Truncated QR factorization with pivoting in mixed precision

Alfredo Buttari, Theo Mary, André Pacteau

TRUNCATED QR FACTORIZATION WITH PIVOTING IN MIXED PRECISION*

ALFREDO BUTTARI†, THEO MARY‡, AND ANDRÉ PACTEAU*

Abstract. Low-rank approximations are widely used to reduce the memory footprint and operational complexity of numerous linear algebra algorithms in scientific computing and data analysis. In some of our recent work we have demonstrated that low-rank approximations can be stored using multiple arithmetic precisions to further reduce the storage and execution time. In this work we present a method that can produce this mixed-precision representation directly; this relies on a mixed-precision truncated rank-revealing QR (RRQR) factorization with pivoting. We present a floating-point error analysis and provide bounds on the error of the approximation demonstrating that the use of multiple precisions does not alter the overall accuracy. Finally, we present experimental results showing the execution time reduction for the cases where either classical or randomized pivoting are used.

Key words. Mixed-precision, QR factorization, low-rank approximations

MSC codes. 65F55, 65G50

1. Introduction. Alongside many classical applications in data analysis, the use of low-rank approximations in computing has become increasingly popular in recent years due to their effectiveness in reducing both the memory consumption and the operational complexity of numerous linear algebra algorithms such as linear system solvers [2, 24, 13]. Essentially, these rely on the idea that a matrix $A \in \mathbb{R}^{m\times n}$ can be approximately represented as a product XY^T where $X \in \mathbb{R}^{m\times k}$ and $Y \in \mathbb{R}^{n\times k}$ are matrices of rank k such that

$$\left\|A - XY^T\right\| \leq \varepsilon \left\|A\right\|$$

in some norm, where ε is a prescribed accuracy tolerance; if k is sufficiently small compared to m and n, this approximation can be used to reduce the storage and operational complexity of operations on A at the cost of a controlled loss of accuracy. For a given ε, an optimal approximation can be obtained by computing the singular value decomposition of A and dropping all the singular values smaller than ε and the corresponding left and right singular vectors [12]. Nevertheless, this method is rarely used in practice due to its high cost and low efficiency. Instead, other methods are preferred which are sufficiently accurate and robust in practice, and more computationally efficient and/or scalable in a parallel setting; among these, we can cite rank-revealing QR factorizations [6] or randomized approaches [22].

Concurrently, low-precision floating-point arithmetic units have become increasingly available and supported not only in specialized computing platforms (such as GPUs) but also in commodity CPUs. This is partly due to the recent explosion of artificial intelligence and machine learning algorithms in science and engineering which work remarkably well with low (e.g., 16-bit) or even very low (8-bit) precisions. As a result, new floating-point arithmetic formats have been proposed and, sometimes, standardized, such as binary16 or BFloat16, which can achieve higher performance

*Submitted to the editors 06/03/2024.

Funding: This work was supported by the France 2030 NumPEx Exa-SofT (ANR-22-EXNU-0003) project managed by the French National Research Agency (ANR) and the ANR MixHPC project (ANR-23-CE46-0005-01).

†Université de Toulouse, CNRS, IRIT, Toulouse France, (alfredo.buttari@irit.fr)

‡Sorbonne Université, CNRS, LIP6, Paris, France, (theo.mary@lip6.fr)

than the traditional 32-bit or 64-bit floating-point formats. Despite their wide adoption in machine learning, these low-precision formats cannot be straightforwardly employed in applications which require relatively high accuracy; this has reignited the interest of the scientific computing community around mixed-precision algorithms. These combine multiple arithmetic formats in order to achieve provably accurate results while maximizing the use of low-precision units in order to improve performance. In particular, in our recent work [1] we have demonstrated that low-rank approximations can be stored in a mixed-precision format which can then be used in a direct method to solve linear systems of equations in a backward stable way. This approach amounts to partitioning the columns of X and Y into block-columns such that the overall accuracy of the approximation

$$\left\| A - X^1 {Y^1}^T - \cdots - X^p {Y^p}^T \right\| \leq \varepsilon \, \|A\| \tag{1.1}$$

is still satisfied when the X^i and Y^i data are stored, from left to right, using different arithmetics of decreasing precision. Intuitively, this can be explained by the fact that the rightmost columns of X and Y are associated with small singular values which carry little information and play a minor role in the error of the low-rank approximation. This splitting is dictated by the spectrum of A, the number and accuracy of the p available precisions and the threshold ε. In the same work we have demonstrated that this mixed-precision format can be used to reduce the execution time of matrix factorizations without harming the backward stability. The format in equation (1.1) can straightforwardly be obtained by first computing the low-rank approximation fully in high precision and then by appropriately casting the block-columns of X and Y into lower precisions; nevertheless, using this naive approach leads to poor performance in contexts where computing the high precision low-rank approximation is the bottleneck.

In the present work, we propose a mixed-precision pivoted QR factorization that can directly compute the mixed-precision low-rank approximation of a matrix A such that the condition in equation (1.1) is satisfied. We make the following contributions:

1. We present, in section 3, a rounding error analysis of the Householder QR factorization where the arithmetic precision is gradually reduced in the course of the algorithm. We produce a theoretical upper bound on the error demonstrating that, if these changes of precision are appropriately operated, the accuracy of the resulting low-rank approximation can be made the same as in the case where only high-precision arithmetic is used.
2. We present, in section 4, two mixed-precision QR factorization algorithms that rely, respectively, on the Businger-Golub and randomized pivoting.
3. Finally, in section 5, we present an experimental evaluation of these algorithms which validates the theoretical findings and demonstrates the higher performance of the mixed-precision algorithms with respect to their full high-precision counterparts.

A mixed-precision algorithm for computing a mixed-precision low-rank approximation of a matrix A was recently proposed by Connolly *et al.* [7]. This, however, relies on the randomized range finder method [22] and the resulting low-rank approximation is stored in high precision despite the computations being carried in mixed-precision.

2. Background. In the reminder of this document, we will use the following notation. Upper-case and lower-case roman letters denote, respectively, matrices and

vectors, while Greek letters denote scalars. Subscripts will be used to denote matrix or vector indices and superscripts to denote data associated with different steps of some algorithm or sequence of operations. To keep the notation simple, when there is no ambiguity, we will denote a column of a matrix using the corresponding lower-case letter with a subscript indicating the column index; for example, the j-th column of matrix A will be denoted a_j.

2.1. Householder QR and error analysis. The Householder QR factorization [19] of an $m \times n$ matrix A proceeds in n steps where, at each step k, an elementary reflector H^k is computed such that all the subdiagonal coefficients in column k are annihilated; as a result, the A matrix is reduced into an upper triangular matrix R and the product of the reflectors defines the orthogonal Q factor:

$$H^n \cdots H^1 A = Q^T A = R, \text{ where } H^k = (I - \tau^k v^k (v^k)^T).$$

Here the Q matrix is never explicitly computed but implicitly represented by the v^k vectors and τ^k scalars; furthermore, the first $k-1$ coefficients in vector v^k are null. The v^k and τ^k vectors can be computed in different ways, for example, to prevent cancellations; we refer the reader to the article by Lehouc [21] for an exhaustive discussion of this argument. Our work relies on the error analysis of Higham [16] (Chapter 19) which assumes that these are computed following the convention used in the LINPACK and LAPACK libraries.

It must be noted that, for the sake of performance, these Householder transformations are not computed and applied to A individually but, rather, in blocks of size $b \ll n$ using the WY representation of Schreiber *et al.* [25]; this allows for using Level-3 BLAS operations for most of the computations which results in a much faster execution due to an efficient use of cache memories.

Our error analysis of section 3 will use the standard model of floating-point arithmetic [16, sect. 2.2]. We put a hat on variables to denote that they represent computed quantities. For any integer k, we define

$$\gamma_k = \frac{ku}{1-ku}$$

where u denotes the unit roundoff of the employed arithmetic; a superscript on γ denotes that u carries that superscript; thus $\gamma_k^f = \frac{ku^f}{(1-ku^f)}$, for example. We also use the notation $\tilde{\gamma}_k = \gamma_{\eta k}$ to hide modest constants η.

Some modern computing devices are equipped with vector or matrix multiplication units where products are done in low precision (for example 16-bit) and accumulation in high precision (for example, 32-bit). For these devices, the error bounds of inner product–based computations can be reduced; see [3] for a detailed analysis of the mixed precision matrix multiply–accumulate available on Google TPUs, NVIDIA GPUs, etc. The reduction of the error bound however only affects the constant part of the bound: thus, a bound nu_{low} becomes $2u_{\text{low}} + nu_{\text{high}}$ if the accumulation is done in precision u_{high} instead of u_{low}. This could be taken into account in our analysis of section 3 by accordingly reducing the precision switch threshold. The use of such hardware would thus in principle allow for a greater use of lower precisions. We do not explore this option in our work.

Note, also, that in writing γ_n we implicitly assume $nu \ll 1$ for all the employed precisions which might not hold for relatively large values of n and u. In practice, though, the actual error rarely attains this worst-case bound, due to several factors.

First, hardware with very low (16 or 8-bit) precisions typically use high precision accumulation; as mentioned above, this reduces the constant in the bound [3]. Second, blocked algorithms benefit from a reduction of the error bound by a factor proportional to the block size b [16, p.64], [4]. Third, probabilistic effects usually allow for replacing the constants by their square root [18]. We refer to [17] for a thorough discussion on why extreme-scale low-precision computations are much more accurate than what worst-case analysis would suggest.

We will make extensive use of the results in lemmas 19.1, 19.2 and 19.3 and theorem 19.4 of Higham [16] which we will report below (with some slight adaptations and notation changes) for the sake of self-completeness prior to extending them to the case where the factorization is truncated and conducted using multiple precisions.

LEMMA 2.1 (19.1 and 19.2 of Higham [16]). *Assume a Householder transformation H is computed and applied to a vector b using precision u. The computed result $\widehat{y}$ satisfies*

$$\widehat{y} = (H + \Delta H)b, \quad \|\Delta H\|_F \leq \tilde{\gamma}_m.$$

The previous lemma demonstrates that computing and applying a Householder transformation in finite precision is a backward stable operation. The next one demonstrates that this property also holds when multiple transformations of this type are applied to a vector.

LEMMA 2.2 (19.3 of Higham [16]). *Consider the sequence of transformations*

$$b^i = H^i b^{i-1}, \quad b^0 = b, \ i = 1, \ldots, k.$$

The computed $\widehat{b}^k$ satisfies

$$\widehat{b}^k = H^k \cdots H^1 (b + \Delta b) = Q^T (b + \Delta b), \quad \|\Delta b\|_2 \leq \tilde{\gamma}_{mk} \|b\|_2.$$

A Householder QR factorization simply consists in applying to a matrix A a sequence of transformations that annihilate all the subdiagonal coefficients one column at a time; therefore, Lemma 2.2 applies on all columns, which leads to the following theorem.

THEOREM 2.3 (19.4 of Higham [16]). *Let $\widehat{R} \in \mathbb{R}^{m \times n}$ be the computed upper trapezoidal QR factor of $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) obtained via the Householder QR algorithm. Then there exists an orthogonal $Q \in \mathbb{R}^{m \times m}$ such that*

$$A + \Delta A = Q\widehat{R}$$

where

$$\|\Delta a_j\|_2 \leq \tilde{\gamma}_{mn} \|a_j\|_2, \quad j = 1, \ldots, n.$$

The latter theorem simply says that one such Q matrix exists. In practice, it is more useful to have a bound using the actually computed $\widehat{Q}$, as stated in the next Theorem.

THEOREM 2.4 (from Higham [16]). *Let $\widehat{R} \in \mathbb{R}^{m \times n}$ and $\widehat{Q} \in \mathbb{R}^{m \times m}$ be, respectively, the computed upper trapezoidal and orthogonal QR factors of $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) obtained via the Householder QR algorithm. Then*

$$\left\| A - \widehat{Q}\widehat{R} \right\|_F \leq \sqrt{n} \tilde{\gamma}_{mn} \|A\|_F. \tag{2.1}$$

Proof. It must be noted that $\widehat{Q}$ is obtained by applying the sequence of transformations to the identity matrix; therefore, Theorem 2.3 can be used to derive the following bound

$$\widehat{Q} = Q(I + \Delta I), \quad \|\Delta i_j\|_2 \leq \tilde{\gamma}_{mn}$$

where Δi_j denoted the j-th column of matrix ΔI; this implies

$$\left\| Q - \widehat{Q} \right\|_F \leq \sqrt{n}\tilde{\gamma}_{mn}. \tag{2.2}$$

Now, using equation (2.2) and Theorem 2.3

$$\begin{aligned}
\left\| \left(A - \widehat{Q}\widehat{R} \right)_j \right\|_2 &= \left\| \left(A - Q\widehat{R} \right)_j + \left(\left(Q - \widehat{Q} \right) \widehat{R} \right)_j \right\|_2 \\
&\leq \tilde{\gamma}_{mn} \|a_j\|_2 + \sqrt{n}\tilde{\gamma}_{mn} \|\widehat{r}_j\|_2 \\
&= \tilde{\gamma}_{mn} \|a_j\|_2 + \sqrt{n}\tilde{\gamma}_{mn} \|Q\widehat{r}_j\|_2 \\
&= \tilde{\gamma}_{mn} \|a_j\|_2 + \sqrt{n}\tilde{\gamma}_{mn} \|a_j + \Delta a_j\|_2 \\
&\leq (1 + \sqrt{n} + \sqrt{n}\tilde{\gamma}_{mn})\tilde{\gamma}_{mn} \|a_j\|_2 \\
&= \sqrt{n}\tilde{\gamma}_{mn} \|a_j\|_2
\end{aligned}$$

which implies the result. Note that, in the last step, one second-order term was dropped and the "1" was absorbed by $\tilde{\gamma}_{mn}$. ∎

2.2. QR factorization with Businger-Golub pivoting. A QR factorization capable of revealing the rank of a matrix (RRQR for Rank-Revealing QR) can be obtained using column pivoting as in the method proposed by Businger *et al.*[5]. Essentially, at each step k of the QR factorization, this method permutes the columns of the trailing submatrix such that the pivotal column k is the one that has maximum 2-norm. This implies that the diagonal coefficients of the R factor are of non-increasing absolute value and that the following property holds:

$$AP = QR \text{ where } |R_{k,k}| \geq \|R_{k:m,j}\|_2 \quad j = k, ..., n. \tag{2.3}$$

Note that explicitly recomputing the norm of all the columns in the trailing submatrix not only is expensive but completely prevents the use of blocking (and, thus, of Level-3 BLAS operations) because this requires entirely updating all the remaining columns after every elimination step. This problem can be partially overcome taking advantage of the fact that, once all the column norms of the original A matrix have been computed, these do not have to be recomputed at every step but can be cheaply updated. Essentially, at step k of the factorization, the norm of column j in the trailing submatrix is updated by subtracting the freshly computed $R_{k,j}$ coefficient. This approach is shown in Algorithm 2.1 which we refer to as QRCP; here we have assumed that V is a matrix containing all the computed v^k vectors in its columns, that householder$(x,\ k)$ is a function which computes and applies a Householder reflection that annihilates the bottom $m-k+1$ coefficients of a vector x of size m and that W is a workspace. Although, in this algorithm, a large portion of computations is still done using Level-2 BLAS operations, some Level-3 can be used (in line 13).

Algorithm 2.1 Blocked QR factorization with Businger-Golub pivoting (QRCP).

1: **Input:** $A \in \mathbb{R}^{m \times n}$
2: **Let:** $\eta_j = \|A_{:j}\|_2, \quad j = 1, \ldots, n,\ P = I$
3: **for** $j = 1 : b : n$ **do**
4: **for** $k = j : j + b - 1$ **do**
5: Find $i \in k, \ldots, n$ such that η_i is maximal
6: Swap columns k and i in A and P
7: $A_{k:m,k} = A_{k:m,k} - V_{k:m,j:k-1} W_{j:k-1,k}$
8: $v^k,\ \tau^k = \text{householder}\,(A_{:,k},\ k)$
9: $W_{k+1,k:n} = \tau^k (v^k)^T A_{:,k+1:n} + \tau^k (v^k)^T V_{:,j:k-1} W_{j:k-1,k+1:n}$
10: $A_{k,k+1:n} = A_{k,k+1:n} - V_{k,j:k-1} W_{j:k-1,k+1:n}$
11: $\eta_i = \sqrt{\eta_i^2 - A_{k,i}^2},\ i = k+1, ..., n$
12: **end for**
13: $A_{j+b:m,j+b:n} = A_{j+b:m,j+b:n} - V_{j+b:m,j:j+b-1} W_{j:j+b-1:j+b:n}$
14: **end for**
15: **Output:** $Q = \prod_{k=1,n} (I - \tau^k v^k (v^k)^T),\ R = triu(A),\ P$

It must be noted that the column norm update in line 11 of Algorithm 2.1 is a very delicate step when carried in finite precision as it may be subject to severe cancellations. This problem might be overcome using the approach proposed by Drmač[10] where if, in line 11 the norm of some columns drops by a value larger than a prescribed threshold which depends on the arithmetic unit roundoff, the algorithm breaks out of the inner loop of line 4, fully updates the trailing submatrix (line 13) and explicitly recomputes the norm of the problematic columns. This approach might reduce the portion of Level-3 BLAS operations but renders the method robust. Algorithm 2.1 with this updating technique is implemented in the LAPACK `_GEQP3` routine.

2.3. QR factorization with randomized pivoting. In practice, Algorithm 2.1 achieves very poor performance and parallel scalability because a large portion of computations in Algorithm 2.1 are of Level-2 BLAS type and because of the numerous communications that the Businger-Golub pivoting requires. For this reason alternative pivoting techniques have been proposed in the literature that aim at overcoming these drawbacks. Multiple methods proposed in the literature [11, 23, 8, 9] rely on an approach where, at every step of the factorization, not one but b (the panel size) selected columns of the trailing submatrix are moved upfront, eliminated and the corresponding transformations applied at once using Level-3 BLAS operations as in the WY technique described above. These methods essentially differ in the way these b columns are selected. In the approach proposed by Duersch *et al.* [11] and Martinsson *et al.* [23], illustrated in Algorithm 2.2, this selection is done using randomized sampling, that is, the trailing submatrix is left-multiplied by a i.i.d. Gaussian matrix $\Omega \in \mathcal{N}(0,1)^{(b+p)\times(m-j)}$ which produces a sample matrix S; here we assume that $j-1$ columns of A have already been eliminated and p is an *oversampling* parameter of moderate value (less than ten). Because of the properties that connect S to the trailing submatrix, the "important" b columns can be selected by applying QRCP to the sample matrix S. This drastically improves the amount of level-3 BLAS operations because S has a much smaller row-dimension than the trailing submatrix. As a matter of fact, the sample matrix S does not have to be recomputed at every factorization step but it can be computed only once and cheaply updated [11, 23].

Algorithm 2.2 Blocked QR factorization with randomized pivoting (QRRP).

Input: $A \in \mathbb{R}^{m \times n}$
Let: $P = I,\ \Omega \in \mathcal{N}(0,1)^{(b+p)\times(m-j)},\ S = \Omega A$
for $j = 1 : b : n$ **do**
  $\tilde{Q}, \tilde{R}, \tilde{P} = \text{QRCP}(S_{:,j:n})$
  $A_{:,j:n} = A_{:,j:n}\tilde{P},\ P_{:,j:n} = P_{:,j:n}\tilde{P}$
  $Q^j,\ R^j = QR(A_{j:m,j:j+b-1})$
  $A_{j:m,j+b:n} = (Q^j)^T A_{j:m,j+b:n}$
  Update S
end for
Output: $Q = \prod Q^j,\ R = triu(A),\ P$

[26] demonstrate that the property of equation (2.3) does not apply formally to the result of QRRP but holds in a probabilistic sense. Given $\varepsilon,\ \Delta \in (0,1)$ and an oversampling parameter $p \geq \lceil \frac{4}{\varepsilon^2 - \varepsilon^3} \log(\frac{2nk}{\Delta}) \rceil$ the following property

$$|R_{k,k}| \geq \sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\, \|R_{k:m,j}\|_2\,, \quad i+1 \leq j \leq n \tag{2.4}$$

holds with probability at least $1 - \Delta$.

3. Error analysis. Let us assume that we have p precisions such that the respective unit roundoff satisfy

$$u^1 \leq u^2 \leq \cdots \leq u^p$$

and for each precision i a sequence of k^i transformations are computed and applied to matrix A such that

$$\sum_{i=1}^{p} k^i = k \leq n.$$

Therefore, first k^1 transformations are computed and applied with precision u^1, then k^2 with precision u^2 and so forth:

$$\overbrace{\left(H^{p,k^p} \cdots H^{p,1}\right)}^{u^p} \cdots \overbrace{\left(H^{1,k^1} \cdots H^{1,1}\right)}^{u^1} A = Q^T A = \begin{bmatrix} R \\ A^{p+1} \end{bmatrix}. \tag{3.1}$$

Here we use an extra superscript to denote the precision at which each transformation is computed and applied. Furthermore, we denote as A^{i+1} the trailing submatrix after K^i of the above transformations are applied where

$$K^i = \sum_{j=1}^{i} k^j.$$

Note that, because we are interested in a truncated QR factorization, k might be smaller than n, in which case A^{p+1} corresponds to a $(m-k) \times n$ matrix with the rightmost $n-k$ columns being non-zero.

Each A^i matrix has $m - K^{i-1}$ rows and n columns, the first K^{i-1} columns being equal to zero; A^1 corresponds to the initial matrix A. The Q and R matrices in

equation (3.1) can be split in block-columns and block-rows, respectively, such that

$$A = [Q_1 \cdots Q_p Q_{p+1}] \begin{bmatrix} R_1 \\ \vdots \\ R_p \\ A^{p+1} \end{bmatrix}. \tag{3.2}$$

Note that each Q_i is a $m \times k^i$ submatrix of Q and is the result of computations carried in precisions 1 through i. Equivalently, R_i is a $k^i \times n$ submatrix of R and is the result of computations carried in precisions 1 through i. Our mixed-precision low-rank approximation of A is obtained by dropping Q_{p+1} and A^{p+1} in the above equation.

Our analysis will rely on the observation that in the QR factorization the i-th of such transformations has the following structure

$$\begin{bmatrix} I^{i-1} & \\ & \bar{H}^i \end{bmatrix}$$

where I^{i-1} is the identity matrix of size $i-1$; this is because the i-th transformation is computed so as to annihilate all the subdiagonal coefficients in the i-th column of A and implies that the application of such transformation will only concern the bottom $m-i+1$ rows of the matrix. In the analysis of Higham [16] this property was not used because in the case where a single precision is used it does not yield any significant improvement of the bounds. For our analysis, it is not necessary to consider the structure of each single transformation but, rather, it is enough to note that all transformations at precision i have the same structure

$$\begin{bmatrix} I^{K^{i-1}} & \\ & \bar{H}^{i,j} \end{bmatrix}, \quad j = 1, \ldots, k^i.$$

LEMMA 3.1 (equivalent of Lemma 2.2 in mixed precision). *Consider the sequence of transformations*

$$b^{K^i} = \prod_{j=1}^{k^i} H^{i,j} b^{K^{i-1}}, \qquad b^0 = b, \ i = 1, \ldots, p, \quad K^p = k.$$

where all transformations $H^{i,j}$, $j = 1, ..., k^i$ are computed and applied in precision u^i and have the following structure

$$H^{i,j} = \begin{bmatrix} I^{K^{i-1}} & \\ & \bar{H}^{i,j} \end{bmatrix}. \tag{3.3}$$

The computed $\widehat{b}^k$ satisfies

$$\widehat{b}^k = Q^T(b + \Delta b), \qquad \|\Delta b\|_2 \leq \sum_{i=1}^{p} \tilde{\gamma}^i_{mk^i} \left\| b^{K^{i-1}}_{K^{i-1}+1:m} \right\|_2.$$

Proof. The result follows by using Lemma 2.2 "in packets" where, in each packet, the Householder transformations have the structure of equation (3.3). Each packet of k^i transformations at precision u^i only concerns rows $K^{i-1}+1, \ldots, m$ of $b^{K^{i-1}}$ and, by Lemma 2.2, introduces an error bounded by $\tilde{\gamma}^i_{mk^i} \left\| b^{K^{i-1}}_{K^{i-1}+1:m} \right\|_2$. Note that it would be more appropriate to use $\tilde{\gamma}^i_{(m-K^{i-1})k^i}$ but we use m instead of $m - K^{i-1}$ to keep the notation simple. ∎

Based on Lemma 3.1, we are now ready to derive a columnwise error bound for a truncated QR factorization in mixed precision.

LEMMA 3.2 (equivalent of Theorem 2.3 with truncation and mixed precision). *Assume that a truncated QR factorization is computed such that $k \leq n$ Householder transformations are computed and applied to a matrix $A \in \mathbb{R}^{m\times n}$ using p different precisions of increasing unit roundoff u^i. Let k^i be the number of transformations that are computed using precision i. Then there exist matrices $Q_1, \ldots, Q_{p+1}$ such that the computed $\widehat{R}^i$ and $\widehat{A}^{p+1}$ satisfy*

$$(A + \Delta A) = [Q_1 \cdots Q_p Q_{p+1}] \begin{bmatrix} \widehat{R}_1 \\ \vdots \\ \widehat{R}_p \\ \widehat{A}^{p+1} \end{bmatrix}, \qquad \|\Delta a_j\|_2 \leq \sum_{i=1}^{p} \tilde{\gamma}^i_{mk^i} \left\|a^i_j\right\|_2 \tag{3.4}$$

and, consequently,

$$\left\| \left(A - \sum_{i=1}^{p} Q_i \widehat{R}_i \right)_j \right\|_2 \leq \left\| a^{p+1}_j \right\|_2 + \sum_{i=1}^{p} \tilde{\gamma}^i_{mk^i} \left\|a^i_j\right\|_2 . \tag{3.5}$$

Proof. Equation (3.4) straightforwardly results from the application of Lemma 3.1 to the case where the sequence of Householder transformations is computed so as to annihilate all the subdiagonal coefficients of A and, therefore, have the structure defined in equation (3.3). Equation (3.5), instead, follows from the observation that

$$\widehat{a}^{p+1}_j = a^{p+1}_j + \Delta a^{p+1}_j, \quad \left\|\Delta a^{p+1}_j\right\|_2 \leq \sum_{i=1}^{p} \tilde{\gamma}^i_{mk^i} \left\|a^i_j\right\|_2 .$$

∎

THEOREM 3.3 (equivalent of Theorem 2.4 with truncation and mixed precision). *Assume that a truncated QR factorization is computed such that $k \leq n$ Householder transformations are computed and applied to a matrix $A \in \mathbb{R}^{m\times n}$ using p different precisions of increasing unit roundoff u^i. Let k^i be the number of transformations that are computed using precision i. The computed $\widehat{R}_i$ and $\widehat{Q}_i$ satisfy*

$$\left\| A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i \right\|_F \leq \left\|A^{p+1}\right\|_F + \sum_{i=1}^{p} \sqrt{k^i} \tilde{\gamma}^i_{mk^i} \left\|A^i\right\|_F . \tag{3.6}$$

Proof. Following the same path that led us to equation (2.2), in the case where multiple precisions are used we obtain

$$\left\| Q_i - \widehat{Q}_i \right\|_F \leq \sqrt{k^i} \sum_{j=1}^{i} \tilde{\gamma}^j_{mk^j} = \sqrt{k^i} \tilde{\gamma}^i_{mk^i}$$

which allows us to derive a bound on the quality of the approximation using the

actually computed $\widehat{Q}_i$ and $\widehat{R}_i$

$$\begin{aligned}
\left\| \left(A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i \right)_j \right\|_2 &= \left\| \left(A - \sum_{i=1}^{p} Q_i \widehat{R}_i \right)_j + \sum_{i=1}^{p} \left(\left(Q_i - \widehat{Q}_i \right) \widehat{R}_i \right)_j \right\|_2 \\
&\leq \left\| \left(A - \sum_{i=1}^{p} Q_i \widehat{R}_i \right)_j \right\|_2 + \sum_{i=1}^{p} \left\| Q_i - \widehat{Q}_i \right\|_F \left\| \left(\widehat{R}_i \right)_j \right\|_2 \\
&\leq \left\| a_j^{p+1} \right\|_2 + \sum_{i=1}^{p} \tilde{\gamma}^i_{mk^i} \left\| a_j^i \right\|_2 + \sum_{i=1}^{p} \sqrt{k^i} \tilde{\gamma}^i_{mk^i} \left\| \left(\widehat{R}_i \right)_j \right\|_2 \\
&\leq \left\| a_j^{p+1} \right\|_2 + \sum_{i=1}^{p} \sqrt{k^i} \tilde{\gamma}^i_{mk^i} \left\| a_j^i \right\|_2
\end{aligned}$$

which implies the result. ∎

4. Pivoted QR factorization in mixed precision. Based on the theoretical findings of the previous section, we are now ready to formulate a truncated rank-revealing QR factorization in mixed precision. We will first introduce two truncated QR factorization algorithms based, respectively, on the Businger-Golub and randomized pivoting, and then a mixed-precision algorithm that can use either of these.

Specifically, the truncated mixed-precision algorithm relies on the use of the error bound in equation (3.6) to gradually switch to lower precision and, eventually, to halt the factorization as soon as the prescribed accuracy ε is reached. The terms of this bound, however, are difficult, in practice, to compute accurately. In order to make this bound more usable in practice, we can proceed to some simplifications. First of all we can ignore the constants related to the accumulation of rounding errors, which are often pessimistic [15]; this amounts to replacing each $\tilde{\gamma}^i_{mk^i}$ with the unit roundoff of the corresponding arithmetic precision u^i. Second, because in the course of the factorization it is not known beforehand how many transformations will be computed with each precision i, we will replace k^i with $n - K^{i-1}$. The error bound thus becomes

$$\left\| A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i \right\|_F \leq \left\| A^{p+1} \right\|_F + \sum_{i=1}^{p} \sqrt{n - K^{i-1}} u^i \left\| A^i \right\|_F . \tag{4.1}$$

Assuming that a low-rank approximation of relative accuracy ε is to be computed

$$\left\| A - \sum_{i=1}^{p} \widehat{Q}_i \widehat{R}_i \right\|_F \leq \left\| A^{p+1} \right\|_F + \sum_{i=1}^{p} \sqrt{n - K^{i-1}} u^i \left\| A^i \right\|_F \leq \varepsilon \left\| A \right\|_F$$

and, once again, ignoring the constants, it will be enough to ensure that all the terms in the bound are smaller than or equal to $\varepsilon \left\| A \right\|_F$. That is to say, it will be possible to switch from precision i to precision $i+1$ at step j of the factorization when

$$\sqrt{n - j} u^{i+1} \left\| A_{j:m, j:n} \right\|_F \leq \varepsilon \left\| A \right\|_F \tag{4.2}$$

and the factorization can be truncated at step k when

$$\left\| A_{k:m, k:n} \right\|_F \leq \varepsilon \left\| A \right\|_F . \tag{4.3}$$

It must be noted that our error analysis does not make any assumption on how the pivoting is done. Actually, for our method to work, it is only required that the norm

of the trailing submatrix decreases in the course of the factorization which happens even in the case where no pivoting is applied. Obviously, the use of pivoting will lead to a faster decay of the trailing submatrix norm and, consequently, to an earlier truncation and change of precisions.

4.1. Truncated QR factorization with Businger-Golub pivoting. The two criteria in equations (4.2) and (4.3) can be straightforwardly used to halt Algorithm 2.1 because the Frobenius norm of the trailing submatrix is readily available as the 2-norm of part of the vector storing the η_i, that is, the 2-norm of the corresponding columns. The resulting algorithm is presented in Algorithm 4.1 and amounts to a simple modification of Algorithm 2.1 where the factorization is interrupted as soon as either equation (4.2) or (4.3) is satisfied. It must be noted that, when the truncation happens because equation (4.2) is satisfied, it is implicitly assumed that the factorization will be continued using a lower precision (see the details in subsection 4.3) and, therefore, the trailing submatrix update on line 17 is necessary. If, instead, the factorization is interrupted because equation (4.3) is satisfied, the trailing submatrix is discarded and therefore need not be updated; for the sake of conciseness we do not include this optimization in Algorithm 4.1 although it is implemented in the code we used for the experimental evaluation of subsection 5.2.

Algorithm 4.1 Truncated blocked QR factorization with Businger-Golub pivoting (TQRCP).

1: **Input:** $A \in \mathbb{R}^{m\times n},\ \varepsilon_t,\ \varepsilon_p$
2: **Let:** $\eta_j = \|A_{:j}\|_2,\ \ j = 1,\ldots,n,\ P = I$
3: **for** $j = 1 : b : n$ **do**
4: **for** $k = j : j + b - 1$ **do**
5: **if** $\|\eta_{k:n}\|_2 \le \varepsilon_t$ **or** $\sqrt{n-j}\,\|\eta_{k:n}\|_2 \le \varepsilon_p$ **then**
6: $b = k - j; k = k - 1$
7: **goto** 17 and **break** j loop
8: **end if**
9: $\arg\min_i \{\eta_i,\ i = k : n\}$
10: Swap column k and i in A and P
11: $A_{k:m,k} \mathrel{-}= V_{k:m,j:k-1}W_{j:k-1,k}$
12: $v^k,\ \tau^k = \text{householder}\,(A_{:,k}))$
13: $W_{k+1,k:n} = \tau^k (v^k)^T A_{:,k+1:n} + \tau^k (v^k)^T V_{:,j:k-1} W_{j:k-1,k+1:n}$
14: $A_{k,k+1:n} = A_{k,k+1:n)} - V_{k,j:k-1} W_{j:k-1,k+1:n}$
15: $\eta_i = \sqrt{\eta_i^2 - A_{k,i}^2},\ i = k+1,...,n$
16: **end for**
17: $A_{j+b:m,j+b:n} \mathrel{-}= V_{j+b:m,j:j+b-1} W_{j:j+b-1:j+b:n}$
18: **end for**
19: **Output:** $Q_{1:m,1:k} = \prod_{i=1,k} (I - \tau^i v^i (v^i)^T),\ R = triu(A_{1:k,1:n}),\ P,\ k$

It must be noted that, thanks to the property in equation (2.3), the Frobenius norm of the trailing submatrix at step k can be bounded using $|R_{k,k}|$, that is, $\|A_{k:n,k:n}\|_F \le \sqrt{n-k}|R_{k,k}|$; this criterion, however, can largely overestimate the norm of the trailing submatrix and result in a late switch of precision, or truncation leading to sub-optimal performance.

4.2. Truncated QR factorization with randomized pivoting. In the case of the QR factorization with randomized pivoting in Algorithm 2.2, the norm of the

trailing submatrix cannot be easily computed. Nevertheless, it is possible to check for precision switch and truncation within the QRCP factorization of the sample matrix based on the norm of the trailing submatrix of the sample. According to Theorem 3.1 by [11], after i transformations the trailing 2-norm squared of the sample, that is, the norm of the columns in the trailing submatrix of the sample, can be written as a factor of the actual trailing column norms. Assuming S is a sample of row-dimension $l = b + p$ for a matrix A and S^i and A^i the corresponding trailing submatrices after i Householder transformations, $\left\|s_j^i\right\|_2^2 = x_j \left\|a_j^i\right\|_2^2$ where x_j has a truncated chi-squared distribution with $l-i$ degrees of freedom. Although the expectation of x_j can theoretically be smaller than $l-i$, in practice we have found that assuming $x_j = l-i$ or, more conservatively, $x_j = p$ works very well on our experimental test set (see experiments in subsection 5.2). Alternatively, the property in equation (2.4) can be used to obtain a probabilistic bound on the norm of the trailing submatrix but this would likely result in an excessively pessimistic criterion.

Based on this discussion, we propose the truncated QR factorization with randomized pivoting in Algorithm 4.2; this relies on Algorithm 4.1 for the factorization of the sample matrix.

Algorithm 4.2 Truncated blocked QR factorization with randomized pivoting (TQRRP).

Input: $A \in \mathbb{R}^{m\times n},\ \varepsilon_t,\ \varepsilon_p$
Let: $P = I,\ \Omega \in \mathcal{N}(0,1)^{(b+p)\times(m-j)},\ S = \Omega A$
for $j = 1 : b : n$ **do**
  $\tilde{Q}, \tilde{R}, \tilde{P}, \tilde{k} = \mathrm{TQRCP}(S_{:,j:n}, \sqrt{p}\varepsilon_t, \sqrt{p}\varepsilon_p)$
  $A_{:,j:n} = A_{:,j:n}\tilde{P},\ P_{:,j:n} = P_{:,j:n}\tilde{P}$
  $Q^j,\ R^j = QR(A_{j:m,j:j+\tilde{k}-1})$
  $A_{j:m,j+\tilde{k}:n} = (Q^j)^T A_{j:m,j+\tilde{k}:n}$
  if $\tilde{k} \leq b$ **then**
    break
  end if
  Update S
end for
Output: $Q_{:,1:k} = \prod Q^j,\ R = triu(A_{1:k,1:n}),\ P,\ k = j + \tilde{k}$

4.3. Truncated QR factorization in mixed precision. The truncated QR factorization in mixed precision is illustrated in Algorithm 4.3. Essentially, assuming p precisions are to be used, the algorithm proceeds in p iterations where, at iteration i, the current trailing submatrix A^i is factorized in precision u^i using either Algorithm 4.1 or Algorithm 4.2 and, then, the resulting trailing submatrix A^{i+1} is cast into precision u^{i+1}.

5. Numerical experiments. In this section we present an experimental evaluation of the theoretical results and algorithms presented in the previous sections. This evaluation is twofold. First, in subsection 5.1 we validate the correcteness of the error analysis in section 3 using up to three different precisions, that is, double (fp64), single (fp32) and BFloat16 (bf16). Second, in subsection 5.2, we assess the performance of the mixed-precision variants compared with their high-precision counterpart; for these experiments, we will use precisions for which support is available in commodity CPUs and optimized LAPACK and BLAS libraries, namely double and

Algorithm 4.3 Truncated RRQR factorization in mixed precision (MPTRRQR).

1: **Input:** $A \in \mathbb{R}^{m\times n},\ \varepsilon, \left[u^1, ..., u^p\right]$
2: **Let:** $A^1 = A,\ u^{p+1} = \infty$
3: **for** $i = 1, \ldots, p$ **do**
4: Set working precision to u^i
5: $Q^i,\ R^i,\ P^i,\ k^i = \text{TQR*P}(A^i,\ \varepsilon \left\|A\right\|_F, \varepsilon \left\|A\right\|_F / u^{i+1})$
6: **if** $i < p$, $A^{i+1} = \text{cast}(A^i_{k^i+1:,k^i+1:},\ u^{i+1})$
7: **end for**

single precision. All these codes, along with instructions on how to reproduce our experimental results, are available in a public Git repository[1].

Experiments were conducted on the following problems:

- **randsvd** [16]: random matrices with geometrically distributed singular values between 1 and 10^{-16};
- **gravity** [14]: Discretization of a 1-D model problem in gravity surveying;
- **heat** [14]: inverse heat equation;
- **kahan** [20]: this is a triangular matrix with columns of decreasing norms; therefore, the pivoted QR factorization does not perform any computations. To avoid this trivial behavior, we scaled the columns of this matrix in such a way that column j is multiplied by $(1-\tau)^{n-j}$ with $\tau = 10^{-10}$ and n being the number of columns in the matrix;
- **phillips** [14]: Phillips' famous test problem;

for all problems, only square matrices were generated of varying sizes.

5.1. Theory validation with a Julia prototype. In this section we present an experimental analysis aiming at validating the theoretical analysis of section 3. For this purpose, we have implemented a prototype using the Julia languague which provides half precision, namely, the BFloat16 arithmetic through the BFloat16s[2] package. Because the only purpose of this prototype is to validate the theoretical results, it does not actually implement Algorithm 4.1 and 4.2 but, instead, proceeds through the following steps:

1. Fully factorizes the input matrix A using Householder QR with Businger-Golub pivoting (i.e., Algorithm 2.1);
2. explicitly permutes the input matrix $\tilde{A} = AP$ using the permutation resulting from the previous step;
3. computes the k^i based on criteria in equations (4.2) and (4.3) using, instead of A, the R factor resulting from step 1;
4. factorizes $\tilde{A}$ using Algorithm 4.3 with standard, unpivoted QR on line 5.

Note that this procedure is consistent with the fact that our error analysis does not make any assumptions on whether and how the pivoting is done, as explained in section 4.

Table 1 and Table 2 show experimental results for the randsvd and phillips matrices, respectively, of size 2048. In the first column we report the value of the chosen ε threshold, in the second, the measured error when the factorization is computed entirely in double precision, in the third the error bound computed using equation (4.1) assuming three precisions are used, in the fourth the actual error and in the fifth, sixth

[1] https://gitlab.com/mpqr/mpqr
[2] https://github.com/JuliaMath/BFloat16s.jl

and seventh, respectively, the number of transformations computed in fp64, fp32, and bfloat16; the last column shows the truncation point which corresponds to the sum of the previous three columns and which we have reported for convenience. Note that the truncation happens on the same step both in full double precision and mixed precision. The following conclusions can be drawn by these results. First, the error bound slightly exceeds but closely tracks the ε threshold; this is perfectly expected because in our error analysis some constants were ignored. Second, the actual error never exceeds the bound, which validates our rounding error analysis. Third, as the ε threshold grows, an increasingly large amount of factorization steps are computed using lower-precision arithmetics; clearly, as ε exceeds $u^s\sqrt{n}$, fp64 is not needed anymore and all computations are done in fp32 and bf16. All these observations are confirmed by experiments conducted on the other matrices of our test set which we omit for the sake of space.

	fp64	fp64/fp32/bf16				
ε	Error	Bound	Error	fp64	fp32	bf16	Trunc.
1.0e-14	9.80e-15	3.42e-14	1.04e-14	1183	597	117	1897
1.0e-12	9.72e-13	2.40e-12	1.04e-12	928	609	99	1636
1.0e-10	9.79e-11	2.27e-10	1.02e-10	669	617	88	1374
1.0e-08	9.75e-09	2.18e-08	1.03e-08	403	629	79	1111
1.0e-06	9.78e-07	2.17e-06	1.02e-06	124	650	75	849
1.0e-05	9.75e-06	1.81e-05	1.02e-05	0	644	72	716
1.0e-04	9.73e-05	1.53e-04	1.01e-04	0	513	70	585
1.0e-03	9.74e-04	1.51e-03	1.02e-03	0	378	69	447
1.0e-02	9.75e-03	1.53e-02	1.02e-02	0	242	67	309

TABLE 1
Error and use of precisions for the randsvd matrix of size 2048.

	fp64	fp64/fp32/bf16				
ε	Error	Bound	Error	fp64	fp32	bf16	Trunc.
1.0e-14	1.20e-15	2.44e-13	8.36e-14	1849	198	1	2048
1.0e-12	1.20e-15	2.02e-12	6.50e-13	774	1271	3	2048
1.0e-10	4.07e-11	2.76e-10	5.54e-11	140	1848	58	2046
1.0e-08	9.92e-09	2.75e-08	1.13e-08	24	1242	482	1748
1.0e-06	9.93e-07	2.89e-06	1.05e-06	6	284	151	441
1.0e-05	9.83e-06	2.59e-05	1.02e-05	0	116	71	187
1.0e-04	9.55e-05	2.14e-04	1.05e-04	0	50	25	75
1.0e-03	9.12e-04	2.33e-03	1.11e-03	0	21	9	30
1.0e-02	7.62e-03	2.20e-02	1.19e-02	0	10	4	14

TABLE 2
Error and use of precisions for the phillips matrix of size 2048.

In Figure 1 we report results of an image compression experiment for an image of size 1600×1057. In this figure we compare the original image (*top-left*) with images that are compressed using a truncated QR factorization in full fp32 (*top-right*), fp32/bf16 (*bottom-left*) and full bf16 (*bottom-right*) and then reconstraucted

in fp32. Here, the truncation threshold ε was set to 0.04 and thus only fp32 and bf16 arithmetics were used; the truncation happened on column 191 in all the three reconstructed images. It can clearly be seen that using bf16 only it is not possible to achieve a satisfactory compression. Both the other two approaches, instead, achieve a satisfactory result although, in the mixed-precision case, only 12 out of 190 columns are computed and stored in fp32 and the rest in bf16 which might result in considerable time savings especially on modern GPUs where bf16 computations are much faster than fp32 ones.

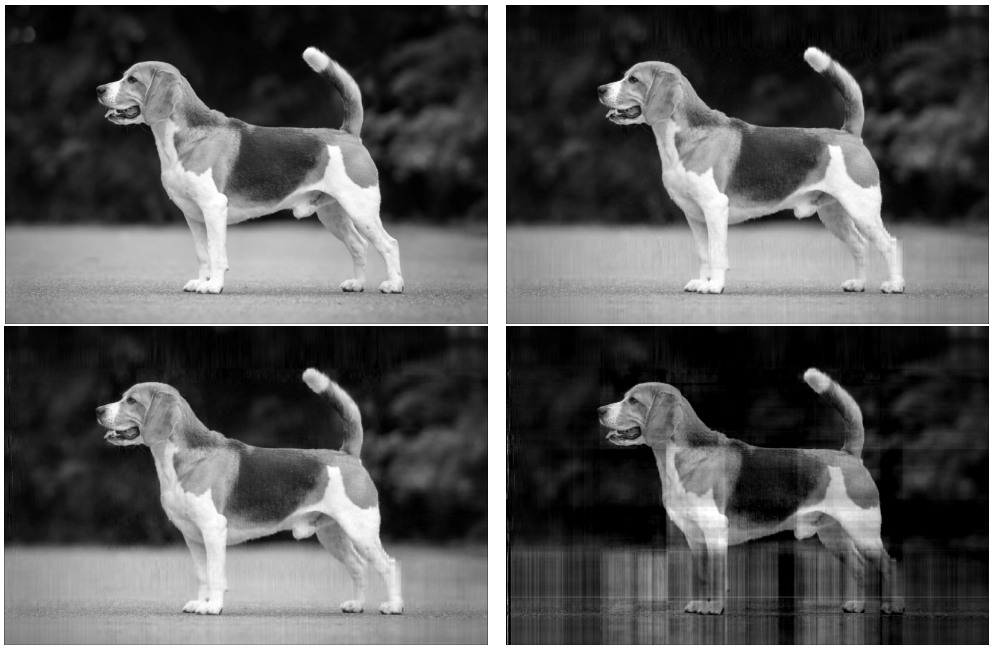

FIG. 1. *Image compression. Original image (top-left); reconstructed image after compression in full fp32 (top-right), in fp32/bf16 (bottom-left) and full bf16 (bottom-right). ε was set to 0.04 and the truncation happened at column 190 but in the mixed-precision case only 12 transformations were computed in fp32 and the rest in bf16. Image size is* 1057×1600.

5.2. Performance analysis with optimized code. In this section we evaluate the performance improvement brought by the use of the mixed-precision algorithm proposed in subsection 4.3 using both Businger-Golub and randomized pivoting compared with the full, high-precision corresponding variants. For these experiments we have implemented Algorithm 4.1, Algorithm 4.2 and Algorithm 4.3 in the Fortran language. The code for Algorithm 4.1 is a straightforward adaptation of the DGEQP3/SGEQP3 and DLAQPS/SLAQPS routines in LAPACK. Algorithm 4.2, instead, is implemented with an entirely new code. Both codes heavily rely on some advanced LAPACK and BLAS routines for which no low-precision (e.g., fp16 or bf16) implementation is currently available neither for CPUs or GPUs. For this reason, our performance analysis is limited to variants that employ fp64 and fp32.

Our experiments were conducted on an AMD Zen3 EPYC 7763 processor, using the BLAS and LAPACK routines in the Intel MKL 2020 package and the GNU gfortran 11.2 Fortran compiler. All experiments are sequential; although parallelism could be straightforwardly used within BLAS operations, we have chosen not to do

so because this does not bring any specific insight on the behavior of the proposed methods.

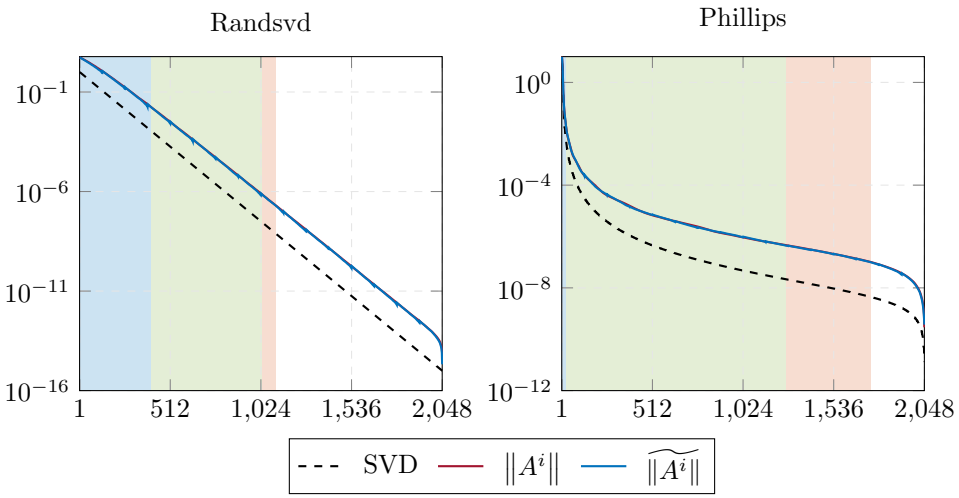

FIG. 2. *The black curves show the spectrum of the randsvd and phillips (left and right, respectively) matrices of size 2048. The red and blue curves show the trailing submatrix norm and its estimate as computed in the factorization with Businger-Golub and randomized pivoting, respectively; the two curves are indistinguishable at low resolution and therefore only the blue one is visible. The vertical bands show number of transformations computed in fp64, fp32 and bf16, respectively, for* $\varepsilon = 10^{-8}$ *and correspond to the values reported in Tables 1 and 2.*

As a further validation of the numerical behavior of our methods, in Figure 2, we report the spectrum (in black) of the randsvd and phillips matrices of size 2048 along with the two criteria that are used within the factorization to decide when to switch precision and when to halt the process. Namely, in red we report the norm of the trailing submatrix as computed within the factorization with Businger-Golub pivoting and, in blue, its estimate computed within the factorization with randomized pivoting, as described in the end of section 4.2. These two curves are almost superposed, which means that the estimate is very accurate; this is actually the case for all of our test problems. Upon a closer inspection, our experiment reveal that the value computed in the factorization with randomized pivoting slightly underestimates the norm of the trailing submatrix which translates into the fact that the precision switch and the truncation happen, on average, slightly earlier than in the case where Businger-Golub pivoting is used. When only two precisions (double and single) are used, on all our test matrices, the median underestimation of the precision switch and the rank is below 1% and in the worst case less than 4%. Despite this fact, the final low-rank approximation accuracy obtained by the factorization with randomized pivoting was always consistent with the error bound and essentially the same as that obtained in the case where Businger-Golub pivoting is used; as explained in the end of section 4.2, a more conservative criterion can be used in the case of randomized pivoting which can offer better robustness although it would likely lead to an excessively late precision switch and truncation.

Figure 3 shows, for the matrices in our test set, the execution time for double-precision (dark color) and mixed single/double-precision (light color) truncated QR factorization algorithms with Businger-Golub (yellow) and randomized (blue) pivoting

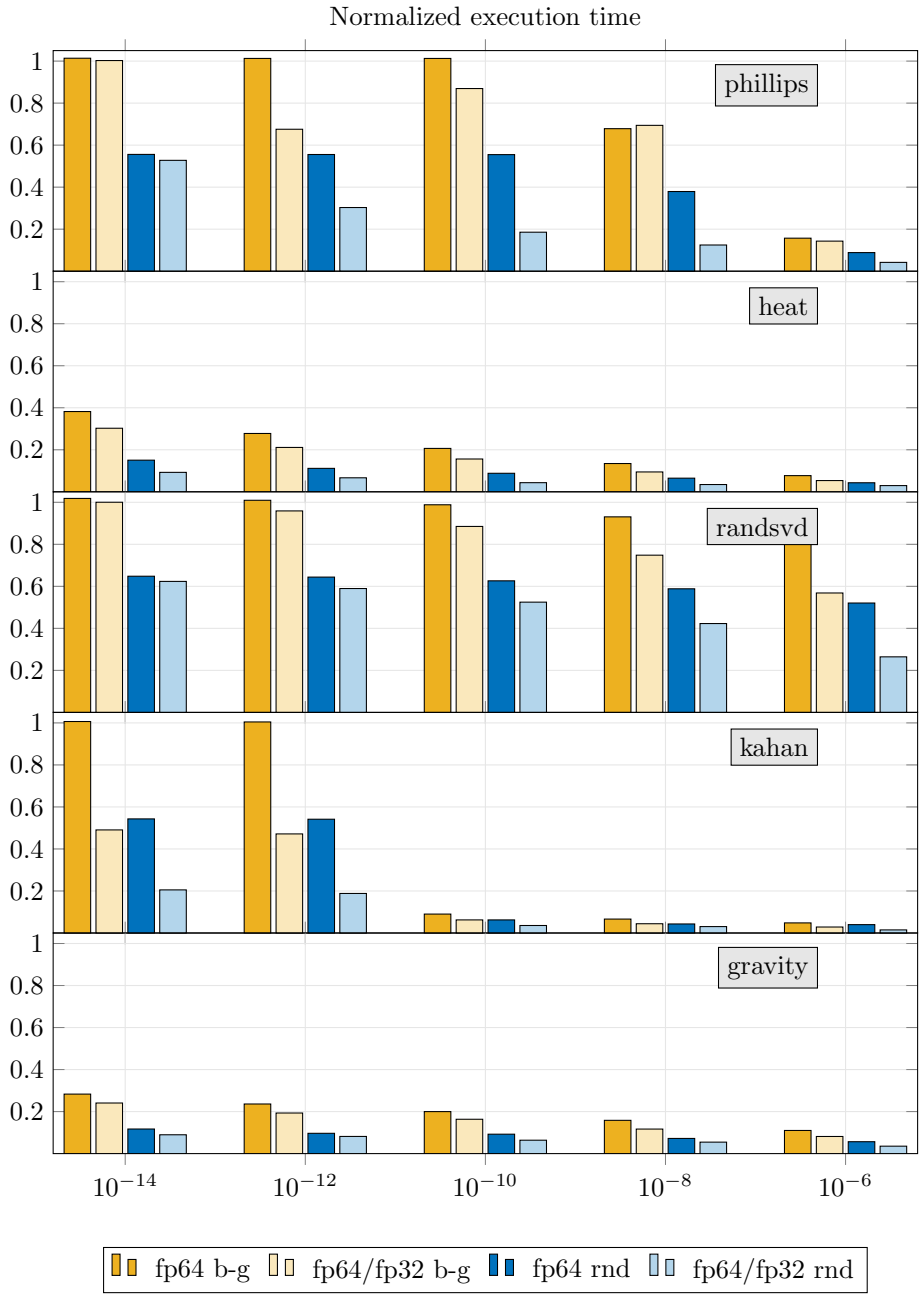

FIG. 3. *Execution times for double-precision (dark color) and mixed single/double-precision (light color) truncated QR factorization algorithms with Businger-Golub (yellow) and randomized (blue) pivoting with respect to the truncation threshold ε. All values are normalized to the execution time of the corresponding full double-precision QR factorization with Businger-Golub pivoting. All the matrices are of size 8192.*

with respect to the truncation threshold ε. For each matrix, the values are normalized to the execution time of the corresponding full double-precision QR factorization with

Businger-Golub pivoting.

The behavior of the proposed algorithms varies considerably across the test problems. Nevertheless, some conclusions can be drawn.

Obviously, the benefit of using the mixed-precision algorithms heavily depends on the spectrum of the problem and the distribution of its singular values which utlimately determines the ratio of operations that are done in double and single precision. As the truncation threshold increases, this ratio normally evolves favorably making the potential benefit of mixed precision higher. This is clearly visible on the randsvd matrix. Although this trend also applies to the other matrices, the execution time does not always evolve correspondigly. This is due to the fact that the ratio of double and single-precision operations alone does not entirely describe performance but other factors must be taken into account. One important factor is the arithmetic intensity of operations. Despite the fact that a fixed panel size is chosen for all algorithms, as explained in subsection 2.2, a panel reduction may be interrupted if the norm of some columns drops beyond a prescribed value which eventually reduces the granularity of operations and, consequently, their speed. This happens in a hardly predictable way and may adversely affect the speed of computation in either double or single precision. This behavior is clearly visible on the phillips matrix comparing the results obtained on the Businger-Golub case with $\varepsilon = 10^{-12}$ and $\varepsilon = 10^{-8}$: in the second case, the single to double-precision operations ratio is more favorable but a large number of restarts happens during the single-precision computations. As a result, the mixed-precision algorithm is slightly slower than the double-precision one, whereas, in the first case, it is 33% faster. Note that in the variant with randomized pivoting, restarts happen in the pivoted factorization of the sample matrix; however, this operation only accounts for a small fraction of the overall operational complexity and, therefore, there restarts have a limited impact. Note that, on standard CPUs, single-precision computations are expected to be twice as fast as double-precision ones. For CPU-bound operations, this is mainly related to the use of vector units (one vector instruction can do twice as many fp32 operations as fp64 ones) whereas for memory-bound operations this is due to a better use of the memory bandwidth (twice as many fp32 coefficients can be transferred as fp64 in the same time). Nevertheless, this assumption does not take into account other factors related to the use of cache memories. Although we cannot assume that the original matrix fits into cache, at some point of the factorization the trailing submatrix, whose size is smaller and smaller, will; when single precision is used, this will happen at an earlier step of the factorization with respect to the double-precision case. Although we haven't conducted dedicated experiments to validate this effect, we speculate that it can explain the fact that in some cases the mixed-precision algorithm is more than twice as fast as the full double-precision one (for example the randomized algorithm on the phillips matrix at $\varepsilon = 10^{-10}$ or the kahan matrix at $\varepsilon = 10^{-12}$).

6. Conclusions and future work. In this work we have introduced a mixed-precision truncated Householder QR factorization where the arithmetic precision of computations is gradually reduced as the norm of the trailing submatrix decreases. We presented an error analysis that results in an error bound demonstrating that if these changes of precision are appropriately operated, the resulting mixed-precision low-rank representation has the same accuracy as in the case where all computations are done in high precision.

Based on our theoretical findings, we have presented two Householder QR factorization algorithms based on Businger-Golub and randomized pivoting, respectively.

Pivoting is not necessary for the mixed-precision algorithm to work because this simply relies on the assumption that the Frobenius norm of the trailing submatrix decreases, which holds true regardless of pivoting; nevertheless, if pivoting is applied, the trailing submatrix norm decays much faster, which ultimately leads to more compact low-rank representations and more efficient use of low-precision arithmetics.

We have presented a twofold experimental analysis. First we focused on validating the theoretical analysis. We did this using a prototype written in the Julia language where we could use up to three different precisions. The corresponding experiments validate the presented theoretical analysis. Second, we evaluated the performance of the two proposed mixed-precision algorithms using double and single-precision arithmetics. Our experimental results on synthetic matrices with different spectra demonstrate that the mixed-precision algorithms can achieve better performance than the full double-precision counterparts, sometimes exceeding a factor of two.

Some opportunities can be identified for pushing the presented ideas further. First, the performance of the mixed-precision algorithms must be evaluated using even lower-precision arithmetic such as Float16 or BFloat16 which, on some hardware, achieve much higher performance than single precision; these are supported in hardware on modern CPUs and GPUs but the corresponding LAPACK and BLAS libraries are still lacking, which prevents us from implementing the mixed-precision algorithms. Second, we must investigate whether and how our approach can be used with other pivoting strategies such as tournament pivoting which might be better suited to parallel implementations. Finally, we would like to study scaling algorithms to prevent issues related to overflow and underflow in low-precision computations.

7. Acknowledgments. This work was supported by the France 2030 NumPEx Exa-SofT (ANR-22-EXNU-0003) project managed by the French National Research Agency (ANR) and the ANR MixHPC project (ANR-23-CE46-0005-01).

Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine [3].

André Pacteau was supported by the National Polytechnic Institute of Toulouse (Toulouse INP) through the EIT program.

[3] https://www.plafrim.fr

587 REFERENCES

588 [1] P. Amestoy, O. Boiteau, A. Buttari, M. Gerest, F. Jézéquel, J.-Y. L'Excellent, and
589 T. Mary, *Mixed precision low-rank approximations and their application to block low-rank*
590 *LU factorization*, IMA Journal of Numerical Analysis, (2022), https://doi.org/10.1093/
591 imanum/drac037, https://doi.org/10.1093/imanum/drac037, https://arxiv.org/abs/https:
592 //hal.archives-ouvertes.fr/hal-03251738v2. drac037.
593 [2] P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary, *On the complexity of the block*
594 *low-rank multifrontal factorization*, SIAM Journal on Scientific Computing, 39 (2017),
595 pp. A1710–A1740, https://doi.org/10.1137/16M1077192, https://arxiv.org/abs/https://
596 hal.archives-ouvertes.fr/hal-01322230v3.
597 [3] P. Blanchard, N. J. Higham, F. Lopez, T. Mary, and S. Pranesh, *Mixed precision*
598 *block fused multiply-add: Error analysis and application to GPU tensor cores*, 42 (2020),
599 pp. C124–C141, https://doi.org/10.1137/19M1289546.
600 [4] P. Blanchard, N. J. Higham, and T. Mary, *A class of fast and accurate summation algo-*
601 *rithms*, 42 (2020), pp. A1541–A1557, https://doi.org/10.1137/19M1257780.
602 [5] P. Businger and G. H. Golub, *Linear least squares solutions by Householder transformations*,
603 Numer. Math., 7 (1965), pp. 269–276, https://doi.org/10.1007/BF01436084, http://dx.doi.
604 org/10.1007/BF01436084.
605 [6] T. F. Chan, *Rank revealing QR factorizations*, Linear Algebra and its Applications, 88-89
606 (1987), pp. 67–82, https://doi.org/https://doi.org/10.1016/0024-3795(87)90103-0, https:
607 //www.sciencedirect.com/science/article/pii/0024379587901030.
608 [7] M. P. Connolly, N. J. Higham, and S. Pranesh, *Randomized low rank matrix approxima-*
609 *tion: Rounding error analysis and a mixed precision algorithm*, Tech. Report 2022.10, The
610 University of Manchester, 2022.
611 [8] J. W. Demmel, L. Grigori, M. Gu, and H. Xiang, *Communication avoiding rank revealing*
612 *QR factorization with column pivoting*, SIAM Journal on Matrix Analysis and Applica-
613 tions, 36 (2015), pp. 55–89, https://doi.org/10.1137/13092157X, https://doi.org/10.1137/
614 13092157X, https://arxiv.org/abs/https://doi.org/10.1137/13092157X.
615 [9] M. Dessole and F. Marcuzzi, *Deviation maximization for rank-revealing QR factorizations*,
616 Numer. Algorithms, 91 (2022), p. 1047–1079, https://doi.org/10.1007/s11075-022-01291-1,
617 https://doi.org/10.1007/s11075-022-01291-1.
618 [10] Z. Drmač and Z. Bujanović, *On the failure of rank-revealing QR factorization software*
619 *– a case study*, ACM Trans. Math. Softw., 35 (2008), https://doi.org/10.1145/1377612.
620 1377616, https://doi.org/10.1145/1377612.1377616.
621 [11] J. A. Duersch and M. Gu, *Randomized QR with column pivoting*, SIAM Journal on Scientific
622 Computing, 39 (2017), pp. C263–C291, https://doi.org/10.1137/15M1044680, https://doi.
623 org/10.1137/15M1044680, https://arxiv.org/abs/https://doi.org/10.1137/15M1044680.
624 [12] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psy-
625 chometrika, 1 (1936), pp. 211–218, https://doi.org/10.1007/BF02288367, https://doi.org/
626 10.1007/BF02288367.
627 [13] P. Ghysels, X. S. Li, F.-H. Rouet, S. Williams, and A. Napov, *An efficient multicore*
628 *implementation of a novel HSS-structured multifrontal solver using randomized sam-*
629 *pling*, SIAM Journal on Scientific Computing, 38 (2016), pp. S358–S384, https://doi.org/
630 10.1137/15M1010117, https://doi.org/10.1137/15M1010117, https://arxiv.org/abs/https:
631 //doi.org/10.1137/15M1010117.
632 [14] P. C. Hansen, *Regularization tools version 4.0 for Matlab 7.3*, Numerical Algorithms, 46
633 (2007), pp. 189–194, https://doi.org/10.1007/s11075-007-9136-9, https://doi.org/10.1007/
634 s11075-007-9136-9.
635 [15] N. Higham and T. Mary, *A new approach to probabilistic rounding error analysis*, SIAM
636 Journal on Scientific Computing, 41 (2019), pp. A2815–A2835, https://doi.org/10.1137/
637 18M1226312.
638 [16] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and
639 Applied Mathematics, USA, 2nd ed., 2002.
640 [17] N. J. Higham, *Numerical stability of algorithms at extreme scale and low precisions*, in In-
641 ternational Congress of Mathematicians, EMS Press, Dec. 2023, p. 5098–5117, https:
642 //doi.org/10.4171/ICM2022/74.
643 [18] N. J. Higham and T. Mary, *A new approach to probabilistic rounding error analysis*, 41
644 (2019), pp. A2815–A2835, https://doi.org/10.1137/18M1226312.
645 [19] A. S. Householder, *Unitary triangularization of a nonsymmetric matrix*, J. ACM, 5 (1958),
646 pp. 339–342, https://doi.org/10.1145/320941.320947, http://doi.acm.org/10.1145/320941.
647 320947.

[20] W. Kahan, *Numerical linear algebra*, Canadian Mathematical Bulletin, 9 (1966), p. 757–801, https://doi.org/10.4153/CMB-1966-083-2.
[21] R. B. Lehoucq, *The computation of elementary unitary matrices*, ACM Trans. Math. Softw., 22 (1996), p. 393–400, https://doi.org/10.1145/235815.235817, https://doi.org/10.1145/235815.235817.
[22] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 20167–20172.
[23] P.-G. Martinsson, G. Quintana Ortí, N. Heavner, and R. van de Geijn, *Householder QR factorization with randomization for column pivoting (HQRRP)*, SIAM Journal on Scientific Computing, 39 (2017), pp. C96–C115, https://doi.org/10.1137/16M1081270, https://doi.org/10.1137/16M1081270, https://arxiv.org/abs/https://doi.org/10.1137/16M1081270.
[24] G. Pichon, E. Darve, M. Faverge, P. Ramet, and J. Roman, *Sparse Supernodal Solver Using Block Low-Rank Compression*, in 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), May 2017, pp. 1138–1147, https://doi.org/10.1109/IPDPSW.2017.86.
[25] R. Schreiber and C. Van Loan, *A storage-efficient WY representation for products of Householder transformations*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 52–57.
[26] J. Xiao, M. Gu, and J. Langou, *Fast parallel randomized QR with column pivoting algorithms for reliable low-rank matrix approximations*, 2017 IEEE 24th International Conference on High Performance Computing (HiPC), (2017), pp. 233–242, https://api.semanticscholar.org/CorpusID:3444436.