Numerical Algorithms
https://doi.org/10.1007/s11075-022-01291-1

ORIGINAL PAPER

# Deviation maximization for rank-revealing QR factorizations

Monica Dessole[1] · Fabio Marcuzzi[1]

Received: 4 June 2021 / Accepted: 28 February 2022

**Abstract**
In this paper, we introduce a new column selection strategy, named here "Deviation Maximization", and apply it to compute rank-revealing QR factorizations as an alternative to the well-known block version of the QR factorization with the column pivoting method, called QP3 and currently implemented in LAPACK's `xgeqp3` routine. We show that the resulting algorithm, named QRDM, has similar rank-revealing properties of QP3 and better execution times. We present experimental results on a wide data set of numerically singular matrices, which has become a reference in the recent literature.

**Keywords** QR factorization · Rank revealing · Column pivoting · Block algorithm · Correlation

## 1 Introduction

The Rank-Revealing QR (RRQR) factorization was introduced by [16] and it is nowadays a classic topic in numerical linear algebra; for example, [17] introduce RRQR factorization for least squares problems where the matrix has not full column rank: in such a case, a plain QR computation may lead to an $R$ factor in which the number of nonzeros on the diagonal does not equal the rank and the matrix $Q$ does not reveal the range nor the null space of the original matrix. Here, the SVD decomposition is the safest and most expensive solution method, while approaches based on a modified QR factorization can be seen as cheaper alternatives. Since

✉ Monica Dessole
monica.dessole.ext@leonardo.com

Fabio Marcuzzi
marcuzzi@math.unipd.it

1 Department of Mathematics Tullio Levi Civita, University of Padova, Via Trieste 63, 35121 Padua, Italy

the QR factorization is essentially unique once the column ordering is fixed, these techniques all amount to finding an appropriate column permutation. The first algorithm was proposed in [7] and it is referred as QR factorization with column pivoting (QRP). It should be noticed that, if the matrix of the least squares problem has not full column rank, then there is an infinite number of solutions and we must resort to rank-revealing techniques which identify a particular solution as “special”. QR with column pivoting identify a particular *basic* solution (with at most $r$ nonzero entries, where $r$ is the rank of the matrix), while biorthogonalization methods [17], identify the minimum $\ell_2$ solution. Rank-revealing decompositions can be used in a number of other applications [20]. The QR factorization with column pivoting works pretty well in practice, even if there are some examples in which it fails, see, e.g., the Kahan matrix [23]. However, further improvements are possible, see, e.g., [8] and [15]: the idea here is to identify and remove small singular values one by one. Gu and Eisenstat [19] introduced the strong RRQR factorization, a stable algorithm for computing a RRQR factorization with a good approximation of the null space, which is not guaranteed by QR factorization with column pivoting. Both can be used as optional improvements to the QR factorization with column pivoting. Rank-revealing QR factorizations were also treated in [9, 18, 22].

Column pivoting makes it more difficult to achieve high performances in QR computation, see [3–6, 27]. The state-of-the-art algorithm for computing RRQR, named QP3, is a block version [27] of the standard column pivoting and it is currently implemented in LAPACK [1]. Other recent high-performance approaches are tournament pivoting [10] and randomized pivoting [14, 25, 31]. In this paper we present a technique based on correlation analysis we call “Deviation Maximization”, that selects a subset of sufficiently linearly independent columns. The deviation maximization may be adopted as a block pivoting strategy in more complex applications that require subset selection. We successfully apply the deviation maximization to the problem of finding a rank-revealing QR decomposition, but, e.g., the authors experimented also a preliminary version of this procedure in the context of active set methods, see [11, 12]. The rest of this paper is organized as follows. In Section 2 we motivate and describe this novel column selection technique. In Section 3 we define the rank-revealing factorization, we review the QRP algorithm and then we introduce a block algorithm for RRQR by means of deviation maximization; furthermore, we give theoretical worst case bounds for the smallest singular value of the $R$ factor of the RRQR factorizations obtained with these two methods. In Section 4 we discuss the algorithm QRDM and some fundamental issues regarding its implementation. Section 5 compares QP3 and QRDM against a relevant database of singular matrices, and finally, the paper concludes with Section 6.

### 1.1 Notation

For any matrix $A$ of size $m \times n$, we denote by $[A]_{I,J}$ the submatrix of $A$ obtained considering the entries with row and columns indices ranging in the sets $I$ and $J$, respectively. We make use of the so-called colon notation, that is we denote by $[A]_{k:l,p:q}$ the submatrix of $A$ obtained considering the entries with row indices $k \le i \le l$ and column indices $p \le j \le q$. When using colon notation, we write

$[A]_{:,p:q}$ ($[A]_{k:l,:}$) as a shorthand for $[A]_{1:m,p:q}$ ($[A]_{k:l,1:n}$). We also denote the $(i, j)$-th entry as $a_{ij}$ or $[A]_{ij}$. The singular values of a matrix $A$ are denoted as

$$\sigma_{\max}(A) = \sigma_1(A) \geq \sigma_2(A) \geq \cdots \geq \sigma_{\min}(A) = \sigma_{\min(m,n)}(A) \geq 0.$$

Given the vector norm $\|x\|_p = (|x_1|^p + \ldots |x_n|^p)^{1/p}$, $p \geq 1$, we denote the family of $p$-norms as

$$\|A\|_p = \sup_{\|x\|_p=1} \|Ax\|_p.$$

We denote the operator norm by $\|A\|_2 = \sigma_{\max}(A)$. When the context allows it, we drop the subscript on the 2-norm. With a little abuse of notation, we define the max-norm of $A$ as $\|A\|_{\max} = \max_{i,j} |a_{ij}|$. Recall that the max-norm is not a matrix norm (it is not submultiplicative), and it should not be confused with the $\infty$-norm $\|A\|_\infty = \max_i \sum_j |a_{ij}|$.

## 2 Column selection by deviation maximization

Consider an $m \times n$ matrix $A$ which has not full column rank, that is $\operatorname{rank}(A) = r < n$, and consider the problem of finding a subset of well-conditioned columns of $A$. Before presenting a strategy to solve this problem, let us recall that for an $m \times k$ matrix $C = (\mathbf{c}_1 \ \ldots \ \mathbf{c}_k)$ whose columns $\mathbf{c}_j$ are non-null, the correlation matrix $\Theta$ has entries

$$\theta_{ij} = \frac{\mathbf{c}_i^T \mathbf{c}_j}{\|\mathbf{c}_i\| \|\mathbf{c}_j\|}, \quad 1 \leq i, j \leq k. \tag{1}$$

In particular, we have $\Theta = \left(CD^{-1}\right)^T CD^{-1} = D^{-1}C^TCD^{-1}$, where $D$ is the diagonal matrix with entries $d_i = \|\mathbf{c}_i\|$ , $1 \leq i \leq k$. It is immediate to see that $\Theta$ is symmetric positive semidefinite, it has only ones on the diagonal, and its entries range from $-1$ to 1. Notice that $\theta_{ij}$ is the cosine of $\alpha_{ij} = \alpha(\mathbf{c}_i, \mathbf{c}_j) \in [0, \pi)$, the angle (modulo $\pi$) between $\mathbf{c}_i$ and $\mathbf{c}_j$. In order to emphasize this geometric interpretation, from now on we refer to $\Theta$ as the *cosine matrix*.

Let us first recall a few definitions taken from [2]. A squared matrix $A = \Delta + N$, where $\Delta$ is diagonal and $N$ has a zero diagonal, is said to be $\tau$*-diagonally dominant* with respect to a norm $\|\cdot\|$ if $\|N\| \leq \tau \min_i |\Delta_{ii}|$ for some $0 \leq \tau < 1$. A matrix $A = D_1(\Delta + N)D_2$, where $\Delta, D_1, D_2$ are diagonal and $N$ has a zero diagonal, is said to be $\tau$*-scaled diagonally dominant* with respect to a norm $\|\cdot\|$ if $\Delta + N$ is $\tau$*-diagonally dominant* with respect to same norm, for some $0 \leq \tau < 1$. If $A$ is symmetric, then $\Delta + N$ is symmetric and we have $D := D_1 = D_2$, with diagonal entries $d_i = |A_{ii}|^{1/2}$. The main idea behind the deviation maximization is based on the following result.

**Lemma 1** *Let $C = (\mathbf{c}_1 \ \ldots \ \mathbf{c}_k)$ be an $m \times k$ matrix such that $\|\mathbf{c}_1\| = \max_j \|\mathbf{c}_j\| > 0$. Suppose there exists $1 > \tau > 0$ such that $\|\mathbf{c}_j\| \geq \tau\|\mathbf{c}_1\|$ for all $j$, and that $C^TC$ is $\tau$-scaled diagonally dominant matrix with respect to the $\infty$-norm. Then*

$$\sigma_{\min}(C) \geq \|\mathbf{c}_1\|\sqrt{\tau(\tau - \|N\|_\infty)}, \tag{2}$$

$$\sigma_{\min}(C) \geq \|\mathbf{c}_1\|\tau\sqrt{1 - \tau}. \tag{3}$$

*Proof* Let us write $A = C^T C = D\Theta D$, where $D = \mathrm{diag}(d_j)$, with $d_j = \|\mathbf{c}_j\|$, and the cosine matrix $\Theta$ decomposes as $\Theta = (I + N)$.

We first prove (2). Let us show that $A$ is diagonally dominant in the classic sense, that is for $i$ we have

$$|a_{ii}| - \sum_{j\neq i} |a_{ij}| = d_i^2 - \sum_{j\neq i} |d_i d_j \theta_{ij}| > 0.$$

For all $1 \le i \le k$, we have

$$\sum_{j\neq i} |d_i d_j \theta_{ij}| = |d_i| \sum_{j\neq i} |d_j \theta_{ij}| \le |d_i| \max_j |d_j| \sum_{j\neq i} |\theta_{ij}| = |d_i| \|\mathbf{c}_1\| \sum_{j\neq i} |\theta_{ij}|,$$

and hence

$$d_i^2 - \sum_{j\neq i} |d_i d_j \theta_{ij}| \ge |di| \|\mathbf{c}_1\| \tau - |d_i| \|\mathbf{c}_1\| \sum_{j\neq i} |\theta_{ij}| = |di| \|\mathbf{c}_1\| \left( \tau - \sum_{j\neq i} |\theta_{ij}| \right).$$

Since $|d_i| > 0$ for all $i$, and $\|\mathbf{c}_1\| > 0$, the right-hand side is positive if and only if

$$\tau > \max_i \sum_{j\neq i} |\theta_{ij}| = \|N\|_\infty,$$

that is true by assumption since the cosine matrix $\Theta$ is $\tau$-diagonally dominant with respect to the $\infty$-norm. Moreover, we have

$$\begin{aligned} \min_i \left\{ d_i^2 - \sum_{j\neq i} |d_i d_j \theta_{ij}| \right\} &\ge \tau \|\mathbf{c}_1\|^2 \left( \tau - \max_i \sum_{j\neq i} |\theta_{ij}| \right) = \tau \|\mathbf{c}_1\|^2 \left(\tau - \|N\|_\infty\right) \\ &= \tau \|\mathbf{c}_1\|^2 \left(\tau - \|N\|_\infty\right). \end{aligned}$$

For any strictly diagonally dominant matrix $A$ with $\alpha = \min_i \left\{ |a_{ii}| - \sum_{j\neq i} |a_{ij}| \right\} > 0$, we have (see [30])

$$\|A^{-1}\| < \frac{1}{\alpha} \quad \Rightarrow \quad \|A^{-1}\|^{-1} = \sigma_{\min}(A) > \alpha.$$

Then

$$\sigma_{\min}(C^T C) \ge \tau \|\mathbf{c}_1\|^2 \left(\tau - \|N\|_\infty\right) \quad \Rightarrow \quad \sigma_{\min}(C) \ge \|\mathbf{c}_1\| \sqrt{\tau \left(\tau - \|N\|_\infty\right)}.$$

Let us now prove (3). First notice that the cosine matrix $\Theta = I + N$ is symmetric, hence $\|N\|_\infty = \|N\|_1$. In particular, $\Theta$ is $\tau$-diagonally dominant also with respect to the 2-norm, since by Hölder's inequality we get

$$\|N\|_2 \le \left(\|N\|_1 \|N\|_\infty\right)^{1/2} = \|N\|_\infty < \tau.$$

Moreover, assume without loss of generality that $d_1 \geq d_2 \geq \cdots \geq d_k$, where $d_i = \|\mathbf{c}_i\|$. Recall the variational characterization (Courant-Fischer Theorem) of the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_k$ of a symmetric matrix $A$ of order $k$

$$\lambda_i = \min_{\substack{S \subseteq \mathbb{R}^n \\ \dim(S)=i-1}} \max_{\substack{\mathbf{x} \in S^\perp \\ \|\mathbf{x}\|=1}} \mathbf{x}^T A \mathbf{x}.$$

Let $S_{i-1} \subseteq \mathbb{R}^k$ be the subspace spanned by the first $i-1$ elements of the canonical basis. Its orthogonal complement $S_{i-1}^\perp$ is then the subspace spanned by the last $k-i+1$ elements of the canonical basis. We have

$$\lambda_i \leq \max_{\substack{\mathbf{x} \in S_{i-1}^\perp \\ \|\mathbf{x}\|=1}} \mathbf{x}^T A \mathbf{x} = \max_{\|\hat{\mathbf{x}}\|=1} \hat{\mathbf{x}}^T A_{i-1} \hat{\mathbf{x}},$$

where $\hat{\mathbf{x}} \in \mathbb{R}^{k-i+1}$ is the vector obtained by deleting the first $i-1$ entries of $\mathbf{x}$ and $A_{i-1}$ is the square submatrix of order $k-i+1$ obtained by deleting the first $i-1$ rows and columns of $A$. Consider the eigenvalues $\lambda_i$ of the symmetric matrix $A = C^T C$. Take $\mathbf{x} \in S_{i-1}^\perp$ with $\|\mathbf{x}\| = 1$, then by Cauchy-Schwarz inequality

$$\begin{aligned}
\hat{\mathbf{x}}^T A_{i-1} \hat{\mathbf{x}} &= \hat{\mathbf{x}}^T D_{i-1} \Theta_{i-1} D_{i-1} \hat{\mathbf{x}} = \hat{\mathbf{x}}^T D_{i-1} (I + N_{i-1}) D_{i-1} \hat{\mathbf{x}} \\
&= \|D_{i-1}\hat{\mathbf{x}}\|^2 + (D_{i-1}\hat{\mathbf{x}})^T N_{i-1} D_{i-1} \hat{\mathbf{x}} \leq \|D_{i-1}\hat{\mathbf{x}}\|^2 + \|N_{i-1} D_{i-1}\hat{\mathbf{x}}\| \, \|D_{i-1}\hat{\mathbf{x}}\| \\
&\leq \|D_{i-1}\hat{\mathbf{x}}\|^2 + \tau \|D_{i-1}\hat{\mathbf{x}}\|^2 \leq (1+\tau)\|D_{i-1}\|^2 \\
&= (1+\tau) \max_{i \leq j \leq k} \|\mathbf{c}_j\|^2 = (1+\tau)\|\mathbf{c}_i\|^2,
\end{aligned}$$

and thus $\lambda_i \leq (1+\tau)\|\mathbf{c}_i\|^2 \leq (1+\tau)\|\mathbf{c}_1\|^2$. Considering $-A$ instead, we get

$$\begin{aligned}
\hat{\mathbf{x}}^T(-A_{i-1})\hat{\mathbf{x}} &= \hat{\mathbf{x}}^T D_{i-1}(-I - N_{i-1}) D_{i-1}\hat{\mathbf{x}} = -\|D_{i-1}\hat{\mathbf{x}}\|^2 - (D_{i-1}\hat{\mathbf{x}})^T N_{i-1} D_{i-1}\hat{\mathbf{x}} \\
&\leq (-1+\tau)\|D_{i-1}\hat{\mathbf{x}}\|^2 \\
&\leq (-1+\tau)\|\mathbf{c}_i\|^2,
\end{aligned}$$

and thus $\lambda_i \geq (1-\tau)\|\mathbf{c}_i\|^2$. Since $\lambda_i = \sigma_i^2$, we have

$$\sigma_i \geq \|\mathbf{c}_1\| \tau \sqrt{1-\tau}.$$

□

The proof of the bound (3) is mainly based on results contained in [2]. Inequalities (2)–(3) show quite clearly that the bound on the smallest singular value of $C$ depends on the norms of the column vectors and on the angles between each pair of such columns. This suggests to choose $k$ columns of $A$, namely those with indices $J = \{j_1, \ldots, j_k\}$, $k \leq r$, such that the submatrix $C = [A]_{:,J}$ has columns with large euclidean norms, i.e., larger than the length defined by a parameter $\tau > 0$, and with large deviations, meaning that each pair of columns form an angle whose cosine in absolute value is bounded by a parameter $\delta \geq 0$. For these reasons, the overall procedure is called deviation maximization and it is presented in Algorithm 1.

**Algorithm 1** Deviation Maximization: $[J] = \mathrm{DM}(A, \mathbf{u}, \tau, \delta)$.

```
1: $J = \{j : j \in \arg\max \mathbf{u}\}$
2: $I = \{i : u_i \geq \tau \max \mathbf{u},\ i \neq j\}$
3: set $k_{\max} = \mathrm{card}(I)$
4: compute the cosine matrix $\Theta$ associated to $[A]_{:,I}$
5: **for** $i \in I$ **do**
6:     **if** $|\theta_{i,j}| < \delta, \forall j \in J$ **then**
7:         $J = J \cup \{i\}$
8:     **end if**
9: **end for**
```

Let us detail the above procedure. Define the vector $\mathbf{u}$ containing the column norms of $A$, namely $\mathbf{u} = (u_i) = (\|\mathbf{a}_i\|)$, for $i = 1, \dots, n$. The set $J$ of column indices is initialized at step 1 with a column index corresponding to the maximum column norm, namely

$$J = \{j : j \in \arg\max \mathbf{u}\}.$$

At step 2, a set of "candidate" column indices $I$ to be added in $J$ is identified by selecting those columns with a large norm with respect to the parameter $\tau$, that is

$$I = \{i : u_i \geq \tau \max \mathbf{u},\ i \neq j\}, \tag{4}$$

and then the cosine matrix associated to the corresponding submatrix $\Theta$, i.e., the cosine matrix of $[A]_{:,I}$, is computed at step 4. With a loop over the indices of the candidate set $I$, an index $i \in I$ is inserted in $J$ only if the $i$-th column has a large deviation from the columns whose index is already in $J$. In formulae, we ask

$$|\theta_{ij}| < \delta, \qquad \text{for all } j \in J,$$

i.e., the columns $\mathbf{a}_i$ and $\mathbf{a}_j$ are orthogonal up to the factor $\delta$. At the end of the iterations, we have $J = \{j_1, \dots, j_k\}$, with $1 \leq k \leq k_{max}$, where $k_{\max}$ is the cardinality of the candidate set $I$, and we set $C = [A]_{:,J}$. The following static choice of the parameter $\delta$, namely

$$\delta = \frac{\tau}{k_{\max} - 1}, \tag{5}$$

yields a submatrix $C = [A]_{:,J}$ that satisfies the hypotheses of Lemma 1 for a fixed choice of parameters $\delta$ and $\tau$. Indeed, for every $j \in J$, this choice ensures

$$\sum_{\substack{i \in J \\ i \neq j}} |\theta_{ij}| < (k-1)\delta = (k-1)\frac{\tau}{k_{\max} - 1} \leq (k_{\max} - 1)\frac{\tau}{k_{\max} - 1} = \tau,$$

and hence the cosine matrix $\Theta$ is $\tau$-diagonally dominant. Other strategies are possible: for example, at each iteration, an index $i$ can be added to $J$ if

$$|\theta_{ij}| < \tau - \sum_{\substack{l \in J \\ l \neq j}} |\theta_{lj}|,$$

for all $j \in J$, suggesting that the value $\delta$ can be updated dynamically as follows

$$\delta = \tau - \max_{j \in J} \sum_{\substack{l \in J \\ l \neq j}} |\theta_{lj}|. \tag{6}$$

In both (5) and (6), we have $0 \le \delta < \tau < 1$. In practice, the value of $\delta$ can be chosen independently from $\tau$, as we detail in Section 4, and this is why it is kept as an input parameter.

### 2.1 Computing the cosine matrix

Let us focus on some details of the implementation of the deviation maximization presented in Algorithm 1. First, the candidate set $I$ defined in (4) can be computed with a fast sorting algorithm, e.g., quicksort, applied to the array of column norms. The most expensive operation in Algorithm 1 is the computation of the cosine matrix $\Theta$ in step 4. The cosine matrix of the columns indexed in the candidate set $I$ is given by

$$\Theta = D^{-1}[A]_{:,I}^T[A]_{:,I}D^{-1}, \tag{7}$$

where $D = \operatorname{diag}(\|\mathbf{a}_j\|)$, with $j \in I$. The matrix $\Theta$ is symmetric, thus we only need its upper (lower) triangular part. This can be computed as

$$\Theta = U_1^T U_1, \qquad U_1 = [A]_{:,I}D^{-1}, \tag{8}$$

or

$$\Theta = D^{-1}U_2D^{-1}, \qquad U_2 = [A]_{:,I}^T[A]_{:,I}. \tag{9}$$

The former approach requires $m \times n$ additional memory to store $U_1$ and it requires $m^2k_{\max}^2$ flops to compute $U_1$ and $(2m-1)k_{\max}(k_{\max}-1)/2$ flops for the upper triangular part of $U_1^T U_1$, while the latter does not require additional memory since the matrix $U_2$ can be stored in the same memory space used for the cosine matrix $\Theta$, and it requires $(2m-1)k_{\max}(k_{\max}-1)/2$ flops the upper triangular part of $U_2$ and $k_{\max}(k_{\max}-1)$ flops for the upper triangular part of $D^{-1}U_2D^{-1}$. The cheapest strategy is to compute $\Theta$ according to (9), even if it requires to write an *ad hoc* low level routine which is not implemented in the BLAS library. It should be noted that both (8) and involve a symmetric matrix–matrix multiplication, which can be efficiently computed with the BLAS subroutine `xsyrk`.

In order to limit the cost and the amount of additional memory of Algorithm 1, we propose a restricted version of the deviation maximization pivoting. If the candidate is given by $I = \{j_l : l = 1, \ldots, k_{\max}\}$, we limit its cardinality to be smaller or equal to a machine dependent parameter $k_{DM}$, that is

$$I = \{j_l : l = 1, \ldots, \min(k_{\max}, k_{DM})\}. \tag{10}$$

We refer to the value $k_{DM}$ as block size, and we discuss its value in terms of achieved performances in Section 5.

## 3 Rank-revealing QR decompositions

In exact arithmetic, we say that an $m \times n$ matrix $A$ is rank-deficient if $0 = \sigma_{r+1}(A) < \sigma_r(A)$, where $r < \min(m, n)$ is its rank. However, rank determination is nontrivial in presence of errors in the matrix elements. Golub and Van Loan [17] define $\varepsilon$-rank of a matrix $A$ as

$$\operatorname{rank}(A, \varepsilon) = \min_{\|A-B\|<\varepsilon} \operatorname{rank}(B), \tag{11}$$

for some small $\varepsilon > 0$. Thus, if the input data have an initial uncertainty of a known order $\eta$, then it has sense to look at $\operatorname{rank}(A, \eta)$. Similarly, for a floating point matrix $A$ it is reasonable to regard $A$ as numerically rank-deficient if $\operatorname{rank}(A, \varepsilon) < \min(m, n)$, where $\varepsilon = u\|A\|$ and $u$ is the unit roundoff. This issue is discussed more in detail in Section 3.4.

Let us now introduce the mathematical formulation for the problem of finding a rank-revealing decomposition of a matrix $A$ of size $m \times n$ with numerical rank $r$, defined up to a certain tolerance $\varepsilon$ as discussed above. Let $\Pi$ denote a permutation matrix of size $n$, then we can compute

$$A\Pi = QR = (Q_1\ Q_2)\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \tag{12}$$

where $Q$ is an orthogonal matrix of order $m$, $Q_1 \in m \times r$ and $Q_2 \in m \times (m - r)$, $R_{11}$ is upper triangular of order $r$, $R_{12} \in r \times (n - r)$ and $R_{22} \in (m - r) \times (n - r)$. The QR factorization above is called *rank-revealing* if

$$\sigma_{\min}(R_{11}) = \sigma_r(R_{11}) \approx \sigma_r(A),$$

or

$$\sigma_{\max}(R_{22}) = \sigma_1(R_{22}) \approx \sigma_{r+1}(A),$$

or both conditions hold. Notice that if $\sigma_{\min}(R_{11}) \gg \varepsilon$ and $\|R_{22}\|$ is small, then the matrix $A$ has numerical rank $r$, but the converse is not true. In other words, even if $A$ has $(\min(m, n) - r)$ small singular values, it does not follow that any permutation $\Pi$ yields a small $\|R_{22}\|$, even if there exist strategies that ensure a small value of $\|R_{22}\|$ by identifying and removing small singular values, see, e.g., [8, 15]. It is easy to show that for any factorization like (12) the following relations hold

$$\sigma_{\min}(R_{11}) \le \sigma_r(A), \tag{13}$$

$$\sigma_{\max}(R_{22}) \ge \sigma_{r+1}(A). \tag{14}$$

The proof is an easy application of the interlacing inequalities for singular values [29], namely

$$\sigma_k(A) \ge \sigma_k(B) \ge \sigma_{k+r+s}(A), \quad k \ge 1,$$

which hold for any $(m - s) \times (n - r)$ submatrix $B$ of $A$. In fact we have

$$\sigma_{\min}(R_{11}) = \sigma_{\min}\begin{pmatrix} R_{11} \\ 0 \end{pmatrix} = \sigma_r([Q^T A\Pi]_{1:m,1:r}) \le \sigma_r(Q^T A\Pi) = \sigma_r(A),$$

$$\sigma_{\max}(R_{22}) = \sigma_{\max}(0\ R_{22}) = \sigma_1([Q^T A\Pi]_{r+1:m,1:n}) \ge \sigma_{r+1}(Q^T A\Pi) = \sigma_{r+1}(A).$$

Ideally, the best rank-revealing QR decomposition is obtained by the column permutation $\Pi$ which solves

$$\max_{\Pi} \sigma_{\min}(R_{11}), \tag{15}$$

for a fixed rank $r$. Recall that the volume of a rectangular real matrix $A$ is defined as $\sqrt{\det(A^T A)}$ or $\sqrt{\det(AA^T)}$ depending on the shape of $A$ [26], that is the volume of $A$ equals the square root of the product of the singular values of $A$. It is not difficult to show that problem (15) is equivalent to the problem of selecting $r$ columns such that the volume of the corresponding submatrix $[\Pi A]_{:,1:r}$ is maximal. Problem (15) clearly has a combinatorial nature, thus algorithms that compute RRQR usually provide (see, e.g., [9, 22]) at least one of the following bounds

$$\sigma_{\min}(R_{11}) \geq \frac{\sigma_r(A)}{p(n)}, \tag{16}$$

$$\sigma_{\max}(R_{22}) \leq \sigma_{r+1}(A) q(n), \tag{17}$$

where $p(n)$ and $q(n)$ are low degree polynomials in $n$. These are worst case bounds and are usually not sharp. We provide a bound of type (16) in Section 3.3.

## 3.1 The standard column pivoting

Let us introduce the QR factorization with column pivoting proposed by [7], which can be labeled as a greedy approach in order to cope with the combinatorial optimization problem (15). Suppose at the $s$-th algorithmic step we have already selected $s < r$ well-conditioned columns of $A$, which are moved to the leading positions by the permutation matrix $\Pi^{(s)}$ as follows

$$A\Pi^{(s)} = Q^{(s)} R^{(s)} = Q^{(s)} \begin{pmatrix} R_{11}^{(s)} & R_{12}^{(s)} \\ & R_{22}^{(s)} \end{pmatrix}, \tag{18}$$

where $R_{11}^{(s)}$ is an upper triangular block of order $s$, and the blocks $R_{12}^{(s)}$ and $R_{22}^{(s)}$ have size $s \times (n-s)$ and $(m-s) \times (n-s)$, respectively. The block $R_{22}^{(s)}$ is what is left to be processed, and it is often called the "trailing matrix". Let us introduce the following column partitions for $R_{12}^{(s)}$, $R_{22}^{(s)}$ respectively

$$R_{12}^{(s)} = (\mathbf{b}_1 \ldots \mathbf{b}_{n-s}),$$
$$R_{22}^{(s)} = (\mathbf{c}_1 \ldots \mathbf{c}_{n-s}).$$

We aim at selecting, within the $n-s$ remaining columns, that column such that the condition number of the next block $R_{11}^{(s+1)}$ is kept the largest possible. Formally, we would like to solve

$$\sigma_{\min}\left(R_{11}^{(s+1)}\right) = \sigma_{\min}\begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_j \\ & \mathbf{c}_j \end{pmatrix} = \max_{1\leq i\leq n-s} \sigma_{\min}\begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_i \\ & \mathbf{c}_i \end{pmatrix}. \tag{19}$$

Using the following fact

$$\sigma_{\min}\begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_j \\ & \mathbf{c}_j \end{pmatrix} = \sigma_{\min}\begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_j \\ & \|\mathbf{c}_j\| \end{pmatrix},$$

which is a simple consequence of the invariance of singular values under left multiplication by orthogonal matrices and the insertion of null rows, and using the bound

$$\sigma_{\min}(A) \le \min_i(\|\mathbf{e}_i^T A^{-1}\|_2^{-1}) \le \sqrt{n}\sigma_{\min}(A), \tag{20}$$

which holds for any nonsingular matrix $A$, we can approximate up to a factor $\sqrt{s+1}$ the smallest singular value as

$$\sigma_{\min}\begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_j \\ & \mathbf{c}_j \end{pmatrix} \approx \min_h \left\| \mathbf{e}_h^T \begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_j \\ & \|\mathbf{c}_j\| \end{pmatrix}^{-1} \right\|^{-1},$$

where $\mathbf{e}_h$ is the $h$-th element of the canonical basis of $\mathbb{R}^{s+1}$.
Using this result, as argued in [9], the maximization problem (19) can be solved approximately by solving

$$j = \arg \max_{1\le i\le n-n_s} \|\mathbf{c}_j\| \approx \arg \max_{1\le i\le n-n_s} \sigma_{\min}\begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_i \\ & \mathbf{c}_i \end{pmatrix}.$$

The resulting procedure is referred as QR factorization with column pivoting, and it is presented in Algorithm 2.

**Algorithm 2** QR with column pivoting: $[A] = \mathrm{QRP}(A)$.

```
1: initialize the vector $\mathbf{u}$ of squared norms
2: for $s = 0, \dots, n-1$ do
3:     $j = \arg\max_i [\mathbf{u}]_{s+1:n}$
4:     move the $j$-th index to the leading position on $A$ and $\mathbf{u}$
5:     compute the Householder reflector $\mathbf{v}^{(s)}$ w.r.t. $[A]_{s:m,s}$
6:     update the trailing matrix $[A]_{s:m,s+1:n} -= \mathbf{v}^{(s)}(\mathbf{v}^{(s)})^T\ [A]_{s:m,s+1:n}$
7:     update the partial column norms $\mathbf{u}$
8: end for
```

This algorithm can be efficiently implemented since the column norms of the trailing matrix can be updated at each iteration instead of being recomputed from scratch. This can be done [17] by exploiting the following property

$$Q\mathbf{a} = \begin{pmatrix} \beta \\ \mathbf{c} \end{pmatrix} \begin{matrix} 1 \\ m-1 \end{matrix} \Rightarrow \|\mathbf{a}\|^2 = \|Q\mathbf{a}\|^2 = \beta^2 + \|\mathbf{c}\|^2,$$

which holds for any orthogonal matrix $Q$ and any vector $\mathbf{a}$ of order $m$. Therefore, once defined the vector $\mathbf{u}^{(s)}$ whose entry $u_j^{(s)}$ is the $j$-th partial column norm of $A\Pi^{(s)}$, that is the norm of the subcolumn with row indices ranging from $m-n_s$ to $m$, and initialized $u_j^{(1)} = \|\mathbf{a}_j\|^2$, with $1 \le j \le n$, we can perform the following update

$$u_j^{(s+1)} = \begin{cases} \sqrt{\left(u_j^{(s)}\right)^2 - r_{sj}^2}, & s+1 \le j \le n, \quad 2 \le s \le n, \\ 0, & j < s+1, \end{cases} \tag{21}$$

where $r_{ij}$ is the entry of indices $(i, j)$ in $R^{(s)}$, $1 \leq i \leq m$, $1 \leq j \leq n$. The partial column norm update allows to reduce the operation count from $\mathscr{O}(mn^2)$ to $\mathscr{O}(mn)$. Actually, the formula (21) cannot be applied as it is because of numerical cancellation, and it needs to modified, see, e.g., [13] for a robust implementation. The pivoting strategy just presented produces a factor $R$ that satisfies [21]

$$|r_{kk}|^2 \geq \sum_{i=k}^{j} |r_{ij}|^2, \qquad k \leq j \leq n,\ 1 \leq k \leq n, \tag{22}$$

and, in particular,

$$|r_{11}| \geq |r_{22}| \geq \cdots \geq |r_{nn}|, \tag{23}$$

$$|r_{kk}| \geq |r_{kj}|, \quad k \leq j \leq n,\ 1 \leq k \leq n. \tag{24}$$

A block version of Algorithm 2 has been proposed [27], and it is currently implemented in LAPACK's `xgeqp3` routine, that we will use in the numerical section for comparison.

*Remark 1* Geometric interpretation: Introduce the following block column partitioning $R^{(s)} = (R_1\ R_2)$ and $Q^{(s)} = (Q_1\ Q_2)$, where $R_1$ and $Q_1$ have $s$ columns. By the properties of the QR decomposition, we have

$$\mathscr{R}(Q_1) = \mathscr{R}(R_1) \quad \mathscr{R}(Q_2) = \mathscr{R}(R_1)^{\perp}.$$

where $\mathscr{R}(B)$ denotes the subspace spanned by the columns of a matrix $B$. Every unprocessed column of $A$ rewrites as

$$\mathbf{a}_j = Q_1\mathbf{b}_{j-s} + Q_2\mathbf{c}_{j-s},$$

where $Q_1\mathbf{b}_{j-s}$ and $Q_2\mathbf{c}_{j-s}$ are the orthogonal projection of $\mathbf{a}_j$ on the subspace $\mathscr{R}(Q_1)$ and on its orthogonal complement $\mathscr{R}(Q_2)$, respectively. The most linearly independent column $\mathbf{a}_i$ from the columns in $R_1$ can be seen as the one with the largest orthogonal projection of the complement on the subspace spanned by such columns, namely

$$\max_{i \geq s} \left\| \mathscr{P}_{\mathscr{R}(Q_2)}\mathbf{a}_i \right\| = \max_{i \geq 1} \| Q_2\mathbf{c}_i \| .$$

However, the matrix $Q^{(s)}$ is never directly available unless it is explicitly computed. We then settle for the index $j$ such that

$$\|\mathbf{c}_{j-s}\| = \max_{i \geq 1} \|\mathbf{c}_i\|.$$

## 3.2 The deviation maximization pivoting

Consider the partial factorization in (18), and now suppose at the $s$-th algorithmic step we have already selected $n_s$, with $s \leq n_s < r$, well-conditioned columns of $A$, so that $R_{11}^{(s)}$ has size $n_s \times n_s$, while blocks $R_{12}^{(s)}$ and $R_{22}^{(s)}$ have size $n_s \times (n - n_s)$ and $(m - n_s) \times (n - n_s)$ respectively. The idea is to pick $k_s$, with $n_{s+1} = n_s + k_s \leq r$, linearly independent and well-conditioned columns from the remaining $n - n_s$ columns of $A$, which are also sufficiently linearly independent from the $n_s$ columns already selected, in order to keep the smallest singular value of the $R_{11}$ block as large as possible. We aim at selecting those columns with indices $j_1, \ldots, j_{k_s}$ that solve

$$\sigma_{\min}\begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_{j_1} & \ldots & \mathbf{b}_{j_{k_s}} \\ & \mathbf{c}_{j_1} & \ldots & \mathbf{c}_{j_{k_s}} \end{pmatrix} = \max_{1 \le i_1, \ldots, i_{k_s} \le n-n_s} \sigma_{\min}\begin{pmatrix} R_{11}^{(s)} & \mathbf{b}_{i_1} & \ldots & \mathbf{b}_{i_{k_s}} \\ & \mathbf{c}_{i_1} & \ldots & \mathbf{c}_{i_{k_s}} \end{pmatrix}. \tag{25}$$

Of course, this maximization problem has the same combinatorial nature as problem (15), so we rather solve it approximately. We propose to approximate the indices $\left\{j_1, \ldots, j_{k_s}\right\}$ that solves problem (25) with the indices selected by means of the deviation maximization procedure presented in Algorithm 1 applied to the trailing matrix $R_{22}^{(s)}$. For the moment, let $\tau > 0$ and $\delta$ be fixed accordingly to (5) or (6). More efficient choices will be widely discussed in Section 5. For sake of brevity, we will denote by $B = (\mathbf{b}_{j_1} \ldots \mathbf{b}_{j_{k_s}})$ and $C = (\mathbf{c}_{j_1} \ldots \mathbf{c}_{j_{k_s}})$ the matrices made up of the columns selected, and by $\bar{B}$ and $\bar{C}$ the matrices made up by the remaining columns. The rest of the block update, which we detail below, proceeds in a way similar to the recursive block QR. Let $\tilde{Q}^{(s+1)}$ be an orthogonal matrix of order $(m - n_s)$ such that

$$\left(\tilde{Q}^{(s)}\right)^T C = \begin{pmatrix} T \\ 0 \end{pmatrix} \in \mathbb{R}^{(m-n_s)\times k_s}, \tag{26}$$

where $T$ is an upper triangular matrix of order $k_s$. The matrix $\tilde{Q}^{(s+1)}$ is obtained as a product of $k_s$ Householder reflectors, that we represent by mean of the so-called compact WY form [28] as

$$\tilde{Q}^{(s)} = I - Y^{(s)} W^{(s)} (Y^{(s)})^T,$$

where $Y^{(s)}$ is lower trapezoidal with $k_s$ columns and $W^{(s)}$ is upper triangular of order $k_s$. This allows us to carry out the update of the rest of trailing matrix, that is

$$\left(\tilde{Q}^{(s)}\right)^T \bar{C} = \begin{pmatrix} \bar{T} \\ R_{22}^{(s+1)} \end{pmatrix} \in \mathbb{R}^{(m-n_s)\times(n-n_s-k_s)}, \tag{27}$$

by means of BLAS-3 kernels, for performance efficiency. Denoting by $\tilde{\Pi}^{(s)}$ a permutation matrix that moves columns with indices $j_1, \ldots, j_{k_s}$ to the current leading positions, we set $\Pi^{(s+1)} = \Pi^{(s)}\tilde{\Pi}^{(s)}$ and

$$Q^{(s+1)} = Q^{(s)} \begin{pmatrix} I & \\ & \tilde{Q}^{(s)} \end{pmatrix} \in \mathbb{R}^{m\times m},$$

then the overall factorization of $A\Pi^{(s+1)}$ takes the form

$$Q^{(s)} \begin{pmatrix} R_{11}^{(s)} & B & \bar{B} \\ & C & \bar{C} \end{pmatrix} = Q^{(s+1)} \begin{pmatrix} R_{11}^{(s)} & B & \bar{B} \\ & T & \bar{T} \\ & & R_{22}^{(s+1)} \end{pmatrix}, \tag{28}$$

where, for the successive iteration, we set

$$R_{11}^{(s+1)} = \begin{pmatrix} R_{11}^{(s)} & B \\ & T \end{pmatrix} \in \mathbb{R}^{n_{s+1}\times n_{s+1}},$$

$$R_{12}^{(s+1)} = \begin{pmatrix} \bar{B} \\ \bar{T} \end{pmatrix} \in \mathbb{R}^{n_{s+1}\times(n-n_{s+1})},$$

with $n_{s+1} = n_s + k_s$. The resulting procedure is presented in Algorithm 3.

**Algorithm 3**

```
1:  Inputs: $A$, $\tau$, $\delta$
2:  set $n_s = 0$ and initialize the vector $\mathbf{u}$ of squared norms
3:  while $n_s < n$ do
4:      $\{j_1, \ldots, j_{k_s}\} = \mathrm{DM}([A]_{n_s+1:m,n_s+1:n}, [\mathbf{u}]_{n_s+1:n}, \tau, \delta)$
5:      move columns $\{j_1, \ldots, j_{k_s}\}$ to the leading positions of $A$ and $\mathbf{u}$
6:      for $l = 1, \ldots, k_s$ do
7:          compute the Householder reflector $\mathbf{v}^{(n_s+l)}$ w.r.t. $[A]_{n_s+l:m,n_s+l}$
8:          update the remaining columns $[A]_{n_s+l:m,n_s+l+1:n_s+k_s} - \; = \mathbf{v}^{(n_s+l)}(\mathbf{v}^{(n_s+l)})^T [A]_{n_s+l:m,n_s+l+1:n_s+k_s}$
9:      end for
10:     compute the compact WY representation of $\mathbf{v}^{(n_s+1)}, \ldots, \mathbf{v}^{(n_s+k_s)}$
11:     block update $[A]_{n_s+1:m,n_s+k_s+1:n} - \; = Y^{(s)} (W^{(s)})^T (Y^{(s)})^T [A]_{n_s+1:m,n_s+k_s+1:n}$
12:     update the partial column norms $\mathbf{u}$
13:     $n_s = n_s + k_s$
14: end while
15: Output: $A$
```

Last, we point out that the partial column norms can be updated at each iteration also in this case with some straightforward changes of (21), namely

$$u_j^{(s+1)} = \begin{cases} \sqrt{\left(u_j^{(s)}\right)^2 - \sum_{l=n_s}^{n_{s+1}} r_{lj}^2}, & n_{s+1} < j \le n, \quad n_{s+1} \le n, \\ 0, & j \le n_{s+1}. \end{cases} \tag{29}$$

The above formula cannot be applied as it is because of numerical cancellation, like (21). Thus, we apply safety switch from [13] for a robust implementation.

Algorithm 2 has the particular feature that the diagonal elements of the final upper triangular factor $R$ are monotonically non-increasing in modulus, i.e., they satisfy (23). This is because we have (22), that also implies that the diagonal element is larger than any other extra diagonal entry in modulus, see (24). For what concerns Algorithm 3, an analogous of (22) cannot hold in general. Suppose that Algorithm 3 terminates in $S \ge 1$ steps for a given matrix $A$ and parameters $\tau$, $\delta$, and the dimension of the $R_{11}^{(s)}$ factor at the $s$-th algorithmic step is $n_s$, so that $0 = n_0 < n_1 < \cdots < n_S = n$. It is easy to show that a weaker version of (22) holds, namely we have

$$|r_{n_s+1,n_s+1}|^2 \ge \sum_{i=n_s+1}^{j} |r_{ij}|^2, \qquad n_s + 1 < j \le n,\ 0 \le s < S, \tag{30}$$

which essentially establishes that we have diagonally dominance only for the first pivot of each block, while the standard pivoting ensures it for all pivots (22). In particular, we can only ensure that

$$|r_{n_s,n_s}| \geq |r_{n_s+j,n_s+j}|, \; 1 \leq j \leq n_{s+1} - n_s, \; 1 \leq s < S \tag{31}$$

$$|r_{n_s,n_s}| \geq |r_{n_s,n_s+j}|, \; 1 \leq j \leq n_{s+1} - n_s, \; 1 \leq s < S. \tag{32}$$

Thus Algorithm 3 does not ensure that the factor $R$ will have a monotonically decreasing diagonal, as it is not the case for other recently proposed pivoting strategies [10].

*Remark 2* Geometric interpretation: We pointed out in Remark 1 that the standard pivoting can be seen as an approximate procedure to compute at each iteration the most linearly independent column from the columns already processed. Following this line, Algorithm 3 is an approximate procedure to compute at each iteration a set of linearly independent columns which are the most linearly independent from the columns already processed. In fact, the first task is achieved by selecting vector columns $\{\mathbf{c}_{j_1}, \ldots, \mathbf{c}_{j_k}\}$ which are pairwise orthogonal up to a factor $\delta$, i.e.,

$$\left| \frac{\mathbf{c}_{j_i}^T \mathbf{c}_{j_l}}{\|\mathbf{c}_{j_i}\| \|\mathbf{c}_{j_l}\|} \right| < \delta,$$

for all $1 \leq i, l \leq k$, $i \neq l$. The second task is achieved by selecting columns with the largest norm up to a factor $\tau$, i.e.,

$$\|\mathbf{c}_{j_l}\| \geq \tau \max_i \|\mathbf{c}_i\|,$$

for all $1 \leq l \leq k$.

### 3.3 Worst-case bound on the smallest singular value

Let us denote by $\bar{\sigma}^{(s)}$ the smallest singular value of the computed $R_{11}^{(s)}$ block at step $s$, that is

$$\bar{\sigma}^{(s)} = \sigma_{\min}\left(R_{11}^{(s)}\right).$$

Let us first report from [9] an estimate of $\bar{\sigma}^{(s+1)}$ for QRP.

**Theorem 1** *Let $R_{11}^{(s)}$ be the upper triangular factor of order $s$ computed by Algorithm 2. We have*

$$\bar{\sigma}^{(s+1)} \geq \sigma_{s+1}(A) \frac{\bar{\sigma}^{(s)}}{\sigma_1(A)} \frac{1}{\sqrt{2(n-s)(s+1)}}.$$

Before coming to the main result, we introduce the following auxiliary Lemma.

**Lemma 2** *With reference to the notation used for introducing the block partition in* (28)*, we have*

$$\sigma_{\min}(T) \geq \frac{\tau\sqrt{1-\tau}}{\sqrt{n - n_{s+1} + 1}} \sigma_{n_{s+1}}(A). \tag{33}$$

*Proof* Consider following column partitions $T = (\mathbf{t}_1 \dots \mathbf{t}_k)$, $\bar{T} = (\mathbf{t}_{k+1} \dots \mathbf{t}_{n-n_s})$, $R_{22}^{(s+1)} = (\mathbf{r}_{k+1} \dots \mathbf{r}_{n-n_s})$, and set $\mathbf{r}_j = 0$, for $1 \le j \le k$. Moreover, let $T = (t_{i,j})$, with $1 \le i \le j \le k$, and $\bar{T} = (t_{i,j})$ with $1 \le i \le k$, $1 \le j \le n - n_s$. First, notice that by (14) we have

$$\left\| \begin{matrix} t_{k,k} & t_{k,k+1}, \dots, t_{k,n-n_s} \\ 0 & R_{22}^{(s+1)} \end{matrix} \right\| \ge \sigma_{n_{s+1}}(A).$$

We have

$$\left\| \begin{matrix} t_{k,k} & t_{k,k+1}, \dots, t_{k,n-n_s} \\ 0 & R_{22}^{(s+1)} \end{matrix} \right\|^2 \le (n - n_{s+1} + 1) \max \left\{ t_{k,k}^2, \max_{j \ge k+1} \left( \|\mathbf{r}_j\|^2 + t_{k,j}^2 \right) \right\}.$$

Since $t_{k,j}^2 \le \|\mathbf{t}_j\|^2$, for all $1 \le j \le n - n_s$, and computing the maximum on a larger set of indices we have

$$\begin{aligned} \max \left\{ t_{k,k}^2, \max_{j \ge k+1} \left( \|\mathbf{r}_j\|^2 + t_{k,j}^2 \right) \right\} &\le \max \left\{ \|\mathbf{t}_k\|^2, \max_{j \ge k+1} \left( \|\mathbf{r}_j\|^2 + \|\mathbf{t}_j\|^2 \right) \right\} \\ &\le \max_{j \ge 1} \left( \|\mathbf{r}_j\|^2 + \|\mathbf{t}_j\|^2 \right). \end{aligned}$$

From equations (26–27), for all $1 \le j \le n - n_s$, we have

$$\|\mathbf{c}_j\|^2 = \|\mathbf{r}_j\|^2 + \|\mathbf{t}_j\|^2,$$

and, finally, since $\|\mathbf{t}_1\|^2 = \|\mathbf{c}_1\|^2 = \max_j \|\mathbf{c}_j\|^2$ and by using Lemma 1, we get

$$\left\| \begin{matrix} t_{k,k} & t_{k,k+1}, \dots, t_{k,n-n_s} \\ 0 & R_{22}^{(s+1)} \end{matrix} \right\|^2 \le (n - n_{s+1} + 1)\|\mathbf{c}_1\|^2 \le \frac{n - n_{s+1} + 1}{\tau^2(1-\tau)} \sigma_{\min}^2(C).$$

We can conclude by noticing that $\sigma_{\min}(T) = \sigma_{\min}(C)$, since the two matrices differ by a left multiplication by an orthogonal matrix. $\square$

By the interlacing property of singular values, we have

$$\bar{\sigma}^{(s+1)} \le \min \left\{ \bar{\sigma}^{(s)}, \sigma_{\min} \begin{pmatrix} B \\ T \end{pmatrix} \right\},$$

thus the bounds on $\bar{\sigma}^{(s)}$ and $\sigma_{\min}(T)$ are, by themselves, not a sufficient condition. Let us introduce the following result, which provides a bound of type (16) for Algorithm 3.

**Theorem 2** *Let $R_{11}^{(s)}$ be the upper triangular factor of order $n_s$ computed by Algorithm 3. We have*

$$\bar{\sigma}^{(s+1)} \ge \sigma_{n_{s+1}}(A) \frac{\bar{\sigma}^{(s)}}{\sigma_1(A)} \frac{1}{\sqrt{2(n - n_{s+1})n_{s+1}}} \frac{\tau\sqrt{1-\tau}}{k^2 n_s}.$$

*Proof* Let us drop the subscript and the superscript on the inverse of $R_{11}^{(s)}$ and its inverse $\left(R_{11}^{(s)}\right)^{-1}$, which will be denoted as $R$ and $R^{-1}$ respectively. Then, the inverse of matrix $R_{11}^{(s+1)}$ is given by

$$\left(R_{11}^{(s+1)}\right)^{-1} = \begin{pmatrix} R^{-1} & -R^{-1}BT^{-1} \\ & T^{-1} \end{pmatrix}.$$

Let us introduce the following partitions into rows

$$F = R^{-1}BT^{-1} = \begin{pmatrix} \mathbf{f}_1^T \\ \vdots \\ \mathbf{f}_{n_s}^T \end{pmatrix}, \quad R^{-1} = \begin{pmatrix} \mathbf{g}_1^T \\ \vdots \\ \mathbf{g}_{n_s}^T \end{pmatrix}, \quad T^{-1} = \begin{pmatrix} \mathbf{h}_1^T \\ \vdots \\ \mathbf{h}_k^T \end{pmatrix}.$$

The idea is to use (20), that is

$$\bar{\sigma}^{(s+1)} \leq \min_h \left\| \mathbf{e}_h^T \begin{pmatrix} R^{-1} & F \\ & T^{-1} \end{pmatrix} \right\|^{-1} \leq \sqrt{n_{s+1}}\sigma_{\min}\bar{\sigma}^{(s+1)},$$

to estimate the minimum singular value up to a factor $\sqrt{n_{s+1}}$. For $1 \leq h \leq n_{s+1}$ we have

$$\left\| \mathbf{e}_h^T \begin{pmatrix} R^{-1} & F \\ & T^{-1} \end{pmatrix} \right\|^2 = \begin{cases} \|\mathbf{g}_h\|^2 + \|\mathbf{f}_h\|^2, & h \leq n_s, \\ \|\mathbf{h}_{h-n_s}\|^2, & h > n_s. \end{cases}$$

We can bound $\|\mathbf{g}_h\|$ using (20) again, which gives

$$\bar{\sigma}^{(s)} \leq \min_h \left(\|\mathbf{g}_h\|^{-1}\right) \leq \sqrt{n_s}\bar{\sigma}^{(s)}.$$

In particular, for every $1 \leq h \leq n_s$, we get

$$\bar{\sigma}^{(s)} \leq \min_h \left(\|\mathbf{g}_h\|^{-1}\right) \leq \|\mathbf{g}_h\|^{-1},$$

and thus we have

$$\|\mathbf{g}_h\| \leq \frac{1}{\bar{\sigma}^{(s)}} = \frac{1}{\sigma_{\min}(R)} = \sigma_{\max}(R^{-1}) = \|R^{-1}\|.$$

Similarly, we can bound $\|\mathbf{h}_{h-n_s}\|$ by $\|T^{-1}\|$. Let us now concentrate on bounding $\|\mathbf{f}_h\|$. We have

$$\begin{aligned}
\|\mathbf{f}_h\|_2 \le \|\mathbf{f}_h\|_1 &= \sum_{l=1}^{k} |f_{hl}| = \sum_{l=1}^{k} \left| \sum_{i=1}^{k} [R^{-1}B]_{hi} [T^{-1}]_{il} \right| \\
&= \sum_{l=1}^{k} \left| \sum_{i=1}^{k} \sum_{j=1}^{n_s} [R^{-1}]_{hj} [B]_{ji} [T^{-1}]_{il} \right| \\
&\le \sum_{l=1}^{k} \sum_{i=1}^{k} \sum_{j=1}^{n_s} \left| [R^{-1}]_{hj} \right| \, \left| [B]_{ji} \right| \, \left| [T^{-1}]_{il} \right| \\
&\le \sum_{l=1}^{k} \sum_{i=1}^{k} \sum_{j=1}^{n_s} \left\| R^{-1} \right\|_{\max} \|B\|_{\max} \left\| T^{-1} \right\|_{\max} \\
&= k^2 n_s \left\| R^{-1} \right\|_{\max} \|B\|_{\max} \left\| T^{-1} \right\|_{\max} \\
&\le k^2 n_s \left\| R^{-1} \right\| \|B\| \left\| T^{-1} \right\| \\
&= \frac{k^2 n_s}{\bar{\sigma}^{(s)}} \|B\| \left\| T^{-1} \right\| ,
\end{aligned}$$

where we use the following well-known inequalities $\|\mathbf{x}\|_2 \le \|\mathbf{x}\|_1$, $\|A\|_{\max} \le \|A\|$. Moreover, we can write

$$\begin{aligned}
&\|\mathbf{g}_h\|^2 + \|\mathbf{f}_h\|^2 \le \frac{1}{(\bar{\sigma}^{(s)})^2} + \frac{k^4 n_s^2}{(\bar{\sigma}^{(s)})^2} \|B\|^2 \left\| T^{-1} \right\|^2 \\
&= \frac{\sigma_{\min}^2(T) + k^4 n_s^2 \|B\|^2}{(\bar{\sigma}^{(s)} \sigma_{\min}(T))^2} \le \frac{\|T\|^2 + k^4 n_s^2 \|B\|^2}{(\bar{\sigma}^{(s)} \sigma_{\min}(T))^2} \\
&\le \frac{2k^4 n_s^2}{(\bar{\sigma}^{(s)} \sigma_{\min}(T))^2} \max \left\{ \|T\|^2, \|B\|^2 \right\} \le \frac{2k^4 n_s^2}{(\bar{\sigma}^{(s)} \sigma_{\min}(T))^2} \|A\|^2,
\end{aligned}$$

where, in the last inequality, we used the interlacing property and the invariance under matrix transposition of the singular values. In fact

$$\sigma_1(A) \ge \sigma_1 \begin{pmatrix} B \\ T \end{pmatrix} = \sigma_1 \begin{pmatrix} B^T & T^T \end{pmatrix} \ge \max \left\{ \sigma_1(B), \sigma_1(T) \right\}.$$

Hence, we get

$$\frac{1}{\sqrt{\|\mathbf{g}_h\|^2 + \|\mathbf{f}_h\|^2}} \ge \frac{\bar{\sigma}^{(s)} \sigma_{\min}(T)}{\sqrt{2} k^2 n_s \sigma_1(A)}.$$

If $\bar{\sigma}^{(s)}$ is a good approximation of $\sigma_{n_s}(A)$, we can suppose that $\bar{\sigma}^{(s)}/\sigma_{n_s}(A) \approx 1$, and we can write

$$\begin{aligned}\sqrt{n_{s+1}}\bar{\sigma}^{(s+1)} &\geq \min\left\{\min_h \|\mathbf{h}_h\|^{-1}, \min_h \frac{1}{\sqrt{\|\mathbf{g}_h\|^2 + \|\mathbf{f}_h\|^2}}\right\} \\ &\geq \min\left\{1, \frac{\bar{\sigma}^{(s)}}{\sqrt{2}k^2 n_s \sigma_1(A)}\right\}\sigma_{\min}(T) \\ &= \frac{\bar{\sigma}^{(s)}}{\sqrt{2}k^2 n_s \sigma_1(A)}\sigma_{\min}(T).\end{aligned}$$

Finally, using Lemma 2, we get

$$\bar{\sigma}^{(s+1)} \geq \sigma_{n_{s+1}}(A)\frac{\bar{\sigma}^{(s)}}{\sigma_1(A)}\frac{1}{\sqrt{2(n-n_{s+1})n_{s+1}}}\frac{\tau\sqrt{1-\tau}}{k^2 n_s},$$

which is the desired bound. □

This shows that even if the leading $n_s$ columns have been carefully selected, so that $\bar{\sigma}^{(s)}$ is an accurate approximation of $\sigma_{n_s}(A)$, there could be a potentially dramatic loss of accuracy in the estimation of the successive block of singular values, namely $\sigma_{n_s+1}(A), \ldots, \sigma_{n_{s+1}}(A)$, just like for the standard column pivoting. In fact, it is well known that failure of QRP algorithm may occur (one such example is the Kahan matrix [23]), as well as for other greedy algorithms, but it is very unlikely in practice.

## 3.4 Termination criteria

In principle, both Algorithms 2 and 3 reveal the rank of a matrix. In finite arithmetic we have

$$\begin{pmatrix} \hat{R}_{11}^{(s)} & \hat{R}_{12}^{(s)} \\ & \hat{R}_{22}^{(s)} \end{pmatrix}, \tag{34}$$

where $\hat{R}_{ij}^{(s)}$ is the block $R_{ij}^{(s)}$ computed in finite representation, for $i = 1, 2$, $j = 2$. If the block $\hat{R}_{22}^{(s)}$ is small in norm, then it is reasonable to say that the matrix $A$ has rank $n_s$, where $n_s$ is the order of the upper triangular block $\hat{R}_{11}^{(s)}$. [17] propose the following termination criterion

$$\left\|\hat{R}_{22}^{(s)}\right\| \leq f(n)\varepsilon \|A\|, \tag{35}$$

where $\varepsilon$ is the machine precision and $f(n)$ is a modestly growing function of the number $n$ of columns. Notice that even if a block $\hat{R}_{22}^{(s)}$ with small norm implies numerical rank-deficiency, the converse is not true in general: an example is the Kahan matrix [23], discussed in Section 5.1. Let us write the column partition $\hat{R}_{22}^{(s)} = (\hat{\mathbf{c}}_1 \ \ldots \ \hat{\mathbf{c}}_{n-n_s})$. We have

$$\left\|\hat{R}_{22}^{(s)}\right\| \leq \sqrt{n-n_s}\max_i \left\|\hat{\mathbf{c}}_i\right\|, \quad \max_i \|\mathbf{a}_i\| \leq \|A\|.$$

Therefore, the stopping criterion (35) holds if

$$\sqrt{n-n_s}\max_i \left\|\hat{\mathbf{c}}_i\right\| \leq f(n)\varepsilon \max_i \|\mathbf{a}_i\|\,, \tag{36}$$

but the converse is not true in general. Suppose now to have input data with an initial uncertainty of a known order $\eta$ in $A$. In this case, the numerical rank may be defined up to a perturbation of order $\eta$, see (11), and the stopping criterion (36) is replaced as follows

$$\sqrt{n-n_s}\max_i \left\|\hat{\mathbf{c}}_i\right\| \leq \eta \max_i \|\mathbf{a}_i\|\,, \tag{37}$$

where $\eta$ is a user defined input parameter. We do not investigate this case, however it is left as an option in the software. In Section 5 we test the practical stopping criterion (36) and discuss the following two choices:

$$f(n) = n, \tag{38}$$
$$f(n) = \sqrt{n}. \tag{39}$$

## 4 The QR with deviation maximization algorithm

In this section we introduce the QR with deviation maximization (QRDM) algorithm and discuss some crucial aspects related to its implementation.

The deviation maximization procedure exploits diagonal dominance in order to ensure linear independence. Diagonal dominance is sufficient but obviously not necessary and it often turns out to be a too strong condition to be satisfied in practice. Let us briefly comment the choice of the parameter $\tau$: on the one hand, its value should be small in order to get a large candidate set $I$; on the other hand, a small value of $\tau$ implies a small value of $\delta < \tau$ if (5) or (6) are applied, likely yielding fewer selected columns. Notice that when the value of $\delta$ is close to zero, only pairwise nearly orthogonal columns are selected, and it is unlikely to find such matrices in real world problems. However, the value of $\delta$ can be chosen independently from $\tau$, as we now detail. Suppose to give up the constraint $\delta < \tau$ and to settle for any value of $\tau$ and $\delta$ in the interval (0, 1). Then the deviation maximization may identify a set of numerically linearly dependent columns. In order to overcome this issue, we incorporate a filtering procedure in the Householder triangularization. The selected columns $\left\{j_1,\ldots,j_{k_s}\right\}$ at the $s$-th algorithmic step satisfy

$$\|[A]_{n_s:m,n_s+j}\| \geq \varepsilon_s := \tau \max_{i>n_s} \|[A]_{n_s:m,i}\|, \quad j = j_1,\ldots,j_{k_s}, \tag{40}$$

before being reduced to triangular form. If a partial column norm becomes too small during the Householder triangularization, then that column is not sufficiently linearly independent from the columns already processed and the procedure is interrupted. In general, the converse is not true. For instance, we demand that the partial column norms $\|[A]_{n_s+l:m,n_s+l}\|$ do not become smaller than the parameter $\varepsilon_s$ defined above in order to compute the related Householder reflectors.

The QR computation obtained in this way is called QRDM and it is presented in Algorithm 4, where the filtering procedure on the partial column norms appears at step 9.

**Algorithm 4** QR with Deviation Maximization: $[A] = \text{QRDM}(A, \tau, \delta)$.

```
1: set $n_s = 0$ and initialize the vector $\mathbf{u}$ of squared norms
2: while $n_s < n$ do
3:     $\{j_1, \ldots, j_{k_s}\} = \text{DM}([A]_{n_s+1:m, n_s+1:n}, [\mathbf{u}]_{n_s+1:n}, \tau, \delta)$
4:     choose $\{j_1, \ldots, j_{k_s}\}$ by applying DM to $[A]_{n_s+1:m, n_s+1:n}$
5:     move columns $\{j_1, \ldots, j_{k_s}\}$ to the leading positions of $A$ and $\mathbf{u}$
6:     for $l = 1, \ldots, k_s$ do
7:         compute the Householder reflector $\mathbf{v}^{(n_s+l)}$ w.r.t. $[A]_{n_s+l:m, n_s+l}$
8:         update the remaining columns $[A]_{n_s+l:m, n_s+l+1:n_s+k_s} - = \mathbf{v}^{(n_s+l)}(\mathbf{v}^{(n_s+l)})^T\ [A]_{n_s+l:m, n_s+l+1:n_s+k_s}$
9:         if $l + 1 < k_s$ and $\|[A]_{n_s+l+1:m, n_s+l+1}\| < \varepsilon_s$ then break
10:        end if
11:    end for
12:    compute the compact WY representation of $\mathbf{v}^{(n_s+1)}, \ldots, \mathbf{v}^{(n_s+l)}$
13:    update the trailing matrix $[A]_{n_s+1:m, n_s+k_s+1:n} - = Y^{(s)}\ (W^{(s)})^T\ (Y^{(s)})^T\ [A]_{n_s+1:m, n_s+k_s+1:n}$
14:    update the partial column norms $\mathbf{u}$
15:    $n_s = n_s + l$
16: end while
```

Other values of $\varepsilon_s$ are possible, e.g., a small and constant threshold. However, numerical tests show that the choice (40) works well in practice. In this case, the Householder reduction to triangular form terminates with $l < k$ Householder reflectors, and the algorithm continues with the computation and the application of the compact WY representation of these $l$ reflectors. At the next iteration, the pivoting strategy moves the rejected column away from the leading position, if necessary. As we show in Section 5, this break mechanism enables us to independently set values for $\tau$ and $\delta$, and thus to obtain the best results in execution times.

## 4.1 Minimizing memory communication

The performance of an algorithm is highly impacted by the amount of communication performed during its execution, as explicitly pointed out in the literature, see, e.g., [10], where communication refers to data movement within a memory hierarchy of a processor or even between different processors of a parallel computer. In this context, the goal of this section is to design a pivoting strategy that is effective in revealing the rank of a matrix but also minimizes communication. Each time step 5 in Algorithm 3 is reached, the deviation maximization selects $k_s$ columns $\{j_1, \ldots, j_{k_s}\}$ to be moved to the leading positions $\{n_s + 1, \ldots, n_s + k_s\}$. However, if one of more columns are already situated within the leading position indices then it is not necessary to move them from their current positions: in this case, since the columns $\{j_1, \ldots, j_{k_s}\}$ are placed in the leading positions with a different ordering, the smallest singular value $\bar{\sigma}^{(s)}$ of the $R_{11}^{(s)}$ factor is unchanged, i.e., Theorem 2 still holds. This change on the pivoting strategy allows a huge memory saving in terms of data movement without affecting the rank-revealing properties of the resulting decomposition. This result does not come for free, namely we lose any monotonically decreasing trend in the magnitude of the diagonal elements of the $R_{11}$ factor and the weak diagonally dominance established in (30)–(32) does not hold anymore. Let us briefly describe the structure of the permutations employed. For every $i = 1, \ldots, k_s$, the column $j_i$ is not moved if it is within the $k_s$ leading positions. Otherwise, it is moved in place of the first free spot within the $k_s$ leading positions, namely we swap the columns $j_i$ and $n_s + l$, where $l$ is the minimum integer $1 \leq l \leq k_s$ such that $n_s + l \notin \left\{ j_1, \ldots, j_{k_s} \right\}$. In this way, the memory communication is minimized and the pivoting strategy requires only $m$ additional memory slots. Let us stress that this communication avoiding pivoting strategy if possible only when multiple columns are selected at once, and hence it cannot be extended to the QR decomposition with standard column pivoting.

## 5 Numerical experiments

In this section we discuss the numerical accuracy of QRDM against the SVD decomposition and the block QRP algorithm, briefly called QP3 [27]. We report experimental results comparing the double precision codes `dgeqrf` and `dgeqp3` from LAPACK, and `dgeqrdm`, a double precision C implementation of our block algorithm QRDM available online at the URL: https://github.com/mdessole/qrdm. All tests are carried out on a computer with an Intel(R) Core(TM) i7-2700K processor and a 8 GB system memory, employing CBLAS and LAPACKE, the C reference interfaces to BLAS and LAPACK implementations on Netlib, respectively. All codes have been compiled through a GNU Compiler Collection or a GNU Fortran compiler on a Linux system. The libraries BLAS and LAPACK have been installed from the package `libatlas-base-dev` for Linux Ubuntu, derived from the well-known ATLAS project (Automatically Tuned Linear Algebra Software), http://math-atlas.sourceforge.net/. It must be pointed out that the absolute timings of the algorithms here discussed are sensitive to the particular optimization adopted for the BLAS library, but it does not change the significance of the results here presented. Actually, a better optimization of the BLAS means more efficient BLAS-3 operations and QRDM increases the speedup with respect to QP3.

Particular importance is given to the values on the diagonal of the upper triangular factor $R$ of the RRQR factorization, which are compared with the singular values of the $R_{11}$ block and with the singular values of the input matrix $A$. The tests are carried out on several instances of the Kahan matrix [23] and on a subset of matrices from the San Jose State University Singular matrix database, which were used in other previous papers on the topic, see, e.g., [10, 19].

### 5.1 Kahan matrix

We first discuss the Kahan matrix [24, p. 31], that is defined as follows

$$K(n,\varphi) = \operatorname{diag}\left(1, \varsigma, \ldots, \varsigma^{n-1}\right) \begin{pmatrix} 1 & -\varphi & \ldots & -\varphi \\ & 1 & \ddots & \vdots \\ & & \ddots & -\varphi \\ & & & 1 \end{pmatrix}, \tag{41}$$

where $\varsigma^2 + \varphi^2 = 1$, and, in general we have

$$K(n,\varphi) = \begin{pmatrix} 1 & -\varphi \ \ldots \ -\varphi \\ 0 & \varsigma K(n-1,\varphi) \end{pmatrix}, \qquad \varphi = \cos(\alpha),\ \ \varsigma = \sin(\alpha). \tag{42}$$

For $\varphi = 0$, the singular values are all equal to one. An increasing gap between the last two singular values is obtained when the value of $\varphi$ is increased. The QRP algorithm does not perform any pivoting, producing a RRQR factorization in which the Q factor is the identity matrix and thus leaving these matrices unchanged [17]. This implies that $\left\| R_{22}^{(s)} \right\| \geq \varsigma^{n-1}$, for $1 \leq s \leq n-1$. For example, the matrix $K(300, 0.99)$ has no particular small trailing matrix though, since $\varsigma^{299} \approx 0.5$. In such case $\sigma_{\min}(R_{11}^{(n-1)})$ can be much smaller than $\sigma_{n-1}(K(n,\varphi))$ [19]. It is not difficult to see that the QRDM algorithm does not perform any pivoting on these matrices too. Let us show this fact by induction on the algorithmic step $s$. Let $(\mathbf{k}_1 \ldots \mathbf{k}_n)$ be the column partition of $K(n,\varphi)$. It is easy to see that all columns of the Kahan matrix have unit norm. Moreover, take $i < j$ and we have

$$\begin{aligned} \theta_{ij} &= \mathbf{k}_i^T \mathbf{k}_j = \sum_{l=1}^{n} k_{li} k_{lj} = \sum_{l=1}^{i} k_{li} k_{lj} = \varphi^2 \sum_{l=1}^{i-1} \varsigma^{2(l-1)} + \left(-\varphi \varsigma^{i-1}\right) \varsigma^{i-1} \\ &= 1 - \varsigma^{2(i-1)}(1+\varphi), \end{aligned}$$

that is the cosine of the angle $\alpha_{ij} \in [0, \pi)$ between $\mathbf{k}_i$ and $\mathbf{k}_j$ does not depend on $j$. In other words, the column $\mathbf{k}_i$ forms the same angle (modulo $\pi$) with all columns $\mathbf{k}_j$, with $j > i$, thus no column permutations are necessary in the first iteration of QRDM. Suppose no permutations are necessary in the first $s$ iterations, then (42) allows to write

$$K(n,\varphi) = \begin{pmatrix} K(n_s,\varphi) & \mathbf{b}_1 \ \ldots \ \mathbf{b}_{n-n_s} \\ & \mathbf{c}_1 \ \ldots \ \mathbf{c}_{n-n_s} \end{pmatrix} = \begin{pmatrix} K(n_s,\varphi) & \mathbf{b}_1 \ \ldots \ \mathbf{b}_{n-n_s} \\ & \varsigma^{n_s} K(n-n_s,\varphi) \end{pmatrix}.$$

At the $s$-th algorithmic step the trailing matrix is then $R_{22}^{(s)} = \varsigma^{n_s} K(n-n_s,\varphi)$, whose columns all have the same norm equal to $\varsigma^{n_s}$. Moreover, the column $\mathbf{c}_i$ forms the same angle (modulo $\pi$) with all other columns $\mathbf{c}_j$, for $1 \leq i < j \leq (n-n_s)$, hence no permutations are necessary. However, the matrix $K(n,\varphi)$ may not be in rank-revealed form. In this case, QRDM shows poor rank-reveling properties, similarly to QRP. It is well known that rounding errors due to finite precision may cause nontrivial permutation and the QRP algorithm may reveal the rank. Following [10], in order to avoid this issue we consider instead the matrix

$$\hat{K}(n,\varphi,\xi) = K(n,\varphi)\begin{pmatrix} (1-\xi) & & & \\ & (1-\xi)^2 & & \\ & & \ddots & \\ & & & (1-\xi)^n \end{pmatrix}, \qquad (43)$$

where $1 > \xi > 0$. In other words, the $j$-th column of the matrix $\hat{K}(n,\varphi,\xi)$ is the $j$-th column of $K(n,\varphi)$ scaled by $(1-\xi)^j$, for $j = 1,\dots,n$. Tables 1 and 2 show results for scaled Kahan matrices $\hat{K}(n,\varphi,\xi)$, with size $n = 128$, for several values of $\varphi$. For each test case, we show the last two singular values $\sigma_{n-1}, \sigma_n$ and the last two diagonal entries $k_{n-1,n-1}, k_{n,n}$ of the current instance of $\hat{K}(n,\varphi,\xi)$. For both algorithms QRP and QRDM, we report the $(n-1)$-th singular value $\bar{\sigma}_{n-1}$ of the $R_{11}$ block of order $n-1$ and the absolute value of the last two diagonal entries $d_{n-1}, d_n$ of the factor $R$. The singular values here presented computed with the `xgejsv` subroutine of LAPACK. Here, we use the following setting for the hyperparameters $\tau = 0.15$ and $\delta = 0.9$, whose choice is motivated in the next section. When $\xi$ is small (see Table 1), e.g., $\xi = 10^{-15}$, both algorithms reveal the rank for some values of $\varphi$. However, when the parameter $\xi$ is increased (see Table 2), e.g., $\xi = 10^{-7}$, the algorithms do not perform any pivoting for any value of $\varphi$, thus resulting in poor rank revealing, according to results in [10]. This fact can be deduced by comparing the diagonal values $d_{n-1}, d_n$ of the $R$ factor with the corresponding singular values $\sigma_{n-1}, \sigma_n$, and the singular value $\bar{\sigma}_{n-1}$ of the computed $R_{11}$ block with the corresponding singular value $\sigma_{n-1}$.

## 5.2 SJSU matrices

We now discuss results coming from two subsets of the San Jose State University Singular matrix database, that we call:

1. "small matrices": it consists of the 261 matrices with $m \le 1024$, $32 < n \le 2048$, sorted in ascending order with respect to the number of columns $n$;
2. "big matrices": it consists of the first 247 matrices with $m > 1024$, $n > 2048$, sorted in ascending order with respect to the number of columns $n$.

These datasets consist of "fat" ($m > n$), "tall" ($m < n$) and square matrices, the results presented hereafter do not depend on this characteristic. For each matrix $A$, we denote by $\sigma_i$ the $i$-th singular value of $A$ computed with the `xgejsv` subroutine of LAPACK, and by $r$ the numerical rank computed with the option `JOBA='A'`: in this case, small singular values are comparable with roundoff noise and the matrix is treated as numerically rank deficient. Deviation maximization does not guarantee that the diagonal values of the factor $R$ are monotonically decreasing in modulus, therefore we do not sort the diagonal entries and we denote by $d_i$ the $i$-th diagonal entry with positive sign. As an example, we show in Fig. 1 the singular values $\sigma_i$ and diagonal values $d_i$ computed with both QP3 and QRDM for the instance n. 3 of the set "small matrices". Figure 1a shows that the diagonal values $d_i$ computed with QP3 are monotonically decreasing, while the diagonal values $d_i$ computed with QRDM

**Table 1** Numerical tests on scaled Kahan matrices $\hat{K}(n, \varphi, \xi)$, where $n = 128$ and $\xi = 10^{-15}$

| $\varphi$ | Singular values | | Diagonal values | | QRP | | | QRDM | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\sigma_{n-1}$ | $\sigma_n$ | $k_{n-1,n-1}$ | $k_{n,n}$ | $\bar{\sigma}_{n-1}$ | $d_{n-1}$ | $d_n$ | $\bar{\sigma}_{n-1}$ | $d_{n-1}$ | $d_n$ |
| 0.1 | 5.57E-01 | 5.71E-06 | 5.31E-01 | 5.28E-01 | 6.32E-06 | 5.31E-01 | 5.28E-01 | 6.32E-06 | 5.31E-01 | 5.28E-01 |
| 0.2 | 8.37E-02 | 1.26E-11 | 7.64E-02 | 7.49E-02 | 1.54E-11 | 7.64E-02 | 7.49E-02 | 6.58E-07 | 9.17E-02 | 5.52E-06 |
| 0.3 | 3.00E-03 | 1.59E-17 | 2.63E-03 | 2.51E-03 | 8.72E-04 | 3.42E-03 | 8.02E-15 | 1.79E-06 | 3.42E-03 | 1.52E-12 |
| 0.4 | 2.01E-05 | 7.96E-24 | 1.70E-05 | 1.56E-05 | 2.01E-05 | 2.38E-05 | 6.45E-22 | 1.20E-06 | 2.38E-05 | 5.40E-19 |
| 0.5 | 1.65E-08 | 9.21E-31 | 1.35E-08 | 1.17E-08 | 1.65E-08 | 2.02E-08 | 3.61E-28 | 1.65E-08 | 2.02E-08 | 6.16E-27 |
| 0.6 | 7.79E-13 | 1.06E-38 | 6.16E-13 | 4.93E-13 | 7.79E-13 | 9.85E-13 | 9.33E-37 | 7.79E-13 | 9.85E-13 | 6.41E-35 |

**Table 2** Numerical tests on scaled Kahan matrices $\hat{K}(n, \varphi, \xi)$, where $n = 128$ and $\xi = 10^{-7}$

| $\varphi$ | Singular values | | Diagonal values | | QRP | | | QRDM | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\sigma_{n-1}$ | $\sigma_n$ | $k_{n-1,n-1}$ | $k_{n,n}$ | $\bar{\sigma}_{n-1}$ | $d_{n-1}$ | $d_n$ | $\bar{\sigma}_{n-1}$ | $d_{n-1}$ | $d_n$ |
| 0.1 | 5.57E-01 | 5.71E-06 | 5.31E-01 | 5.28E-01 | 6.32E-06 | 5.31E-01 | 5.28E-01 | 6.32E-06 | 5.31E-01 | 5.28E-01 |
| 0.2 | 8.37E-02 | 1.26E-11 | 7.64E-02 | 7.49E-02 | 1.54E-11 | 7.64E-02 | 7.49E-02 | 1.54E-11 | 7.64E-02 | 7.49E-02 |
| 0.3 | 3.00E-03 | 1.55E-17 | 2.63E-03 | 2.51E-03 | 2.17E-17 | 2.63E-03 | 2.51E-03 | 2.22E-17 | 2.63E-03 | 2.51E-03 |
| 0.4 | 2.01E-05 | 5.43E-21 | 1.70E-05 | 1.55E-05 | 9.02E-22 | 1.70E-05 | 1.55E-05 | 9.02E-22 | 1.70E-05 | 1.55E-05 |
| 0.5 | 1.65E-08 | 1.91E-24 | 1.35E-08 | 1.17E-08 | 1.90E-24 | 1.35E-08 | 1.17E-08 | 1.20E-24 | 2.02E-08 | 2.20E-10 |
| 0.6 | 7.79E-13 | 9.28E-29 | 6.16E-13 | 4.93E-13 | 3.08E-29 | 6.16E-13 | 4.93E-13 | 1.37E-28 | 9.85E-13 | 2.65E-14 |

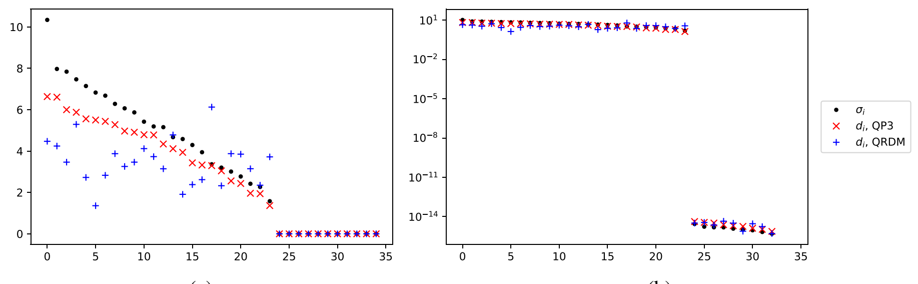

**Fig. 1** Singular values $\sigma_i$ $(\cdot)$ and diagonal values $d_i$ computed with QP3 $(\times)$ and QRDM $(+)$ for the 3rd instance of the set "small matrices", natural (a) and logarithmic (b) scale

are not ordered. However, as it is highlighted in Fig. 1b, the order of magnitude of $\sigma_i$ is well approximated by that of the corresponding $d_i$ for both methods.

Let us first discuss results provided by QP3. Figure 2 compares the positive diagonal entry $d_i$ and the correspoding singular value $\sigma_i$ for each matrix in the two collections, by taking into account the maximum and minimum value of the ratios $d_i/\sigma_i$. Results show that the positive diagonal value $d_i$ approximates the corresponding singular value $\sigma_i$ up to a factor 10, for $i = 1, \ldots, r$. Moreover, Fig. 3 compares $\sigma_i(R_{11})$, that is the $i$-th singular value of $R_{11} = [R]_{1:r,1:r}$ computed by LAPACK's `xgejsv`, with $\sigma_i$ for each matrix in the two collections, by taking into account the ratios $\sigma_i(R_{11})/\sigma_i$. These results confirm that QP3 provides an approximation of the singular value $\sigma_r$ up to a factor 10.

Before providing similar results for QRDM, let us discuss the sensitivity of parameters $\tau$ and $\delta$ to the rank-revealing property (16). To this aim, we set a grid $\mathscr{G}$ of

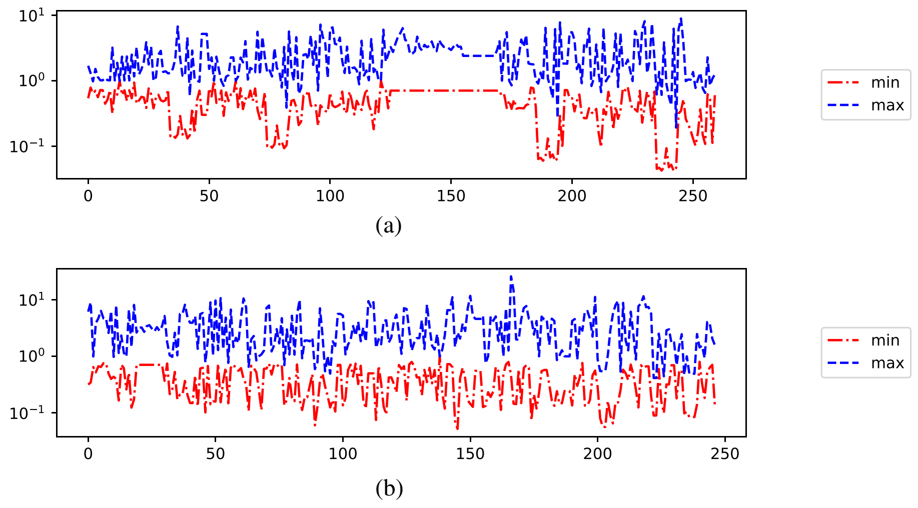

**Fig. 2** Ratio $d_i/\sigma_i$, minimum and maximum values for QP3 on the dataset "small matrices" (a) and "big matrices" (b)

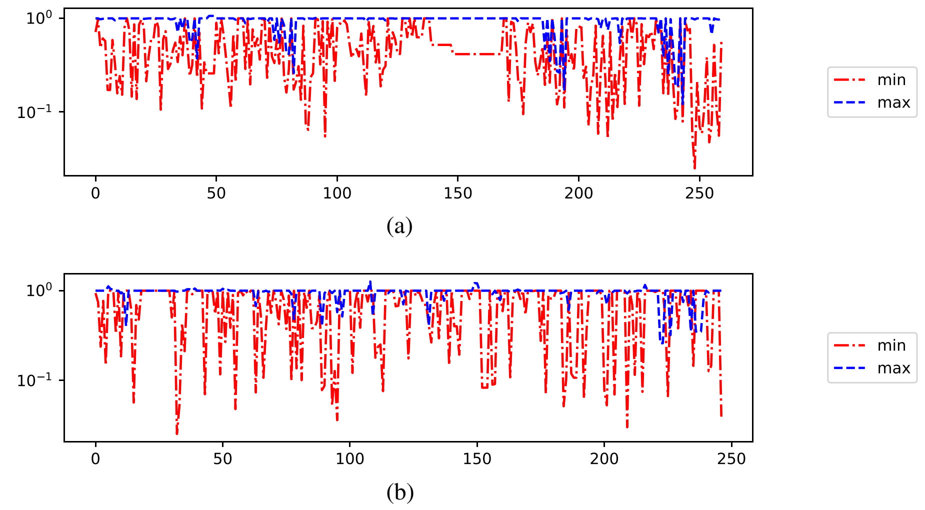

**Fig. 3** Ratio $\sigma_i(R_{11})/\sigma_i$, minimum and maximum values for QP3 on the dataset "small matrices" (a) and "big matrices" (b)

values $\mathscr{G}(i, j) = (\delta_i, \tau_j) = (0.05\, i, 0.05\, j)$, with $i, j = 0, \dots, 20$, and we consider the $R$ factor obtained by QRDM. Figure 4a shows the order of magnitude of

$$\min_A \min_{1 \le i \le r} \frac{d_i}{\sigma_i(A)},$$

where $A$ ranges in the collection "small matrices", for each grid point $(\delta_i, \tau_j)$. We see that the positive diagonal elements provide an approximation up to a factor 10 of the singular values for a wide range of parameters, corresponding to the light gray region of the grid that we call stability region. In practice, any choice of $1 \ge \tau > 0$ and $1 > \delta \ge 0$ leads to a rank-revealing QR decomposition.

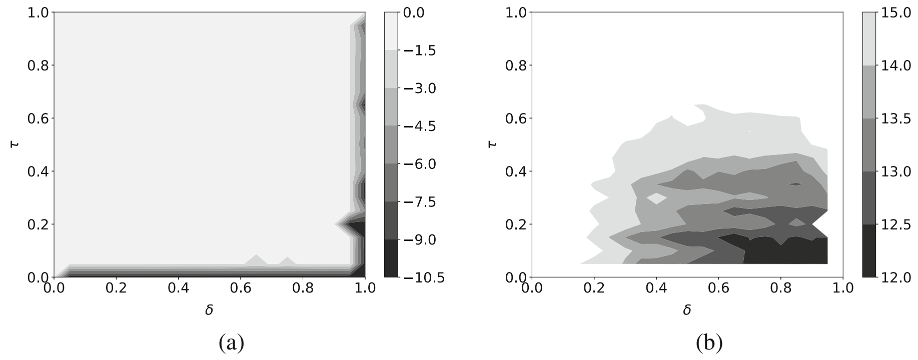

**Fig. 4** Order of magnitude of the minimum $\min(r_i/\sigma_i)$ over all matrices (a) and cumulative execution times for QRDM (b) in function of the parameters $\tau$ and $\delta$ on the "small matrices" data set

Therefore, for an optimal parameters' choice, we look at execution times. Figure 4b shows the cumulative execution times (in seconds) $\sum_A t_{QRDM}(A)$ where $A$ ranges in the "small matrices" collection and where $t_{QRDM}(A)$ is the execution time of QRDM, for each grid point $(\delta_i, \tau_j)$ in the stability region. It is evident that best performances are obtained toward the right-bottom corner, in correspondence of the dark gray region. Hence, we set $\tau = 0.15$ and $\delta = 0.9$, which are the optimal values for the validation set here considered.

We can now analyze the quality of the RRQR factorization obtained by QRDM with the choice of parameters just discussed. Figure 5 shows that the positive diagonal entries approximate the singular values up to a factor 10, and Fig. 6 shows that the singular values of $R_{11}$ provide an approximation up to a factor $10^2$, loosing an order of approximation with respect to QP3 in very few cases.

Let us now consider QRDM with a stopping criterion. We show the accuracy in the determination of the numerical rank, and the benefits in terms of execution times, when the matrix rank is much smaller than its number of columns. We consider the stopping criterion in (36)–(38): the numerical rank is this case is given by the number of columns processed by QRDM and we denote it by $\tilde{r}$. The matrix

$$A_{\tilde{r}} = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \Pi^T$$

denotes the corresponding rank-$\tilde{r}$ approximation of $A$. Figure 7 shows the ratios $\sigma_{\tilde{r}+1}/\|A\|$ (in red) and $\|A - A_{\tilde{r}}\|/\|A\|$ (in blue), for all matrices in the "small matrices" (Fig. 7a) and "big matrices" (Fig. 7b) collections. Whenever the $i$-th matrix has full rank, i.e., it has rank $\tilde{r}$, the singular value $\sigma_{\tilde{r}+1}$ does not exist and we replaced its value with $\varepsilon = 10^{-16}$. We also considered the stopping criterion in (36) with the

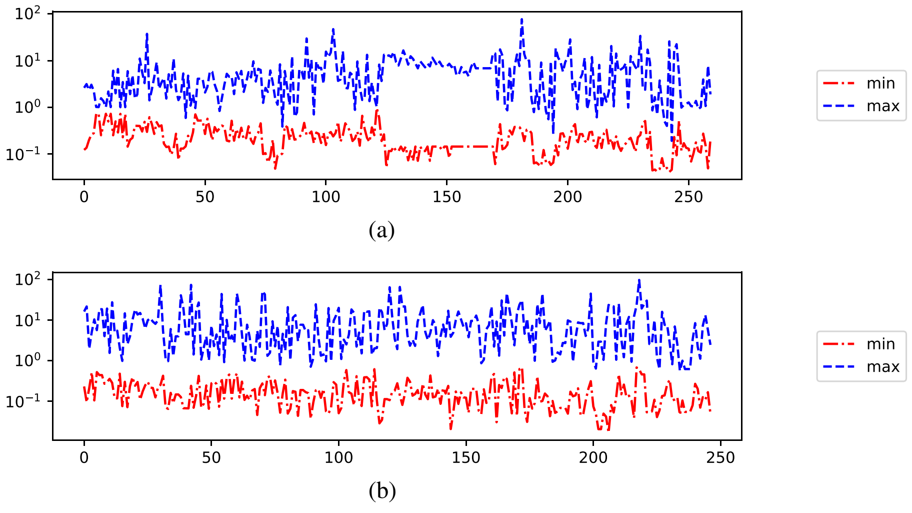

**Fig. 5** Ratio $d_i/\sigma_i$, minimum and maximum values for QRDM on the dataset "small matrices" (a) and "big matrices" (b)

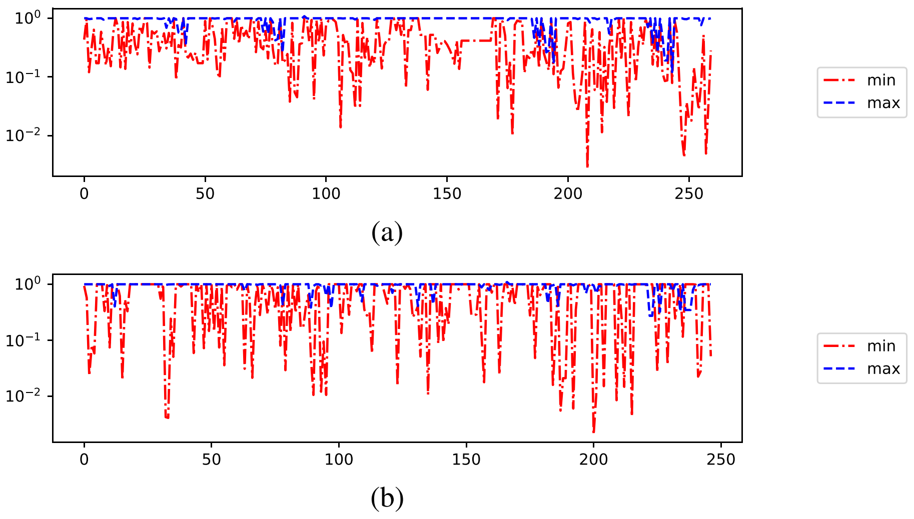

**Fig. 6** Ratio $\sigma_i(R_{11})/\sigma_i$, minimum and maximum values for QRDM on the dataset "small matrices" (a) and "big matrices" (b)

choice (39), which turned out to be less accurate and for this reason we omit the results.

Finally, we compare the execution times of QR computations for the matrices of the set "big matrices". Here, the instances have been ordered accordingly to the total number of entries $nm$. Figure 8 shows the speedup of QRDM (Fig. 8a) and QRDM with stopping criterion (Fig. 8b) over QP3, namely the ratio $t_{QP3}/t_{QRDM}$, where

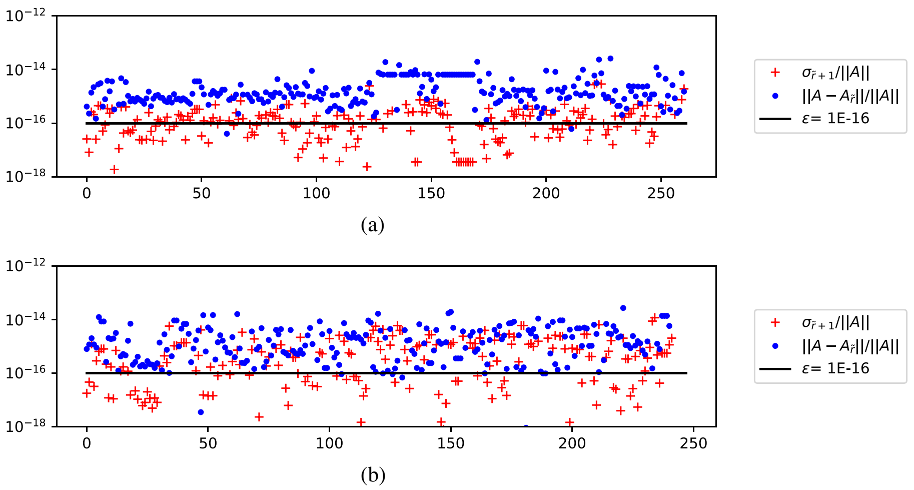

**Fig. 7** Relative error on the computed numerical rank $\tilde{r}$ for QRDM expressed as the ratios $\sigma_{\tilde{r}+1}/\|A\|$ (+) and $\|A - A_{\tilde{r}}\|/\|A\|$ (·), on the set "small matrices" (a) and "big matrices" (b)

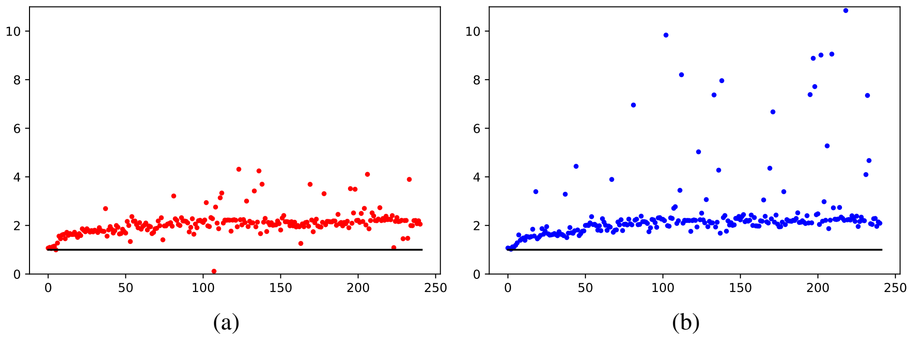

**Fig. 8** Speedup of QRDM (a) and QRDM with stopping criterion (b) over QP3

$t_{QP3}$ and $t_{QRDM}$ are the execution times (in seconds) of QP3 and QRDM respectively. The algorithm QRDM achieves an average speedup of $2.1\times$, as a consequence of the lower amount of memory communication employed to carry out the pivoting, as detailed in Section 4.1. In a stopping criterion is adopted, the average speedup reached is $2.5\times$. It may also be interesting to consider a comparison with an implementation of QP3 with a termination criterion, but this is beyond the scope of the present work.

Figure 9 compares QP3 and QRDM with the QR without pivoting, briefly called QR, implemented by the `dgeqrf` subroutine of LAPACK. We display the ratios $t_{QRDM}/t_{QR}$ and $t_{QP3}/t_{QR}$, where $t_{QR}$ is the array of execution times (in seconds) of QR. The standard QR is indeed faster, since it does not involve any column permutation, and it is in average $3\times$ faster than QP3 while it is only $1.3\times$ faster than QRDM. This result is obtained thanks to the permutation strategy described in Section 4.1.

Last, let us discuss briefly the effect of the block size $k_{DM}$ introduced to limit the cardinality of the candidate set in (10). This parameter depends on the specific architecture, mainly in terms of cache-memory size, and typical values are $k_{DM} =$

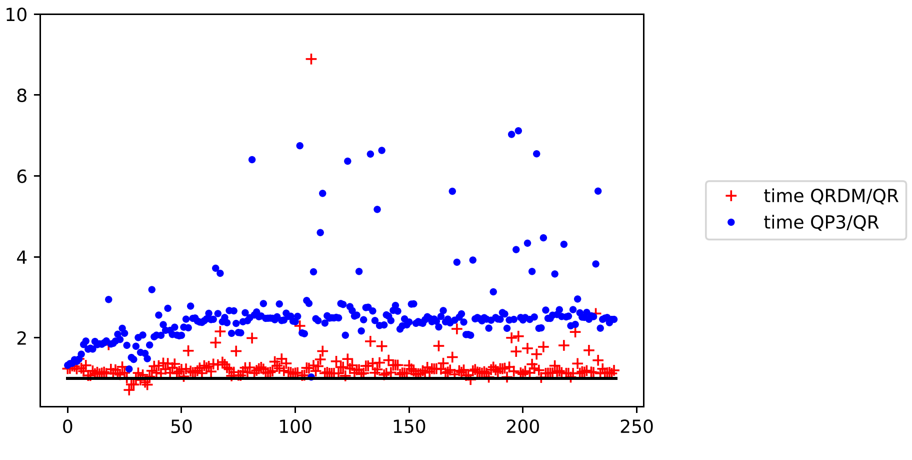

**Fig. 9** Overhead of QRDM (+) and QP3 (·) over the QR without pivoting

32, 64, 128. We observed that there is an optimal value of $k_{DM}$, in sense that it gives the smallest for a fixed experimental setting, and its computation is similar to the well-known computation practice of the BLAS block size, which is out of scope of this paper. For sake of clarity we say that in our test environment we observed the optimal value $k_{DM} = 64$, but other choices exhibit a similar behavior, e.g., $k_{DM} = 32$.

## 6 Conclusions

In this work we have presented a new subset selection strategy we called "Deviation Maximization". Our method relies on correlation analysis in order to select a subset of sufficiently linearly independent vectors. Despite this strategy is not sufficient by itself to identify a maximal subset of linearly independent columns for a given numerically rank deficient matrix, it can be adopted as a column pivoting strategy. We introduced the rank-revealing QR factorization with Deviation Maximization pivoting, briefly called QRDM, and we compared it with the rank-revealing QR factorization with standard column pivoting, briefly QRP. We have provided a theoretical worst case bound on the smallest singular value for QRDM and we have shown it is similar to available results for QRP. Extensive numerical experiments confirmed that QRDM reveals the rank similarly to QRP and provides a good approximation of the singular values obtained with LAPACK's `xgejsv` routine. Moreover, we have shown that QRDM has better execution times than those of QP3 implemented in LAPACK's double precision `dgeqp3` routine for a large number of test cases, thanks to the lower amount of memory communication involved. The software implementation of QRDM used in this article is available at the URL: https://github.com/mdessole/qrdm.

Our future work will focus on applying deviation maximization as pivoting strategy to other problems which require column selection, e.g., constrained optimization problems, on which the authors successfully experimented a preliminary version in the context of active set methods for NonNegative Least Squares problems, see [11, 12].

**Acknowledgements** The authors sincerely thank the anonymous referees for their constructive suggestions that improved the quality of the paper.

**Funding** M. Dessole gratefully acknowledges the company beanTech S.r.l. for funding the doctoral grant "GPU computing for modeling, nonlinear optimization and machine learning". This work was partially supported by the Project BIRD192932 of the University of Padova.

**Data availability** The data that support the findings of this study are available from https://github.com/mdessole/qrdm, the datasets analyzed are available from http://www.math.sjsu.edu/singular/matrices.

## Declarations

**Conflict of interest** The authors declare no competing interests.

## References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA, 3rd edn. ISBN 0-89871-447-8 (paperback) (1999)
2. Barlow, J., Demmel, J.: Computing accurate eigensystems of scaled diagonally dominant matrices. SIAM J. Numer. Anal. **27**, 11 (1990). https://doi.org/10.1137/0727045
3. Bischof, C., Hansen, P.: A block algorithm for computing rank-revealing QR factorizations. Numer. Algo. **2**, 371–391,10 (1992). https://doi.org/10.1007/BF02139475
4. Bischof, C., Quintana-Ortí, G.: Computing rank-revealing QR factorizations of dense matrices. ACM Trans. Math. Softw. **24**, 226–253, 06 (1998a). https://doi.org/10.1145/290200.287637
5. Bischof, C., Quintana-Ortí, G.: Algorithm 782: codes for Rank-Revealing QR factorizations of dense matrices. ACM Trans. Math. Softw. **24**, 254–257, 07 (1998b). https://doi.org/10.1145/290200.287638
6. Bischof, J.R.: A block QR factorization algorithm using restricted pivoting. In: Supercomputing '89:Proceedings of the 1989 ACM/IEEE Conference on Supercomputing, pp. 248–256 (1989). https://doi.org/10.1145/76263.76290
7. Businger, P., Golub, G.H.: Linear Least Squares Solutions by Householder Transformations. Numer. Math. **7**(3), 269–276 (1965). ISSN 0029-599X. https://doi.org/10.1007/BF01436084
8. Chan, T.F.: Rank revealing QR factorizations. Linear Algebra Appl. **88-89**, 67–82 (1987). ISSN 0024-3795. https://doi.org/10.1016/0024-3795(87)90103-0. http://www.sciencedirect.com/science/article/pii/0024379587901030
9. Chandrasekaran, S., Ipsen, I.C.F.: On Rank-Revealing factorisations. SIAM J. Matrix Anal. Appl. **15**(2), 592–622 (1994). https://doi.org/10.1137/S0895479891223781
10. Demmel, J., Grigori, L., Gu, M., Xiang, H.: Communication avoiding rank revealing QR factorization with column pivoting. SIAM J. Matrix Anal. Appl. **36**, 55–89, 01 (2015). https://doi.org/10.1137/13092157X
11. Dessole, M., Marcuzzi, F., Vianello, M.: Accelerating the Lawson-Hanson NNLS solver for large-scale Tchakaloff regression designs. Dolomites Research Notes on Approximation **13**, 20–29 (2020a). ISSN 2035-6803. https://doi.org/10.14658/PUPJ-DRNA-2020-1-3. https://drna.padovauniversitypress.it/2020/1/3
12. Dessole, M., Marcuzzi, F., Vianello, M.: DCATCH—a numerical package for d-variate near g-optimal Tchakaloff regression via fast NNLS. Mathematics **8**, 7 (2020b). https://doi.org/10.3390/math8071122
13. Drmač, Z., Bujanović, Z.: On the Failure of Rank-Revealing QR Factorization Software – A Case Study. ACM Trans. Math. Softw. 35(2). ISSN 0098-3500. https://doi.org/10.1145/1377612.1377616 (2008)
14. Duersch, J.A., Gu, M.: Randomized QR with column pivoting. SIAM J. Sci. Comput. **39**(4), C263–C291 (2017). https://doi.org/10.1137/15M1044680
15. Foster, L.V.: Rank and null space calculations using matrix decomposition without column interchanges. Linear Algebra Appl. **74**, 47–71 (1986). ISSN 0024-3795. https://doi.org/10.1016/0024-3795(86)90115-1. https://www.sciencedirect.com/science/article/pii/0024379586901151
16. Golub, G.: Numerical methods for solving linear least squares problems. Numer. Math. **7**(3), 206–216 (1965). ISSN 0029-599X. https://doi.org/10.1007/BF01436075
17. Golub, G., Van Loan, C.: Matrix Computations (4th ed.). Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore (2013). ISBN 9781421407944

18. Golub, G., Klema, V., Stewart, G.W.: Rank degeneracy and least squares problems. Technical Report STAN-CS-76-559. Department of Computer Science Stanford University, Stanford (1976)
19. Gu, M., Eisenstat, S.C.: Efficient algorithms for computing a strong Rank-Revealing QR factorization. SIAM J. Sci. Comput. **17**(4), 848–869 (1996). https://doi.org/10.1137/0917055
20. Hansen, P.C.: Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion. Society for Industrial and Applied Mathematics, USA (1999). ISBN 0898714036
21. Higham, N.J.: A survey of condition number estimation for triangular matrices. SIAM Rev. **29**(4), 575–596 (1987). ISSN 0036-1445. https://doi.org/10.1137/1029112
22. Hong, Y.P., Pan, C.-T.: Rank-revealing QR factorizations and the singular value decomposition. Math. Comput. **58**(197), 213–232 (1992). ISSN 00255718, 10886842. http://www.jstor.org/stable/2153029
23. Kahan, W.: Numerical linear algebra. Can. Math. Bull. **9**, 757–801 (1966)
24. Lawson, C.L., Hanson, R.J.: Solving least squares problems, vol. 15. SIAM, Bangkok (1995)
25. Martinsson, P.G.: Blocked rank-revealing QR factorizations: How randomized sampling can be used to avoid single-vector pivoting. Report, 05. arXiv:1505.08115 (2015)
26. Mikhalev, A., Oseledets, I.: Rectangular maximum-volume submatrices and their applications. Linear Algebra Appl. **538**, 187–211 (2018). ISSN 0024-3795. https://doi.org/10.1016/j.laa.2017.10.014. https://www.sciencedirect.com/science/article/pii/S0024379517305931
27. Quintana-Ortí, G., Sun, X., Bischof, C.H.: A BLAS-3 version of the QR factorization with column pivoting. SIAM J. Sci. Comput. **19**(5), 1486–1494 (1998). https://doi.org/10.1137/S1064827595296732
28. Schreiber, R., VanLoan, C.: A Storage-Efficient WY representation for products of householder transformations. SIAM J. Sci. Stat. Comput. **10**, 02 (1989). https://doi.org/10.1137/0910005
29. Thompson, R.: Principal submatrices IX: Interlacing inequalities for singular values of submatrices. Linear Algebra Appl. **5**(1), 1–12 (1972). ISSN 0024-3795. https://doi.org/10.1016/0024-3795(72)90013-4. https://www.sciencedirect.com/science/article/pii/0024379572900134
30. Varah, J.: A lower bound for the smallest singular value of a matrix. Linear Algebra Appl. **11**(1), 3–5 (1975). ISSN 0024-3795. https://doi.org/10.1016/0024-3795(75)90112-3. http://www.sciencedirect.com/science/article/pii/0024379575901123
31. Xiao, J., Gu, M., Langou, J.: Fast Parallel Randomized QR with Column Pivoting Algorithms for Reliable Low-Rank Matrix Approximations. In: 2017 IEEE 24Th International Conference on High Performance Computing (HiPC), pp. 233–242, 12 (2017). https://doi.org/10.1109/HiPC.2017.00035