Mixed Precision Low Rank Approximations and their Application to Block Low Rank LU Factorization

Patrick Amestoy, Olivier Boiteau, Alfredo Buttari, Matthieu Gerest, Fabienne Jézéquel, Jean-Yves L'Excellent, Théo Mary

MIXED PRECISION LOW RANK APPROXIMATIONS AND THEIR APPLICATION TO BLOCK LOW RANK MATRIX FACTORIZATION*

PATRICK AMESTOY†, OLIVIER BOITEAU‡, ALFREDO BUTTARI§, MATTHIEU GEREST‡, FABIENNE JÉZÉQUEL¶, JEAN-YVES L'EXCELLENT†, AND THEO MARY‖

Abstract. We introduce a novel approach to exploit mixed precision arithmetic for low rank approximations. Our approach is based on the observation that singular vectors associated with small singular values can be stored in lower precisions while preserving high accuracy overall. We provide an explicit criterion to determine which level of precision is needed for each singular vector. We apply this approach to block low-rank (BLR) matrices, most of whose off-diagonal blocks have low rank. We propose a new BLR LU factorization algorithm that exploits the mixed precision representation of the blocks. We carry out the rounding error analysis of this algorithm and prove that the use of mixed precision arithmetic does not compromise the numerical stability of BLR LU factorization. Moreover our analysis determines which level of precision is needed for each floating-point operation (flop), and therefore guides us towards an implementation that is both robust and efficient. We evaluate the potential of this new algorithm on a range of matrices coming from real-life problems in industrial and academic applications. We show that a large fraction of the entries in the LU factors and flops to perform the BLR LU factorization can be safely switched to lower precisions, leading to significant reductions of the storage and flop costs, of up to a factor three using fp64, fp32, and bfloat16 arithmetics.

Key words. numerical linear algebra, rounding error analysis, floating-point arithmetic, mixed precision algorithms, multiprecision algorithms, block low-rank matrices, data sparse matrices, LU factorization, linear systems, low-rank approximations, singular value decomposition

AMS subject classifications. 65G50, 65F05, 65F08, 65F10, 65F50

1. Introduction.

The emergence of low precision arithmetics on modern hardware, such as the half precision floating-point formats fp16 and bfloat16, has generated a renewed interest in mixed precision algorithms. These algorithms combine low precisions with higher ones in order to improve the performance of the computations (speed, memory and energy consumption) without compromising their accuracy or stability. New mixed precision variants of numerical linear algebra algorithms have been recently proposed, for example, for matrix multiplication [11, 12, 27, 29, 4], iterative refinement [13, 14, 21, 6], LU [11, 26] and QR [34, 35] matrix factorizations, Krylov solvers [19, 3], least squares problems [15], and many others [1].

In this article, we investigate the use of mixed precision arithmetic in a new class of algorithms: low-rank approximations, in particular those obtained with a singular value decomposition or a rank-revealing decomposition (such as QR with column pivoting). Let $A \in \mathbb{R}^{m\times n}$ and let $X\Sigma Y^T = \sum_{i=1}^{\min(m,n)} x_i\sigma_i y_i^T$ be its singular value decomposition (SVD)[1]. Given a target accuracy ε, a low-rank approximation T of A satisfying $\|T - A\| \leq \varepsilon\|A\|$ can be built from the truncated SVD $T = \sum_{i=1}^{r} x_i\sigma_i y_i^T$, where r is the rank of T.

---

*Version of June 8, 2021.

†Mumps Technologies, ENS Lyon, 46 Allée d'Italie, F-69007 Lyon, France (patrick.amestoy@mumps-tech.com; jean-yves.l.excellent@mumps-tech.com)

‡EDF R&D (olivier.boiteau@edf.fr; matthieu.gerest@edf.fr)

§CNRS, IRIT, 2 Rue Charles Camichel, F-31071 Toulouse, France (alfredo.buttari@irit.fr)

¶Sorbonne Université, CNRS, LIP6, and Université Panthéon-Assas, Paris, F-75005, France (fabienne.jezequel@lip6.fr)

‖Sorbonne Université, CNRS, LIP6, Paris, F-75005, France (theo.mary@lip6.fr)

[1]We write $X\Sigma Y^T$ rather than the usual $U\Sigma V^T$ to avoid any confusion between the left singular vectors, the unit roundoffs u_i (see (2.4)), and the upper factor from LU factorization.

Our starting idea for this work is to ask what precision should be used to store T and to operate on it. In the literature, the truncated SVD (or indeed any other form of low-rank decomposition) is stored in the lowest possible precision with unit roundoff safely smaller than ε; for example, if $\varepsilon = 10^{-12}$ and we have access to the floating-point arithmetics defined by the IEEE standard, existing algorithms would use double precision (for which the unit roundoff is $u_d \approx 1 \times 10^{-16}$), because the next lower precision, single precision, has a unit roundoff $u_s \approx 6 \times 10^{-8}$ much larger than the prescribed ε.

However, in this article we explain why and how we can actually exploit much lower precisions, with almost no loss of accuracy. We show that singular vectors associated with sufficiently small singular values can be stored at precisions with unit roundoff larger than $\varepsilon\|A\|$ while maintaining an overall approximation accuracy of order $\varepsilon\|A\|$. For example, with $\varepsilon = 10^{-12}$, any singular vector x_i whose associated singular value σ_i is smaller than $\varepsilon\|A\|/u_s \approx 2 \times 10^{-5}\|A\|$ can be stored in single precision. Indeed, the single precision vector $\widehat{x}_i$ satisfies $\|\widehat{x}_i - x_i\| \leq u_s$, but the overall error introduced by replacing x_i by $\widehat{x}_i$ is bounded by $\|(\widehat{x}_i - x_i)\sigma_i y_i^T\| \leq (\varepsilon\|A\|/u_s)u_s = \varepsilon\|A\|$. As can be seen from this example, the reason we can afford to convert some singular vectors to lower precision is because the error introduced by this conversion is demagnified by the singular value; hence the error may be safely bounded if σ_i is small enough.

In the following we formalize this intuition with an error analysis considering an arbitrary number of precisions. Moreover our analysis applies to any low-rank decomposition of the form $T = XY^T$ where X has orthonormal columns. Indeed the mixed-precision approach presented here is general and can be used for several other low-rank approximations, in particular rank-revealing QR decompositions.

Clearly, the potential of the proposed approach depends on whether the singular values of the matrices to be approximated decay rapidly. In the second part of this article, we apply this approach to an important class of matrices: data sparse, rank-structured matrices, whose off-diagonal blocks have low numerical rank [10]. We focus in particular on the block low-rank (BLR) format [5, 7], although the approach is also applicable to hierarchical formats. Our numerical experiments demonstrate that the proposed mixed precision low-rank representation presents a very high potential in this context: a large fraction of both the entries needed to represent BLR matrices and the floating-point operations (flops) needed to compute their LU factorization can be switched to lower precisions.

The rest of this article is organized as follows. In section 2, we describe the proposed mixed precision low-rank representation and show that the loss of accuracy introduced by the use of lower precisions can be rigorously bounded. We then apply this representation to BLR matrices in section 3. In section 4, we analyze how to compute the LU factorization of a BLR matrix using mixed precision arithmetic. We present numerical experiments on a range of real-life matrices in section 5, before concluding in section 6.

Throughout the article, the unsubscripted norm $\|\cdot\|$ denotes the Frobenius norm

$$\|A\| = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2} = \left(\sum_{i=1}^{\min(m,n)} \sigma_i^2\right)^{1/2}. \tag{1.1}$$

We also define $\gamma_k^{(\ell)} = ku_\ell/(1-ku_\ell)$ for any $k > 0$ and for any unit roundoff u_ℓ.

2. Mixed precision low-rank approximations. Let $A \in \mathbb{R}^{m\times n}$ and let T be a low-rank approximation of A satisfying

$$\|A - T\| \leq \varepsilon\beta, \tag{2.1}$$

where $\varepsilon > 0$ is the target accuracy and where β is a scaling parameter chosen by the user: a natural choice is $\beta = \|A\|$, which leads to an accuracy of ε relative to $\|A\|$, but other choices are possible and will be explored in the next sections on BLR matrices.

Hereinafter, we refer to the precision that T is stored in as the working precision, and we assume that its unit roundoff u_1 is safely smaller than ε, that is, $u_1 \ll \varepsilon$.

Given the SVD $A = \sum_{i=1}^{n} x_i\sigma_i y_i^T$, it is well known that the approximation of A of lowest rank is given by the truncated SVD

$$T = X\Sigma Y^T = \sum_{i=1}^{r} x_i\sigma_i y_i^T, \quad X \in \mathbb{R}^{m\times r},\ Y \in \mathbb{R}^{n\times r}, \tag{2.2}$$

where the rank r is the smallest integer such that (2.1) is satisfied. Neglecting any noise associated with the working precision, r is given by

$$r = \min\left\{k\colon \left(\sum_{i=k+1}^{n} \sigma_i^2\right)^{1/2} \leq \varepsilon\beta\right\}. \tag{2.3}$$

The goal of this section is to prove that depending on the singular values of T, we can use lower precisions than the working precision (with unit roundoff larger than ε) and still preserve an overall approximation error of order ε. We first carry out our analysis for the SVD, but also provide at the end of this section its extension to other types of low-rank approximation methods, such as rank-revealing QR.

In this article, we only consider the use of arithmetics with less significant bits. We assume that the use of lower precision arithmetics with less exponent bits than the working precision does not lead to any overflow or underflow. To ensure that this assumption is satisfied, in our experiments, we focus on the use of bfloat16 arithmetic (which has the same range as fp32), rather than fp16 (which has a much narrower range).

Let us assume that p different floating-point arithmetics are available (including the working precision u_1), and that their unit roundoffs satisfy

$$u_1 \ll \varepsilon < u_2 < \ldots < u_p. \tag{2.4}$$

Let us consider a partition of matrix T into p groups

$$T = X\Sigma Y^T = \begin{bmatrix} X_1 \ldots X_p \end{bmatrix} \begin{bmatrix} \Sigma_1 & & \\ & \ddots & \\ & & \Sigma_p \end{bmatrix} \begin{bmatrix} Y_1 \ldots Y_p \end{bmatrix}^T, \tag{2.5}$$

where $T_k = X_k\Sigma_k Y_k^T$ is formed of a subset of the singular values and vectors of T. We denote by r_k the rank of T_k, which is the number of singular values assigned to group k.

We now analyze the effect of converting T_k to precision u_k. We assume that only the singular vectors X_k and Y_k are converted, whereas the singular values Σ_k are kept in precision u_1. This is because the storage for Σ_k is negligible compared with that of

X_k and Y_k. We however note that adapting the analysis to the case where Σ_k is also converted to precision u_k is straightforward and only slightly increases the constants in the error bounds. We write $\widehat{X}_k$ and $\widehat{Y}_k$ the converted vectors, and $\widehat{T}_k = \widehat{X}_k \Sigma_k \widehat{Y}_k$ (note that since T_1 is already in precision u_1, $\widehat{T}_1 = T_1$). The following lemma bounds $\|\widehat{T}_k - T_k\|$ for $k \geq 2$.

LEMMA 2.1. *Let $T_k = X_k \Sigma_k Y_k^T$ where X_k and Y_k have orthonormal columns, and let $\widehat{T}_k = \widehat{X}_k \Sigma_k \widehat{Y}_k$ be obtained by converting X_k and Y_k to precision u_k. Then*

$$\|\widehat{T}_k - T_k\| \leq (2 + \sqrt{r_k} u_k) u_k \|\Sigma_k\|. \tag{2.6}$$

Proof. The converted $\widehat{X}_k$ and $\widehat{Y}_k$ satisfy, for $k \geq 2$,

$$\widehat{X}_k = X_k + E_k, \quad |E_k| \leq u_k |X_k|, \tag{2.7}$$

$$\widehat{Y}_k = Y_k + F_k, \quad |F_k| \leq u_k |Y_k|. \tag{2.8}$$

Therefore we have

$$\|T_k - \widehat{T}_k\| \leq \|E_k \Sigma_k Y_k^T\| + \|X_k \Sigma_k F_k^T\| + \|E_k \Sigma_k F_k^T\|. \tag{2.9}$$

For the first term, we observe that

$$\|E_k \Sigma_k Y_k^T\|^2 = \|E_k \Sigma_k\|^2 \tag{2.10}$$

$$= \sum_j \sigma_j^2 \sum_i e_{ij}^2 \tag{2.11}$$

$$\leq \sum_j \sigma_j^2 \sum_i u_k^2 x_{ij}^2 \tag{2.12}$$

$$= u_k^2 \sum_j \sigma_j^2 = u_k^2 \|\Sigma_k\|^2, \tag{2.13}$$

where we have used the fact that the columns of X_k have a norm of 1. Therefore

$$\|E_k \Sigma_k Y_k^T\| \leq u_k \|\Sigma_k\|. \tag{2.14}$$

Similarly, we also have

$$\|X_k \Sigma_k F_k^T\| \leq u_k \|\Sigma_k\|. \tag{2.15}$$

Finally, for the third term, we have $\|E_k \Sigma_k F_k^T\| \leq \|E_k \Sigma_k\| \|F_k\|$ and so

$$\|E_k \Sigma_k F_k^T\| \leq \sqrt{r_k} u_k^2 \|\Sigma_k\|. \tag{2.16}$$

Reinjecting (2.14), (2.15), and (2.16) into (2.9) yields the result. ∎

Lemma 2.1 shows that converting T_k to precision u_k introduces an error of order $u_k \|\Sigma_k\|$, which is thus proportional to the size of the singular values in Σ_k. This fundamental observation is at the foundation of the mixed precision representation that we propose in this article. Indeed, Lemma 2.1 suggests that we can preserve an overall accuracy of order $\varepsilon\beta$ by partitioning the singular values in such a way that $\|\Sigma_k\| \approx \varepsilon\beta/u_k$.

In order to build such a partitioning where the size of the groups stored in lower precision is as large as possible, it is easy to see that we should start by including the smallest singular values in the last group first, until its norm exceeds $\varepsilon\beta/u_p$; at this

point, we can start building group $p-1$ with the remaining singular values, and so on. Therefore, the Σ_k are formed of consecutive singular values:

$$\Sigma_k = \mathrm{diag}(\sigma_i), \quad i = i_k \colon i_{k+1} - 1, \tag{2.17}$$

where the indices i_k and i_{k+1} define which singular values are part of Σ_k, and can be easily computed by the recursive formula:

$$i_k = \min \left\{ i \colon \left(\sum_{j=i}^{i_{k+1}-1} \sigma_j^2 \right)^{1/2} \leq \varepsilon\beta/u_k \right\}, \quad k \in [2:p], \tag{2.18}$$

starting with $i_{p+1} = r+1$ and ending with $i_1 = 1$. We thus end up with a partitioning of the SVD as defined by (2.17)–(2.18). We note that this partitioning is similar to the Method 3 proposed in [30]. Our analysis justifies the use of this partitioning and gives a precise rule to define the p groups depending on the singular values and on the precisions.

This partitioning guarantees that $\|\Sigma_k\| \leq \varepsilon\beta/u_k$ for all $k \geq 2$ and so, by Lemma 2.1, converting T_k to precision u_k introduces an error bounded by

$$\|T_k - \widehat{T}_k\| \leq (2 + \sqrt{r_k}u_k)\varepsilon\beta. \tag{2.19}$$

By combining (2.19) over $k = 2 \colon p$, we readily obtain a bound on the overall error introduced by converting each T_k to precision u_k.

THEOREM 2.2. *Let T be a low-rank approximation of A satisfying $\|A - T\| \leq \varepsilon\beta$. If T is partitioned into p groups $T_k = X_k\Sigma_k Y_k^T$ as defined by (2.17)–(2.18), and the X_k and Y_k are converted to precision u_k, the resulting matrix $\widehat{T} = \sum_{k=1}^p \widehat{T}_k = \sum_{k=1}^p \widehat{X}_k\Sigma_k\widehat{Y}_k$ satisfies*

$$\|A - \widehat{T}\| \leq \left(2p - 1 + \sum_{k=2}^{p} \sqrt{r_k}u_k\right)\varepsilon\beta. \tag{2.20}$$

Proof. The triangle inequality $\|A - \widehat{T}\| \leq \|A - T\| + \sum_{k=2}^p \|T_k - \widehat{T}_k\|$ together with (2.19) readily yields the result. ∎

Theorem 2.2 shows that the mixed precision low-rank matrix $\widehat{T}$ approximates A with an accuracy of order ε. To first order, the constant in this error bound is $2p-1$, instead of 1 for the uniform precision matrix T: the introduction of lower precisions therefore only increases the overall error by a very modest quantity. Moreover, we note that by means of a more sophisticated proof that avoids the use of the triangle inequality, this constant can be reduced to $2\sqrt{p-1}$. However, we will not use such proofs for the sake of readability, and because the precise value of the constants in the error bounds is unimportant, as long as they are not too large.

From a practical point of view, the SVD is expensive to compute and for this reason other low-rank decompositions are often preferred, such as rank-revealing QR decompositions. Theorem 2.2 can be easily extended to more general low-rank matrix decompositions. For example, with a decomposition of the form XBY^T, where X and Y have orthonormal columns (the difference with the SVD being that B is not diagonal), bound (2.20) holds with a slightly larger constant; one example of this form is the UTV decomposition [18]. Our analysis can also be adapted to decompositions of the form XY^T, where X has orthonormal columns (but Y does not). This second form is particularly of interest because it applies to rank-revealing QR decompositions. We state the analogue to Lemma 2.1 for XY^T decompositions below.

LEMMA 2.3. *Let $T_k = X_k Y_k^T$ where X_k has orthonormal columns, and let $\widehat{T}_k = \widehat{X}_k \widehat{Y}_k$ be obtained by converting X_k and Y_k to precision u_k. Then*

$$\|T_k - \widehat{T}_k\| \leq (2 + \sqrt{r_k} u_k) u_k \|Y_k\|. \tag{2.21}$$

Thus, the error introduced by converting group k now depends on $\|Y_k\|$, and this means that the $X\Sigma Y^T$ partitioning (2.17)–(2.18) should be adapted by replacing $\|\Sigma_k\|$ by $\|Y_k\|$. Then, it is easy to show that Theorem 2.2 still holds. In the rest of this article, we will focus on low-rank decompositions of the form XY^T, computed by means of a truncated QR factorization with column pivoting.

An important question is under what condition the low-rank compression is beneficial, that is, when does the low-rank approximation T require less storage than the original matrix $A \in \mathbb{R}^{m \times n}$. In the standard uniform precision case, $T = XY^T$ can be represented with $r(m+n)$ entries, and so the condition is

$$r(m+n) \leq mn. \tag{2.22}$$

With a mixed precision representation, this condition changes due to the fact that entries belonging to groups $k \geq 2$ are stored in lower precision. The condition becomes

$$(m+n) \sum_{k=1}^{p} c_k r_k \leq mn, \tag{2.23}$$

where c_k ponderates the cost of storing a floating-point number in precision u_k instead of u_1. For example, if we use three precisions, fp64, fp32, and bfloat16, (2.23) takes the form $(m+n)(r_1 + 0.5 r_2 + 0.25 r_3) \leq mn$. Interestingly, the difference between conditions (2.22) and (2.23) means that a matrix that is not "low-rank enough" in uniform precision can become so when using mixed precision arithmetic.

Crucially, the size r_k of each group depends on the singular values. Indeed, group k must satisfy $\|\Sigma_k\| \leq \varepsilon\beta/u_k$, so if A possesses many small singular values, the low precision groups will be much larger than the first group stored in the working precision u_1. Conversely, if the singular values decay slowly, most of them must be kept in the first group. Therefore, the potential gains achieved by the proposed mixed precision representation completely depend on the spectrum of the matrix. In the rest of this article, we focus on an important class of matrices that exhibit off-diagonal blocks with rapidly decaying singular values, and therefore present a high potential for mixed precision arithmetic.

3. Mixed precision BLR compression.

Data sparse matrices are rank-structured matrices most of whose off-diagonal blocks have low numerical rank. In this section, we show how this property can be exploited to represent these matrices in mixed precision. We focus on a specific class of data sparse matrices, called the block low-rank (BLR) format [5, 7, 9]. The approach described here could easily be extended to other formats, such as hierarchical [20] or multilevel [8] ones.

3.1. Background on BLR matrices.

A block low-rank (BLR) representation T of a dense square matrix $A \in \mathbb{R}^{n \times n}$ has the block $q \times q$ form

$$T = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1q} \\ T_{21} & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ T_{q1} & \cdots & \cdots & T_{qq} \end{bmatrix}, \tag{3.1}$$

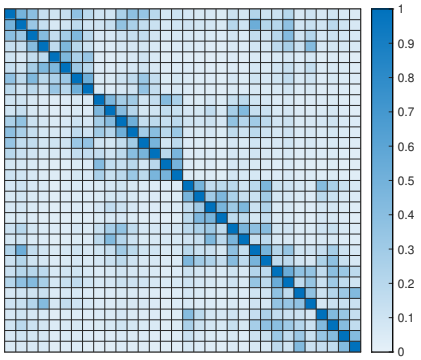


Fig. 3.1: Ranks of the blocks T_{ij} of the BLR approximation of matrix P64 for $\varepsilon = 10^{-10}$. The ranks are given as a percentage of the block size.

where $T_{ii} = A_{ii}$ (the diagonal blocks are not compressed), and where the off-diagonal blocks A_{ij} of size $b \times b$ are approximated by matrices T_{ij} satisfying

$$\|A_{ij} - T_{ij}\| \leq \varepsilon \beta_{ij}, \tag{3.2}$$

where $\beta_{ij} > 0$. The T_{ij} matrices, of rank r_{ij}, are given by

$$T_{ij} = \begin{cases} X_{ij} Y_{ij}^T, & i > j, \\ Y_{ij} X_{ij}^T, & i < j, \end{cases} \tag{3.3}$$

where X_{ij} and Y_{ij} are $b \times r_{ij}$ matrices, and where X_{ij} has orthornormal columns. Even though, in general, each block can be of different dimensions, we assume for simplicity that they are all of the same dimensions $b \times b$, and so $n = qb$. Representation (3.3) guarantees the so-called outer orthonormality property [24, 28], which allows for efficient intermediate recompressions during the factorization of a BLR matrix.

Importantly, the β_{ij} parameters in (3.2) are used to distinguish two types of BLR compression, local and global, depending on whether block T_{ij} approximates A_{ij} with error ε relative to the local norm $\beta_{ij} = \|A_{ij}\|$ or relative to the global norm $\beta_{ij} = \|A\|$. In their error analysis of the BLR factorization, Higham and Mary [24] show that global compression achieves a better tradeoff between compression and accuracy, and is therefore to be preferred. Throughout this article, we will thus use global compression, for which we have the global error bound

$$\|A - T\| \leq q\varepsilon \|A\|. \tag{3.4}$$

Matrices amenable to BLR compression arise in a variety of applications, such as partial differential equations, integral equations, and covariance matrices. In this article, we focus on a range of dense matrices that are Schur complements of sparse matrices. Our test matrices are listed in Table 5.1, and we will illustrate some aspects of our analysis with the matrix P64. Figure 3.1 plots a heatmap of the numerical ranks of the blocks of this matrix.

3.2. Error analysis of mixed precision BLR compression. We now seek to combine BLR compression with the mixed precision representation proposed in section 2. The natural approach is to simply use this mixed precision representation

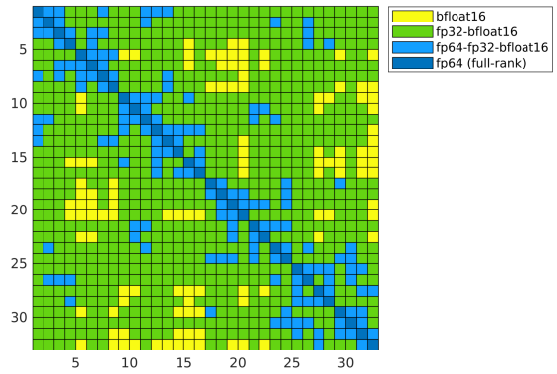


Fig. 3.2: Precisions used for representing each block of a mixed precision BLR matrix (matrix P64, $\varepsilon = 10^{-10}$).

on every low-rank off-diagonal block of the BLR matrix, leaving the full-rank blocks in the working precision u_1. Then, it is easy to show that (2.20) becomes

$$\|A_{ij} - \widehat{T}_{ij}\| \leq \big(2p - 1 + \sum_{k=2}^{p} \sqrt{r_{ij}^{(k)}}\, u_k\big)\varepsilon\beta_{ij}, \tag{3.5}$$

where $r_{ij}^{(k)}$ is the rank of the matrix $\widehat{T}_{ij}^{(k)} = \widehat{X}_{ij}^{(k)}(\widehat{Y}_{ij}^{(k)})^T$, that is, the number of columns of X_{ij} and Y_{ij} stored in precision u_k. The next result bounds the error introduced by mixed precision BLR compression.

THEOREM 3.1 (mixed precision BLR compression). *Let T be a BLR approximation of A defined by (3.1)–(3.2) with $\beta_{ij} = \|A\|$ (global compression). If the off-diagonal blocks T_{ij} are represented with the mixed precision representation $\widehat{T}_{ij}$ described in section 2, the resulting BLR matrix $\widehat{T}$ satisfies*

$$\|A - \widehat{T}\| \leq q\left(2p - 1 + \sum_{k=2}^{p} c_k u_k\right)\varepsilon\|A\|, \tag{3.6}$$

with $c_k = \max_{i,j} \sqrt{r_{ij}^{(k)}}$.

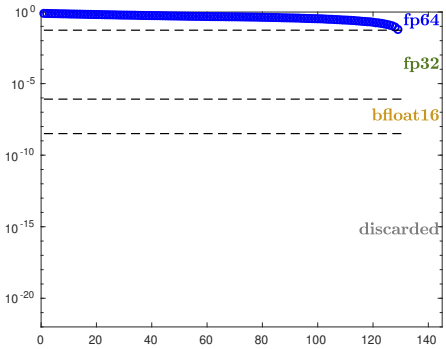
Proof. Using

$$\|A - \widehat{T}\|^2 = \sum_{i=1}^{q}\sum_{j=1}^{q} \|A_{ij} - \widehat{T}_{ij}\|^2 \tag{3.7}$$

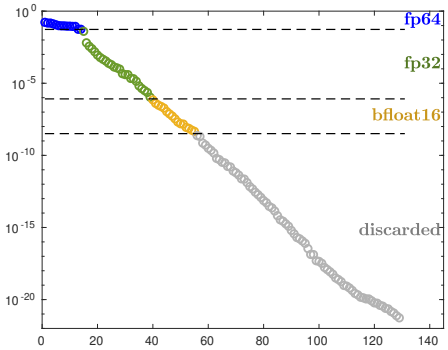
and (3.5), we readily obtain the result. ∎

Compared with the uniform precision bound (3.4), the mixed precision bound (3.6) is thus larger by a modest factor of about $2p - 1$. Theorem 3.1 therefore shows that we can exploit mixed precision arithmetic in the BLR compression while preserving an accuracy of order ε.

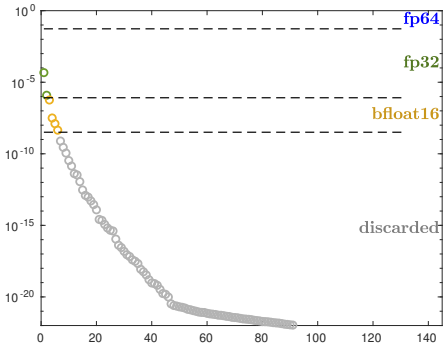
3.3. Types of mixed precision blocks. Figure 3.2 shows an example of a mixed precision BLR matrix, plotting for each of its blocks the precisions that are effectively used to represent it. With $\varepsilon = 10^{-10}$ and with three available precisions



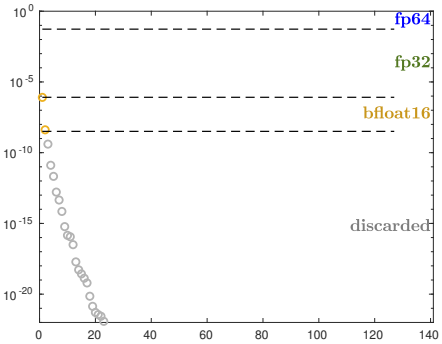
(a) Diagonal block in position (15,15).



(b) Near field block in position (15,16).



(c) Mid field block in position (15,22).



(d) Far field block in position (15,27).

Fig. 3.3: Distribution of the singular values of four blocks of different type in Figure 3.2. The dashed lines indicate the thresholds $\varepsilon\beta/u_s$, $\varepsilon\beta/u_h$, and $\varepsilon\beta$, with $\beta = \|A\|$ and where u_s and u_h denote the unit roundoffs of the fp32 and bfloat16 arithmetics, respectively.

(fp64, fp32, and bfloat16), we can distinguish four types of blocks. The singular values of a representative example of each type are plotted in Figure 3.3.

First, the full-rank blocks (type 1, dark blue blocks in Figure 3.2, consisting of only the diagonal blocks here) are stored in the working precision (fp64). An example of diagonal block is given in Figure 3.3a, showing that its singular values decay too slowly to benefit from the use of a mixed precision representation. However, there is only a small number of such blocks: the majority of the blocks therefore benefits from the use of lower precisions.

Interestingly, the number of low-rank blocks that effectively use all three precisions is quite small (type 2, light blue blocks, example given by Figure 3.3b). Most blocks actually do not need to store any of their entries in fp64. This is a consequence of using global compression: if $\beta_{ij} = \|A\| \gg \|A_{ij}\|$, fp64 is not needed to represent A_{ij}. In other words, blocks of sufficiently small norm can be stored entirely in lower precision. Among these blocks, we can further distinguish two types: those that are represented in mixed precision using both fp32 and bfloat16 (type 3, green blocks, example given by Figure 3.3c) and those that are stored entirely in bfloat16 (type 4, yellow blocks, example given by Figure 3.3d). The type-4 blocks are those whose norm is smaller than $\varepsilon\|A\|/u_h$, where $u_h = 2^{-8}$ is the unit roundoff of bfloat16. Note

that a fifth type of block could arise, those whose norm is smaller than $\varepsilon\|A\|$: these blocks can simply be dropped, that is, replaced by zero (but no such blocks appear in the example of Figure 3.2).

The observation that blocks of small norm can be stored entirely in lower precision is important. It justifies why the simpler approach proposed by [2, 17] can already achieve significant gains. Their approach consists in storing each block in uniform precision, but possibly differing from one block to another. For example, for the matrix in Figure 3.2, all type-2 blocks (light blue) would need to be stored entirely in double precision, but type-3 blocks (green) could be stored entirely in single precision. Moreover, our error analysis provides the criterion that should be used to choose each block's precision: blocks A_{ij} such that $\|A_{ij}\| \leq \varepsilon\|A\|/u_i$ can be stored in precision u_i.

4. Mixed precision BLR LU factorization. We now present how to exploit the mixed precision BLR representation described previously in order to accelerate the LU factorization of BLR matrices. There exists several BLR LU factorization algorithms, and here we focus on the so-called UCF (a.k.a. UCFS or FCSU) variant described in Algorithm 4.1. This variant has been successfully used in the literature, for example in the MUMPS [9] and PaStiX [31] sparse direct solvers, and its rounding error analysis in the uniform precision case has been carried out in [24, sect. 4.2]. We note that BLR LU factorization can and usually does incorporate numerical pivoting for stability, but we describe Algorithm 4.1 without pivoting for simplicity.

Algorithm 4.1 BLR LU factorization.

{**Input**: a $q \times q$ block matrix A with blocks A_{ij} of size $b \times b$.}
{**Output**: its BLR LDU factors LDU.}
for $k = 1$ **to** q **do**
  UPDATE:
    $R_{kk} = A_{kk} - \sum_{j=1}^{k-1} L_{kj} D_{jj} U_{jk}$.
    for $i = k+1$ **to** q **do**
      $R_{ik} = A_{ik} - \sum_{j=1}^{k-1} L_{ij} D_{jj} U_{jk}$ and $R_{ki} = A_{ki} - \sum_{j=1}^{k-1} L_{kj} D_{jj} U_{ji}$.
    end for
  COMPRESS:
    for $i = k+1$ **to** q **do**
      Compute low-rank approximations $T_{ik} \approx R_{ik}$ and $T_{ki} \approx R_{ki}$.
    end for
  FACTOR:
    Compute the LU factorization $L_{kk} D_{kk} U_{kk} = R_{kk}$.
    for $i = k+1$ **to** q **do**
      Solve $L_{ik} D_{kk} U_{kk} = T_{ik}$ for L_{ik} and $L_{kk} D_{kk} U_{ki} = T_{ki}$ for U_{ki}.
    end for
end for

The error analysis presented in this section has a double purpose. First, it proves that the numerical stability of the uniform precision BLR LU factorization (proven by Higham and Mary [24]) is preserved in mixed precision arithmetic. Second, for each operation required by Algorithm 4.1, it determines which level of precision is needed to maintain the overall error of order ε. Our analysis therefore guides us towards an implementation of mixed precision BLR LU that is both robust and efficient.

One technical difficulty of this analysis is the handling of the scaling factors hidden inside the U factor. To make these details more apparent, we analyze instead the LDU

factorization, where L and U are unitriangular matrices (with ones on the diagonal). For the sake of simplicity, we however do not take into account any rounding errors incurred by applying D or its inverse. Moreover, we will not always keep track of lower order error terms; we use the notations $\approx$ and $\lesssim$ to indicate when these terms (of order at most $u_p\varepsilon$) have been dropped.

We first analyze each kernel separately. Algorithm 4.1 requires computing products of the form $L_{ij}D_{jj}U_{jk}$ (on line 7), where L_{ij} and U_{jk} may be either uniform precision full-rank blocks or mixed precision low-rank blocks (analysis of sections 4.1 and 4.2). We also analyze in section 4.3 the solution of a triangular system $L_{kk}D_{kk}U_{ki} = T_{ki}$ (needed on line 16), where the right-hand side T_{ki} is a mixed precision low-rank matrix. Finally, we combine the analysis of these kernels to obtain a backward error bound on the mixed precision BLR LU factorization in section 4.4.

4.1. Low-rank matrix times full-rank matrix. Let us begin with the computation of a product $P = BC$, where C is a full-rank matrix and $B = XY^T$ is a mixed precision low-rank matrix partitioned into p groups $B_\ell = X_\ell Y_\ell^T$ satisfying $\|B_\ell\| \leq \varepsilon\beta/u_\ell$ for $\ell > 1$, and where the output P is needed under full-rank form.

In which precision should we compute $P = BC$? The natural approach is to compute each product $P_\ell = B_\ell C$ in precision u_ℓ. Then, using [24, Lemma 3.2], the computed $\widehat{P}_\ell$ satisfies

$$\widehat{P}_\ell = B_\ell C + \Delta P_\ell, \quad \|\Delta P_\ell\| \leq \gamma_{c_\ell}^{(\ell)} \|B_\ell\|\|C\| \tag{4.1}$$

with $c_\ell = b + r_\ell^{3/2}$. For $\ell > 1$, we thus obtain

$$\widehat{P}_\ell = B_\ell C + \Delta P_\ell, \quad \|\Delta P_\ell\| \lesssim c_\ell \varepsilon\beta\|C\|, \tag{4.2}$$

since $\gamma_{c_\ell}^{(\ell)}/u_\ell = c_\ell(1 + \gamma_{c_\ell}^{(\ell)}) \approx c_\ell$. This shows that the partial product P_ℓ associated with the part of B stored in precision u_ℓ can itself be computed in precision u_ℓ, since the introduced error remains of order ε.

The next question is what precision should be used to combine the partial results into $P = \sum_{\ell=1}^p P_\ell$. Since for $\ell > 1$ $\|P_\ell\| \leq \varepsilon/u_\ell\beta\|C\|$, it is easy to see that $P_i + P_j$ must be computed in precision $\min(u_i, u_j) = u_{\min(i,j)}$. Knowing this, in order to maximize the performance gains associated with the use of lower precisions, the best approach is to compute $\sum_{\ell=1}^p \widehat{P}_\ell$ in reverse order. The approach to compute P suggested by our analysis is summarized in Algorithm 4.2.

Algorithm 4.2 Mixed precision low-rank matrix times full-rank matrix.

1: {**Input**: a mixed precision low-rank matrix B and a full-rank matrix C.}
2: {**Output**: $P = BC$.}
3: Initialize P to zero.
4: **for** $\ell = p$ **to** 1 **do**
5: Compute $P_\ell = B_\ell C$ in precision u_ℓ.
6: Update $P \leftarrow P + P_\ell$ in precision u_ℓ.
7: **end for**

With this algorithm, each component of $\widehat{P}_\ell$ is involved in exactly $\min(\ell, p-1)$ additions, one in each precision $u_1, \ldots, u_{\min(\ell,p-1)}$. Therefore the computed $\widehat{P}$ satisfies:

$$\widehat{P} = \sum_{\ell=1}^{p} \widehat{P}_\ell \circ (J + \Theta_\ell), \quad |\Theta_\ell| \lesssim u_\ell, \tag{4.3}$$

where J is the matrix of ones, $\circ$ denotes the Hadamard product, and the inequality $|\Theta_\ell| \lesssim u_\ell$ holds componentwise. Overall, we obtain

$$\widehat{P} = \sum_{\ell=1}^{p} (B_\ell C + \Delta P_\ell) \circ (J + \Theta_\ell) \tag{4.4}$$

$$= BC + \sum_{\ell=1}^{p} B_\ell C \circ \Theta_\ell + \Delta P_\ell + \Delta P_\ell \circ \Theta_\ell \tag{4.5}$$

$$= BC + \Delta P, \quad \|\Delta P\| \lesssim \big((c_1+1)u_1\|B^{(1)}\| + \sum_{\ell=2}^{p}(c_\ell+1)\varepsilon\beta\big)\|C\| \tag{4.6}$$

$$= BC + \Delta P, \quad \|\Delta P\| \lesssim (pb + r^{3/2} + p)\max(u_1\|B^{(1)}\|, \varepsilon\beta)\|C\|. \tag{4.7}$$

We summarize this analysis in the next theorem.

THEOREM 4.1. *Let $B = \sum_{\ell=1}^{p} B_\ell \in \mathbb{R}^{b\times b}$ be a mixed precision low-rank matrix of rank r such that $\|B_\ell\| \le \varepsilon\beta/u_\ell$ for $\ell > 1$, and let $C \in \mathbb{R}^{b\times b}$. If $P = BC$ is computed as described by Algorithm 4.2, then the computed $\widehat{P}$ satisfies*

$$\|\widehat{P} - BC\| \lesssim c\max(u_1\|B\|, \varepsilon\beta)\|C\|, \tag{4.8}$$

with $c = pb + r^{3/2} + p$.

Theorem 4.1 shows that we can perform many of the flops in Algorithm 4.2 in lower precisions and still maintain an error of order ε. We now prove similar results for the other kernels.

4.2. Low-rank matrix times low-rank matrix. Next we analyze the product $P = BDC$ of two mixed precision low-rank matrices $B = X_B Y_B^T$ and $C = Y_C X_C^T$, where we also incorporate a diagonal scaling matrix D, which will be useful for the LU factorization analysis of section 4.4.

The product P, which is needed in full-rank form, can be computed in the following three steps:

1. Compute the inner product $M = (Y_B)^T D Y_C$.
2. Compute the middle product $W = X_B M$ (or $W = M X_C^T$).
3. Compute the outer product $P = W X_C^T$ (or $P = X_B W$).

A trivial extension of [24, Lem. 3.2] to incorporate D shows that if P is computed in uniform precision u, the computed $\widehat{P}$ satisfies

$$\widehat{P} = BDC + \Delta P, \quad \|\Delta P\| \lesssim cu\|B\|\|D\|\|C\|, \tag{4.9}$$

where $c = b + 2r^{3/2}$.

We now consider the case where B and C are partitioned into p groups $B_\ell = X_{B\ell} Y_{B\ell}^T$ and $C_m = Y_{Cm} X_{Cm}^T$, stored in precision u_ℓ and u_m, respectively. We assume that matrices B and C satisfy $\|B_\ell D\| \le \varepsilon\beta_B/u_\ell$ and $\|DC_m\| \le \varepsilon\beta_C/u_m$ for $\ell, m > 1$. We analyze each of the three steps separately.

4.2.1. Inner product $M = Y_B^T D Y_C$. Let us first analyze the computation of the inner product M. Assume $M_{\ell m} = Y_{B\ell}^T D Y_{Cm}$ is computed in a given precision denoted as $u_{\ell m}^M$. The computed $\widehat{M}_{\ell m}$ satisfies $\widehat{M}_{\ell m} = M_{\ell m} + \Delta M_{\ell m}$, with

$$|\Delta M_{\ell m}| \lesssim b u_{\ell m}^M |Y_{B\ell}|^T |D| |Y_{Cm}|. \tag{4.10}$$

By taking norms, we obtain

$$\|\Delta M_{\ell m}\| \lesssim b u_{\ell m}^M \min(\alpha_1, \alpha_2, \alpha_3), \tag{4.11}$$

where

$$\alpha_1 = \|B_\ell D\|\|C_m\|, \qquad \alpha_1 \le \varepsilon\beta_B\|C_m\|/u_\ell \ \text{ if } \ell > 1, \tag{4.12}$$
$$\alpha_2 = \|B_\ell\|\|DC_m\|, \qquad \alpha_2 \le \varepsilon\|B_\ell\|\beta_C/u_m \ \text{ if } m > 1, \tag{4.13}$$
$$\alpha_3 = \|B_\ell D\|\|D^{-1}\|\|DC_m\|, \quad \alpha_3 \le \varepsilon^2\beta_B\beta_C\|D^{-1}\|/(u_\ell u_m) \ \text{ if } \ell, m > 1. \tag{4.14}$$

From this we can deduce the optimal choices of precisions $u_{\ell m}^M$ that still guarantee an error of order ε.

- If $\ell = m = 1$, in general we must take $u_{11}^M = u_1$ since $\|B_1\|$ and $\|C_1\|$ are not bounded in terms of ε. We obtain

$$\|\Delta M_{11}\| \lesssim b u_1\|B_1\|\|D\|\|C_1\|. \tag{4.15}$$

- If $\ell = 1$ and $m > 1$, $\alpha_2 \le \varepsilon/u_m\|B\|\beta_C$, and so taking $u_{1m}^M = u_m$ yields an error of order ε:

$$\|\Delta M_{1m}\| \lesssim b\varepsilon\|B_1\|\beta_C. \tag{4.16}$$

  Similarly, we can take $u_{\ell 1}^M = u_\ell$ and obtain

$$\|\Delta M_{\ell 1}\| \lesssim b\varepsilon\beta_B\|C_1\|. \tag{4.17}$$

- If $\ell, m > 1$, we can safely take $u_{\ell m}^M = \max(u_\ell, u_m) = u_{\max(\ell,m)}$. Indeed, if $\ell \ge m$ we can use (4.12) and if $\ell < m$ we can use (4.13), and so, in any case, we have

$$\|\Delta M_{\ell m}\| \lesssim b\varepsilon \max(\beta_B\|C_1\|, \|B_1\|\beta_C). \tag{4.18}$$

Combining (4.15), (4.16), (4.17), and (4.18), we obtain for $\ell, m \ge 1$

$$\|\Delta M_{\ell m}\| \lesssim b \max(\varepsilon\beta_B\|C_1\|, \varepsilon\|B_1\|\beta_C, u_1\|B_1\|\|D\|\|C_1\|). \tag{4.19}$$

In summary, for any value of ℓ and m, we can compute the product between the part of B stored in precision u_ℓ and the part of C stored in precision u_m in the *lower* of the two precisions. This is a crucial observation that allows us to maximize the use of lower precision.

Moreover, in some cases we may actually take $u_{\ell m}^M > \max(u_\ell, u_m)$ because of (4.14). To see why, let us take an example where $\|D^{-1}\|$, β_B, β_C, and $\|A\|$ are all approximately equal to 1. In this case, for $\ell, m > 1$, (4.11) reduces to

$$\|\Delta M_{\ell m}\| \lesssim b u_{\ell m}^M \varepsilon^2/(u_\ell u_m) \tag{4.20}$$

and so the requirement to obtain an error of order ε is

$$b u_{\ell m}^M \varepsilon \le u_\ell u_m, \tag{4.21}$$

which thus depends not only on u_ℓ and u_m, but also on ε. If ε is small enough, (4.21) may be satisfied even for $u_{\ell m}^M > \max(u_\ell, u_m)$. For example, assume we have three precisions $u_1 = u_d = 2^{-53}$, $u_2 = u_s = 2^{-24}$, and $u_3 = u_h = 2^{-8}$. Then, ignoring the constant b in (4.21):

- The condition $u_{22}^M \varepsilon \le u_2^2$ is satisfied for $u_{22}^M = u_3$ if $\varepsilon \le u_s^2/u_h \approx 9 \times 10^{-13}$. Thus, if ε is small enough, M_{22} need only be computed in half precision.

- The condition $u_{23}^M \varepsilon \le u_2 u_3$ is satisfied for $u_{23}^M = 1$ if $\varepsilon \le u_s u_h \approx 2 \times 10^{-10}$. The same holds for u_{32}^M. Thus, if ε is small enough, the computation of M_{23} and M_{32} may be skipped altogether.
- Finally, the condition $u_{33}^M \varepsilon \le u_3^2$ is satisfied for $u_{33}^M = 1$ if $\varepsilon \le u_h^2 \approx 2 \times 10^{-5}$. Again, the computation of M_{33} may be skipped in this case.

Going back to the general case, the precise requirement on $u_{\ell m}^M$ depends on u_ℓ, u_m, ε, β_B, β_C, and $\|D^{-1}\|$. For global compression ($\beta_B = \beta_C = \|A\|$), we obtain

$$\|\Delta M_{\ell m}\| \lesssim b u_{\ell m}^M \varepsilon^2 \|A\|^2 \|D^{-1}\| / (u_\ell u_m). \tag{4.22}$$

4.2.2. Middle product $W = X_B M$ (or $W = M Y_C^T$). We analyze the product $W = X_B M$, the case of $W = M Y_C^T$ being analogous. Let $W_m = \sum_{\ell=1}^p X_{B\ell} M_{\ell m}$, for $m = 1\colon p$. Assume the product $W_m^{(\ell)} = X_{B\ell} M_{\ell m}$ is computed in precision $u_{\ell m}^W$, then the computed $\widehat{W}_m^{(\ell)}$ satisfies

$$\widehat{W}_m^{(\ell)} = X_{B\ell} \widehat{M}_{\ell m} + \Delta W_m^{(\ell)}, \tag{4.23}$$

$$\|\Delta W_m^{(\ell)}\| \lesssim r_\ell u_{\ell m}^W \|X_{B\ell}\| \|\widehat{M}_{\ell m}\| \lesssim r_\ell^{3/2} u_{\ell m}^W \|B_\ell D C_m\|. \tag{4.24}$$

This bound on $\|\Delta W_m^{(\ell)}\|$ is similar to the bound (4.11) on $\|\Delta M_{\ell m}\|$, and we should therefore set $u_{\ell m}^W = u_{\ell m}^M$. Then, similarly to Algorithm 4.2, the partial results $W_m^{(\ell)}$ should be summed in reverse order and in increasing precision, since $W_m^{(\ell)} + W_m^{(\ell+1)}$ must be computed in precision $u_{\ell m}^W$. Overall, with $u_{\ell m}^W = u_{\ell m}^M = \max(u_\ell, u_m)$, the computed $\widehat{W}_m$ satisfies

$$\widehat{W}_m = \sum_{\ell=1}^p \widehat{W}_m^{(\ell)} \circ (J + \Theta_\ell), \quad |\Theta_\ell| \lesssim u_\ell, \tag{4.25}$$

$$= \sum_{\ell=1}^p (X_{B\ell} M_{\ell m} + X_{B\ell} \Delta M_{\ell m} + \Delta W_m^{(\ell)}) \circ (J + \Theta_\ell), \tag{4.26}$$

$$= \sum_{\ell=1}^p W_m^{(\ell)} + \Delta \widehat{W}_m^{(\ell)} = W_m + \Delta W_m, \tag{4.27}$$

with

$$\|\Delta \widehat{W}_m^{(\ell)}\| \lesssim (b + r_\ell^{3/2} + 1) \max(\varepsilon \beta_B \|C\|, \varepsilon \|B\| \beta_C, u_1 \|B\| \|D\| \|C\|) \tag{4.28}$$

and so

$$\|\Delta W_m\| \lesssim (pb + r^{3/2} + p) \max(\varepsilon \beta_B \|C\|, \varepsilon \|B\| \beta_C, u_1 \|B\| \|D\| \|C\|). \tag{4.29}$$

4.2.3. Outer product $P = W X_C^T$ (or $P = X_B W$). It remains to analyze the final product $P = W X_C^T$ (or $P = X_B W$, which is analogous). Let $P_m = W_m X_{Cm}^T$ be computed in precision u_m^P. Then the computed $\widehat{P}_m$ satisfies

$$\widehat{P}_m = \widehat{W}_m X_{Cm}^T + \Delta P_m, \tag{4.30}$$

$$\|\Delta P_m\| \lesssim r_m u_m^P \|\widehat{W}_m\| \|X_{Cm}\| \le r_m^{3/2} u_m^P \|\widehat{W}_m\| \lesssim r_m^{3/2} u_m^P \sum_{\ell=1}^p \|B_\ell D C_m\|. \tag{4.31}$$

Since $\sum_{\ell=1}^{p} \|B_\ell D C_m\|$ is at least as large as $\|B_1 D C_m\|$, by (4.13) we must take $u_m^P = u_m$. Then, (4.31) becomes

$$\|\Delta P_m\| \lesssim r_m^{3/2} \max(\varepsilon \|B\| \beta_C, u_1 \|B\| \|D\| \|C\|). \tag{4.32}$$

Finally, as previously for W_m, we sum P_m over m in reverse order and in increasing precision, to obtain a computed $\widehat{P}$ satisfying

$$\widehat{P} = \sum_{m=1}^{p} \widehat{P}_m \circ (J + \Theta_m), \quad |\Theta_m| \lesssim u_m, \tag{4.33}$$

$$= \sum_{m=1}^{p} (\widehat{W}_m (X_{Cm})^T + \Delta P_m) \circ (J + \Theta_m), \tag{4.34}$$

$$= \sum_{m=1}^{p} P_m + (\Delta W_m X_{Cm}^T + \Delta P_m) \circ (J + \Theta_m), \tag{4.35}$$

$$= P + \Delta P, \tag{4.36}$$

with

$$\|\Delta P\| \lesssim (p^2 b + (p+1) r^{3/2} + p^2 + p) \max(\varepsilon \beta_B \|C\|, \varepsilon \|B\| \beta_C, u_1 \|B\| \|D\| \|C\|). \tag{4.37}$$

This concludes the analysis of the product P. We summarize the approach suggested by this analysis in Algorithm 4.3, for which the following theorem holds.

Algorithm 4.3 Mixed precision low-rank matrix times mixed precision low-rank matrix.

1: {**Input**: mixed precision low-rank matrices $B = X_B Y_B^T$ and $C = Y_C X_C^T$ and a diagonal matrix D.}
2: {**Output**: $P = BDC$.}
3: Initialize P to zero.
4: **for** $m = p$ **to** 1 **do**
5: Initialize W_m to zero.
6: **for** $\ell = p$ **to** 1 **do**
7: Compute $M_{\ell m} = Y_{B\ell} D Y_{Cm}^T$ in precision $\max(u_\ell, u_m)$.
8: Compute $W_m^{(\ell)} = X_{B\ell} M_{\ell m}$ in precision $\max(u_\ell, u_m)$.
9: Update $W_m \leftarrow W_m + W_m^{(\ell)}$ in precision $\max(u_\ell, u_m)$.
10: **end for**
11: Compute $P_m = W_m X_{Cm}^T$ in precision u_m.
12: Update $P \leftarrow P + P_m$ in precision u_m.
13: **end for**

THEOREM 4.2 (Low-rank times low-rank). *Let $B = \sum_{\ell=1}^{p} B_\ell$ and $C = \sum_{m=1}^{p} C_m$ be two mixed precision low-rank matrices satisfying*

$$\|B_\ell D\| \le \varepsilon \beta_B / u_\ell \quad \text{for } \ell > 1, \tag{4.38}$$

$$\|D C_m\| \le \varepsilon \beta_C / u_m \text{ for } m > 1, \tag{4.39}$$

and D a diagonal matrix, and let $P = BDC$ be computed as described in Algorithm 4.3. Then, the computed $\widehat{P}$ satisfies

$$\widehat{P} = BDC + \Delta P, \quad \|\Delta P\| \lesssim c \max\left(\varepsilon \beta_B \|C\|,\ \varepsilon \|B\| \beta_C,\ u_1 \|B\| \|D\| \|C\|\right), \tag{4.40}$$

with $c = p^2 b + (p+1) r^{3/2} + p^2 + p$.

4.3. Triangular system with low-rank right-hand side. The last kernel that we need to analyze is the solution of a triangular system $LDZ = B$, where $L \in \mathbb{R}^{b\times b}$ is lower triangular, D is diagonal, and the right-hand side $B = YX^T$ is a mixed precision low-rank matrix. We analyze the kernel for a lower triangular matrix L, the upper triangular case ($ZDU = B$) being analogous. For this kernel, the output (the solution Z) is needed under low-rank form.

In the uniform precision case, if the system $LDZ = B$ is solved in uniform precision u, the computed solution $\widehat{Z}$ satisfies [24, Lemma 3.5]

$$L\widehat{D}\widehat{Z} = B + \Delta B, \quad \|\Delta B\| \lesssim bu\|L\|\|D\|\|\widehat{Z}\|. \tag{4.41}$$

Let $B_\ell = Y_\ell X_\ell^T$ be the part of B that is stored in precision u_ℓ, satisfying $\|B_\ell\| = \|Y_\ell\| \le \varepsilon\beta/u_\ell$ for $\ell > 1$. Then, the natural approach to solve $LDZ = B$ in mixed precision is to solve each system $LDV_\ell = Y_\ell$ in precision u_ℓ and to define $Z_\ell = V_\ell X_\ell^T$, which yields the mixed precision low-rank solution $Z = \sum_{\ell=1}^{p} Z_\ell$. However, a traditional normwise analysis based on (4.41) does not provide a satisfactory bound here: if we apply (4.41) to $LDV_\ell = Y_\ell$ and use $\widehat{V}_\ell \approx D^{-1}L^{-1}Y_\ell$, we obtain the bound

$$L\widehat{D}\widehat{V}_\ell = Y_\ell + \Delta Y_\ell, \quad \|\Delta Y_\ell\| \lesssim b\varepsilon\beta\kappa(L)\kappa(D). \tag{4.42}$$

This bound is very weak due to the presence of the normwise condition numbers $\kappa(L)\kappa(D) = \|L\|\|L^{-1}\|\|D\|\|D^{-1}\|$.

A stronger bound can be obtained by using a componentwise analysis:

$$L\widehat{D}\widehat{V}_\ell = Y_\ell + \Delta Y_\ell, \quad |\Delta Y_\ell| \lesssim bu_\ell|L||D||\widehat{V}_\ell|. \tag{4.43}$$

Replacing $\widehat{V}_\ell$ by $D^{-1}L^{-1}(Y_\ell + \Delta Y_\ell)$ in the bound on ΔY_ℓ yields

$$|\Delta Y_\ell| \lesssim bu_\ell|L||D||D^{-1}L^{-1}Y_\ell| \le bu_\ell|L||L^{-1}||Y_\ell|. \tag{4.44}$$

We can now take norms, obtaining for $\ell > 1$

$$\|\Delta Y_\ell\| \lesssim bu_\ell \operatorname{cond}(L, Y_\ell)\|Y_\ell\| \le b\varepsilon\beta \operatorname{cond}(L, Y_\ell) \tag{4.45}$$

where $\operatorname{cond}(L, Y_\ell)$ is the condition number introduced by Skeel [33], [23, Eq. (7.13)]:

$$\operatorname{cond}(L, Y_\ell) = \frac{\||L||L^{-1}||Y_\ell|\|}{\|Y_\ell\|}. \tag{4.46}$$

Multiplying both sides of (4.43) by X_ℓ^T on the right yields

$$L\widehat{D}\widehat{Z}_\ell = B_\ell + \Delta Y_\ell X_\ell^T. \tag{4.47}$$

Summing (4.47) over ℓ, we obtain

$$L\widehat{D}\widehat{Z} = B + \sum_{\ell=1}^{p} \Delta Y_\ell X_\ell^T = B + \Delta B, \tag{4.48}$$

$$\|\Delta B\| \lesssim bu_1\|L\|\|D\|\|\widehat{Z}_1\| + pb\varepsilon\beta \operatorname{cond}(L), \tag{4.49}$$

where $\operatorname{cond}(L) = \||L^{-1}||L|\|$.

The use of intermediate componentwise bounds presents two advantages: first, we obtain a bound with $\operatorname{cond}(L)$, which is potentially much smaller than $\kappa(L)$; second,

we have dropped the matrix D from the term proportional to $\varepsilon\beta$, which shows that this term is invariant under scaling. Importantly, in the case of an LDU factorization with partial pivoting, both L and U are well conditioned, and $\mathrm{cond}(L)$ is in practice a small constant. Therefore, in the context of Algorithm 4.1, the mixed precision triangular solution analyzed here is backward stable. However, for a general system $LDZ = B$, bound (4.49) does not guarantee backward stability, and indeed some examples can be built where the use of mixed precision arithmetic in the solution of the system leads to a large increase of the backward error (we note however that such examples are very hard to find and we were only able to construct one using direct search optimization [22]).

We summarize the proposed approach to solve $LDZ = B$ in Algorithm 4.4 and its error analysis in the following theorem.

Algorithm 4.4 Solution to $LDZ = B$ (triangular system with low-rank RHS).

1: {**Input**: a mixed precision low-rank matrix $B = YX^T$, a lower triangular matrix L, and a diagonal matrix D.}
2: {**Output**: a mixed precision low-rank matrix Z, solution to $LDZ = B$.}
3: **for** $\ell = p$ **to** 1 **do**
4: Solve the triangular system $LDV_\ell = Y_\ell$ in precision u_ℓ.
5: Define $Z_\ell = V_\ell X_\ell^T$ (no computation performed: output is low-rank).
6: **end for**

THEOREM 4.3. *Let $L \in \mathbb{R}^{b\times b}$ be a lower triangular full-rank matrix and let $B = \sum_{\ell=1}^{p} B_\ell$ be a mixed precision low-rank matrix satisfying $\|B_\ell\| \le \beta\varepsilon/u_\ell$. If the system $LDZ = B$ is solved by Algorithm 4.4, the computed solution $\widehat{Z}$ satisfies*

$$L\widehat{D}\widehat{Z} = B + \Delta B, \quad \|\Delta B\| \lesssim pb\varepsilon\beta\,\mathrm{cond}(L) + bu_1\|L\|\|D\|\|\widehat{Z}\|. \tag{4.50}$$

4.4. Putting everything together: error analysis of mixed precision BLR LU factorization. Now that we have analyzed all the kernels of Algorithm 4.1, we are ready to prove the backward stability of the BLR LU factorization in mixed precision arithmetic. We define

$$\lambda_1 = \max_{k=1:\,q} \max\left(\|L_{kk}^{-1}\|, \|U_{kk}^{-1}\|, \mathrm{cond}(L_{kk}), \mathrm{cond}(U_{kk})\right). \tag{4.51}$$

If partial pivoting is performed, λ_1 is almost always small in practice and of order a constant [23, Chapter 8], even though in theory it can only be bounded by $2^b - 1$ [23, Lemma 8.6]. We also define

$$\lambda_2 = \max_{i\ge j} \max\left(\|L_{ij}\|, \|U_{ji}\|\right). \tag{4.52}$$

If partial pivoting is performed, $\lambda_2 \le b$.

Let us bound the error incurred in the computation of some (i,k) block of the L factor, the U factor analysis being similar. For $i < k$, L_{ik} is obtained by solving

$$L_{ik}D_{kk}U_{kk} = T_{ik}, \tag{4.53}$$

where T_{ik} is the compressed form of

$$R_{ik} = A_{ik} - \sum_{j=1}^{k-1} \widehat{L}_{ij}D_{jj}\widehat{U}_{jk}, \tag{4.54}$$

where $\widehat{L}_{ij}$ and $\widehat{U}_{jk}$ are the LU factors computed at the previous steps, and are represented as mixed precision low-rank matrices, and the product $\widehat{L}_{ij}D_{jj}\widehat{U}_{jk}$ is computed with Algorithm 4.3. Note that if one of $\widehat{L}_{ij}$ or $\widehat{U}_{jk}$ is a full-rank matrix, the analysis is similar and relies on Theorem 4.1; if both are full-rank, the computation is done in uniform precision $u_1 \ll \varepsilon$ and introduces an error term $bu_1\|\widehat{L}_{ij}\|\|D_{jj}\|\|\widehat{U}_{jk}\|$. The difficulty is that $\widehat{L}_{ij}$ and $\widehat{U}_{jk}$ are not directly the result of a compression, and so we cannot directly control the norms of $\widehat{L}_{ij}^{(\ell)}$ and $\widehat{U}_{jk}^{(m)}$, the parts of $\widehat{L}_{ij}$ and $\widehat{U}_{jk}$ stored in precision u_ℓ and u_m, respectively. Instead, they are given by

$$\widehat{L}_{ij}^{(\ell)} \approx T_{ij}^{(\ell)}\widehat{U}_{jj}^{-1}D_{jj}^{-1}, \tag{4.55}$$

$$\widehat{U}_{jk}^{(m)} \approx D_{jj}^{-1}\widehat{L}_{jj}^{-1}T_{jk}^{(m)}, \tag{4.56}$$

where T_{ij} and T_{jk} have been compressed such that

$$\|T_{ij}^{(\ell)}\| \leq \varepsilon\beta_{ij}/u_\ell \quad \text{for } \ell > 1, \tag{4.57}$$

$$\|T_{jk}^{(m)}\| \leq \varepsilon\beta_{jk}/u_m \text{ for } m > 1. \tag{4.58}$$

Therefore, the norms of $\widehat{L}_{ij}^{(\ell)}$ and $\widehat{U}_{jk}^{(m)}$ depend on β_{ij} and β_{jk}, respectively, but also on the scaling factors in D_{jj}. However, one of the two D_{jj}^{-1} in (4.55)–(4.56) is cancelled by the D_{jj} in (4.54) and $\|\widehat{L}_{jj}^{-1}\|$ and $\|\widehat{U}_{jj}^{-1}\|$ are both bounded by λ_1. As a result, we can rewrite the product $R_{ik,j} = \widehat{L}_{ij}D_{jj}\widehat{U}_{jk}$ as BDC, where

$$\|B_\ell D\| \lesssim \lambda_1\varepsilon\beta_{ij}/u_\ell \quad \text{for } \ell > 1, \tag{4.59}$$

$$\|DC_m\| \lesssim \lambda_1\varepsilon\beta_{jk}/u_m \text{ for } m > 1. \tag{4.60}$$

By Theorem 4.2, the computed $\widehat{R}_{ik,j}$ satisfies

$$\widehat{R}_{ik,j} = \widehat{L}_{ij}D_{jj}\widehat{U}_{jk} + \Delta R_{ik,j}, \tag{4.61}$$

$$\|\Delta R_{ik,j}\| \lesssim c\max(\lambda_1\varepsilon\beta_{ij}\|\widehat{U}_{jk}\|, \lambda_1\varepsilon\beta_{jk}\|\widehat{L}_{ij}\|, u_1\|\widehat{L}_{ij}\|\|D_{jj}\|\|\widehat{U}_{jk}\|). \tag{4.62}$$

By (4.54), we obtain a computed $\widehat{R}_{ik}$

$$\widehat{R}_{ik} = A_{ik} \circ (J + \Theta_k) - \sum_{j=1}^{k-1} \left(\widehat{L}_{ij}D_{jj}\widehat{U}_{jk} + \Delta R_{ik,j}\right) \circ (J + \Theta_j), \tag{4.63}$$

where $|\Theta_j| \leq \gamma_j^{(1)}J$ accounts for the errors in the additions of the products $\widehat{R}_{ik,j}$ to A_{ik}. We thus obtain

$$\widehat{R}_{ik} = A_{ik} - \sum_{j=1}^{k-1} \widehat{L}_{ij}D_{jj}\widehat{U}_{jk} + \Delta R_{ik}, \tag{4.64}$$

$$\|\Delta R_{ik}\| \lesssim ku_1\|A_{ik}\| + \sum_{j=1}^{k-1} \max\left(\lambda_1\lambda_2 c\varepsilon\max(\beta_{ij},\beta_{jk}),\ (\lambda_2^2 c + j)u_1\|D_{jj}\|\right). \tag{4.65}$$

$\widehat{R}_{ik}$ is then compressed into T_{ik} such that the part of T_{ik} stored in precision u_ℓ satisfies

$$T_{ik}^{(\ell)} = \widehat{R}_{ik}^{(\ell)} + E_{ik}^{(\ell)}, \quad \|E_{ik}^{(\ell)}\| \leq \varepsilon\beta_{ik}/u_\ell, \tag{4.66}$$

and so overall, by Theorem 2.2,

$$T_{ik} = \widehat{R}_{ik} + E_{ik}, \quad \|E_{ik}\| \lesssim (2p-1)\varepsilon\beta_{ik}. \tag{4.67}$$

Finally, we solve (4.53) for L_{ik}, and by Theorem 4.3, the computed $\widehat{L}_{ik}$ satisfies

$$\widehat{L}_{ik}D_{kk}\widehat{U}_{kk} = T_{ik} + F_{ik}, \quad \|F_{ik}\| \lesssim pb\lambda_1\varepsilon\beta_{ik} + bu_1\lambda_2^2\|D_{kk}\|. \tag{4.68}$$

Putting together (4.68), (4.67), (4.65), we obtain

$$\widehat{L}_{ik}D_{kk}\widehat{U}_{kk} = A_{ik} - \sum_{j=1}^{k-1}\widehat{L}_{ij}D_{jj}\widehat{U}_{jk} + \Delta R_{ik} + E_{ik} + F_{ik}, \tag{4.69}$$

and so

$$A_{ik} = \sum_{j=1}^{k}\widehat{L}_{ij}D_{jj}\widehat{U}_{jk} + \Delta A_{ik}, \tag{4.70}$$

$$\|\Delta A_{ik}\| \lesssim ku_1\|A_{ik}\| + \sum_{j=1}^{k}\max\left(\lambda_1\lambda_2 c\varepsilon\max(\beta_{ij},\beta_{jk}),\ (\lambda_2^2 c + j)u_1\|D_{jj}\|\right). \tag{4.71}$$

With the choice $\beta_{ij} = \beta_{jk} = \|A\|$ for $j = 1\colon k$, and since $k \le q$, we obtain

$$\|\Delta A_{ik}\| \lesssim \lambda_1\lambda_2 cq\varepsilon\|A\| + qu_1\|A_{ik}\| + q(\lambda_2^2 c + q)\rho u_1\|A\|, \tag{4.72}$$

where we have used $\|D_{jj}\| \le \rho\|A\|$, where ρ denotes the growth factor. This concludes the case $i < k$. Bounds analogous to (4.72) hold for $i = k$ and $i > k$, and so overall we have

$$A = \widehat{L}D\widehat{U} + \Delta A, \quad \|\Delta A\| \lesssim q^2\left(\lambda_1\lambda_2 c\varepsilon + (\lambda_2^2 c + q)\rho u_1\right)\|A\|. \tag{4.73}$$

We summarize this analysis in the next theorem.

THEOREM 4.4 (Mixed precision BLR LU factorization). *Let $A \in \mathbb{R}^{n\times n}$ be a BLR matrix partitioned into q^2 blocks of order b. If the BLR LU factorization of A in p precisions described by Algorithms 4.1–4.4 runs to completion, the computed LU factors satisfy*

$$A = \widehat{L}D\widehat{U} + \Delta A, \quad \|\Delta A\| \lesssim q^2\left(\lambda_1\lambda_2 c\varepsilon + (\lambda_2^2 c + q)\rho u_1\right)\|A\|. \tag{4.74}$$

where λ_1 and λ_2 are defined by (4.51)–(4.52), ρ is the growth factor, and $c = p^2 b + (p+1)r^{3/2} + p$.

Theorem 4.4 therefore proves the backward stability of the mixed precision BLR LU factorization: the computed LU factors give an exact LU decomposition of a perturbed matrix, where the norm of the perturbation $\|\Delta A\|$ is of order ε. The precise value of the constants $q^2\lambda_1\lambda_2 c$ and $q^2(\lambda_2^2 c + q)\rho$ in (4.74) is not of great importance but, as a check, we compare it against the uniform precision bound [24, Thm. 4.3]

$$A = \widehat{L}\widehat{U} + \Delta A, \quad \|\Delta A\| \lesssim q\varepsilon\|A\| + (b + 2r^{3/2} + q)u_1\|\widehat{L}\|\|\widehat{U}\|. \tag{4.75}$$

Since $\|\widehat{L}\|\|\widehat{U}\| \lesssim n^2\rho\|A\|$ and, with partial pivoting, $\lambda_2 \le b$, we see that both (4.74) and (4.75) grow as $O(n^2(b+q)\rho)$.

After Theorem 4.4, not much additional effort is needed to prove the backward stability of the solution of linear systems $Ax = v$ by mixed precision BLR LU factorization. We note that mixed precision arithmetic can also be used in the solution of the triangular systems with the LU factors, but in the interest of space, we omit these details.

Table 5.1: List of matrices used in our experiments. We use their Schur complement corresponding to the root separator in their multifrontal factorization, whose order n is given in the second column.

Matrix	n	b	Application
nd24k	8k	128	2D/3D problem
audikw_1	4k	128	Structural problem
perf009d	2k	64	Elastic computation of a pump with internal pressure
Transport	5k	256	3D finite element flow and transport
P64	4k	128	Poisson equation (3D, mesh size=64)
nlpkkt80	14k	256	3D PDE-constrained optimization problem
Fault_639	8k	128	Contact mechanics for a faulted gas reservoir
Geo_1438	13k	256	Geomechanical model of earth crust
Serena	16k	256	Gas reservoir simulation for CO2 sequestration
Cube_Coup_dt0	21k	256	3D coupled consolidation problem (3D cube)

5. Experimental results.

5.1. Experimental setting. We have written a MATLAB code that implements a mixed precision variant of Algorithm 4.1 that uses Algorithms 4.2–4.4. Our implementation can use any number of arbitrary precisions, where the lower precisions are simulated using the `chop` function of Higham and Pranesh [25]. To compress the blocks, we use a mixed precision truncated QR decomposition with column pivoting—we omit a detailed description of this algorithm, which we plan to investigate more in depth in future work. We have made our code publicly available online[2].

For our experiments, we use the matrices listed in Table 5.1. These matrices are all obtained as the Schur complement of larger sparse matrices (specifically, the root separators of their multifrontal factorization) arising in various applications: P64 comes from the discretization of a Poisson equation, perf009d comes from a structural mechanics problem from EDF (French electricity supplier), the others come from the SuiteSparse collection [16].

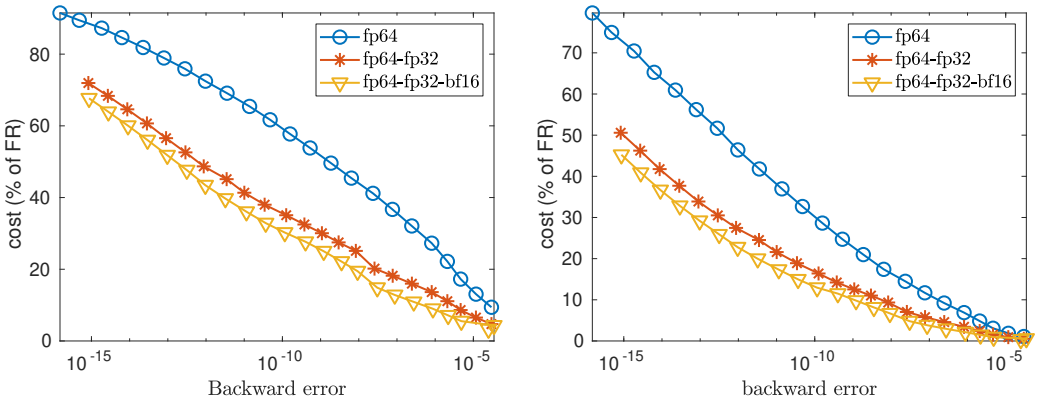
To confirm experimentally the numerical stability of the algorithms, and to assess the impact of mixed precision arithmetic on their accuracy, we measure backward errors. Rather than computing the backward error for the LU factorization, which is expensive, we use the computed LU factors to solve a linear system $Ax = v$, where x is the vector of ones (and v is computed as Ax), and we use the computed solution $\widehat{x}$ to measure the backward error

$$\frac{\|A\widehat{x} - v\|}{\|A\|\|\widehat{x}\|} \tag{5.1}$$

given by the Rigal–Gaches theorem [23, Thm 7.1], [32].

5.2. Performance–accuracy tradeoff. The analytical error bounds obtained in section 4 show that the use of mixed precision arithmetic should only increase the backward error by a small constant. In this first experiment, we check experimentally (i) that the error increase is indeed small, and (ii) whether the flops and storage gains obtained by the use of mixed precision justify this error increase, that is, whether

[2] https://gitlab.com/mgerest/mixedblr



(a) Storage. (b) Flops.

Fig. 5.1: Storage and flops for three variants of the BLR LU factorization of matrix perf009d, given as a percentage of the Full-Rank factorization, and as a function of the backward error.

the mixed precision variant achieves a better performance–accuracy tradeoff than the uniform precision variant. Indeed, since the mixed precision variant achieves a slightly larger error, to be completely fair, we should compare it against the uniform precision variant with a correspondingly larger ε.

To answer this question, we perform the following experiment in Figure 5.1: taking several values of ε ranging from 10^{-16} to 10^{-5}, we plot the storage and flop costs as a function of the error (5.1). We compare three variants of the BLR factorization: the standard uniform precision variant run entirely with fp64 arithmetic, a two-precision variant using both fp64 and fp32, and a three-precision variant using bfloat16 as well. For the flops, we assume that their cost is proportional to the number of bits of each arithmetic. The figure shows that the two-precision variant achieves a much better performance–accuracy tradeoff than the uniform precision one, and that the three-precision variant further improves this tradeoff. Indeed, using lower precisions slightly increases the error, but the experiment shows that this increase is largely compensated by the flops and storage gains. Indeed, the closer a variant is to the bottom left corner of the plots, the better its tradeoff is: for a given accuracy, it requires less flops and storage than the other variants, or, equivalently, for a given flops or storage cost, it achieves an improved accuracy.

In light of this experiment, and to avoid hand-tuning ε for every variant and every matrix, in the remainder of our experiments we directly compare the variants with the same value of ε.

5.3. Results on real-life matrices. In this section we experiment on the real-life matrices listed in Table 5.1. Figure 5.2 compares the backward error achieved by the BLR factorization for three values of ε: 10^{-12}, 10^{-9}, and 10^{-6}. For $\varepsilon = 10^{-6}$, we compare the uniform fp32 precision BLR factorization with the two-precision one using fp32 and bfloat16; for $\varepsilon = 10^{-9}$ and 10^{-12}, we compare the uniform fp64 precision factorization with both a two-precision variant (using fp64 and fp32) and a three-precision one (also using bfloat16). The figure shows that the use of mixed precision arithmetic does not significantly impact the backward error, leading to an increase of at most an order of magnitude in the worst case (and very often much less than that).

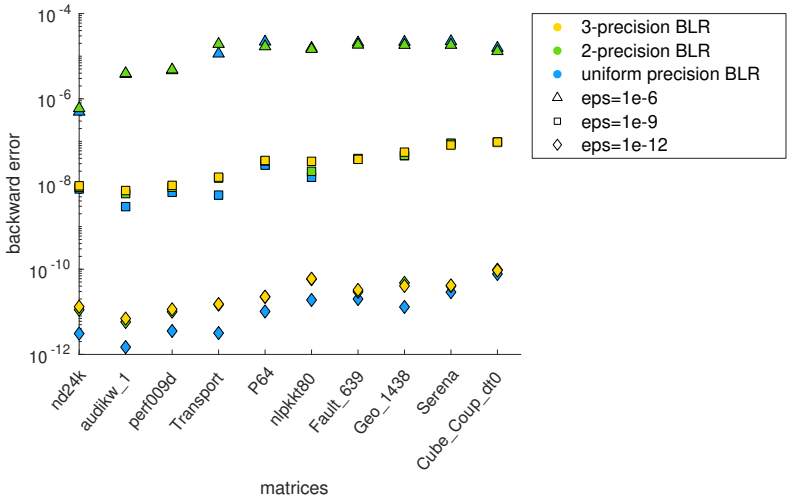


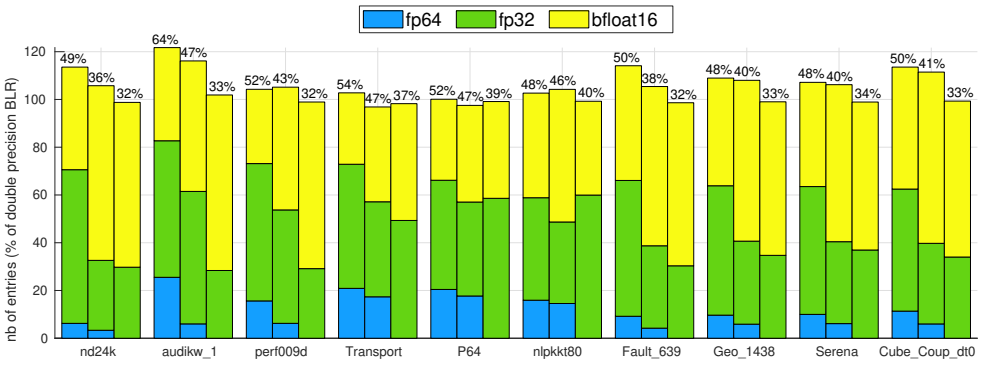
Fig. 5.2: Backward error for the uniform and mixed precision BLR factorizations, for $\varepsilon = 10^{-12}$, 10^{-9}, and 10^{-6}.

Figure 5.3 shows the associated storage and flops gains obtained by the use of mixed precision arithmetic. For each matrix, each bar corresponds to a different value of ε. For $\varepsilon = 10^{-12}$ and 10^{-9}, we focus on the gains achieved by the three-precision variant. The y-axis (height of the bars) indicates the number of entries (or number of flops) required by the mixed precision variant as a percentage of the uniform precision variant. For storage, this percentage is at least 100%, but can be slightly larger because of the difference between conditions (2.22) and (2.23). Indeed, as explained in section 2, there are some blocks that satisfy (2.23) but do not satisfy (2.22): that is, we allow the mixed precision variant to store more entries, because we expect this increase to pay off thanks to the use of lower precisions. As a result, the percentage for the flops can also differ from 100%, either being larger or smaller.

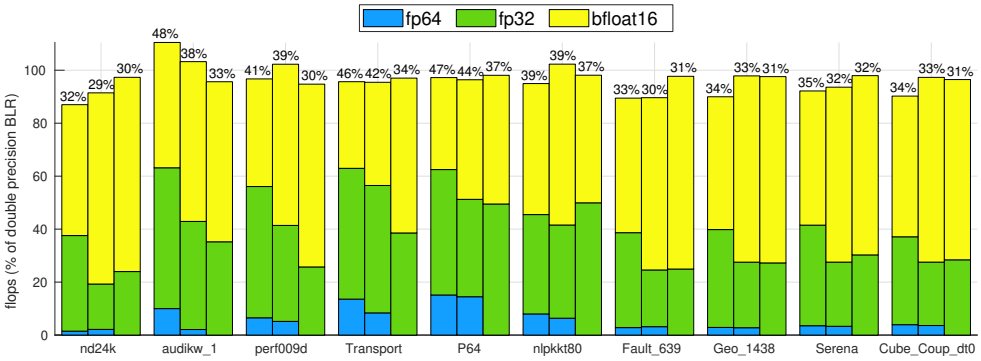
The colors breakdown in Figure 5.3 shows the proportion of entries that are stored in each precision, and the proportion of flops that are performed in each precision. For $\varepsilon = 10^{-6}$, note that the mixed precision algorithm recovers the fact that we do not need any entries or flops in fp64 arithmetic, since $\varepsilon > u_s$. For all matrices there is a significant fraction of the entries that can be stored in lower precision, even for $\varepsilon = 10^{-12}$. This is a very positive result that confirms that BLR matrices are amenable to the use of mixed precision arithmetic and that the proposed mixed precision BLR representation can achieve very significant gains with respect to the uniform precision one.

The precise gains in storage and flops depend on the relative performance of each arithmetic. For storage, it is easy to measure this gain since the storage cost of each arithmetic is proportional to the number of bits it uses: thus, an fp32 number requires half the storage of an fp64 one, and a bfloat16 number requires a quarter of the storage. The number on top of each bar in Figure 5.3a indicates the resulting storage of the three-precision variant, as a percentage of the uniform precision fp64 variant (for example, "33%" means that we expect the three-precision variant to reduce the overall storage by a factor three). The figure shows very significant reductions, of up to a factor 2.8× for $\varepsilon = 10^{-9}$.

As for the flops results of Figure 5.3b, we can use them to estimate the expected



(a) Storage.

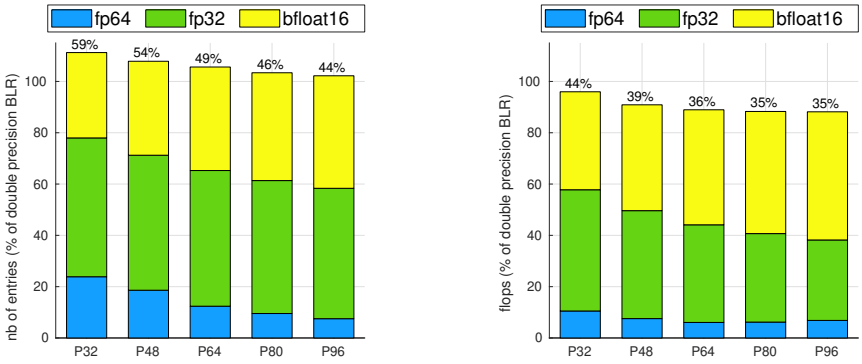


(b) Flops.

Fig. 5.3: Storage and flops for the mixed precision BLR factorization. For each matrix, the 3 bars correspond to $\varepsilon = 10^{-12}$, $\varepsilon = 10^{-9}$ and $\varepsilon = 10^{-6}$. The y-axis shows the number of entries and the number of flops required by the mixed precision variant with respect to the uniform precision variant in fp64 (which can be slightly different from 100% because of the different conditions to represent a block under low-rank form (2.22) and (2.23)). The color breakdown gives the proportion of entries/flops in each precision, and the number above each bar indicates the resulting expected performance of the mixed precision variant as a percentage of the uniform precision variant.

performance gains in execution time. This is a more complex issue because the relative speed of each arithmetic strongly depends on the hardware, the matrix, and several other factors. A practical high performance implementation of the mixed precision BLR factorization is outside our scope but, as a rough indicator, we plot on top of each bar the expected time (again, as a percentage of the uniform precision fp64 variant) under the assumption that the speed of each arithmetic is also proportional to the number of bits. The results are once more very positive, showing expected gains of up to a factor 3.4× for $\varepsilon = 10^{-9}$.

Finally, in Figure 5.4 we perform a similar experiment as in Figure 5.3 for matrices of increasing size belonging to the same Poisson problem class. This experiment highlights an important and valuable property of the mixed precision BLR factorization: the storage and flops gains increase with the problem size, as a larger and larger fraction of the entries and flops can be safely switched to lower precisions.



(a) Storage. (b) Flops.

Fig. 5.4: Storage and flops for the three-precision BLR factorization for Poisson matrices of increasing size. We have set $\varepsilon = 10^{-12}$ and $b = 64$.

6. Conclusion. We have introduced a novel approach to exploit mixed precision arithmetic for low-rank approximations. Given a prescribed accuracy ε, we have proved in Theorem 2.2 that singular vectors associated with sufficiently small singular values can be stored in precisions with unit roundoff larger than ε while preserving an overall accuracy of order ε. This approach is not only applicable to low-rank matrices built with a singular value decomposition, but also to many other low-rank decompositions, in particular rank-revealing QR (Lemma 2.3).

We have applied this approach to block low-rank (BLR) matrices, for which this new mixed precision low-rank approximation presents a high potential. We have adapted the existing uniform precision BLR LU factorization algorithm (Algorithm 4.1) to exploit the mixed precision representation of the blocks. We carried out the rounding error analysis of this new algorithm and obtained two keys results. First, we proved in Theorem 4.4 that the use of mixed precision arithmetic does not compromise the numerical stability of BLR LU factorization recently proven by Higham and Mary [24]. Second, our analysis determines which level of precision is needed for each floating-point operation, and therefore guides us towards a method that is both robust and efficient. The resulting mixed precision BLR algorithms are summarized in Algorithms 4.2, 4.3, and 4.4.

We have evaluated the potential of this mixed precision BLR LU factorization on a range of matrices coming from real-life problems from industrial and academic applications. We have shown that a large fraction of the entries and flops can be safely switched to lower precisions. For $\varepsilon = 10^{-9}$, by mixing fp64, fp32, and bfloat16 arithmetics, we obtain reductions in storage of up to 2.8× with respect to uniform precision BLR. Moreover, assuming fp32 and bfloat16 flops are, respectively, twice and four times faster than fp64 ones, we estimate the expected time gains, predicting reductions of up to 3.4× with respect to uniform precision BLR. We emphasize that these gains are not achieved at the expense of accuracy: for the same accuracy, the mixed precision variant is less expensive than the uniform precision one, or, equivalently, for a fixed storage or work budget, the mixed precision variant is more accurate (Figure 5.1).

Given the very promising results obtained with this new mixed precision BLR approach, we plan as future work to develop its high performance implementation and integrate it within the sparse direct solver MUMPS [9], which already exploits

uniform precision BLR compression.

Acknowledgements. This work was done in the context of the CIFRE PhD thesis of Matthieu Gerest funded by EDF. We thank Cleve Ashcraft for insightful discussions that led to the genesis of this work.

REFERENCES

[1] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, A. Fox, M. Gates, N. J. Higham, X. S. Li, J. Loe, P. Luszczek, S. Pranesh, S. Rajamanickam, T. Ribizel, B. F. Smith, K. Swirydowicz, S. Thomas, S. Tomov, Y. M. Tsai, and U. M. Yang, *A survey of numerical linear algebra methods utilizing mixed-precision arithmetic*, Int. J. High Performance Computing Applications, (2021), p. 109434202110033, https://doi.org/10.1177/10943420211003313.

[2] S. Abdulah, H. Ltaief, Y. Sun, M. G. Genton, and D. E. Keyes, *Geostatistical modeling and prediction using mixed precision tile Cholesky factorization*, in 2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC), IEEE, Dec. 2019, https://doi.org/10.1109/hipc.2019.00028.

[3] E. Agullo, F. Cappello, S. Di, L. Giraud, X. Liang, and N. Schenkels, *Exploring variable accuracy storage through lossy compression techniques in numerical linear algebra: a first application to flexible GMRES*, Research Report RR-9342, Inria Bordeaux Sud-Ouest, May 2020, https://hal.inria.fr/hal-02572910.

[4] K. Ahmad, H. Sundar, and M. Hall, *Data-driven mixed precision sparse matrix vector multiplication for GPUs*, ACM Trans. Archit. Code Optim., 16 (2019), pp. 51:1–51:24, https://doi.org/10.1145/3371275.

[5] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker, *Improving multifrontal methods by means of block low-rank representations*, SIAM J. Sci. Comput., 37 (2015), pp. A1451–A1474, https://doi.org/10.1137/120903476.

[6] P. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. Mary, and B. Vieublé, *Five-precision GMRES-based iterative refinement*, MIMS EPrint 2021.5, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Apr. 2021, http://eprints.maths.manchester.ac.uk/id/eprint/2807.

[7] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary, *On the Complexity of the Block Low-Rank Multifrontal Factorization*, SIAM J. Sci. Comput., 39 (2017), pp. A1710–A1740, https://doi.org/10.1137/16M1077192.

[8] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary, *Bridging the Gap between Flat and Hierarchical Low-rank Matrix Formats: the Multilevel Block Low-Rank Format*, SIAM J. Sci. Comput., 41 (2019), pp. A1414–A1442, https://doi.org/10.1137/18M1182760.

[9] P. R. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary, *Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures*, ACM Trans. Math. Software, 45 (2019), pp. 2:1–2:26, https://doi.org/10.1145/3242094.

[10] M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, vol. 63 of Lecture Notes in Computational Science and Engineering (LNCSE), Springer-Verlag, 2008.

[11] P. Blanchard, N. J. Higham, F. Lopez, T. Mary, and S. Pranesh, *Mixed precision block fused multiply-add: Error analysis and application to GPU tensor cores*, SIAM J. Sci. Comput., 42 (2020), pp. C124–C141, https://doi.org/10.1137/19M1289546.

[12] P. Blanchard, N. J. Higham, and T. Mary, *A class of fast and accurate summation algorithms*, SIAM J. Sci. Comput., 42 (2020), pp. A1541–A1557, https://doi.org/10.1137/19M1257780.

[13] E. Carson and N. J. Higham, *A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems*, SIAM J. Sci. Comput., 39 (2017), pp. A2834–A2856, https://doi.org/10.1137/17M1122918.

[14] E. Carson and N. J. Higham, *Accelerating the solution of linear systems by iterative refinement in three precisions*, SIAM J. Sci. Comput., 40 (2018), pp. A817–A847, https://doi.org/10.1137/17M1140819.

[15] E. Carson, N. J. Higham, and S. Pranesh, *Three-precision GMRES-based iterative refinement for least squares problems*, SIAM J. Sci. Comput., 42 (2020), pp. A4063–A4083, https://doi.org/10.1137/20m1316822.

[16] T. A. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), pp. 1:1–1:25, https://doi.org/10.1145/2049662.2049663, http:

//doi.acm.org/10.1145/2049662.2049663.
[17] N. Doucet, H. Ltaief, D. Gratadour, and D. Keyes, *Mixed-precision tomographic reconstructor computations on hardware accelerators*, in 2019 IEEE/ACM 9th Workshop on Irregular Applications: Architectures and Algorithms (IA3), Nov. 2019, pp. 31–38, https://doi.org/10.1109/IA349570.2019.00011.
[18] R. D. Fierro and P. C. Hansen, *Low-rank revealing utv decompositions*, Numerical Algorithms, 15 (1997), pp. 37–55.
[19] S. Gratton, E. Simon, D. Titley-Peloquin, and P. Toint, *Exploiting variable precision in GMRES*, ArXiv:1907.10550, July 2019, https://arxiv.org/abs/1907.10550. Revised February 2020.
[20] W. Hackbusch, *Hierarchical Matrices : Algorithms and Analysis*, vol. 49 of Springer series in computational mathematics, Springer, Berlin, 2015, https://doi.org/10.1007/978-3-662-47324-5.
[21] A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, *Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC18 (Dallas, TX), Piscataway, NJ, USA, 2018, pp. 47:1–47:11, https://doi.org/10.1109/SC.2018.00050.
[22] N. J. Higham, *Optimization by direct search in matrix computations*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 317–333, https://doi.org/10.1137/0614023.
[23] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second ed., 2002, https://doi.org/10.1137/1.9780898718027.
[24] N. J. Higham and T. Mary, *Solving block low-rank linear systems by LU factorization is numerically stable*, IMA J. Numer. Anal., (2020), pp. 1–30, https://doi.org/10.1093/imanum/drab020.
[25] N. J. Higham and S. Pranesh, *Simulating low precision floating-point arithmetic*, SIAM J. Sci. Comput., 41 (2019), pp. C585–C602, https://doi.org/10.1137/19M1251308.
[26] F. Lopez and T. Mary, *Mixed Precision LU Factorization on GPU Tensor Cores: Reducing Data Movement and Memory Footprint*, http://eprints.maths.manchester.ac.uk/2782/. MIMS EPrint 2020.20, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, September 2020.
[27] S. Markidis, S. W. D. Chien, E. Laure, I. B. Peng, and J. S. Vetter, *Nvidia tensor core programmability, performance precision*, in 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2018, pp. 522–531, https://doi.org/10.1109/IPDPSW.2018.00091.
[28] T. Mary, *Block Low-Rank multifrontal solvers: complexity, performance, and scalability*, PhD thesis, Université de Toulouse, Nov. 2017.
[29] D. Mukunoki, K. Ozaki, T. Ogita, and T. Imamura, *DGEMM using tensor cores, and its accurate and reproducible versions*, in High Performance Computing, Cham, 2020, Springer International Publishing, pp. 230–248, https://doi.org/10.1007%2F978-3-030-50743-5_12.
[30] R. Ooi, T. Iwashita, T. Fukaya, A. Ida, and R. Yokota, *Effect of mixed precision computing on h-matrix vector multiplication in BEM analysis*, in Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, ACM Press, New York, Jan. 2020, https://doi.org/10.1145/3368474.3368479.
[31] G. Pichon, E. Darve, M. Faverge, P. Ramet, and J. Roman, *Sparse supernodal solver using block low-rank compression: Design, performance and analysis*, Journal of Computational Science, 27 (2018), pp. 255–270, https://doi.org/https://doi.org/10.1016/j.jocs.2018.06.007, http://www.sciencedirect.com/science/article/pii/S1877750317314497.
[32] J. L. Rigal and J. Gaches, *On the compatibility of a given solution with the data of a linear system*, J. Assoc. Comput. Mach., 14 (1967), pp. 543–548, https://doi.org/10.1145/321406.321416, http://doi.acm.org/10.1145/321406.321416.
[33] R. D. Skeel, *Scaling for numerical stability in gaussian elimination*, J. Assoc. Comput. Mach., 26 (1979), pp. 494–526, https://doi.org/10.1145/322139.322148.
[34] I. Yamazaki, S. Tomov, and J. Dongarra, *Mixed-precision Cholesky QR factorization and its case studies on multicore CPU with multiple GPUs*, SIAM J. Sci. Comput., 37 (2015), pp. C307–C330, https://doi.org/10.1137/14M0973773.
[35] L. M. Yang, A. Fox, and G. Sanders, *Rounding error analysis of mixed precision block householder qr algorithms*, arXiv preprint arXiv:1912.06217, (2019).