{t-1} \left(1 + \sqrt{b} c_j\right) = \|M^{-1}\|_1 . \square$$

We are now ready to derive global upper bounds. Suppose that at least t rounds of tournament pivoting have been computed:

$$B\Pi = Q \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1t} & R_{1,t+1} \\ & R_{22} & \cdots & R_{2t} & R_{2,t+1} \\ & & \ddots & \vdots & \vdots \\ & & & R_{tt} & R_{t,t+1} \\ & & & & R_{t+1,t+1} \end{bmatrix},$$

where R_{ii} is $b \times b$ for $1 \leq i \leq t$, and $R_{t+1,t+1}$ is $(n-tb) \times (n-tb)$, i.e. the first t tournaments selected the tb columns yielding R_{11} through R_{tt}. In light of equation (2.3), we need to develop an upper bound on

$$\tau^2 \stackrel{def}{=} \gamma_j^2 \left(\begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1t} \\ & R_{22} & \cdots & R_{2t} \\ & & \ddots & \vdots \\ & & & R_{tt} \end{bmatrix}^{-1} \begin{bmatrix} R_{1,t+1} \\ R_{2,t+1} \\ \vdots \\ R_{t,t+1} \end{bmatrix} \right)$$
$$+ \gamma_j^2 \left(R_{t+1,t+1} \right) / \sigma_{\min}^2 \left(\begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1t} \\ & R_{22} & \cdots & R_{2t} \\ & & \ddots & \vdots \\ & & & R_{tt} \end{bmatrix} \right).$$

Let N be the matrix defined in equation (2.20) with $N_{ij} = R_{ii}^{-1} R_{ij}$. Then N satisfies $\gamma_k(N_{ij}) \leq c_i$, where the exact value of c_i depends on the reduction tree:

$$c_i = \begin{cases} F^B_{\log_2((n-(i-1)b)/b)} & \text{for binary tree (see Corollary 2.6)} \\ \\ F^F_{n-(i-1)b} & \text{for flat tree (see Corollary 2.7).} \end{cases}$$

Let $y = \begin{bmatrix} y_1 \\ \vdots \\ y_t \end{bmatrix}$ be the j-th column of the matrix $\begin{bmatrix} R_{11}^{-1} R_{1,t+1} \\ \vdots \\ R_{tt}^{-1} R_{t,t+1} \end{bmatrix}$. We rewrite τ^2 as

$$\tau^2 = \|N^{-1}y\|_2^2 + \gamma_j^2 \left(R_{t+1,t+1}\right) / \sigma_{\min}^2 \left(\operatorname{diag}\left(R_{11}, \cdots, R_{tt}\right) N\right).$$

By Corollaries 2.6 and 2.7, $\|y_i\|_2$ is bounded above by the same c_i as defined above. Repeating the same arguments made in the proof for Lemma 2.9, it is easy to show that

$$\|N^{-1}y\|_2 \leq \|N^{-1}y\|_1 \leq \|M^{-1} \begin{bmatrix} -\frac{c_1}{\sqrt{b}} e \\ \vdots \\ -\frac{c_t}{\sqrt{b}} e \end{bmatrix} \|_1 = \prod_{j=1}^{t} \left(1 + \sqrt{b} c_j\right) - 1 < \prod_{j=1}^{t} \left(1 + \sqrt{b} c_j\right).$$

For the second term in τ^2, we note by Corollaries 2.6 and 2.7 that

$$\gamma_j \left(R_{t+1,t+1}\right) / \sigma_{\min} \left(R_{ii}\right) \leq c_i \leq c_1$$

for all $1 \leq i \leq t$. This implies that

$$\gamma_j \left(R_{t+1,t+1}\right) / \sigma_{\min} \left(\operatorname{diag}\left(R_{11}, \cdots, R_{tt}\right)\right) \leq c_1.$$

Hence

$$\begin{aligned} & \gamma_j \left(R_{t+1,t+1}\right) / \sigma_{\min} \left(\operatorname{diag}\left(R_{11}, \cdots, R_{tt}\right) N\right) \\ \leq & \gamma_j \left(R_{t+1,t+1}\right) / \sigma_{\min} \left(\operatorname{diag}\left(R_{11}, \cdots, R_{tt}\right)\right) \|N^{-1}\|_2 \\ \leq & c_1 \sqrt{n} \|N^{-1}\|_1 \leq c_1 \sqrt{n} \|M^{-1}\|_1 = \sqrt{n} c_1 \prod_{j=1}^{t-1} \left(1 + \sqrt{b} c_j\right). \end{aligned}$$

Putting it all together,

$$\begin{aligned} \tau^2 &\leq \left(\prod_{j=1}^{t} \left(1 + \sqrt{b} c_j\right)\right)^2 + \left(\sqrt{n} c_1 \prod_{j=1}^{t-1} \left(1 + \sqrt{b} c_j\right)\right)^2 \\ &= \left(\left(1 + \sqrt{b} c_t\right)^2 + n c_1^2\right) \left(\prod_{j=1}^{t-1} \left(1 + \sqrt{b} c_j\right)\right)^2 \\ &\leq \left(1 + \sqrt{b+n} c_1\right)^2 \left(\prod_{j=1}^{t-1} \left(1 + \sqrt{b} c_j\right)\right)^2. \end{aligned}$$

To further simplify the upper bound on τ^2, we consider the two special cases of the reduction tree:

Binary tree: In this case, we simply use the relation

$$c_j \leq c_1 = \frac{1}{\sqrt{2b}} \left(\sqrt{2} f b\right)^{\log_2(n/b)}$$

so that

$$\tau^2 \leq \left(1 + \sqrt{b+n} c_1\right)^2 \left(1 + \sqrt{b} c_1\right)^{2n/b} = \left(1 + \sqrt{b+n} c_1\right)^2 \left(1 + \frac{1}{\sqrt{2}} \left(\sqrt{2} f b\right)^{\log_2(n/b)}\right)^{2n/b}.$$

Flat tree: In this case, we note that

$$\tau^2 \le \left(1+\sqrt{b+nc_1}\right)^2 \left(\prod_{j=1}^{t-1}\sqrt{b}c_j\right)^2 e^{\sum_{j=1}^{t-1}\frac{2}{\sqrt{b}c_j}}$$
$$= \left(1+\sqrt{b+nc_1}\right)^2 b^{t-1}\left(\prod_{j=1}^{t-1}c_j\right)^2 e^{\frac{2}{\sqrt{b}}\sum_{j=1}^{t-1}\frac{1}{c_j}}.$$

Since

$$\sum_{j=1}^{t-1}\frac{1}{c_j} = \sqrt{2b}\sum_{j=1}^{t-1}\left(\sqrt{2}fb\right)^{-(n/b-j)} \le \sqrt{2b}/\left(1-1/(\sqrt{2}fb)\right) \le 4\sqrt{b},$$

it follows that

$$\tau^2 \le \left(1+\sqrt{b+nc_1}\right)^2 b^{n/b}\left(\frac{1}{\sqrt{2b}}\right)^{2n/b}\left(e^{\frac{2}{\sqrt{b}}4\sqrt{b}}\right)\left(\prod_{j=1}^{n/b}\left(\sqrt{2}fb\right)^{2(n/b-j)}\right)$$
$$= e^8\left(1+\sqrt{b+nc_1}\right)^2\left(1/\sqrt{2}\right)^{2n/b}\left(\sqrt{2}fb\right)^{n/b(n/b+1)}.$$

We are now ready to state the main theorem of this section.

THEOREM 2.10. *Let A be $m \times n$ with $m \ge n$, and assume for simplicity that b divides n. Suppose that we do QR factorization with tournament pivoting on A n/b times, each time selecting b columns to pivot to the left of the remaining matrix. Then for ranks k that are multiples of b, this yields a rank-revealing QR factorization of A in the sense of Theorem 2.4, with*

$$F = \begin{cases} \left(1+\sqrt{b+nc_1}\right)\left(1+\frac{1}{\sqrt{2}}\left(\sqrt{2}fb\right)^{\log_2(n/b)}\right)^{n/b}, & \text{for binary tree} \\ e^4\left(1+\sqrt{b+nc_1}\right)\left(1/\sqrt{2}\right)^{n/b}\left(\sqrt{2}fb\right)^{n/b(n/b+1)/2}, & \text{for flat tree.} \end{cases}$$

Although the binary tree bound is relatively smaller, we must note that both bounds in Theorem 2.10 are super-exponentially large. In contrast, our numerical results in section 4 are much better. As with conventional column pivoting (the special case $b = 1$), the worst case must be exponential, but whether these bounds can be significantly tightened is an open question. The matrix M in equation (2.21) was constructed to have the largest 1-norm in the inverse among all matrices that satisfy equation (2.20), which is likely to be a much larger class of matrices than CA-RRQR actually produces.

Example: $b = 1$ (traditional column pivoting). We now evaluate the bound in Theorem 2.10 when $b = 1$, i.e. traditional column pivoting. In this case we know the best bound is $O(2^n)$, which is attained by the upper triangular Kahan matrix, where $K_{ii} = c^{i-1}$ and $K_{ij} = -sc^{i-1}$ where $j > i$, $s^2 + c^2 = 1$ and s and c are positive. In contrast, our bounds are much larger, at $O(n^{n/2})$ and $O(2^{n^2/4})$, respectively.

Finally, we consider the case of k not a multiple of b. We only sketch the derivation of the bounds, since explicit expressions are complicated, and not more illuminating.

Recall that we must assume that strong RRQR has been independently performed on the $b \times b$ diagonal block

$$R_{ii} = \begin{bmatrix} R_{ii11} & R_{ii12} \\ 0 & R_{ii22} \end{bmatrix}$$

of the final result of tournament pivoting, with analogous notation R_{ij1} and R_{ij2}, with $j > i$, for the subblocks of R_{ij} to the right of R_{ii12} and R_{ii22}, resp., and R_{ki1} and R_{ki2}, with $k < i$, for the subblocks of R_{ki} above R_{ii11} and R_{ii12}, resp. yielding

$$\left[\begin{array}{ccc|c||c|ccc} R_{11} & \cdots & R_{1,i-1} & R_{1i1} & R_{1i2} & R_{1,i+1} & \cdots & R_{1t} \\ & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & R_{i-1,i-1} & R_{i-1,i,1} & R_{i-1,i,2} & R_{i-1,i+1} & \cdots & R_{i-1,t} \\ \hline & & & R_{ii11} & R_{ii12} & R_{i,i+1,1} & \cdots & R_{it1} \\ \hline\hline & & & & R_{ii22} & R_{i,i+1,2} & \cdots & R_{it2} \\ \hline & & & & & R_{i+1,i+1} & \cdots & R_{i+1,t} \\ & & & & & & \ddots & \vdots \\ & & & & & & & R_{tt} \end{array}\right]$$

$$\stackrel{def}{=} \left[\begin{array}{c|c||c|c} \hat{R}_{11} & \hat{R}_{12} & \hat{R}_{13} & \hat{R}_{14} \\ \hline & R_{ii11} & R_{ii12} & \hat{R}_{24} \\ \hline\hline & & R_{ii22} & \hat{R}_{34} \\ \hline & & & \hat{R}_{44} \end{array}\right]$$

We first need to bound the 2-norm of each column of the matrix

$$\begin{bmatrix} \hat{R}_{11} & \hat{R}_{12} \\ 0 & R_{ii11} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \hat{R}_{13} & \hat{R}_{14} \\ R_{ii12} & \hat{R}_{24} \end{bmatrix}$$
$$= \begin{bmatrix} \hat{R}_{11}^{-1}\hat{R}_{13} - \hat{R}_{11}^{-1}\hat{R}_{12}R_{ii11}^{-1}R_{ii12} & \hat{R}_{11}^{-1}\hat{R}_{14} - \hat{R}_{11}^{-1}\hat{R}_{12}R_{ii11}^{-1}\hat{R}_{24} \\ R_{ii11}^{-1}R_{ii12} & R_{ii11}^{-1}\hat{R}_{24} \end{bmatrix}$$

If we can bound the 2-norm of every column of all 4 subblocks in the above expression, then their root-sum-of-squares will bound the overall column 2-norms:

- column norms of $\hat{R}_{11}^{-1}\hat{R}_{12}$, $\hat{R}_{11}^{-1}\hat{R}_{13}$, and $\hat{R}_{11}^{-1}\hat{R}_{14}$ are all bounded by our previous result on tournament pivoting alone, since $\hat{R}_{11}$ is $(i-1)b \times (i-1)b$.
- $R_{ii11}^{-1}R_{ii12}$ is bounded since we did strong RRQR on R_{ii}.
- To see that columns of $R_{ii11}^{-1}\hat{R}_{24}$ are bounded we note that columns of $R_{ii}^{-1}R_{ij}$ are bounded (for $j > i$) by our previous result for tournament pivoting alone. Performing strong RRQR on R_{ii} changes R_{ii} to $\hat{Q}R_{ii}\Pi$ and R_{ij} to $\hat{Q}R_{ij}$ for some permutation Π and orthogonal $\hat{Q}$, so columns of

$$\begin{aligned} R_{ii}^{-1}R_{ij} &= \Pi \cdot (\hat{Q}R_{ii}\Pi)^{-1}(\hat{Q}R_{ij}) \\ &= \Pi \cdot \begin{bmatrix} R_{ii11} & R_{ii12} \\ 0 & R_{ii22} \end{bmatrix}^{-1} \begin{bmatrix} \hat{R}_{24} \\ \hat{R}_{34} \end{bmatrix} \\ &= \Pi \cdot \begin{bmatrix} R_{ii11}^{-1}\hat{R}_{24} - R_{ii11}^{-1}R_{ii12}R_{ii22}^{-1}\hat{R}_{34} \\ R_{ii22}^{-1}\hat{R}_{34} \end{bmatrix} \end{aligned}$$

are bounded by our previous result, and so for each j,

$$\begin{aligned}\gamma_j\left(R_{ii11}^{-1}\hat{R}_{24}\right) &\leq \gamma_j\left(R_{ii11}^{-1}\hat{R}_{24} - R_{ii11}^{-1}R_{ii12}R_{ii22}^{-1}\hat{R}_{34}\right) + \gamma_j\left(R_{ii11}^{-1}R_{ii12}R_{ii22}^{-1}\hat{R}_{34}\right)\\ &\leq \gamma_j\left(R_{ii}^{-1}R_{ij}\right) + \|R_{ii11}^{-1}R_{ii12}\|_2 \cdot \gamma_j\left(R_{ii22}^{-1}\hat{R}_{34}\right)\\ &\leq \gamma_j\left(R_{ii}^{-1}R_{ij}\right) + \|R_{ii11}^{-1}R_{ii12}\|_2 \cdot \gamma_j\left(R_{ii}^{-1}R_{ij}\right)\\ &\leq (1 + \|R_{ii11}^{-1}R_{ii12}\|_2)\gamma_j\left(R_{ii}^{-1}R_{ij}\right)\end{aligned}$$

is bounded.

- Finally, we can combine these bounds to bound above

$$\gamma_j\left(\hat{R}_{11}^{-1}\hat{R}_{13} - \hat{R}_{11}^{-1}\hat{R}_{12}R_{ii11}^{-1}R_{ii12}\right) \leq \gamma_j\left(\hat{R}_{11}^{-1}\hat{R}_{13}\right) + \|\hat{R}_{11}^{-1}\hat{R}_{12}\|_2 \cdot \gamma_j\left(R_{ii11}^{-1}R_{ii12}\right)$$

and

$$\gamma_j\left(\hat{R}_{11}^{-1}\hat{R}_{14} - \hat{R}_{11}^{-1}\hat{R}_{12}R_{ii11}^{-1}\hat{R}_{24}\right) \leq \gamma_j\left(\hat{R}_{11}^{-1}\hat{R}_{14}\right) + \|\hat{R}_{11}^{-1}\hat{R}_{12}\|_2 \cdot \gamma_j\left(R_{ii11}^{-1}\hat{R}_{24}\right)$$

Now we must bound from above

$$\gamma_j\left(\begin{bmatrix} R_{ii22} & \hat{R}_{34} \\ 0 & \hat{R}_{44}\end{bmatrix}\right) / \sigma_{\min}\left(\begin{bmatrix} \hat{R}_{11} & \hat{R}_{12} \\ 0 & R_{ii11}\end{bmatrix}\right)$$

Using the previously established bounds on column norms of $\hat{R}_{11}^{-1}\hat{R}_{12}$ and $R_{ii22}^{-1}\hat{R}_{34}$, it is as before enough to bound above

$$\begin{aligned}&\gamma_j\left(\begin{bmatrix} R_{ii22} & 0 \\ 0 & \hat{R}_{44}\end{bmatrix}\right) / \sigma_{\min}\left(\begin{bmatrix} \hat{R}_{11} & 0 \\ 0 & R_{ii11}\end{bmatrix}\right)\\ &= \gamma_j\left(\begin{bmatrix} R_{ii22} & 0 \\ 0 & \hat{R}_{44}\end{bmatrix}\right) / \min(\sigma_{\min}(\hat{R}_{11}), \sigma_{\min}(R_{ii11}))\end{aligned}$$

By the strong rank-revealing property for $k = (i-1)b$, we know that $\sigma_{\min}(\hat{R}_{11})$ cannot be much smaller than $\gamma_j\left(R_{ii11}\right)$, so the denominator in the last fraction can be replaced by $\sigma_{\min}(R_{ii11})$. Similarly, the numerator can be replaced by $\gamma_j\left(R_{ii22}\right)$. This leaves us with $\gamma_j\left(R_{ii22}\right)/\sigma_{\min}(R_{ii11})$, which is bounded above by the strong rank-revealing property of R_{ii}.

3. Performance analysis of QR with tournament pivoting. In this section we analyze the performance of parallel communication avoiding QR with tournament pivoting and we show that it minimizes communication, at the cost of roughly tripling the number of arithmetic operations. Algorithm 1 presents the parallel CARRQR factorization of a matrix A of size $m \times n$. The matrix is distributed block cyclically on a grid of $P = P_r \times P_c$ processors using blocks of size b. For simplicity we suppose n/b is an integer. Consider step j of the factorization, and let $m_b = m - (j-1)b$, $n_b = n-(j-1)b$. The size of the matrix on which the algorithm operates at this step is $m_b \times n_b$. First, tournament pivoting is used to identify b pivot columns, using a binary tree of depth $\log_2 n_b/b_T$. For the ease of the analysis, we consider that tournament pivoting is performed as an all-reduce operation, that is all processors participate at all the stages of the reduction, and the final result is available on all processors.

Depending on the size of b_T, a processor can be involved in operations at more than one node at each level of the binary tree. At the leaves, the b candidate columns

are selected from blocks of size $m_b \times b_T$. For this, the processors in the same process column perform the QR factorization of their blocks of columns, using TSQR [11]. The R factors obtained from this factorization are available on all processors. Each processor selects b column candidates by performing QR with column pivoting on their R factor.

After this step, there are nb/b_T candidate columns. The subsequent steps of the reduction operate on blocks of $2b$ columns, where the new column candidates are chosen similarly by computing TSQR followed by QR with column pivoting on the R factor. In the first $\log_2(n/(bP_c))$ levels of the reduction tree, a processor is involved in the computation of several nodes per level. In other words, it is the owner of several sets of candidate columns. Hence it can form with no communication the matrix of $2b$ column candidates of the next level. At the last $\log_2 P_c$ levels of the reduction tree, a processor is involved in the computation performed at only one node per level. Hence the processors in the same process row need to exchange their local parts of the candidate columns.

Algorithm 1 Parallel QR with tournament pivoting

Input matrix A of size $m \times n$, block size b, b_T
for $j = 1$ to n/b **do**
  Let $m_b = m - (j-1)b$, $n_b = n - (j-1)b$
  /* Perform tournament pivoting to choose b pivot columns */
  for $k = 1$ to $n_b/(2bP_c)$ **do**
    Let A_{0k} be the k-th column block of size $m_b \times 2b$ that belongs to the process column of my processor.
    Processors in the same process column select b column candidates by performing TSQR of their block A_{0k} followed by strong RRQR of the R factor.
  end for
  for $i = 1$ to $log_2(n_b/b_T)$ **do**
    for each node $k = 1$ to $n_b/(2bP_c2^i)$ in the current level assigned to my processor **do**
      if at most one node is assigned per processor **then**
        Each two processors in the same process row exchange their local parts of the b column candidates selected at the sons of their current node k.
      end if
      Let A_{ik} be the matrix obtained by putting next to each other the two sets of b column candidates selected at the sons of the current node k.
      Processors in the same process column select b column candidates by performing TSQR of their block A_{ik} followed by strong RRQR of the R factor.
    end for
  end for
  Swap the b pivot columns to j-th column block: $A(:, j_0 : end) = A(:, j_0 : end)\Pi_j$, where $j_0 = (j-1)b+1$ and Π_j is the permutation matrix returned by tournament pivoting.
  TSQR factorization of the j-th column block $A(j_0 : end, j_0 : j_1) = Q_j R_j$, where $j_1 = jb$, R_j is an upper triangular matrix.
  Update of the trailing matrix $A(j_0 : end, j_1 + 1 : end) = Q_j^T A(j_0 : end, j_1 + 1 : end)$.
end for

To study the theoretical performance of CARRQR, we use a simple model to describe a machine architecture in terms of speed, network latency, and bandwidth. The time needed to send a message of m words between two processors is estimated to be $\alpha+m\beta$, where α specifies the latency and β the inverse of the bandwidth. The cost of performing parallel CARRQR of an $m \times n$ matrix is given in Table 3.1. We display separately the cost of performing communication avoiding QR factorization and the cost of performing tournament pivoting, with $b_T = 2b$ and $b_T = n/P_c$. The total cost of CARRQR is the cost of performing CAQR plus the cost of performing tournament pivoting. The parallel CARRQR factorization based on tournament pivoting with $b_T = 2b$ requires three times more flops than the QR factorization. However for $b_T = n/P_c$, the parallel CARRQR factorization performs a factor of $n/(bP_c)$ more flops than the QR factorization. Hence the former choice of $b_T = 2b$ should lead to a faster algorithm in practice.

	Parallel CAQR
# flops	$\frac{2mn^2-2n^3/3}{P} + \frac{bn^2}{2P_c} + \frac{3bn(2m-n)}{2P_r} + \left(\frac{4b^2n}{3} + \frac{n^2(3b+5)}{2P_c}\right)\log_2 P_r - b^2n$
# words	$\left(\frac{n^2}{P_c} + \frac{bn}{2}\right)\log_2 P_r + \left(\frac{mn-n^2/2}{P_r} + 2n\right)\log_2 P_c$
# messages	$\frac{3n}{b}\log_2 P_r + \frac{2n}{b}\log_2 P_c$
	Tournament pivoting with $b_T = 2b$ (excluding cost of QR factorization)
# flops	$\frac{4mn^2-4n^3/3}{P} + \frac{8n^2b}{3P_c}(\log_2 P_r + 2) + f(m,n,b,P_r,P_c)$
# words	$\frac{n^2}{P_c}\log_2 P_r + \frac{mn-n^2/2}{P_r}(\log_2 P_c + 1)$
# messages	$\frac{n^2}{2b^2P_c}\log_2 P_r + \frac{n}{b}\log_2 P_c(\log_2 P_r + 1)$
	Tournament pivoting with $b_T = n/P_c$ (excluding cost of QR factorization)
# flops	$\frac{2}{3}\frac{mn^2-n^3/4}{P}\frac{n}{bP_c} + \frac{1}{6}\frac{n^3}{P_c^2}\frac{n}{bP_c}(\log_2 P_r + 2) + f(m,n,b,P_r,P_c)$
# words	$\frac{1}{6}\frac{n^2}{P_c}\frac{n}{bP_c}\log_2 P_r + \frac{mn-n^2/2}{P_r}(\log_2 P_c + 1) + 2nb\log_2 P_r \log_2 P_c$
# messages	$\frac{n}{b}(\log_2 P_r + 2\log_2 P_c + \log_2 P_r \log_2 P_c)$

TABLE 3.1

Performance models of parallel CAQR and tournament pivoting when factoring an $m \times n$ matrix, distributed in a 2-D block cyclic layout on a $P_r \times P_c$ grid of processors with square $b \times b$ blocks. All terms are counted along the critical path. We generally assume $m \geq n$. The cost of CARRQR is equal to the cost of tournament pivoting plus the cost of CAQR. In the table, $f(m,n,b,P_r,P_c) = \frac{4(2mnb-n^2b)}{P_r}\log_2 P_c + \frac{16nb^2}{3}\log_2 P_r \log_2 P_c + 11nb^2\log_2 P_c$.

We recall now briefly the lower bounds on communication from [3], which apply to QR under certain hypotheses. On a sequential machine with fast memory of size M and slow memory, a lower bound on the volume of data and on the number of messages transferred between fast and slow memory during the QR factorization of a matrix of size $m \times n$ is

$$\#\text{ words} \geq \Omega\left(\frac{mn^2}{\sqrt{M}}\right), \quad \#\text{ messages} \geq \Omega\left(\frac{mn^2}{M^{3/2}}\right). \tag{3.1}$$

When the matrix is distributed over P processors, the size of the memory per processor is on the order of $O(mn/P)$, and the work is balanced among the P processors, then a lower bound on the volume of data and the number of messages that at least one of the processors must transfer is

$$\#\text{ words} \geq \Omega\left(\sqrt{\frac{mn^3}{P}}\right), \quad \#\text{ messages} \geq \Omega\left(\sqrt{\frac{nP}{m}}\right). \tag{3.2}$$

To minimize communication in parallel CARRQR, we use an optimal layout, with the same values as in [10],

$$P_r = \sqrt{\frac{mP}{n}} \ , \ P_c = \sqrt{\frac{nP}{m}} \ \text{and} \ b = B \cdot \sqrt{\frac{mn}{P}}. \tag{3.3}$$

The number of flops required by tournament pivoting with $b_T = 2b$, which does not include the number of flops required for computing the QR factorization once the pivot columns have been identified, is

$$\begin{aligned} \#flops= & \frac{4mn^2 - 4n^3/3}{P} + \\ + \quad & \frac{mn^2}{P}\left(8B\left(\frac{1}{3}\log_2 P_r + \log_2 P_c + \frac{2}{3}\right) + 32B^2\left(\frac{1}{6}\log_2 P_r \log_2 P_c + \frac{1}{3}\log_2 P_c\right)\right) \end{aligned}$$

By choosing $B = 8^{-1}\log_2^{-1}(P_r)\log_2^{-1}(P_c)$, the first term in $\#flops$ dominates the redundant computation due to tournament pivoting. This is a term we cannot decrease. After dropping some of the lower order terms, the counts of the entire parallel CARRQR factorization become

$$\begin{aligned} \#flops &\approx \frac{6mn^2 - 6n^3/3}{P} + cmn^2, \ \ c < 1, \\ \#words &\approx 2\frac{\sqrt{mn^3}}{\sqrt{P}}\left(\log_2\sqrt{\frac{mP}{n}} + \log_2\sqrt{\frac{nP}{m}}\right), \\ \#messages &\approx 2^7\sqrt{\frac{nP}{m}}\log_2^2\sqrt{\frac{mP}{n}}\log_2^2\sqrt{\frac{nP}{m}}, \end{aligned}$$

and this shows that parallel CARRQR is communication optimal, modulo polylogarithmic factors.

Appendix B describes briefly the performance model of a sequential communication avoiding QR. It shows that sequential CARRQR performs three times more flops than QRCP, transfers $\Theta(mn^2/M^{1/2})$ number of words and exchanges $\Theta(mn^2/M^{3/2})$ number of messages between the slow memory and the fast memory of size M of the sequential computer. Thus sequential QR attains the lower bounds on communication. As explained in the introduction, neither sequential QRCP nor parallel QRCP are communication optimal. We note however that parallel QRCP minimizes the volume of communication, but not the number of messages.

4. Experimental results. In this section we discuss the numerical accuracy of our CARRQR and compare it with the classic QR factorization with column pivoting (QRCP) and the singular value decomposition. We focus in particular on the value of the elements obtained on the diagonal of the upper triangular factor R (referred to as R-values) of the factorization $A\Pi = QR$, and compare them with the singular values of the input matrix. For the CARRQR factorization, we test binary-tree-based CARRQR and flat-tree-based CARRQR, denoted in the figures as CARRQR-B and CARRQR-F respectively. The singular values are computed by first performing a QR factorization with column pivoting of A, and then computing the singular values of the upper triangular factor R with the highly accurate routine DGESVJ [13, 14]. We use this approach to compute the singular values since given a matrix $A = DY$ (or $A = YD$), where D is diagonal and Y is reasonably well conditioned,

the Jacobi SVD algorithm in DGESVJ computes the singular values with relative error of at most $O(\epsilon)\kappa(Y)$, whereas the SVD algorithm based on the QR iteration has a relative error of at most $O(\epsilon)\kappa(A)$. Here $\kappa(A)$ is the condition number of A, $\kappa(A) = ||A||_2 \cdot \left|\left|A^{-1}\right|\right|_2$. This choice affects in particular the accuracy of the small singular values. Not all our input matrices A have the scaling property that A can be expressed as the multiplication of a diagonal matrix and a well-conditioned matrix, but after computing $A\Pi = QR$ by using a rank revealing QR factorization and $R = DY$, we expect to obtain a well conditioned Y. Even when Y is not well conditioned, it is still expected that DGESVJ of R is more accurate than SVD of R.

In the plots displaying R-values and singular values, we also display bounds for trustworthiness computed as,

$$\epsilon \min\{||(A\Pi_0)(:,i)||_2, ||(A\Pi_1)(:,i)||_2, ||(A\Pi_2)(:,i)||_2\} \tag{4.1}$$

$$\epsilon \max\{||(A\Pi_0)(:,i)||_2, ||(A\Pi_1)(:,i)||_2, ||(A\Pi_2)(:,i)||_2\} \tag{4.2}$$

where $\Pi_j (j = 0, 1, 2)$ are the permutation matrices obtained by QRCP, CARRQR-B, and CARRQR-F (represented by the subscripts $j = 0, 1, 2$ respectively), $(A\Pi)(:, i)$ is the i-th column of the permuted matrix $A\Pi$, and ϵ is the machine precision. Since the algorithm can be interpreted as applying orthogonal transformations to each column separately, machine epsilon times the norm of column i is a reasonable estimate of the uncertainty in any entry in that column of R, including the diagonal, our estimate of σ_i. Therefore the quantities in (4.1) and (4.2) describe the range of uncertainties in σ_i, for all three choices of column i, from the three pivoting schemes tested.

For each matrix in our test set we display the ratio $R(i, i)/\sigma_i$, where R is the upper triangular factor obtained by QRCP, CARRQR-B or CARRQR-F, $R(i, i)$ denotes the i-th diagonal element of R, assumed to be nonnegative, and σ_i is the i-th singular value computed as described above. Consider the factorization $A\Pi = QR$ obtained by using QRCP, CARRQR-B, or CARRQR-F. Let $D = diag(diag(R))$ and define Y by $R = DY^T$. Then we have $A\Pi = QDY^T$. The R-values of QRCP decrease monotonically, and it follows straightforwardly from the minimax theorem that the ratio $R(i, i)/\sigma_i$ can be bounded as

$$\frac{1}{||Y||} \leq \frac{R(i,i)}{\sigma_i} \leq ||Y^{-1}||, \tag{4.3}$$

and these lower and upper bounds are also displayed in the corresponding plots.

As described in [26], if after a first factorization $A\Pi = QR$, a second QR factorization of R^T is performed, $R^T\Pi_1 = Q_1L^T$, then the diagonal elements of the lower triangular factor L approximate more accurately the singular values of the input matrix. The obtained factorization $A = Q\Pi_1LQ_1^T\Pi_T$ is called a QLP factorization, a special case of ULV decomposition, where Q and Q_1 are orthogonal matrices that approximate the left and right singular subspaces. Stewart notes that there is no need to perform pivoting in the second QR factorization, and hence a communication optimal (modulo polylogarithnic factors) QLP factorization can be obtained by using QR with tournament pivoting for the first factorization, and then CAQR for the second factorization. In our numerical tests we use the QLP factorization only for two matrices, the devil's stairs and Kahan matrix, two challenging matrices for rank-revealing factorizations also used in [26]. For the other matrices, we test QRCP, CARRQR-B, and CARRQR-F. The entire set of matrices is presented in Table 4.1. Unless otherwise specified, the matrices are of size 256×256 and the block size is

$b = 8$. The binary tree version of CARRQR uses a tournament that has $2b$ columns at the leaves of the reduction tree. These matrices were used in several previous papers focusing on rank revealing factorizations, as [4, 20, 26, 23]. A short description is given in the table. In addition, we describe in more detail here matrices BREAK-1, BREAK-9, H-C, and STEWART, which are random matrices of size $n \times n$ with prescribed singular values $\{\sigma_i\}$. The first three matrices are of the form $A = U\Sigma V^T$, where U, V are random orthogonal matrices and Σ is a diagonal matrix. For BREAK-1, Σ has the following diagonal entries $\sigma_1 = \ldots = \sigma_{n-1} = 1, \sigma_n = 10^{-9}$ [4]. For BREAK-9, the diagonal entries of Σ are $\sigma_1 = \ldots = \sigma_{n-9} = 1, \sigma_{n-8} = \ldots = \sigma_n = 10^{-9}$ [4]. For H-C, Σ has diagonal entries 100, 10, and the following $n-2$ are evenly spaced between 10^{-2} and 10^{-8} [23]. For STEWART, $A = U\Sigma V^T + 0.1\sigma_{50}E$ [26], where Σ is a diagonal matrix with the first 50 diagonals decreasing geometrically from 1 to 10^{-3} and the last $n-50$ diagonals being set to zero, U and V are random orthogonal matrices, E is a matrix with random entries chosen from a uniform distribution in the interval $(0, 1)$. In MATLAB notation, d=linspace(1,1e-3,n); d(51:end)=0; A=orth(rand(n))*diag(d)*orth(rand(n))+0.1*d(50)*rand(n) [1].

We first discuss the Kahan matrix and the devil's stairs. The Kahan matrix is presented in equation (4.4) where $c^2 + s^2 = 1$. For $c = 0$, the singular values are $s^i, i = 0, \ldots, n-1$. An increasing gap between the last two singular values is obtained when the value of c is increased. Since $\sigma_{\min}(R_{11}) \leq \sigma_k(A)$ in (1.1), we would like to find Π such that $\sigma_{\min}(R_{11})$ is sufficiently large. For the Kahan matrix, it is known that $\sigma_k(A)/\sigma_{\min}(R_{11}) \geq \frac{1}{2}c^3(1+c)^{n-4}/s$, and $\sigma_{\min}(R_{11})$ can be much smaller than $\sigma_k(A)$ [20]. Traditional QR with column pivoting does not permute the columns of A in exact arithmetic and fails to reveal the gap between the last two singular values. It is easy to notice that CARRQR does not permute the columns in exact arithmetic either, if we assume that during tournament pivoting, ties are broken by choosing the leftmost column when multiple columns have the same norm. This results in poor rank revealing, similarly to QRCP. However in finite precision, both QRCP and CARRQR might permute the columns of the matrix and in this case both reveal the rank, as the results displayed in Table 4.2 show, where we compare the last two singular values with the last two L-values and R-values obtained from QRCP, CARRQR-B, and CARRQR-F.

To avoid a possible non-trivial permutation in finite precision (a well-known phenomenon [12] of the Kahan matrix), we multiply the j-th column of the Kahan matrix by $(1-\tau)^{j-1}$ for all j and a small $\tau \gg \epsilon$. The additional numerical experiments we have performed revealed that when τ is small, all three algorithms, QRCP, CARRQR-B, and CARRQR-F, pivot the columns of A and are effective in revealing the rank. However, when the parameter τ is increased, for example for $\tau = 10^{-7}$, the three algorithms do not perform any pivoting, and result in poor rank revealing.

$$A = \begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & s & 0 & \cdots & \cdots & 0 \\ 0 & 0 & s^2 & \ddots & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \cdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & 0 & s^{n-1} \end{pmatrix} \begin{pmatrix} 1 & -c & -c & \cdots & \cdots & -c \\ 0 & 1 & -c & \cdots & \cdots & -c \\ 0 & 0 & 1 & \ddots & \cdots & -c \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \cdots & \ddots & \ddots & -c \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}. \tag{4.4}$$

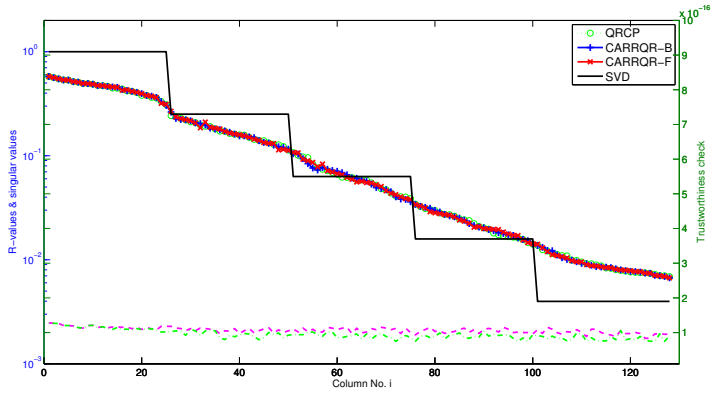
[1] It is not exactly the matrix used in Figure 2.1 in [26].

No.	Matrix	Descriptions
1	BAART	Discretization of the 1st kind Fredholm integral equation [21].
2	BREAK-1	Break 1 distribution, matrix with prescribed singular values, see description in the text and [4].
3	BREAK-9	Break 9 distribution, matrix with prescribed singular values, see description in the text and [4].
4	DERIV2	Computation of second derivative [21].
5	EXPONENTIAL	Exponential Distribution, $\sigma_1 = 1$, $\sigma_i = \alpha^{i-1}$ $(i = 2, \ldots, n)$, $\alpha = 10^{-1/11}$ [4].
6	FOXGOOD	Severely ill-posed test problem [21].
7	GKS	An upper-triangular matrix whose j-th diagonal element is $1/\sqrt{j}$ and whose (i, j) element is $-1/\sqrt{j}$, for $j > i$ [17, 20].
8	GRAVITY	1D gravity surveying problem [21].
9	H-C	Matrix with prescribed singular values, see description in the text and [23].
10	HEAT	Inverse heat equation [21].
11	PHILLIPS	Phillips' famous test problem [21].
12	RANDOM	Random matrix $A = 2 * rand(n) - 1$ [20].
13	SCALE	Scaled random matrix, a random matrix whose i-th row is scaled by the factor $\eta^{i/n}$ [20]. We choose $\eta = 10 \cdot \epsilon$.
14	SHAW	1D image restoration model [21].
15	SPIKES	Test problem with a "spiky" solution [21].
16	STEWART	Matrix $A = U\Sigma V^T + 0.1\sigma_{50} * rand(n)$, see description in the text and [26].
17	URSELL	Integral equation with no square integrable solution [21].
18	WING	Test problem with a discontinuous solution [21].
19	KAHAN	Kahan matrix, see equation (4.4).
20	DEVIL	The devil's stairs, a matrix with gaps in its singular values, see Algorithm 2 in Appendix A [26].

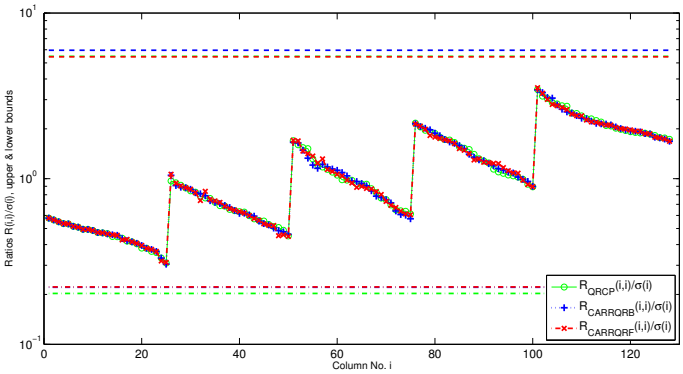
TABLE 4.1
Test matrices.

The devil's stairs [26] is a matrix with multiple gaps in its singular values, generated by the MATLAB code given in Algorithm 2 in Appendix A. From Figure 4.1, we can see that the L-values reveal the gaps in the singular values, while the R-values do not. Both our versions of CARRQR factorizations provide a good first step for the QLP factorization, similar to the traditional QR with column pivoting.

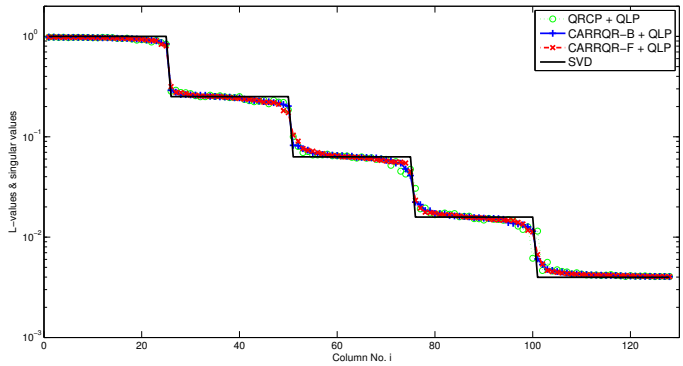
We now discuss the other matrices in our test set, given in alphabetical order in Table 4.1. The results for the first matrix, BAART, are given in Figure 4.2. Figure 4.2(a) shows the R-values of QR with column pivoting (QRCP), CARRQR-B, and CARRQR-F, and the singular values of this matrix, while Figure 4.2(b) displays the ratios of R-values to singular values. In Figure 4.2(a), the curves for R-values and singular values are almost superimposed, and they are well above the bounds of trustworthiness, except for the last few elements. Hence the R-values reveal the rank for this matrix. The ratios $R(i, i)/\sigma_i(R)$ shown in Figure 4.2(b) are close to 1, and well bounded by the formulas in equation (4.3). For all the other matrices in our test set, similar plots with similar behaviour are given in Appendix A, except that the R-values of a few matrices as FOXGOOD lie under the trustworthiness bounds given



(a) R-values of QRCP and CARRQR



(b) Ratios



(c) L-values of QRCP and CARRQR

FIG. 4.1. *The devil's stairs, matrix of size* 128×128. *The roughly horizontal dotted lines in (a) stand for the upper and lower bounds given in* (4.1) *and* (4.2)*; the horizontal dotted lines in (b) denote the upper and lower bounds from* (4.3)*, where the Y factors are obtained from QRCP, CARRQR-B, and CARRQR-F respectively.*

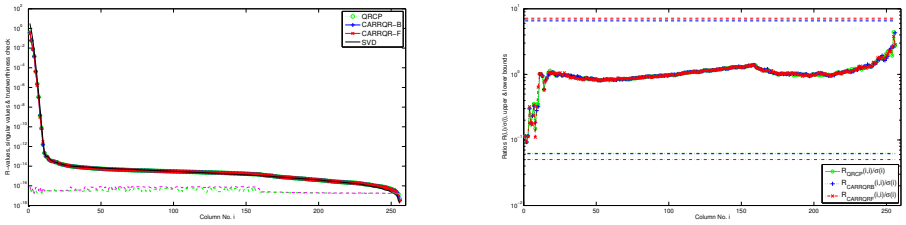
c	singular values		QRCP		CARRQR-B		CARRQR-F	
	σ_{n-1}	σ_n	r_{n-1}/σ_{n-1}	r_n/σ_n	r_{n-1}/σ_{n-1}	r_n/σ_n	r_{n-1}/σ_{n-1}	r_n/σ_n
0.1	5.57E-01	5.71E-06	1.0488	2.4004	1.0488	2.4004	1.0488	2.4004
0.2	8.37E-02	1.26E-11	1.0954	7.7787	1.0954	7.7787	1.0954	7.7787
0.3	3.00E-03	1.59E-17	1.1402	1.5650	1.1402	1.5650	1.1402	1.5650
0.4	2.01E-05	7.96E-24	1.1832	2.8006	1.1832	1.4289	1.1832	1.4350
0.5	1.65E-08	9.26E-31	1.2247	2.0125	1.2247	2.0014	1.2247	2.0021
0.6	7.79E-13	1.06E-38	1.2649	1.2810	1.2649	1.2894	1.2649	1.2811
c	singular values		QRCP		CARRQR-B		CARRQR-F	
	σ_{n-1}	σ_n	l_{n-1}/σ_{n-1}	l_n/σ_n	l_{n-1}/σ_{n-1}	l_n/σ_n	l_{n-1}/σ_{n-1}	l_n/σ_n
0.1	5.57E-01	5.71E-06	1.0223	1.0000	1.0223	1.0000	1.0223	1.0000
0.2	8.37E-02	1.26E-11	1.0392	1.0000	1.0392	1.0000	1.0392	1.0000
0.3	3.00E-03	1.59E-17	1.0512	1.0000	1.0512	1.0000	1.0512	1.0000
0.4	2.01E-05	7.96E-24	1.0583	1.0000	1.0583	1.0000	1.0583	1.0043
0.5	1.65E-08	9.26E-31	1.0607	1.0000	1.0607	0.9945	1.0607	0.9948
0.6	7.79E-13	1.06E-38	1.0583	1.0000	1.0583	1.0065	1.0583	1.0000

TABLE 4.2

Kahan matrix (of size $n = 128$), comparison between last two singular values, R-values, and L-values. In the table, r_{n-1}, r_n are the last two R-values, and l_{n-1}, l_n are the last two L-values.

in equations (4.1) and (4.2).

Figures 4.3, 4.4, and 4.5 summarize these results. Figure 4.3 presents the ratios of the R-values obtained from the different QR factorizations to the singular values. Let r be the vector of R-values obtained from QRCP/CARRQR, and let s be the vector of singular values. The R-values obtained from CARRQR are not in descending order and we do not sort them. The values displayed in Figure 4.3 are the minimum of the ratios $min(r./s)$, the maximum of the ratios $max(r./s)$, and the median of the ratios $median(r./s)$. For all the test cases, we can see that the ratios are close to 1, and are well bounded as in equation (4.3). We conclude that the R-values obtained by communication avoiding RRQR reveal the rank for all the matrices in our test set.



(a) R-values (b) Ratios

FIG. 4.2. *R-values and ratios of QR with column pivoting (QRCP), binary-tree-based CARRQR (CARRQR-B), and flat-tree-based CARRQR (CARRQR-F) of* BAART *matrix. The dotted lines in (a) display the upper and lower bounds given in* (4.1) *and* (4.2)*; the dotted lines in (b) represent the upper and lower bounds in* (4.3)*, where the Y factors are obtained from QRCP, CARRQR-B, and CARRQR-F respectively.*

In the plots displaying R-values and singular values in Figure 4.2 and in Appendix A, it might be hard to distinguish if the computed R-values are above or below the bounds of trustworthiness from equations (4.1) and (4.2). For the purpose of clarity, we display in Figure 4.4 the ratio $\min_i\{\frac{R(i,i)}{\epsilon||(A\Pi)(:,i)||_2}\}$ for the first 18 matrices in our test set. The R factor is obtained from QRCP, CARRQR-B, and CARRQR-F. It can be seen that for the matrices numbered 1, 5 to 8, 10, 14, 15, and 18, this ratio is smaller than 1. But as displayed in Figure 4.3, the ratios $R(i,i)/\sigma_i(R)$ are close to 1,

and well bounded by equation (4.3). In other words, even when $\frac{R(i,i)}{\epsilon||(A\Pi)(:,i)||}$ is small and we do not expect an accurate computation of the R-values due to round-off error, the computed R-values are still very close to the accurately computed singular values of the R matrix.

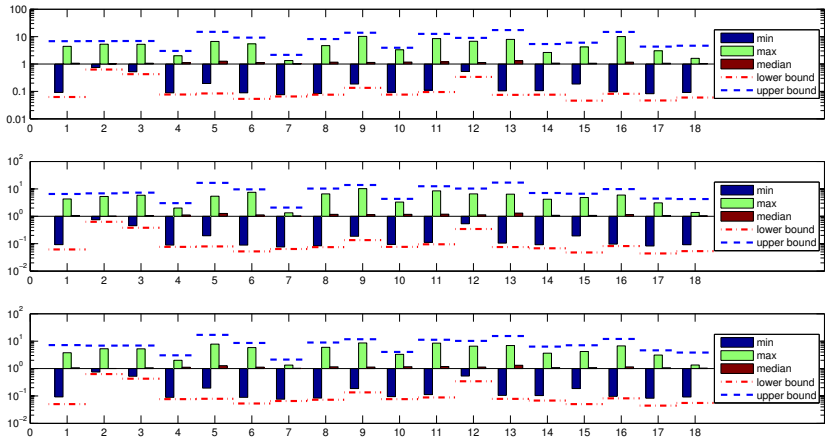


FIG. 4.3. *Minimum, maximum, and median of the ratios $R(i,i)/\sigma_i(R)$, where $i = 1, \ldots, n$ and n is the size of the input matrix. The dotted horizontal lines represent the upper and lower bounds from equation* (4.3)*. The R factor is obtained from QRCP (top plot), CARRQR-B (middle plot), and CARRQR-F (bottom plot).*

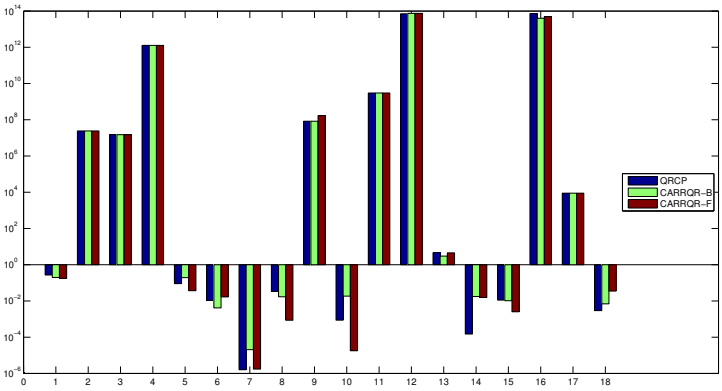


FIG. 4.4. *Ratios $\min_{i=1:n}\{\frac{R(i,i)}{\epsilon||(A\Pi)(:,i)||_2}\}$ for the first* 18 *matrices in our test set, where n is the size of the input matrix. The R factor is obtained from QRCP (left column), CARRQR-B (middle column), and CARRQR-F (right column).*

As explained earlier, tournament pivoting used in CARRQR does not guarantee that the R-values decrease monotonically, while QR with column pivoting does. The

formula (4.3) should be modified for CARRQR as

$$\frac{1}{||Y||} \leq \frac{\rho_i}{\sigma_i} \leq ||Y^{-1}||,$$

where ρ_i is the i-th largest diagonal of the R-factor of CARRQR, that is, the i-th value of the sorted R-values. Since the CARRQR factorizations have good rank-revealing properties in practice, we can expect that (4.3) still holds approximately for CARRQR, even without this modification. And this is indeed verified by the numerical results in Figure 4, where the ratios $R(i,i)/\sigma_i$ are bounded by the bounds in (4.3) for all our test cases. To check the monotonicity of the R-values, Figure 4.5 displays the minimum, the maximum, and the median of the rations $|R(i+1,i+1)|/|R(i,i)|$, where $i = 1, \ldots, n-1$ and n is the size of the input matrix. The factor R is obtained from RRQR with column pivoting (top plot), CARRQR-B (middle plot), and CARRQR-F (bottom plot). It can be seen that for both CARRQR-B and CARRQR-F, the ratios are well below 1 except for a very few cases.

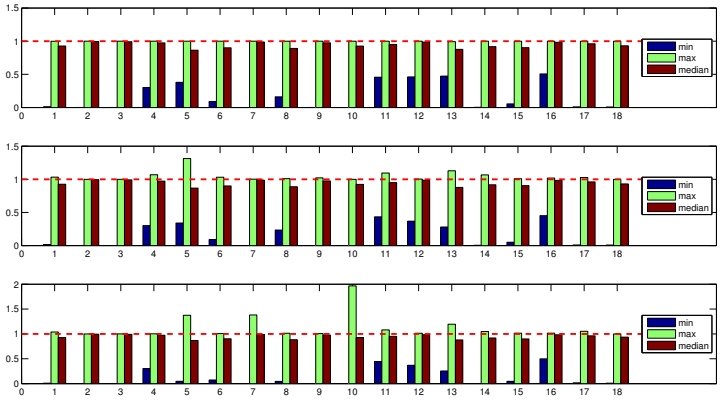


FIG. 4.5. *Ratios of succesive R-values, $|R(i+1,i+1)|/|R(i,i)|$, for $i = 1, \ldots, n-1$ and n is the size of the matrix. The R factor is obtained from QRCP (top plot), CARRQR-B (middle plot), and CARRQR-F (bottom plot).*

5. Tournament pivoting for other factorizations. In this section we describe very briefly how the tournament pivoting can be extended to other factorizations that require some form of pivoting.

Cholesky with diagonal pivoting. Since QR with column pivoting applied to A is mathematically equivalent to Cholesky with diagonal pivoting applied to $A^T A$ (in exact arithmetic), the same general techniques as for RRQR can be used in this context. But now the leaves in our reduction tree are b-by-b diagonal subblocks of the symmetric positive-definite matrix H to which we want to apply Cholesky with pivoting, and at each internal node of the (binary) reduction tree we take the $2b$-by-$2b$ diagonal submatrix enclosing the two b-by-b matrices from the child nodes, and perform some kind of Cholesky with diagonal pivoting on this $2b$-by-$2b$ matrix to select the best b-by-b submatrix to pass up the tree.

Rank-revealing decompositions of products/quotients of matrices. The goal is to compute a rank-revealing QR-like decomposition of an arbitrary product $P = A_1^{\pm 1} \cdot$

$A_2^{\pm 1} \cdots A_k^{\pm 1}$ without actually multiplying or inverting any of the factors A_i. This problem arises, for example, in algorithms for eigenproblems and the SVD that either run asymptotically as fast as fast matrix multiplication ($O(n^\omega)$ for some $\omega < 3$), or minimize communication.

When $P = A_1$, the communication avoiding RRQR algorithm discussed in this paper can be used. But when $P = A_1^{-1}$, or $k > 1$ matrices are involved, the only solution available so far involves randomization, i.e. computing the regular QR decomposition of PV where V is a random orthogonal/unitary matrix. This can be obtained by computing only QR factorizations and multiplying by orthogonal matrices, and without multiplying or inverting other matrices, and so it is stable. The resulting factorization is of the form $P = Q\hat{R}V$ where V is random orthogonal, Q is orthogonal, and $\hat{R}$ is represented as a product of R factors and their inverses. The useful property is that the leading columns of Q do indeed span the desired subspaces with high probability. This approach is used for designing communication-avoiding eigendecompositions and singular value decompositions [2].

GECP - LU with complete pivoting. An approach to extend tournament pivoting to GECP is to perform each panel factorization as following. Tournament pivoting uses RRQR to pick the next best b columns to use, and then uses TSLU on these columns to pick their best b rows. Then the resulting b-by-b submatrix is permuted to the upper left corner of the matrix, and b steps of LU with no pivoting are performed. Then the process repeats on the trailing submatrix. The intuition is that by picking the best b columns, and then the best b rows from these columns, we are in effect picking the best b-by-b submatrix overall, which is then permuted to the upper left corner. The communication costs are clearly minimal, since it just uses previously studied components.

LDL^T factorization with pivoting. The challenge here is preserving symmetry. A possible usage of tournament pivoting is the following:

1. Use the approach for GECP above to find the "best" b-by-b submatrix S of the symmetric matrix A, whether or not it is a principal submatrix (i.e. lies in the same set of b rows and b columns of A). If S is a principal submatrix, symmetrically permute it to lie in the first b rows and columns of A, and use it to perform b steps of symmetric LU without pivoting; otherwise continue to step 2.
2. Expand S to be a smallest-possible principal submatrix of A, called T. The dimension of T can be from $b+1$ up to $2b$, depending on how much S overlaps the diagonal of A.
3. Find a well-conditioned principal submatrix W of T of dimension d where $b \leq d \leq 2b$, permute it into the top left corner of A, and perform d steps of symmetric LU as above.

The conjecture is that such a well-conditioned principal submatrix W of T must exist. Here is a sketch of an algorithm to find W. The b largest singular values of T are at least as large as the b singular values of S, by the interlacing property. Write the eigendecomposition $T = Q\Lambda Q^T = [Q_1, Q_2]\mathrm{diag}(\Lambda_1, \Lambda_2)[Q_1, Q_2]^T$ where Λ_1 has the large eigenvalues, and Λ_2 has the small ones. Do GEPP on Q_1 to identify the most independent rows, and choose these as pivot rows determining W. The above sketch may serve to prove that a well-conditioned principal submatrix W exist, under the condition that there is a big enough gap in the eigenvalues of T. But we may prefer to simply do more conventional LDL^T factorization with pivoting to find it, instead of using an eigendecomposition as above.

Of course one reason for using LDL^T factorization instead of LU, is that it traditionally takes half the storage and half the flops. It is not clear whether we can retain either of these advantages, or even how much the latter one still matters.

6. Conclusions. In this paper we introduce CARRQR, a communication optimal rank revealing QR factorization based on tournament pivoting. CARRQR does asymptotically less communication than the classic rank revealing QR factorization based on column pivoting, at the cost of performing a factor of 3 more floating point operations. We have shown through extensive numerical experiments on challenging matrices that the CARRQR factorization reveals the rank similarly to the QR factorization with column pivoting, and provides results close to the singular values computed with the highly accurate routine DGESVJ. Our future work will focus on implementing CARRQR and studying its performance with respect to current implementations of QR with column pivoting available in LAPACK and ScaLAPACK. We have also outlined how tournament pivoting extends to a variety of other pivoted matrix factorizations, as Cholesky with diagonal pivoting, LU with complete pivoting, or LDL^T factorization with pivoting.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. W. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, USA, 1999.

[2] Grey Ballard, James Demmel, and Ioana Dumitriu, *Minimizing communication for eigenproblems and the singular value decomposition*, Tech. Report UCB/EECS-2011-14, EECS Department, University of California, Berkeley, Feb 2011.

[3] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, *Minimizing communication in numerical linear algebra*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 866–901.

[4] C. H. Bischof, *A parallel QR factorization algorithm with controlled local pivoting*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 36–57.

[5] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. W. Demmel, I. Dhillon, J. J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*, SIAM, Philadelphia, PA, USA, May 1997.

[6] P. A. Businger and G. H. Golub, *Linear least squares solutions by Householder transformations*, Numer. Math., 7 (1965), pp. 269–276.

[7] T. F. Chan, *Rank revealing QR factorization*, Linear Algebra Appl., 88/89 (1987), pp. 67–82.

[8] T. F. Chan and P. C. Hansen, *Some applications of the rank revealing QR factorization*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 727–741.

[9] S. Chandrasekaran and I. C. F. Ipsen, *On Rank-Revealing Factorisations*, SIAM. J. Matrix Anal. Appl., 15 (1994), pp. 592–622.

[10] J. W. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-avoiding parallel and sequential QR and LU factorizations: theory and practice*, Tech. Report UCB/EECS-2008-89, University of California Berkeley, EECS Department, 2008. LAWN #204.

[11] J. W. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM J. Sci. Comput., 34 (2012), pp. 206–239. short version of technical report UCB/EECS-2008-89 from 2008.

[12] Z. Drmač and Z. Bujanović, *On the failure of rank revealing QR factorization software - a case study*, ACM Trans. Math. Softw., 35 (2008), pp. 1–28.

[13] Z. Drmač and K. Veselic, *New fast and accurate Jacobi SVD algorithm I*, SIAM J. Matrix Anal. Appl., 29 (2008), pp. 1322–1342.

[14] ———, *New fast and accurate Jacobi SVD algorithm II*, SIAM J. Matrix Anal. Appl., 29 (2008), pp. 1343–1362.

[15] L. V. Foster and X. Liu, *Comparison of rank revealing algorithms applied to matrices with well defined numerical ranks.* http://www.researchgate.net/publication/228523390_Comparison_of_rank_revealing_algorithms_applied_to_matrices_with_well_defined_nu

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

[17] G. H. Golub, V. Klema, and G. W. Stewart, *Rank degeneracy and least squares problems*, Tech. Report TR-456, Dept. of Computer Science, University of Maryland, 1976.
[18] L. Grigori, J. W. Demmel, and H. Xiang, *Communication avoiding Gaussian elimination*, Proceedings of the ACM/IEEE SC08 Conference, (2008).
[19] ———, *CALU: a communication optimal LU factorization algorithm*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 1317–1350.
[20] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.
[21] P. C. Hansen, *Regularization tools version 4.1 for matlab 7.3*.
[22] Y. P. Hong and C.-T. Pan, *Rank-revealing QR factorizations and the singular value decomposition*, Math. Comp., 58 (1992), pp. 213–232.
[23] D. A. Huckaby and T. F. Chan, *Stewart's pivoted QLP decomposition for low-rank matrices*, Numer. Lin. Algebra Appl., 12 (2005), pp. 153–159.
[24] A. Khabou, J. W. Demmel, L. Grigori, and M. Gu, *Communication avoiding LU factorization with panel rank revealing pivoting*, tech. report, INRIA TR 7867, 2012. submitted to SIMAX.
[25] G. Quintana-Ortí, X. Sun, and C. H. Bischof, *A BLAS-3 version of the QR factorization with column pivoting*, SIAM J. Sci. Comput., 19 (1998), pp. 1486–1494.
[26] G. W. Stewart, *The QLP approximation to the singular value decomposition*, SIAM J. Sci. Comput., 20 (1999), pp. 1336–1348.

7. Appendix A. We present additional numerical results for the matrices in our test set comparing the diagonal values of the R factor obtained after QRCP and CARRQR with the singular values of the input matrices.

Algorithm 2 Matlab code for generating the devil's stairs matrix

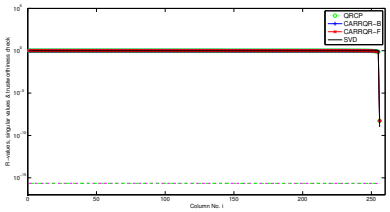
```
Length = 20;
s = zeros(n,1);
Nst = floor(n/Length);
for i = 1 : Nst do
   s(1+Length*(i-1):Length*i) = -0.6*(i-1);
end for
s(Length*Nst:end) = -0.6*(Nst-1);
s = 10. ∧ s;
A = orth(rand(n)) * diag(s) * orth(randn(n));
```

8. Appendix B. We present detailed performance counts for the communication avoiding RRQR algorithms introduced in section 3. In all the counts, we ignore lower order terms.

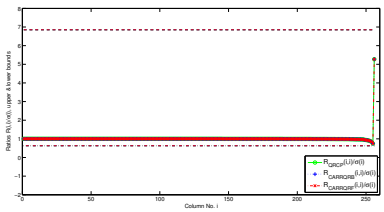
8.1. Performance counts of parallel QR with tournament pivoting. We consider that the input matrix is distributed block cyclically over a grid of $P = P_r \times P_c$ processors. The block size is b. We compute the additional flops introduced in the algorithm due to tournament pivoting. Hence the total cost of the algorithm is the cost of performing tournament pivoting plus the cost of performing CAQR. In the analysis, we give in detail the counts at each step j of the block algorithm, and $m_b = m - jb$, $n_b = n - jb$.

We consider first tournament pivoting with $b_T = n_b/P_c$, that is the matrices at the leaves of the reduction tree have n_b/P_c columns and as many rows as the trailing matrix. At the upper levels of the reduction tree, the matrices have $2b$ columns. Tournament pivoting performs the following operations.

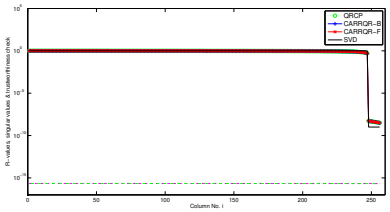
- Each column grid of P_r processors performs TSQR factorization of a matrix



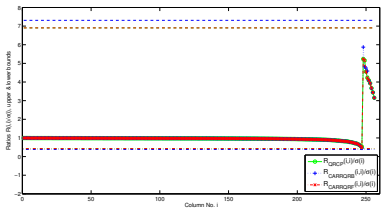
(a) Break-1 - R-values



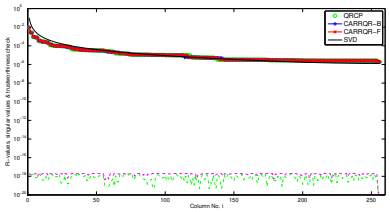
(b) Break-1 - ratios



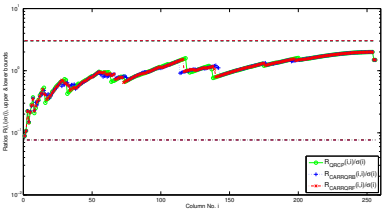
(c) Break-9 - R-values



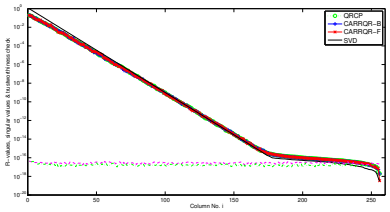
(d) Break-9 - ratios



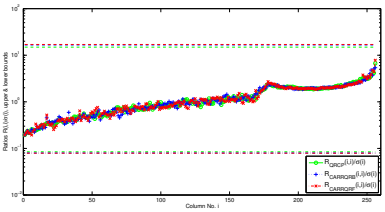
(e) Deriv2 - R-values



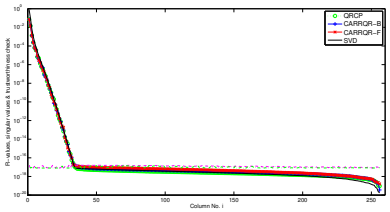
(f) Deriv2 - ratios



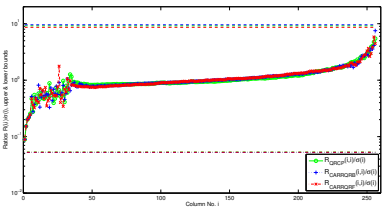
(g) Exponent - R-values



(h) Exponent - ratios

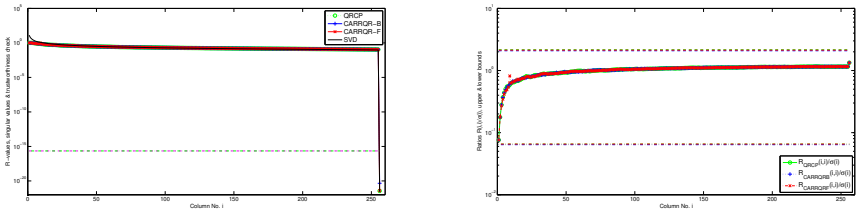


(i) Foxgood - R-values

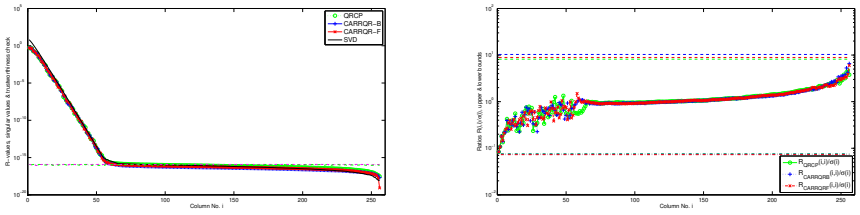


(j) Foxgood - ratios

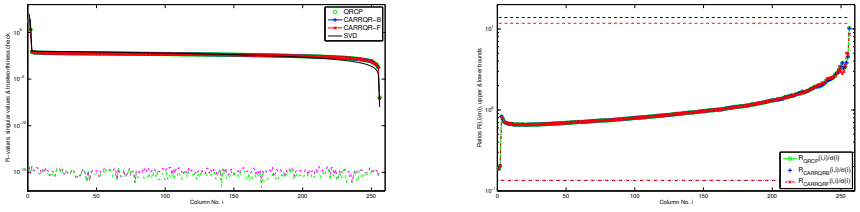
Fig. 7.1. *R-values of QR with column pivoting (QRCP), binary-tree-based CARRQR (CARRQR-B), and flat-tree-based CARRQR (CARRQR-F) and singular values for matrices in our test set.*



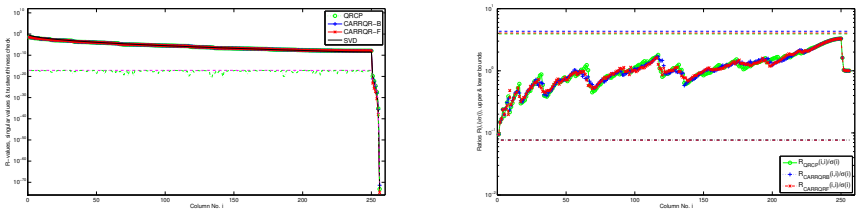
(a) GKS - R-values (b) GKS - ratios



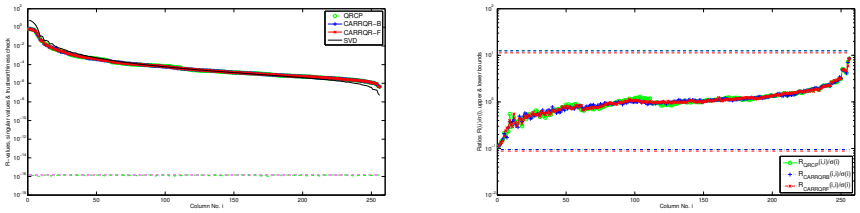
(c) Gravity - R-values (d) Gravity - ratios



(e) HC - R-values (f) HC - ratios

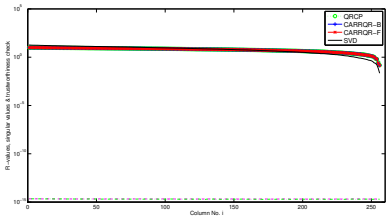


(g) Heat - R-values (h) Heat - ratios

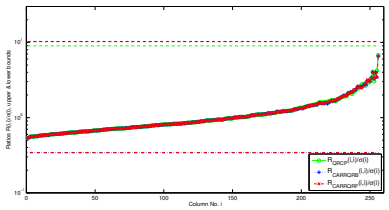


(i) Phillips - R-values (j) Phillips - ratios

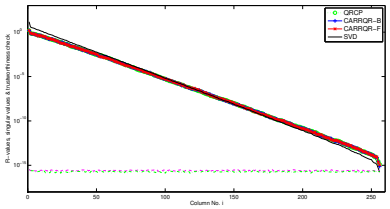
Fig. 7.2. *R-values of QR with column pivoting (QRCP), binary-tree-based CARRQR (CARRQR-B), and flat-tree-based CARRQR (CARRQR-F) and singular values for matrices in our test set.*



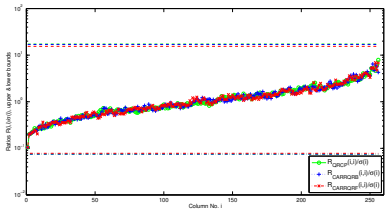
(a) RANDOM - R-values



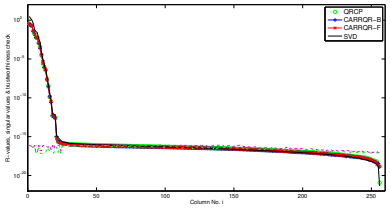
(b) RANDOM - ratios



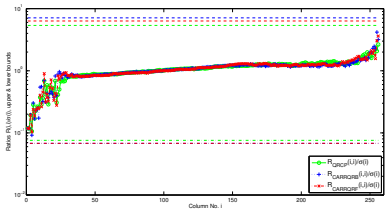
(c) SCALE - R-values



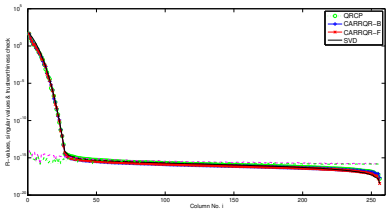
(d) SCALE - ratios



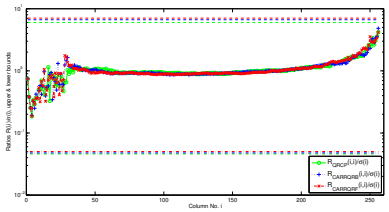
(e) SHAW - R-values



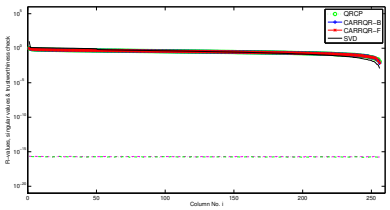
(f) SHAW - ratios



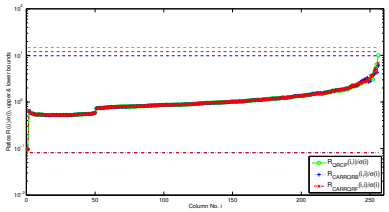
(g) SPIKES - R-values



(h) SPIKES - ratios

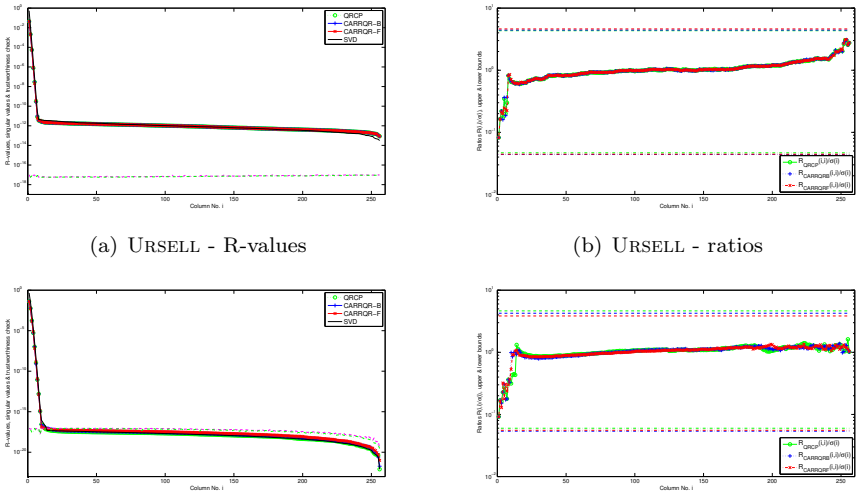


(i) STEWART - R-values



(j) STEWART - ratios

FIG. 7.3. *R-values of QR with column pivoting (QRCP), binary-tree-based CARRQR (CARRQR-B), and flat-tree-based CARRQR (CARRQR-F) and singular values for matrices in our test set.*



(a) URSELL - R-values (b) URSELL - ratios

(c) WING - R-values (d) WING - ratios

FIG. 7.4. *R-values of QR with column pivoting (QRCP), binary-tree-based CARRQR (CARRQR-B), and flat-tree-based CARRQR (CARRQR-F) and singular values for matrices in our test set.*

of size $m_b/P_r \times n_b/P_c$.

$$\#flops = \sum_{j=1}^{n/b} \left(\frac{2m_b n_b^2}{PP_c} + \frac{2n_b^3}{3P_c^3} \log_2 P_r \right) = \frac{2}{3} \frac{mn^2 - n^3/4}{P} \frac{n}{bP_c} + \frac{1}{6} \frac{n^3}{P_c^2} \frac{n}{bP_c} \log_2 P_r$$

$$\#words = \frac{n_b^2}{2P_c^2} \log_2 (P_r) \approx \frac{1}{6} \frac{n^2}{P_c} \frac{n}{bP_c} \log_2 (P_r)$$

$$\#messages = \frac{n}{b} \log_2 P_r$$

- Each processor performs locally QR with column pivoting of a matrix of size $n_b/P_c \times n_b/P_c$ to select b local pivot columns.

$$\#flops = \sum_{j=1}^{n/b} \frac{4n_b^3}{3P_c^3} = \frac{1}{3} \frac{n^3}{P_c^2} \frac{n}{bP_c}$$

- Reduce operation: for each level of the reduction tree of depth $log_2 P_c$ perform:
  - Processors in the same row grid exchange their pivot columns

$$\#words = \sum_{j=1}^{n/b} \frac{m_b b}{P_r} \log_2 P_c \approx \frac{mn - n^2/2}{P_r} \log_2 P_c$$

$$\#messages = \frac{n}{b} \log_2 P_c$$

  - Processors in the same column grid perform TSQR factorization of ma-

trices of size $m_b \times 2b$.

$$\#flops = \sum_{j=1}^{n/b} \left(\frac{2m_b(2b)^2}{P_r} + \frac{2(2b)^3}{3} \log_2 P_r \right) \log_2 P_c = \frac{4(2mnb - n^2 b)}{P_r} \log_2 P_c + \frac{16nb^2}{3} \log_2 P_r \log_2 P_c$$

$$\#words = \sum_{j=1}^{n/b} \frac{4b^2}{2} \log_2 (P_r) \log_2 (P_c) = 2nb \log_2 (P_r) \log_2 (P_c)$$

$$\#messages = \frac{n}{b} \log_2 (P_r) \log_2 (P_c)$$

– Processors perform QR with column pivoting factorization of the matrix of size $2b \times 2b$ of the R factor obtained from TSQR at each node of the reduction tree in the current level.

$$\#flops = \sum_{j=1}^{n/b} \frac{4(2b)^3}{3} \log_2 P_c \approx 11nb^2 \log_2 P_c$$

- The b pivot columns are permuted in the diagonal positions,

$$\#words = \sum_{j=1}^{n/b} \frac{m_b b}{P_r} = \frac{mn - n^2/2}{P_r}$$

$$\#messages = \frac{n}{b} \log_2 P_c$$

Hence the additional cost introduced by tournament pivoting is:

$$\#flops = \frac{2}{3} \frac{mn^2 - n^3/4}{P} \frac{n}{bP_c} + \frac{1}{6} \frac{n^3}{P_c^2} \frac{n}{bP_c} (\log_2 P_r + 2)$$
$$+ \frac{4(2mnb - n^2 b)}{P_r} \log_2 P_c + \frac{16nb^2}{3} \log_2 P_r \log_2 P_c + 11nb^2 \log_2 P_c$$

$$\#words = \frac{1}{6} \frac{n^2}{P_c} \frac{n}{bP_c} \log_2 (P_r) + \frac{mn - n^2/2}{P_r} (\log_2 P_c + 1) + 2nb \log_2 (P_r) \log_2 (P_c)$$

$$\#messages = \frac{n}{b} (\log_2 (P_r) + 2 \log_2 (P_c) + \log_2 (P_r) \log_2 (P_c))$$

We analyze now the algorithm that uses tournament pivoting on column blocks of size $2b$ at each step of the factorization. At each step of the block algorithm, tournament pivoting performs the following operations:

- At the leaves of the reduction tree, the processors in each column grid perform TSQR factorization of matrices of size $m_b \times 2b$, followed by the QR with

column pivoting of the R factor of size $2b \times 2b$, to identify b pivot columns.

$$\begin{aligned}
\#flops &\leq \sum_{j=1}^{n/b} \left(\frac{2m_b 4b^2}{P_r} + \frac{2(2b)^3}{3} \log_2(P_r) + \frac{32}{3} b^3 \right) \frac{n_b}{2bP_c} \\
&\approx \frac{2mn^2 - 2n^3/3}{P} + \frac{4n^2 b}{3P_c} (\log_2(P_r) + 2) \\
\#words &= \sum_{j=1}^{n/b} 2b^2 \frac{n_b}{2bP_c} \log_2(P_r) = \frac{n^2}{2P_c} \log_2(P_r) \\
\#messages &= \sum_{j=1}^{n/b} \frac{n_b}{2bP_c} \log_2(P_r) = \frac{n^2}{4b^2 P_c} \log_2(P_r)
\end{aligned}$$

- For each level of the reduction tree of depth $\log_2\left(\frac{n_b}{2b}\right)$ do
  - Processors in the same row grid exchange their portions of the previously selected b pivot columns. This communication is performed only in the last $\log_2(P_c)$ levels of the reduction tree, since in the previous levels the columns reside on a same processor.

$$\begin{aligned}
\#words &\leq \sum_{j=1}^{n/b} \frac{m_b}{P_r} b \log_2(P_c) = \frac{mn - n^2/2}{P_r} \log_2(P_c) \\
\#messages &\leq \frac{n}{b} \log_2(P_c)
\end{aligned}$$

  - Processors in the same column grid perform TSQR factorization of matrices of size $m_b \times 2b$, followed by the QR with column pivoting of the R factor of size $2b \times 2b$, to identify b pivot columns. The cost at each level is the same as the cost at the leaves of the reduction tree.

$$\begin{aligned}
\#flops &\leq \sum_{j=1}^{n/b} \sum_{i=1}^{\log_2(n_b/(2bP_c))} \left(\frac{2m_b 4b^2}{P_r} + \frac{2(2b)^3}{3} \log_2(P_r) + \frac{32}{3} b^3 \right) \frac{1}{2^i} \frac{n_b}{2bP_c} + \\
&\quad + \sum_{j=1}^{n/b} \left(\frac{2m_b 4b^2}{P_r} + \frac{2(2b)^3}{3} \log_2(P_r) + \frac{32}{3} b^3 \right) \log_2(P_c) \\
&\approx \frac{2mn^2 - 2n^3/3}{P} + \frac{4n^2 b}{3P_c} (\log_2(P_r) + 2) + \\
&\quad + \frac{4(2mnb - n^2 b)}{P_r} \log_2(P_c) + \frac{16}{3} nb^2 \log_2(P_r) \log_2(P_c) + \frac{32}{3} nb^2 \log_2(P_c) \\
\#words &= \frac{n^2}{2P_c} \log_2(P_r) + 2nb \log_2(P_r) \log_2(P_c) \\
\#messages &= \frac{n^2}{4b^2 P_c} \log_2(P_r) + \frac{n}{b} \log_2(P_r) \log_2(P_c)
\end{aligned}$$

- The b pivot columns are permuted in the diagonal positions.

$$\#words = \sum_{j=1}^{n/b} \frac{m_b b}{P_r} = \frac{mn - n^2/2}{P_r}$$
$$\#messages = \frac{n}{b} \log_2 P_c$$

Hence the total cost of tournament pivoting is:

$$\begin{aligned}
\#flops &= \frac{4mn^2 - 4n^3/3}{P} + \frac{8n^2 b}{3P_c}(\log_2(P_r) + 2) + \\
&\quad + \frac{4(2mnb - n^2 b)}{P_r} \log_2(P_c) + \frac{16}{3} nb^2 \log_2(P_r) \log_2(P_c) + \frac{32}{3} nb^2 \log_2(P_c) \\
\#words &= \frac{n^2}{P_c} \log_2(P_r) + \frac{mn - n^2/2}{P_r}(\log_2(P_c) + 1) \\
\#messages &= \frac{n^2}{2b^2 P_c} \log_2(P_r) + \frac{n}{b} \log_2(P_c)(\log_2(P_r) + 1)
\end{aligned}$$

Performance with optimal layout. We analyze the two versions of tournament pivoting when using an optimal layout. As in [10], we use the following values:

$$P_r = \sqrt{\frac{mP}{n}} \ , \ P_c = \sqrt{\frac{nP}{m}} \text{ and } b = B \cdot \sqrt{\frac{mn}{P}}, \tag{8.1}$$

These values ensure that CAQR is communication optimal without increasing the flops count. With these values, for the tournament pivoting with $b_T = 2b$, we obtain:

$$\begin{aligned}
\#flops &= \frac{4mn^2 - 4n^3/3}{P} + \\
&+ \quad \frac{mn^2}{P} \left(8B \left(\frac{1}{3} \log_2(P_r) + \log_2(P_c) + \frac{2}{3} \right) + 32B^2 \left(\frac{1}{6} \log_2(P_r) \log_2(P_c) + \frac{1}{3} \log_2(P_c) \right) \right)
\end{aligned}$$

By choosing $B = 8^{-1} \log^{-1}(P_r) \log^{-1}(P_c)$, the first term in the flops dominates the redundant computation due to tournament pivoting. This is a term we cannot decrease. The counts become:

$$\begin{aligned}
\#flops &\approx \frac{4mn^2 - 4n^3/3}{P} + cmn^2, \ \ c < 1 \\
\#words &= \frac{\sqrt{mn^3}}{\sqrt{P}} \log_2 P - \frac{1}{2} \frac{\sqrt{n^5}}{\sqrt{mP}} \log_2 \sqrt{\frac{nP}{m}} \\
\#messages &\approx \frac{1}{2^7} \sqrt{\frac{nP}{m}}
\end{aligned}$$

We choose the same optimal layout for tournament pivoting with $b_T = n/P_c$. We obtain:

$$\begin{aligned}
\#flops &= \frac{5mn^2 - n^3}{6PB} + \\
&+ \quad \frac{mn^2}{P} \left(8B \log_2(P_c) + 32B^2 \left(\frac{1}{6} \log_2(P_r) \log_2(P_c) + \frac{1}{3} \log_2(P_c) \right) \right)
\end{aligned}$$

The communication becomes:

$$\#words = \frac{1}{6}\frac{\sqrt{mn^3}}{B\sqrt{P}}\log_2 P_r + \frac{\sqrt{mn^3}}{\sqrt{P}}\log_2 P_c - \frac{1}{2}\frac{\sqrt{n^5}}{\sqrt{mP}}\log_2 P_c + 2B\frac{\sqrt{mn^3}}{\sqrt{P}}\log_2 P_r \log_2 P_c$$

$$\#messages = \sqrt{\frac{nP}{m}}\left(\log_2(P_r) + 2\log_2(P_c) + \log_2(P_r)\log_2(P_c)\right)$$

8.2. Performance counts of sequential QR with tournament pivoting. In this section we describe a performance model for sequential QR with tournament pivoting, which is analogous to the current Algorithm 1 (Parallel QR with tournament pivoting). Consider that the matrix A to be factored is of size $m \times n$, and that the sequential algorithm factors the input matrix by traversing panels of b columns. First we count $\#flops$, $\#words$ and $\#messages$ ignoring the cost of permuting columns, then the cost of permuting columns is added.

1. $\#flops$: the total, independent of the shape of the tree (as long as we only combine two b-column panels at a time) cost of the tournament pivoting strategy is $4mn^2 - (4/3)n^3$, so twice the rest of QR. Quick derivation: A single QR factorization of an $r \times 2b$ submatrix costs $2r(2b)^2 - (2/3)(2b)^3 = 8rb^2 - (16/3)b^3$. The subsequent RRQR on the $2b \times 2b$ R matrix costs another $O(b^3)$, so one step of the tournament costs $8rb^2 + O(b^3)$ flops. Whether we use a flat, binary or other kind of tree, if we started with c columns, and so (c/b) panels of b columns each, the cost of the tournament is $(c/b - 1)$ times as much, or $8rcb + O(cb^2)$. Going back to the original mxn matrix, there are $n/b - 1$ tournaments with a total cost of

$$\#flops = \sum_{i=0:n/b-1} [8(m-ib)(n-ib)b + O((n-ib)b^2)] = 4mn^2 - (4/3)n^3 + O(mnb).$$

2. $\#words$: If we choose $b = \Theta(M^{1/2})$ where M is cache size, so that we can do an $r \times 2b$ QR decomposition with a flat tree and read the panel just once, the $\#words$ is $2rb$ for a single $r \times 2b$ QR factorization, $2rc$ for a tournament on an $r \times c$ matrix, and altogether

$$\#words = \sum_{i=0}^{n/b-1} 2(m-ib)(n-ib) = mn^2/b - (1/3)n^3/b + O(mn) = \Theta(mn^2/M^{1/2})$$

   which is the desired lower bound.
3. $\#messages$: If we use a blocked data structure where each $b \times b$ block costs 1 message to access, then one $r \times 2b$ QR decomposition with a flat tree costs $(2rb/b^2) = 2r/b$ messages, so we just divide $\#words$ by $b^2 = \Theta(M)$ to get $\#messages = mn^2/b^3 - (1/3)n^3/b^3 + O(mn/b^2) = \Theta(mn^2/M^{3/2})$ which is the desired lower bound.

Now we add the cost of permuting columns. This occurs both during the tournament, to build the $r \times 2b$ matrix analyzed at each step, and after the b columns are selected by the tournament and must be moved to the front of the matrix. One can imagine interleaving these two operations, or doing them separately. For simplicity, I will describe the costs of doing them separately.

The basic permutation operation during the tournament is to take the list of b selected columns of the $2b$ analyzed at a tournament stage, and build a new packed $r \times b$ matrix from them. We assume $3b \times b$ blocks fit in fast memory at the same time:

```
 for i = 1 to r/b
     read i-th b-by-b blocks of two input r-by-b matrices
     select b columns from these 2b columns of length b
     pack them into a b-by-b block
     write block to slow memory
endfor
```

This costs $\#words = 3rb$ and $\#messages = 3r/b$. An entire tournament on an $r \times c$ matrix costs c/b times as much, independent of the shape of the reduction tree, or $\#words = 3rc$ and $\#messages = 3rc/b^2$. Finally, the cost of all the tournaments on an $m \times n$ matrix is

$$\#words = \sum_{i=0}^{n/b-1} 3(m-ib)(n-ib) = (3/2)mn^2/b - (1/2)n^3/b + O(mn) = \Theta(mn^2/M^{1/2})$$

$$\#messages = (3/2)mn^2/b^3 - (1/2)n^3/b^3 + O(mn/b^2) = \Theta\left(\frac{mn^2}{M^{3/2}}\right)$$

as desired.

Now suppose we have completed a tournament and need to go back to the original matrix and permute its columns correspondingly. Note that we must permute not just within the trailing $(m-ib) \times (n-ib)$ submatrix, but also the trailing $m \times (n-ib)$ submatrix, i.e. the upper rows of the R factor must also be permuted. We organize the permutation in two phases:

1. a collection of transpositions, each one swapping a desired column from a later panel with an undesired column from the current panel
2. any subsequent reordering within the current panel.

So given an $m \times b$ panel at the left of an $m \times c$ submatrix, the algorithm is as follows:

```
for i = 1 to m/b
     read i-th b-by-b block of first m-by-b panel (at left)
     for j = 2 to c/b
           if any transpositions involve j-th m/b p
     endfor
endfor
```