Solving Block Low-Rank Linear Systems by LU Factorization is Numerically Stable[†]

NICHOLAS J. HIGHAM
Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK
(nick.higham@manchester.ac.uk)
THEO MARY
Sorbonne Université, CNRS, LIP6, Paris, France.
(theo.mary@lip6.fr)

Block low-rank (BLR) matrices possess a blockwise low-rank property that can be exploited to reduce the complexity of numerical linear algebra algorithms. The impact of these low-rank approximations on the numerical stability of the algorithms in floating-point arithmetic has not previously been analyzed. We present rounding error analysis for the solution of a linear system by LU factorization of BLR matrices. Assuming that a stable pivoting scheme is used, we prove backward stability: the relative backward error is bounded by a modest constant times ε, where the low-rank threshold ε is the parameter controlling the accuracy of the blockwise low-rank approximations. In addition to this key result, our analysis offers three new insights into the numerical behavior of BLR algorithms. First, we compare the use of a global or local low-rank threshold and find that a global one should be preferred. Second, we show that performing intermediate recompressions during the factorization can significantly reduce its cost without compromising numerical stability. Third, we consider different BLR factorization variants and determine the update–compress–factor (UCF) variant to be the best. Tests on a wide range of matrices from various real-life applications show that the predictions from the analysis are realized in practice.

Keywords: Block low-rank matrices, rounding error analysis, floating-point arithmetic, numerical stability, LU factorization

1. Introduction

In many applications requiring the solution of a linear system $Ax = v$, the coefficient matrix A has been shown to have a blockwise low-rank property: most of its off-diagonal blocks are of low numerical rank and can therefore be well approximated by low-rank (LR) matrices. Several formats have been proposed to exploit this property, differing in how the matrix is partitioned into blocks. In this article, we focus on the block low-rank (BLR) format (Amestoy *et al.*, 2015), which is based on a flat, non-hierarchical partitioning allowing it to reduce both the theoretical complexity (Amestoy *et al.*, 2017) and the practical time and memory costs (Amestoy *et al.*, 2019b) of key numerical linear algebra computations such as solving $Ax = v$ by LU factorization.

Even though the BLR format has been extensively studied and widely used in numerous applications (see, among others, Amestoy *et al.* (2019b), Amestoy *et al.* (2016), Shantsev *et al.* (2017), Pichon *et al.* (2018), Charara *et al.* (2018), and Ida *et al.* (2018)), little is known about its numerical behavior in floating-point arithmetic. Indeed, no rounding error analysis has been published for BLR matrix

[†]Version of January 4, 2021. **Funding**: This work was supported by Engineering and Physical Sciences Research Council grant EP/P020720/1, The MathWorks, and the Royal Society. The opinions and views expressed in this publication are those of the authors, and not necessarily those of the funding bodies.

algorithms. The difficulty of such an analysis lies in the fact that, unlike classical algorithms, there are two kinds of errors to analyze: the floating-point errors (which depend on the unit roundoff u) and the low-rank truncation errors (which depend on the low-rank threshold $\varepsilon > u$). Moreover, these two kinds of errors cannot be easily isolated because the BLR compression and factorization stages are often interlaced. Yet performing such an analysis is crucial to better understand the effect of BLR approximations on the stability of these algorithms and, in particular, to shed light on the following open problems.

1. It has been experimentally observed that the solution to BLR linear systems generally yields a backward error closely related to the low-rank threshold ε. This is an important and valuable property that has, however, never been formally proved. The dependence of the backward error on the unit roundoff should also be investigated.

2. In contrast to hierarchical matrices, the number of block-rows and block-columns in BLR matrices usually grows with the matrix size and may thus become very large. It is therefore important to determine how the error grows as the matrix size increases and whether it depends on the number of blocks.

3. The low-rank approximation $\widetilde{A}_{ij}$ to a block A_{ij} is computed such that $\|A_{ij} - \widetilde{A}_{ij}\| \leqslant \varepsilon\beta_{ij}$, where β_{ij} is a scalar that can be freely chosen and whose impact on the numerical behavior of the algorithms is currently not well understood. In particular, local ($\beta_{ij} = \|A_{ij}\|$) and global ($\beta_{ij} = \|A\|$) low-rank thresholds have both been proposed in the literature and should be compared.

4. Several BLR LU factorization algorithms can be distinguished, depending on when the compression is performed. These algorithms have been compared in terms of asymptotic complexity, performance, and storage requirements (Mary, 2017). However, it is not currently known how they compare in terms of numerical stability.

In this article, we answer these questions by doing detailed rounding error analyses of various BLR matrix algorithms. We begin in section 2 with the preliminary material necessary for the analysis. Section 3 analyzes several kernels involving LR matrices, such as LR matrix–vector and matrix–matrix products. Then, section 4 builds upon these results to analyze two BLR LU factorizations algorithms and their use to solve BLR linear systems. Throughout the article, numerical experiments are interlaced with theoretical results to illustrate them. We provide additional experiments on a wide range of matrices coming from various real-life applications in section 5. We gather our conclusions in section 6.

This article focuses on the *direct* solution of a BLR linear system $Ax = v$ by LU factorization of A. Such systems can be alternatively solved by *iterative* methods, which rely on multiplying the BLR matrix A with a vector. We have also performed the rounding error analysis of matrix–vector multiplication with A replaced by its BLR approximation, which we include as supplementary material.

2. Technical background and experimental setting

2.1 *Low-rank (LR) and block low-rank (BLR) matrices*

Let $A \in \mathbb{R}^{b\times b}$ have the SVD $U\Sigma V^T$, where $\Sigma = \operatorname{diag}(\sigma_i)$ with $\sigma_1 \geqslant \cdots \geqslant \sigma_b \geqslant 0$. Given a target rank $k \leqslant b$, the quantity $\|A - \widetilde{A}\|$ for any rank-k matrix $\widetilde{A}$ is known to be minimized for any unitarily invariant norm by the truncated SVD

$$\widetilde{A} = U_{:,1:k}\Sigma_{1:k,1:k}V^T_{:,1:k}. \tag{2.1}$$

If the singular values of A decay rapidly, $\|A-\widetilde{A}\|$ can be small even for $k \ll b$. In this case, $\widetilde{A}$ is referred to as a low-rank (LR) matrix, and the cost of storing $\widetilde{A}$ and computing with it can be greatly reduced. While $\widetilde{A}$ can directly be represented by the truncated SVD (2.1), in this article we use the alternative form $\widetilde{A}=XY^T$, where $X=U_{:,1:k}$ and $Y=V_{:,1:k}\Sigma^T_{1:k,1:k}$; the matrix X can thus be assumed to have orthonormal columns.

A block low-rank (BLR) representation $\widetilde{A}$ of a dense matrix A has the block $p \times p$ form

$$\widetilde{A} = \begin{bmatrix} A_{11} & \widetilde{A}_{12} & \cdots & \widetilde{A}_{1p} \\ \widetilde{A}_{21} & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ \widetilde{A}_{p1} & \cdots & \cdots & A_{pp} \end{bmatrix}, \tag{2.2}$$

where off-diagonal blocks A_{ij} of size $b \times b$ are approximated by LR matrices $\widetilde{A}_{ij}$ of rank k_{ij} given by

$$\widetilde{A}_{ij} = \begin{cases} X_{ij}Y_{ij}^T, & i>j, \\ Y_{ij}X_{ij}^T, & i<j, \end{cases} \tag{2.3}$$

where X_{ij} and Y_{ij} are $b \times k_{ij}$ matrices, and where X_{ij} has orthornormal columns. Note that 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 $n=pb$. Representation (2.3) allows for efficient intermediate recompressions during the BLR factorization, as will be explained in section 3.2.

We assume that the ranks k_{ij} are chosen as

$$k_{ij} = \min\left\{ \ell_{ij} : \|A_{ij}-\widetilde{A}_{ij}\| \leqslant \varepsilon\beta_{ij},\ \operatorname{rank}\bigl(\widetilde{A}_{ij}\bigr)=\ell_{ij} \right\}, \tag{2.4}$$

where $\varepsilon>0$ is referred to as the *low-rank threshold* and controls the accuracy of the approximations $\widetilde{A}_{ij} \approx A_{ij}$, and where we have either $\beta_{ij}=\|A_{ij}\|$ or $\beta_{ij}=\|A\|$. The parameters β_{ij} therefore control whether the blocks are approximated relative to their own norm $\|A_{ij}\|$ or the norm of the global matrix $\|A\|$. We refer to the low-rank threshold as *local* in the former case and as *global* in the latter case.

With a local threshold, k_{ij} corresponds to the usual definition of numerical rank. Importantly, with a global threshold, blocks that do not have rapidly decaying singular values may still be approximated by LR matrices if they are of small norm compared with $\|A\|$. Indeed, even though such blocks have high numerical rank relative to their own norm, their contribution to the global computation can be considered to be negligible compared with other blocks of larger norm. In the most extreme cases, some blocks may be of norm smaller than ε: these blocks can then approximated by *zero-rank* matrices, that is, they may be dropped entirely. Exploiting this fact can drastically improve the compression, sometimes even leading to an improved asymptotic complexity (Amestoy *et al.*, 2017).

Note that in general, low-rank approximations only hold in a *normwise* sense: that is, given a matrix A and a LR matrix $\widetilde{A}$ satisfying $\|\widetilde{A}-A\| \leqslant \varepsilon\|A\|$, the componentwise inequality $|\widetilde{A}-A| \leqslant \varepsilon|A|$ does not hold. For this reason we perform a normwise error analysis.

Throughout this article, the unsubscripted norm $\|\cdot\|$ denotes the Frobenius norm $\|A\| = (\sum_{i,j}|a_{ij}|^2)^{1/2}$, which we use for all our error analysis. We choose to work with this norm for three of its useful properties. First, it is submultiplicative (also called consistent): $\|AB\| \leqslant \|A\|\|B\|$. Second, it is invariant under multiplication on the left by a matrix with orthonormal columns X: $\|XA\|=\|A\|$ (note that X is not necessarily unitary: $X^TX=I$ must hold but $XX^T=I$ need not). Finally, unlike for the spectral norm, it is easy to switch between blockwise and global bounds using the relation $\|A\| = \left(\sum_{i,j}\|A_{ij}\|^2\right)^{1/2}$ for any block partitioning of A. We will use all these properties of the Frobenius norm without comment. More specific examples of why the Frobenius norm is the best choice for our analysis are given throughout sections 3 and 4.

2.2 *Floating-point arithmetic and rounding error analysis*

Throughout the article, we use the standard model of floating-point arithmetic (Higham, 2002, sec. 2.2)

$$\mathrm{fl}(x \,\mathrm{op}\, y) = (x \,\mathrm{op}\, y)(1+\delta), \quad |\delta| \leqslant u, \quad \mathrm{op} \in \{+,-,\times,/\}. \tag{2.5}$$

We also define $\gamma_k = ku/(1-ku)$ for $ku < 1$. We will use without comment the relations (Higham, 2002, Lem. 3.3)

$$j\gamma_k \leqslant \gamma_{jk}, \quad \gamma_j + \gamma_k + \gamma_j\gamma_k \leqslant \gamma_{j+k},$$

which hold for any $j,k \geqslant 1$ (including non-integer j,k, which we will sometimes use).

We recall normwise error bounds for some basic matrix computations.

LEMMA 2.1 (Error bounds for matrix–vector and matrix–matrix products (Higham, 2002, p. 71)) Let $A \in \mathbb{R}^{a\times b}$, $v \in \mathbb{R}^b$, and $w = Av$. The computed $\widehat{w}$ satisfies

$$\widehat{w} = (A+\Delta A)v, \quad \|\Delta A\| \leqslant \gamma_b\|A\|. \tag{2.6}$$

Let $B \in \mathbb{R}^{b\times c}$ and let $C = AB$. The computed $\widehat{C}$ satisfies

$$\widehat{C} = AB + \Delta C, \quad \|\Delta C\| \leqslant \gamma_b\|A\|\|B\|. \tag{2.7}$$

LEMMA 2.2 (Backward error bound for triangular systems (Higham, 2002, Thm. 8.5)) Let $T \in \mathbb{R}^{b\times b}$ be nonsingular and triangular and let $v \in \mathbb{R}^b$. The computed solution $\widehat{x}$ to the triangular system $Tx = v$ satisfies

$$(T+\Delta T)\widehat{x} = v, \quad \|\Delta T\| \leqslant \gamma_b\|T\|. \tag{2.8}$$

The computed solution $\widehat{X}$ to the multiple right-hand side triangular system $TX = V$, where $V \in \mathbb{R}^{b\times c}$, satisfies

$$T\widehat{X} = B + \Delta B, \quad \|\Delta B\| \leqslant \gamma_b\|T\|\|\widehat{X}\|. \tag{2.9}$$

LEMMA 2.3 (Backward error bound for LU factorization (Higham, 2002, Thm. 9.3)) If the LU factorization of $A \in \mathbb{R}^{b\times b}$ runs to completion then the computed LU factors $\widehat{L}$ and $\widehat{U}$ satisfy

$$\widehat{L}\widehat{U} = A + \Delta A, \quad \|\Delta A\| \leqslant \gamma_b\|\widehat{L}\|\|\widehat{U}\|. \tag{2.10}$$

Finally, we make the assumption that rounding errors can be ignored in the computation of the LR approximations $\widetilde{A}_{ij}$ of all blocks A_{ij} of a BLR matrix A via their truncated SVD.

ASSUMPTION 2.1 (Error bound for the truncated SVD computation) Given a BLR matrix A and two positive parameters ε (the low-rank threshold) and β_{ij}, the LR blocks $\widetilde{A}_{ij}$ computed via the truncated SVD $\widetilde{A}_{ij} = \widehat{X}_{:,1:k_{ij}}\widehat{\Sigma}_{1:k_{ij},1:k_{ij}}\widehat{Y}^T_{:,1:k_{ij}}$ satisfy

$$\widetilde{A}_{ij} = A_{ij} + \Delta A_{ij}, \quad \|\Delta A_{ij}\| \leqslant \varepsilon\beta_{ij}.$$

We recall that $\beta_{ij} = \|A_{ij}\|$ or $\beta_{ij} = \|A\|$ controls whether we use a local or global threshold, as explained in the previous section.

Note that Assumption 2.1 is only satisfied if the unit roundoff u is safely smaller than the low-rank threshold ε, which we assume to be the case throughout the analysis.

2.3 *Experimental setting*

All numerical experiments reported in this article have been performed with MATLAB R2018b. Unless otherwise specified, we use IEEE double precision floating-point arithmetic. In some experiments we also use IEEE single and half precisions. The use of half precision has been simulated as described in Higham & Pranesh (2019). We have made all our codes used for the experiments available online[1].

One of the most common application domains where block low-rank matrices arise is the solution of discretized partial differential equations. These matrices are then sparse. In our analysis we however consider dense matrices, which serve as building blocks for sparse direct solvers. Therefore, in our experiments, we use a set of dense matrices obtained from the Schur complements of sparse matrices: these correspond to the root separator in the context of a nested dissection (George, 1973) solver.

In section 4, we illustrate our analysis by interlacing it with experiments on matrices coming from a Poisson problem $-\Delta u = f$, discretized with a 7-point finite-difference scheme on a 3D domain of dimensions $k \times k \times k$. This leads to a dense $k^2 \times k^2$ matrix. We test variable sizes (from $n = 32^2$ to $n = 128^2$) to explore how the error behaves as $n = k^2$ increases. In section 5, we complement these tests with experiments on matrices from the SuiteSparse collection (Davis & Hu, 2011) coming from various real-life applications.

Throughout sections 4.1 and 4.2 we present experiments on the BLR LU factorization. Instead of measuring the backward error for LU factorization $\|A - \widetilde{L}\widetilde{U}\|/\|A\|$, which is expensive to compute, we solve a linear system $Ax = v$ by forward and backward substitutions with the BLR LU factors (as described in section 4.3), where x is the vector of all ones. We then measure the backward error

$$\frac{\|A\widehat{x} - v\|}{\|A\|\|\widehat{x}\| + \|v\|} \tag{2.11}$$

of the computed $\widehat{x}$, which is much cheaper to compute.

For all experiments, the block size is set to $b = 256$ unless otherwise specified.

3. Rounding error analysis of LR matrix kernels

In this section we analyze some key kernels involving LR matrices, which are necessary to the analysis of the BLR matrix algorithms considered in the subsequent sections.

3.1 *LR matrix times vector or full matrix*

We begin by analyzing the product of an LR matrix $\widetilde{A}$ with a vector v, for which we can establish a backward error bound. We then generalize the analysis to the product of $\widetilde{A}$ with a full matrix B, for which only a forward error bound can be derived.

LEMMA 3.1 (LR matrix times vector) Let $A \in \mathbb{R}^{b\times b}$, $X \in \mathbb{R}^{b\times r}$, $Y \in \mathbb{R}^{b\times r}$, and $v \in \mathbb{R}^{b}$, where X has orthonormal columns and $\widetilde{A} = XY^T$ is an LR approximation of A satisfying $\|A - \widetilde{A}\| \leqslant \varepsilon\beta$ for some $\beta > 0$. If the matrix–vector product Av is computed as $z = X(Y^T v)$, the computed $\widehat{z}$ satisfies

$$\widehat{z} = (\widetilde{A} + \Delta\widetilde{A})v, \quad \|\Delta\widetilde{A}\| \leqslant \gamma_c\|\widetilde{A}\|, \tag{3.1}$$

[1] https://gitlab.com/theo.andreas.mary/BLRstability

where $c = b + r^{3/2}$, and therefore

$$\widehat{z} = (A + \Delta A)v, \quad \|\Delta A\| \leqslant \gamma_c \|A\| + \varepsilon(1+\gamma_c)\beta \\ = \gamma_c\|A\| + \varepsilon\beta + O(u\varepsilon). \tag{3.2}$$

Proof. Let $w = Y^T v$; the computed $\widehat{w}$ satisfies

$$\widehat{w} = (Y + \Delta Y)^T v, \quad \|\Delta Y\| \leqslant \gamma_b \|Y\| = \gamma_b \|\widetilde{A}\|. \tag{3.3}$$

Let $z = X\widehat{w}$; the computed $\widehat{z}$ satisfies

$$\widehat{z} = (X + \Delta X)\widehat{w}, \quad \|\Delta X\| \leqslant \gamma_r \|X\| = \gamma_r \sqrt{r}, \tag{3.4}$$

where $\|X\| = \sqrt{r}$ because X has orthonormal columns. Combining (3.3) and (3.4), we obtain $\widehat{z} = (X+\Delta X)(Y+\Delta Y)^T v = (\widetilde{A} + \Delta\widetilde{A})v$, with $\|\Delta \widetilde{A}\| \leqslant (\gamma_b + \gamma_r\sqrt{r} + \gamma_b\gamma_r\sqrt{r})\|\widetilde{A}\| \leqslant \gamma_c\|\widetilde{A}\|$, yielding (3.1). The bound (3.2) is obtained by replacing $\widetilde{A}$ by $A + E$, where $\|E\| \leqslant \varepsilon\beta$. □

Note that for the particular choice $\beta = \|A\|$, the bound (3.2) simplifies to

$$\widehat{z} = (A + \Delta A)v, \quad \|\Delta A\| \leqslant \big(\varepsilon + \gamma_c + \varepsilon\gamma_c\big)\|A\|.$$

This is a backward error bound, from which the forward error bound

$$\|\widehat{z} - Av\| \leqslant \big(\varepsilon + \gamma_c + \varepsilon\gamma_c\big)\|A\|\|v\|$$

trivially follows.

Before commenting on its significance, we immediately generalize this result to the case where v is a full matrix rather than a vector, in which case only a forward error bound can be obtained.

LEMMA 3.2 (LR matrix times full matrix) Let $A \in \mathbb{R}^{b\times b}$ and $V \in \mathbb{R}^{b\times m}$, and let $\widetilde{A} = XY^T$ be defined as in Lemma 3.1. If the product AV is computed as $Z = X(Y^T V)$, the computed $\widehat{Z}$ satisfies

$$\|\widehat{Z} - \widetilde{A}V\| \leqslant \gamma_c \|\widetilde{A}\|\|V\| \tag{3.5}$$

and therefore

$$\|\widehat{Z} - AV\| \leqslant \gamma_c\|A\|\|V\| + \varepsilon(1+\gamma_c)\beta\|V\|, \\ = \gamma_c\|A\|\|V\| + \varepsilon\beta\|V\| + O(u\varepsilon), \tag{3.6}$$

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

Proof. The result follows from the columnwise bounds

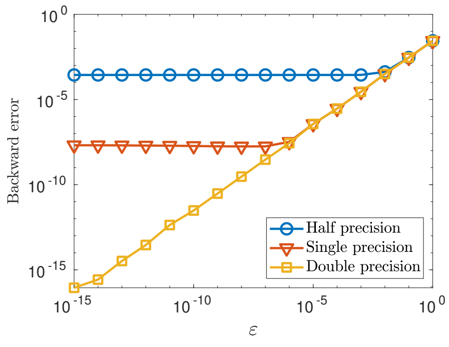
$$\|\widehat{z}_j - \widetilde{A}v_j\| \leqslant \gamma_c\|\widetilde{A}\|\|v_j\|, \quad j = 1\colon m,$$

that are obtained from (3.1). □

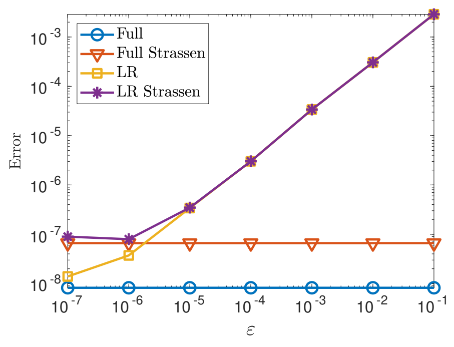
For $\beta = \|A\|$, bound (3.6) simplifies to

$$\|\widehat{Z} - AV\| \leqslant \big(\varepsilon + \gamma_c + \varepsilon\gamma_c\big)\|A\|\|V\|. \tag{3.7}$$

The bounds (3.1) and (3.5) generalize the classical bounds (2.6) and (2.7) to the case where $\widetilde{A}$ is an LR matrix rather than a full one. The bounds (3.2) and (3.6) have more informative forms, as



(A) Backward error (3.8) for three floating-point precisions.



(B) Forward error (3.9) with multiplications performed in single precision, either classically or by Strassen's algorithm.

FIG. 3.1. *Errors for different* ε *for computing* $z = \widetilde{A}v$ *(left) or* $C = \widetilde{A}V$ *(right), with* $\beta = \|A\|$, $b = 1024$, $A =$ `gallery('randsvd',b,1e16,3)`, $v =$ `rand(b,1)`, *and* $V =$ `rand(b,128)`.

they measure not only the effect of floating-point errors but also that of the low-rank truncation errors. They consist of three terms: the term ε, corresponding to the low-rank truncation errors; the term γ_c, corresponding to the floating-point errors; and their product $\varepsilon\gamma_c$, which reflects the fact that the two types of error accumulate, although this $O(u\varepsilon)$ term can be considered to be of lower order and will not always be explicitly tracked in the rest of the analysis below.

Since $u \ll \varepsilon$, the main conclusion of Lemma 3.1 is that the bound (3.2) is dominated by the low-rank truncation error term $\varepsilon\beta$ and is almost independent of both the unit roundoff and the constants b and r. This is a very positive result, since for $\beta = \|A\|$ we have $\|\Delta A\| \lesssim \varepsilon\|A\|$, meaning that the computation is backward stable with respect to the low-rank threshold ε. Moreover, a crucial consequence of this result is that we can and should use the lowest floating-point precision such that u remains safely smaller than ε. This is illustrated by Figure 3.1a, which shows that the backward error is not affected by the use of single rather than double precision arithmetic when $\varepsilon \gg 10^{-8}$. Similarly, half precision arithmetic can be used with no impact on the error when $\varepsilon \gg 10^{-4}$. In this experiment, the backward error is computed by the formula

$$\frac{\|\widehat{z} - Av\|}{\|A\|\|v\|}, \tag{3.8}$$

which is a consequence of the Rigal–Gaches theorem (Higham (2002, Thm. 7.1), Rigal & Gaches (1967)).

It is important to note that Lemma 3.1, as well as all other results in this paper, provides only a normwise error bound. This is unavoidable because low-rank approximations do not satisfy useful componentwise error bounds. Interestingly, this means that algorithms that sacrifice componentwise stability for speed, such as the 3M algorithm to multiply complex matrices with only three real multiplications (Higham, 1992), or fast matrix algorithms such as Strassen's algorithm, are much more attractive when used on LR matrices, since only normwise stability can be expected anyway. Another possible drawback of Strassen's algorithm is that the constants in the error bound are much larger: for example, for matrix multiplication, the constant b in (2.7) increases to $O(b^{\log_2 12}) \approx O(b^{3.6})$ (Higham (1990), Higham (2002, sec. 23.2.2)). This larger constant is insignificant for LR matrices because the low-rank errors dominate the floating-point ones, and so the constant c in (3.2) has essentially no impact on the overall error. This is illustrated in Figure 3.1b, where we compare the error[2]

$$\frac{\|\widehat{Z} - AV\|}{\|A\|\|V\|} \tag{3.9}$$

for computing the product of two full matrices, $Z = AV$, with that for computing the product of an LR matrix with a full one, $Z = \widetilde{A}V$, for multiplication performed classically and by Strassen's algorithm. While the use of Strassen's algorithm leads to an error larger by about an order of magnitude for the product $Z = AV$, the error for the product $Z = \widetilde{A}V$ does not increase as long as the threshold ε is large enough to hide the larger constant of Strassen's algorithm.

Even though the constant $c = b + r^{3/2}$ has no significant impact on the overall error, further insight can nevertheless be gained by analyzing its form. First, note that c heavily depends on the choice of norm used for the analysis: the proof of Lemma 3.1 repeatedly uses the fact that the Frobenius norm is both submultiplicative and unitarily invariant, which helps us to obtain a relative small constant. With the maximum norm $\|A\|_M = \max_{i,j} |a_{ij}|$ (for example) which satisfies neither of these two properties, bound (3.5) holds with a much larger constant $c = b^{3/2} r(b + r)$. Second, it may seem surprising that γ_c is larger than γ_b, the constant in the error bound (2.6) for the classical matrix–vector product, since computing $\widetilde{A}v$ rather than Av reduces the number of flops from $2b^2$ to $4br$. However, not all flops are equal, as is clearly illustrated by the case of inner and outer products $x^T y$ and xy^T of two vectors $x, y \in \mathbb{R}^n$, which require $O(n)$ and $O(n^2)$ flops, but yield error bounds proportional to γ_n and γ_1, respectively. In the case of Lemma 3.1, b rounding errors combine in the first product $w = Y^T v$, but r additional errors combine in the second product Xw. The extra $\sqrt{r}$ factor comes from the term $\|X\|$, and could thus be avoided by using the 2-norm rather than the Frobenius norm; however we need the Frobenius norm when working on BLR matrices, as will become clear in section 4.

We also note the importance of assuming that X has orthonormal columns, as it allows for replacing $\|Y\|$ by $\|\widetilde{A}\|$ in (3.3). Without that assumption, the bound would be proportional to $\|X\|\|Y\|$ instead of $\|\widetilde{A}\|$, reflecting the fact that the computation $X(Y^T v)$ would be subject to possibly severe cancellation when $\|X\|\|Y\| \gg \|\widetilde{A}\|$. Note that assuming the columns of Y, rather than X, to be orthonormal would lead to the same bound. We also mention that using the alternative form $\widetilde{A} = X\Sigma Y^T$, with $\Sigma \in \mathbb{R}^{r \times r}$ and where both X and Y have orthonormal columns yields a similar bound with a slightly different constant c.

3.2 *LR matrix times LR matrix*

Next we analyze the product of two LR matrices $\widetilde{A} = X_A Y_A^T$ and $\widetilde{B} = Y_B X_B^T$, where X_A and X_B have orthonormal columns. Note that we consider $\widetilde{B}$ of the form $Y_B X_B^T$ rather than $X_B Y_B^T$, because of the representation (2.3) whose usefulness is made clear below.

LEMMA 3.3 (LR matrix times LR matrix) Let $A, B \in \mathbb{R}^{b \times b}$ and

$$\widetilde{A} = \underbrace{X_A}_{b\times r}\underbrace{Y_A^T}_{r\times b}, \quad \widetilde{B} = \underbrace{Y_B}_{b\times r}\underbrace{X_B^T}_{r\times b},$$

[2]Note that the quantity (3.9) is not a backward error; it can be interpreted as a combination of the columnwise backward errors (3.8) for each column v_i of V.

where X_A and X_B have orthonormal columns and

$$\|A-\widetilde{A}\| \leqslant \varepsilon\beta_A, \quad \|B-\widetilde{B}\| \leqslant \varepsilon\beta_B.$$

If the product $C=\widetilde{A}\widetilde{B}$ is computed as $C=(X_A(Y_A^TY_B))X_B^T$ or $C=X_A((Y_A^TY_B)X_B^T)$, where the parentheses indicate the order in which the intermediate products are performed, then the computed $\widehat{C}$ satisfies

$$\|\widehat{C}-\widetilde{A}\widetilde{B}\| \leqslant \gamma_c\|\widetilde{A}\|\|\widetilde{B}\|, \tag{3.10}$$

where $c=b+2r^{3/2}$, and therefore

$$\|\widehat{C}-AB\| \leqslant \gamma_c\|A\|\|B\| + \varepsilon(1+\gamma_c)\big(\beta_A\|B\|+\|A\|\beta_B+\varepsilon\beta_A\beta_B\big). \tag{3.11}$$

Proof. We consider the case where the product is computed as $(\widetilde{A}Y_B)X_B^T=(X_A(Y_A^TY_B))X_B^T$, the other case being analogous. Let $W=\widetilde{A}Y_B$; by Lemma 3.2, the computed W satisfies

$$\widehat{W}=\widetilde{A}Y_B+\Delta W, \quad \|\Delta W\| \leqslant \gamma_{b+r^{3/2}}\|\widetilde{A}\|\|Y_B\| = \gamma_{b+r^{3/2}}\|\widetilde{A}\|\|\widetilde{B}\|.$$

Let $C=\widehat{W}X_B^T$; the computed $\widehat{C}$ satisfies

$$\begin{aligned}\widehat{C}&=\widehat{W}X_B^T+\Delta C, \quad \|\Delta C\| \leqslant \gamma_r\|\widehat{W}\|\|X_B\| \leqslant \gamma_{r^{3/2}}(1+\gamma_{b+r^{3/2}})\|\widetilde{A}\|\|\widetilde{B}\|,\\ &=\widetilde{A}\widetilde{B}+\Delta W X_B^T+\Delta C=\widetilde{A}\widetilde{B}+F.\end{aligned}$$

Bounding $\|F\| \leqslant \gamma_c\|\widetilde{A}\|\|\widetilde{B}\|$, with $c=b+2r^{3/2}$, proves (3.10). We then replace $\widetilde{A}$ by $A+E_A$ and $\widetilde{B}$ by $B+E_B$ to obtain

$$\begin{aligned}\widehat{C}=AB+F+G, \quad &\|F\| \leqslant \gamma_c\|A\|\|B\|+\gamma_c\varepsilon\big(\beta_A\|B\|+\|A\|\beta_B+\varepsilon\beta_A\beta_B\big),\\ &\|G\|=\|E_AB+AE_B+E_AE_B\| \leqslant \varepsilon\big(\beta_A\|B\|+\|A\|\beta_B+\varepsilon\beta_A\beta_B\big),\end{aligned}$$

which yields (3.11). □

In the case $\beta_A=\|A\|$, $\beta_B=\|B\|$, the bound (3.11) simplifies to

$$\|\widehat{C}-AB\| \leqslant \big(2\varepsilon+\varepsilon^2+\gamma_c(1+\varepsilon)^2\big)\|A\|\|B\|, \tag{3.12}$$

which is similar to bound (3.7) from Lemma 3.2 for an LR matrix times a full matrix.

In the context of the BLR matrix LU factorization, computing products of LR matrices asymptotically represents the dominant cost. This has generated interest in strategies seeking to reduce this cost. For instance, in Amestoy *et al.* (2017) it is proposed that the middle product $M=Y_A^TY_B$ should be recompressed, that is, we should compute an LR approximation $\widetilde{M}=X_MY_M^T\approx M$. Indeed, matrix M often has a lower numerical rank than A and B, because even though $\sigma_{\min}(\widetilde{A})$ and $\sigma_{\min}(\widetilde{B})$ are both larger than ε, $\sigma_{\min}(M)=\sigma_{\min}(\widetilde{A}\widetilde{B})$ can potentially be as small as ε^2.

The motivation for representation (2.3) is now clear: the above only holds when $\widetilde{A}=X_AY_A^T$ and $\widetilde{B}=Y_BX_B^T$, that is, when the matrices with orthonormal columns X_A and X_B are on the outside of the product $\widetilde{A}\widetilde{B}$.

The cost of computing the product $C=\widetilde{A}\widetilde{B}$ can thus be reduced by replacing M by an LR matrix. The following lemma bounds the additional errors introduced in doing so.

LEMMA 3.4 (LR matrix times LR matrix with intermediate recompression) Let $A, B, \widetilde{A}, \widetilde{B}$ be defined as in Lemma 3.3 and let $M = Y_A^T Y_B$. If the product $C = \widetilde{A}\widetilde{B}$ is computed as $(X_A X_M)(Y_M^T X_B^T)$, where $X_M, Y_M \in \mathbb{R}^{r\times r}$, X_M has orthonormal columns, and $\widetilde{M} = X_M Y_M^T$ satisfies $\|M - \widetilde{M}\| \leqslant \varepsilon\beta_M$, then the computed $\widehat{C}$ satisfies

$$\|\widehat{C} - \widetilde{A}\widetilde{B}\| \leqslant \varepsilon(1+\gamma_c)\beta_M + \gamma_c\|\widetilde{A}\|\|\widetilde{B}\| + O(u^2), \tag{3.13}$$

where $c = b + r^2 + 2r^{3/2}$, and therefore

$$\|\widehat{C} - AB\| \leqslant \gamma_c\|A\|\|B\| + \varepsilon(1+\gamma_c)\big(\beta_M + \beta_A\|B\| + \|A\|\beta_B + \varepsilon\beta_A\beta_B\big) + O(u^2). \tag{3.14}$$

Proof. In order to do the compression, we first compute $M = Y_A^T Y_B$, obtaining $\widehat{M}$ satisfying

$$\widehat{M} = M + \Delta M, \quad \|\Delta M\| \leqslant \gamma_b\|Y_A\|\|Y_B\| = \gamma_b\|\widetilde{A}\|\|\widetilde{B}\|.$$

Then, we compute an LR approximation to $\widehat{M}$ satisfying $\widetilde{M} = \widehat{M} + E_M$, $\|E_M\| \leqslant \varepsilon\beta_M$. Let $W = X_A X_M$ and $Z = Y_M^T X_B^T$; the computed $\widehat{W}$ and $\widehat{Z}$ satisfy

$$\widehat{W} = X_A X_M + \Delta W, \quad \|\Delta W\| \leqslant \gamma_r\|X_A\|\|X_M\| \leqslant \gamma_{r^2}, \tag{3.15}$$

$$\widehat{Z} = Y_M^T X_B^T + \Delta Z, \quad \|\Delta Z\| \leqslant \gamma_r\|X_B\|\|Y_M\| \leqslant \gamma_{r^{3/2}}\|\widetilde{M}\|. \tag{3.16}$$

Finally, let $C = \widehat{W}\widehat{Z}$; the computed $\widehat{C}$ satisfies

$$\begin{aligned}
\widehat{C} &= \widehat{W}\widehat{Z} + F, \quad \|F\| \leqslant \gamma_r\|\widehat{W}\|\|\widehat{Z}\| \leqslant \gamma_{r^{3/2}}\|\widetilde{M}\| + O(u^2),\\
&= X_A X_M Y_M^T X_B^T + \Delta W Y_M^T X_B^T + X_A X_M \Delta Z + \Delta W \Delta Z + F\\
&= X_A\big(\widehat{M} + E_M\big)X_B^T + \Delta W Y_M^T X_B^T + X_A X_M \Delta Z + \Delta W \Delta Z + F\\
&= \widetilde{A}\widetilde{B} + X_A\big(\Delta M + E_M\big)X_B^T + \Delta W Y_M^T X_B^T + X_A X_M \Delta Z + \Delta W \Delta Z + F\\
&= \widetilde{A}\widetilde{B} + \Delta C, \quad \|\Delta C\| \leqslant \varepsilon(1+\gamma_{r^2+2r^{3/2}})\beta_M + \gamma_c\|\widetilde{A}\|\|\widetilde{B}\| + O(u^2),
\end{aligned}$$

where $c = b + r^2 + 2r^{3/2}$. We obtain the slightly weaker bound (3.13) by bounding $\gamma_{r^2+2r^{3/2}}$ by γ_c. Replacing $\widetilde{A}$ by $A + E_A$ and $\widetilde{B}$ by $B + E_B$ yields (3.14) and concludes the proof. □

The introduction of an intermediate low-rank approximation has two consequences. First, the constant c is slightly larger in Lemma 3.4 than in Lemma 3.3, a consequence of computing one more product than previously (four instead of three products). Second, and more importantly, a new low-rank truncation term $\varepsilon\beta_M$ is introduced and is dominant in the bound (3.13) on $\|\widehat{C} - \widetilde{A}\widetilde{B}\|$. However, in the overall bound (3.14) on $\|\widehat{C} - AB\|$, $\varepsilon\beta_M$ is just one more term that adds to $\varepsilon\big(\beta_A\|B\| + \|A\|\beta_A\big)$. For instance, in the case $\beta_A = \|A\|$, $\beta_B = \|B\|$, and $\beta_M = \|A\|\|B\|$, the bound (3.14) simplifies to

$$\|\widehat{C} - AB\| \leqslant \big(3\varepsilon + \varepsilon^2 + \gamma_c(1 + 3\varepsilon + \varepsilon^2)\big)\|A\|\|B\| + O(u^2),$$

which is roughly $3\varepsilon\|A\|\|B\|$, compared with roughly $2\varepsilon\|A\|\|B\|$ in the bound (3.12).

3.3 *Triangular system with LR right-hand side*

We now consider the solution of a triangular system where the right-hand side is an LR matrix $\widetilde{B} = YX^T$, which will be needed to analyze the UCF factorization in section 4.2. Note that we consider $\widetilde{B}$ of the form YX^T rather than XY^T, where X has orthonormal columns, because this is the form that arises in the UCF factorization.

LEMMA 3.5 (Triangular system with LR right-hand side) Let $T \in \mathbb{R}^{b\times b}$ be a triangular matrix and let $B \in \mathbb{R}^{b\times m}$ such that the LR matrix $\widetilde{B} = YX^T$, where X has orthonormal columns, satisfies $\|B-\widetilde{B}\| \leqslant \varepsilon\beta$ for some $\beta > 0$. If the solution to the triangular system $T\widetilde{Z} = \widetilde{B}$ is obtained as the LR matrix $\widetilde{Z} = \widehat{W}X^T$, where $\widehat{W}$ is the computed solution ot the system $TW = Y$, then $\widetilde{Z}$ satisfies

$$T\widetilde{Z} = \widetilde{B} + \Delta\widetilde{B}, \quad \|\Delta\widetilde{B}\| \leqslant \gamma_b\|T\|\|\widetilde{Z}\| \tag{3.17}$$

and therefore

$$T\widetilde{Z} = B + \Delta B, \quad \|\Delta B\| \leqslant \gamma_b\|T\|\|\widetilde{Z}\| + \varepsilon\beta. \tag{3.18}$$

Proof. By Lemma 2.2, the computed solution $\widehat{W}$ to the triangular system $TW = Y$ satisfies $T\widehat{W} = Y + \Delta Y$, where $\|\Delta Y\| \leqslant \gamma_b\|T\|\|\widehat{W}\|$. Defining $\widetilde{Z} = \widehat{W}X^T$, we obtain $T\widetilde{Z} = \widetilde{B} + \Delta\widetilde{B}$, with $\|\Delta\widetilde{B}\| = \|\Delta Y\| \leqslant \gamma_b\|T\|\|\widetilde{Z}\|$, proving (3.17). Replacing $\widetilde{B}$ by $B+E$, with $\|E\| \leqslant \varepsilon\beta$, yields (3.18) and concludes the proof. □

Note that the assumption that X has orthonormal columns is important, as otherwise we could only prove that $\|\Delta\widetilde{B}\| \leqslant \gamma_b\|T\|\|\widehat{W}\|\|X\|$, reflecting the possibility of cancellation if $\|\widetilde{Z}\| \ll \|\widehat{W}\|\|X\|$.

Also note that the solution $\widetilde{Z}$ is given in LR form. If it is needed as a full matrix instead, we must compute the product $\widehat{W}X^T$ and the analysis must therefore be adapted to take into account the errors introduced by this additional computation.

4. Rounding error analysis of solving BLR linear systems by LU factorization

We now turn to BLR matrices, building on the results on LR matrices obtained in the previous section. We first analyze two LU factorization algorithms in sections 4.1 and 4.2, and then turn to their use to solve to BLR linear systems in section 4.3.

Given a BLR matrix A, we define its *BLR rank* as the largest of the ranks of any of its off-diagonal blocks. Throughout this section, we denote by r the BLR rank of the LU factors of A (which is in general larger than the BLR rank of A). We emphasize that the compression of the blocks is controlled solely by the ε and β_{ij} parameters defined in sect. 2. The value of r is not tunable but rather depends on the matrix and on the choice of threshold.

4.1 *BLR matrix LU factorization: UFC algorithm*

To compute the LU factorization of BLR matrices, the classical partitioned LU factorization of full matrices must be adapted by incorporating the compressions of the blocks into LR matrices. Several algorithms have been distinguished depending on when this compression step is performed. In this section, we analyze the UFC algorithm (standing for update, factor, compress) described in Algorithm 4.1. Another algorithm, referred to as UCF (update, compress, factor), is analyzed in section 4.2.

The algorithm is written in a left-looking fashion: at step k of the UFC algorithm, the kth block-row and block-column are first updated (lines 4–7) using the BLR LU factors "to the left" (note that at step $k=1$, the update step therefore does not do anything). Then, this block-row and block-column are factored (lines 9–12) before being finally compressed (lines 14–17). We note that the right-looking variant of this algorithm, used in some of the literature, is numerically equivalent since the same operations are performed in a different order.

Note that the compression and factorization steps are interlaced, unlike other variants such as the CUF algorithm (discussed at the end of section 4.2), where the entire matrix A is initially compressed.

We recall that, for all the experiments on the BLR LU factorization, we measure the backward error by solving a linear system as explained in section 2.3.

Algorithm 4.1 BLR LU factorization: UFC algorithm.

```
1: {Input: a $p \times p$ block matrix $A$. Output: its BLR LU factors $\widetilde{L}$ and $\widetilde{U}$.}
2: for $k = 1$ to $p$ do
3:   UPDATE:
4:       $A_{kk} \leftarrow A_{kk} - \sum_{j=1}^{k-1} \widetilde{L}_{kj}\widetilde{U}_{jk}$.
5:       for $i = k+1$ to $p$ do
6:           $A_{ik} \leftarrow A_{ik} - \sum_{j=1}^{k-1} \widetilde{L}_{ij}\widetilde{U}_{jk}$ and $A_{ki} \leftarrow A_{ki} - \sum_{j=1}^{k-1} \widetilde{L}_{kj}\widetilde{U}_{ji}$.
7:       end for
8:   FACTOR:
9:       Compute the LU factorization $L_{kk}U_{kk} = A_{kk}$.
10:      for $i = k+1$ to $p$ do
11:          Solve $L_{ik}U_{kk} = A_{ik}$ for $L_{ik}$ and $L_{kk}U_{ki} = A_{ki}$ for $U_{ki}$.
12:      end for
13:  COMPRESS:
14:      Set $\widetilde{L}_{kk} = L_{kk}$ and $\widetilde{U}_{kk} = U_{kk}$.
15:      for $i = k+1$ to $p$ do
16:          Compute LR approximations $\widetilde{L}_{ik} \approx L_{ik}$ and $\widetilde{U}_{ki} \approx U_{ki}$.
17:      end for
18: end for
```

The next theorem analyzes the UFC algorithm. We recall that the β_{ik} parameters control the type of threshold (global or local) and are defined by (2.4).

THEOREM 4.1 (BLR LU factorization: UFC algorithm) Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular matrix partitioned into p^2 blocks of order b. If Algorithm 4.1 runs to completion it produces computed BLR LU factors $\widetilde{L}$ and $\widetilde{U}$ of A satisfying

$$A = \widetilde{L}\widetilde{U} + \Delta A + F + G,$$

with $\|\Delta A\| \leqslant \gamma_p \|A\|$ and $\|F\| \leqslant \gamma_c \|\widetilde{L}\|\|\widetilde{U}\| + O(u\varepsilon)$, where $c = b + 2r^{3/2} + p$ and

$$G_{ik} = \begin{cases} E_{ik}\widetilde{U}_{kk} & i > k, \\ 0 & i = k, \\ \widetilde{L}_{ii}E_{ik} & i < k, \end{cases} \qquad \|E_{ik}\| \leqslant \varepsilon\beta_{ik}.$$

Proof. The (i,k) block of the L factor is computed by solving

$$L_{ik}\widetilde{U}_{kk} = R_{ik}, \quad R_{ik} = A_{ik} - \sum_{j=1}^{k-1} \widetilde{L}_{ij}\widetilde{U}_{jk}, \quad i > k, \tag{4.1}$$

where $\widetilde{L}$ and $\widetilde{U}$ are the partial BLR LU factors computed in the previous $k-1$ steps (line 16 of Algorithm 4.1). Let $R_{ik}^{(j)} = \widetilde{L}_{ij}\widetilde{U}_{jk}$; by (3.10), the computed $\widehat{R}_{ik}^{(j)}$ satisfies

$$\widehat{R}_{ik}^{(j)} = \widetilde{L}_{ij}\widetilde{U}_{jk} + \Delta R_{ik}^{(j)}, \quad \|\Delta R_{ik}^{(j)}\| \leqslant \gamma_d \|\widetilde{L}_{ij}\|\|\widetilde{U}_{jk}\| + O(u^2) \tag{4.2}$$

with $d = b + 2r^{3/2}$. The computed $\widehat{R}_{ik}$ then satisfies

$$\widehat{R}_{ik} = A_{ik} \circ (J + \Theta_k) - \sum_{j=1}^{k-1} \widehat{R}_{ik}^{(j)} \circ (J + \Theta_j), \quad |\Theta_j| \leqslant \gamma_p J, \tag{4.3}$$

where J is the matrix of ones, $\circ$ denotes the Hadamard product ($A \circ B = (a_{ij} b_{ij})$), and where the inequality $|\Theta_j| \leqslant \gamma_p J$ holds componentwise (the constant p comes from the additions of the $\widehat{R}_{ik}^{(j)}$ matrices, of which there are at most p). By (2.9) we have

$$\widehat{L}_{ik} \widetilde{U}_{kk} = \widehat{R}_{ik} + \Delta R_{ik}^{(k)}, \quad \|\Delta R_{ik}^{(k)}\| \leqslant \gamma_b \|\widehat{L}_{ik}\| \|\widetilde{U}_{kk}\|. \tag{4.4}$$

After compression, we finally obtain the BLR factor $\widetilde{L}_{ik} = \widehat{L}_{ik} + E_{ik}$, with $\|E_{ik}\| \leqslant \varepsilon \beta_{ik}$. Combining (4.2), (4.3), and (4.4) gives

$$A_{ik} \circ (J + \Theta_k) - \sum_{j=1}^{k} \widetilde{L}_{ij} \widetilde{U}_{jk} = \sum_{j=1}^{k-1} \Delta R_{ik}^{(j)} \circ (J + \Theta_j) - \Delta R_{ik}^{(k)} - E_{ik} \widetilde{U}_{kk}.$$

We therefore obtain

$$A_{ik} - \sum_{j=1}^{k} \widetilde{L}_{ij} \widetilde{U}_{jk} = \Delta A_{ik} + F_{ik} + G_{ik}, \quad \|\Delta A_{ik}\| \leqslant \gamma_p \|A_{ik}\|, \tag{4.5}$$

$$\begin{aligned} \|F_{ik}\| &\leqslant \gamma_{d+p+O(u)} \left(\sum_{j=1}^{k-1} \|\widetilde{L}_{ij}\| \|\widetilde{U}_{jk}\| \right) + \gamma_b \|\widehat{L}_{ik}\| \|\widetilde{U}_{kk}\| \\ &\leqslant \gamma_{d+p+O(\varepsilon)} \left(\sum_{j=1}^{k} \|\widetilde{L}_{ij}\| \|\widetilde{U}_{jk}\| \right), \end{aligned} \tag{4.6}$$

$$\|G_{ik}\| \leqslant \|E_{ik} \widetilde{U}_{kk}\| \leqslant \varepsilon \beta_{ik} \|\widetilde{U}_{kk}\|, \quad i > k. \tag{4.7}$$

This concludes the blocks for $i > k$. For $i = k$, L_{kk} is determined together with U_{kk} on line 9 of Algorithm 4.1, and by Lemma 2.3 we have $\|\widetilde{L}_{kk} \widetilde{U}_{kk} - \widehat{R}_{kk}\| \leqslant \gamma_b \|\widetilde{L}_{kk}\| \|\widetilde{U}_{kk}\|$. Therefore (4.4) holds for $i = k$, too, and hence so does (4.5), with the same bound (4.6) on $\|F_{kk}\|$ and with $G_{kk} = 0$, because diagonal blocks are not compressed. Finally, the case $i < k$ is analogous to the case $i > k$ and yields (4.5) where the bound (4.6) becomes

$$\|F_{ik}\| \leqslant \gamma_{d+p+O(\varepsilon)} \left(\sum_{j=1}^{i} \|\widetilde{L}_{ij}\| \|\widetilde{U}_{jk}\| \right) \tag{4.8}$$

and with

$$\|G_{ik}\| \leqslant \varepsilon \beta_{ik} \|\widetilde{L}_{ii}\|, \quad i < k. \tag{4.9}$$

We have therefore proved that $A - \widetilde{L}\widetilde{U} = \Delta A + F + G$, where $G_{kk} = 0$ and blockwise bounds on $\|\Delta A\|$, $\|F\|$, and $\|G\|$ are given by (4.5)–(4.9). It thus remains to derive global bounds. Bounding $\|G\|$ is delayed to section 4.1.1, as it depends on the choice of the β_{ik} parameters. The bound $\|\Delta A\| \leqslant \gamma_p \|A\|$

trivially holds. Finally, for matrix F, the Cauchy–Schwarz inequality gives

$$\begin{aligned}\|F\| &\leqslant \gamma_{d+p+O(\varepsilon)} \left(\sum_{i=1}^{p} \sum_{k=1}^{p} \left(\sum_{j=1}^{\min(i,k)} \|\widetilde{L}_{ij}\| \|\widetilde{U}_{jk}\| \right)^2 \right)^{1/2} \\ &\leqslant \gamma_{d+p+O(\varepsilon)} \left(\sum_{i=1}^{p} \sum_{j=1}^{i} \|\widetilde{L}_{ij}\|^2 \sum_{k=1}^{p} \sum_{j=1}^{k} \|\widetilde{U}_{jk}\|^2 \right)^{1/2} \\ &\leqslant \gamma_{d+p+O(\varepsilon)} \left(\left(\sum_{i=1}^{p} \|\widetilde{L}_i\|^2 \sum_{k=1}^{p} \|\widetilde{U}_k\|^2 \right) \right)^{1/2} \\ &\leqslant \gamma_{d+p} \|\widetilde{L}\| \|\widetilde{U}\| + O(u\varepsilon), \end{aligned} \tag{4.10}$$

where $\widetilde{L}_i$ and $\widetilde{U}_k$ denote the ith block-row of $\widetilde{L}$ and kth block-column of $\widetilde{U}$, respectively. $\square$

Theorem 4.1 yields a backward error bound that is comparable with the other results obtained so far: we obtain a term $\|\Delta A\| + \|F\|$ proportional to the unit roundoff u, and a term $\|G\|$ depending on the low-rank threshold ε, the latter likely dominating the former. However, before further commenting on the significance of this bound, we must compute a global bound on $\|G\|$ by examining several possible choices for the β_{ik} parameters.

4.1.1 *Bounding $\|G\|$ and choice of β_{ik}.* The blockwise bounds on $\|G_{ik}\|$ given in Theorem 4.1 yield the global bound

$$\|G\|^2 \leqslant \varepsilon^2 \sum_{k=1}^{p} \left(\|\widetilde{L}_{kk}\|^2 \sum_{i=1}^{k-1} \beta_{ik}^2 + \|\widetilde{U}_{kk}\|^2 \sum_{i=k+1}^{p} \beta_{ik}^2 \right).$$

With a local threshold $\beta_{ik} = \|A_{ik}\|$, we have

$$\begin{aligned}\|G\| &\leqslant \varepsilon \left(\sum_{k=1}^{p} \max\left(\|\widetilde{L}_{kk}\|, \|\widetilde{U}_{kk}\|\right)^2 \sum_{i\neq k}^{p} \|A_{ik}\|^2 \right)^{1/2} \\ &\leqslant \varepsilon \max_{k=1:p} \left(\max\left(\|\widetilde{L}_{kk}\|, \|\widetilde{U}_{kk}\|\right) \right) \|A\|. \end{aligned} \tag{4.11}$$

On the other hand, a global threshold $\beta_{ik} = \|A\|$ yields

$$\begin{aligned}\|G\| &\leqslant \varepsilon \left(\sum_{k=1}^{p} \max\left(\|\widetilde{L}_{kk}\|, \|\widetilde{U}_{kk}\|\right)^2 \sum_{i\neq k}^{p} \|A\|^2 \right)^{1/2} \\ &\leqslant (p-1)^{1/2} \varepsilon \|D\| \|A\|, \end{aligned} \tag{4.12}$$

where D is the diagonal matrix defined by $D_{kk} = \max\left(\|\widetilde{L}_{kk}\|, \|\widetilde{U}_{kk}\|\right)$. Bound (4.12) is thus up to a factor $\left(p(p-1)\right)^{1/2} \approx p$ times larger than (4.11).

Two main observations can be made. First, the use of a global threshold leads to a bound about p times larger than with a local threshold. This is illustrated experimentally in Figure 4.1a, which shows that for a fixed ε, a global threshold yields a backward error about two orders of magnitude larger than with a local threshold. However, this experiment is not sufficient to determine which type of threshold is better: we must determine whether the increased error of the global threshold pays off by improving the compression and thus reducing the number of flops for the computation. To answer this question we must assess which type of threshold achieves the best flops–accuracy tradeoff. To do so, we perform the following experiment in Figure 4.1b: taking several values of ε, we plot the error (2.11) as a function of the corresponding number of flops required to compute the factorization. This experiment shows that a global threshold achieves the best tradeoff, since it is always closer to the bottom left corner of the plot: that is, for the same accuracy as a local threshold, a global threshold performs fewer flops or, equivalently, for the same number of flops as a local threshold, a global threshold delivers a more accurate result. A global threshold is therefore the best choice for this matrix and algorithm. We will show in section 5 this remains the case for LU factorization of a wide range of BLR matrices.

The second important observation is that the error depends on the norm of the diagonal blocks of the LU factors ($\max(\|\widetilde{L}_{kk}\|, \|\widetilde{U}_{kk}\|)$ in (4.11) and $\|D\|$ in (4.12)). This is due to the compress step (lines 14–17 of Algorithm 4.1) being performed after the factor step (lines 9–12). Indeed, the blocks that are compressed are equal to $A_{ik}U_{kk}^{-1}$ and $L_{kk}^{-1}A_{ki}$ and so their norms depend on those of the diagonal blocks. This property of the UFC factorization is undesirable because the L and U factors are often not scaled comparably: the entries of L are bounded by 1 with partial pivoting, whereas those of U are scaled similarly to those of A. As a consequence, the average ranks of the blocks differ depending on whether they belong to the L or U factors, even for symmetric matrices where $U = L^T$. For example, on the Poisson matrix used in Figure 4.1, which is symmetric, the L factor requires up to 15% more storage than the U factor.

There exist several solutions to avoid the dependence of the ranks of the BLR LU factors and the backward error on the norms of the diagonal blocks. One solution consists in scaling the β_{ik} differently for the L and U factors: specifically, setting $\beta_{ik} = \|A_{ik}\|/\|\widetilde{U}_{kk}\|$ (if $i > k$) and $\beta_{ik} = \|A_{ik}\|/\|\widetilde{L}_{ii}\|$ (if $i < k$) changes bound (4.11) to

$$\|G\| \leqslant \varepsilon \|A\| \tag{4.13}$$

and setting $\beta_{ik} = \|A\|/\|\widetilde{U}_{kk}\|$ (if $i > k$) and $\beta_{ik} = \|A\|/\|\widetilde{L}_{ii}\|$ (if $i < k$) changes bound (4.12) to

$$\|G\| \leqslant p\varepsilon \|A\|. \tag{4.14}$$

This scaling of the threshold yields similar ranks in both L and U. Even though Figure 4.1b shows that this strategy does not achieve a visibly better flops–accuracy tradeoff, in the following we consider scaled thresholds since they simplify the bounds.

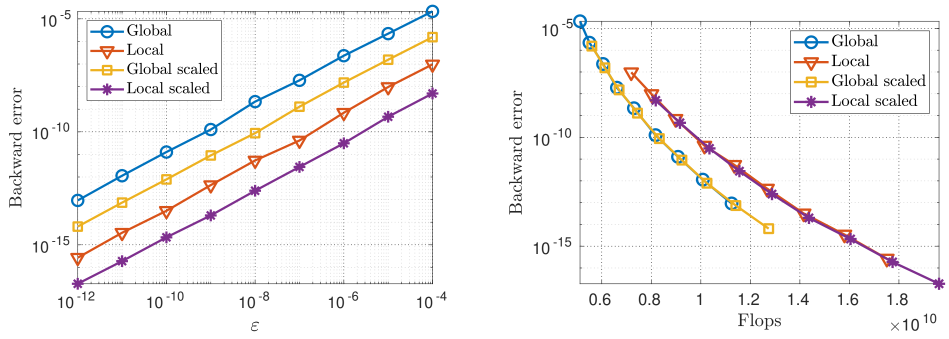
Note that there are alternative strategies to scaling the threshold. For example computing an LDU rather than LU factorization, where both the L and U factors have entries bounded by one. Interestingly, as we show in section 4.2, another solution is to perform a UCF factorization, which avoids this issue by compressing the blocks before factorizing them.

4.1.2 *General comments on Theorem 4.1.* Outside the technical discussion of the previous section on how to choose the β_{ik} parameters, some higher level conclusions can be drawn from Theorem 4.1, which we summarize in the following result.

COROLLARY 4.1 Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular matrix partitioned into p^2 blocks of order b. If Algorithm 4.1 runs to completion, it produces BLR LU factors $\widetilde{L}$ and $\widetilde{U}$ of A satisfying

$$A = \widetilde{L}\widetilde{U} + \Delta A, \quad \|\Delta A\| \leqslant \big(\xi_p \varepsilon + \gamma_p\big)\|A\| + \gamma_c \|\widetilde{L}\|\|\widetilde{U}\| + O(u\varepsilon), \tag{4.15}$$

where $c = b + 2r^{3/2} + p$, and $\xi_p = 1$ or $\xi_p = p$ for a scaled local or global threshold, respectively.



(A) Backward error for varying ε threshold.

(B) Flops–accuracy tradeoff. Each point corresponds to a different ε threshold.

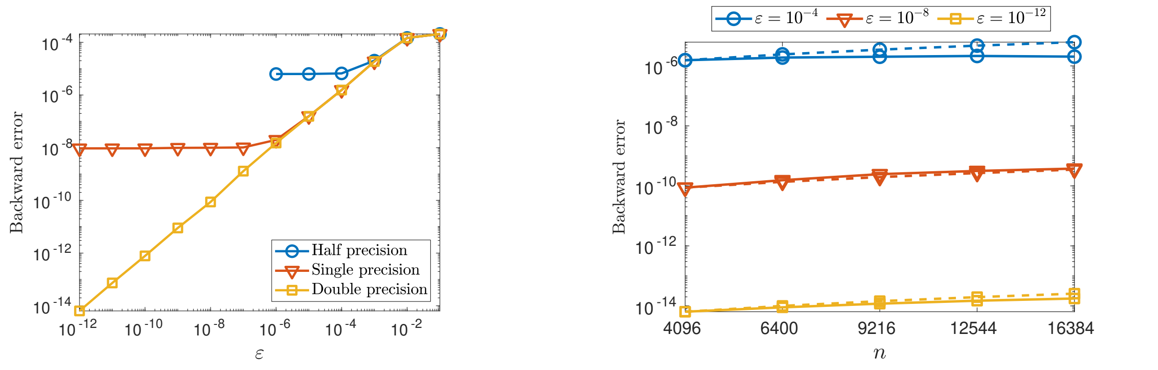
FIG. 4.1. *Comparison of several choices for β_{ik} in the BLR UFC factorization (Algorithm 4.1) of a Poisson matrix of order $n = 4096$. Local: $\beta_{ik} = \|A_{ik}\|$; local scaled: $\beta_{ik} = \|A_{ik}\|/\|\widetilde{U}_{kk}\|$ $(i > k)$ and $\beta_{ik} = \|A_{ik}\|/\|\widetilde{L}_{ii}\|$ $(i < k)$; global: $\beta_{ik} = \|A\|$; global scaled: $\beta_{ik} = \|A\|/\|\widetilde{U}_{kk}\|$ $(i > k)$ and $\beta_{ik} = \|A\|/\|\widetilde{L}_{ii}\|$ $(i < k)$.*

Proof. Directly follows from Theorem 4.1, (4.10), (4.13), and (4.14). □

Corollary 4.1 states that the backward error $\|A - \widetilde{L}\widetilde{U}\|$ is of order $O(\varepsilon\|A\| + u\|\widetilde{L}\|\|\widetilde{U}\|)$. The first term corresponds to the low-rank truncations errors and the second term to the floating-point errors. If we set $\varepsilon = 0$, we recover the backward error bound (2.10) for classical LU factorization (with a slightly higher constant). If we set $u = 0$, we obtain a bound on the error introduced by BLR approximations in exact arithmetic. In the general case, since $u \ll \varepsilon$, the low-rank error term dominates if $\|\widetilde{L}\|\|\widetilde{U}\|$ is not too large compared with $\|A\|$. For a stable LU factorization, using for example partial pivoting, $\|\widetilde{L}\|\|\widetilde{U}\|$ is bounded and therefore the BLR factorization is also stable. This is the main conclusion drawn from Corollary 4.1: it proves the long conjectured rule of thumb that the backward error for the BLR LU factorization is proportional to the low-rank threshold ε. This is a very desirable theoretical guarantee that can now be given to the users of BLR solvers. This is illustrated by the numerical experiments on a Poisson matrix reported in Figure 4.2a, which additionally show that the unit roundoff u has no impact on the error as long as $u \ll \varepsilon$. We will verify experimentally that this crucial result holds for a wide range of matrices in section 5.

The low-rank error term $\xi_p \varepsilon\|A\|$ grows at most linearly with the number of blocks p. Figure 4.2b illustrates that a roughly linear growth (indicated by the dashed lines) is indeed observed in practice when using a global threshold (except for the largest of the tested thresholds, $\varepsilon = 10^{-4}$, for which the error growth is sublinear). This seems quite acceptable, especially considering that the constant in the error bound for traditional LU factorization is usually of order n^3 (Higham, 2002, Thm. 9.5).

We also briefly comment on the use of fast matrix operations, such as Strassen's algorithm. Similarly to the LR matrix algorithms analyzed in section 3.1, fast matrix algorithms are especially attractive with BLR matrices since only normwise stability is expected, and because they only affect the floating-point error term, which is negligible compared with the low-rank error term for large enough ε. Our analysis therefore theoretically explains why the stability of the BLR factorization is much less sensitive to the use of fast matrix arithmetic, as experimentally observed in Jeannerod *et al.* (2019). We mention that the algorithm proposed in Jeannerod *et al.* (2019) recasts the operations so as to work on matrices of larger dimensions, which allows for exploiting fast matrix arithmetic more efficiently. The error analysis of this new algorithm is outside our scope, but we expect it to retain the same backward stability as the



(A) Error for varying low-rank thresholds ε and floating-point precisions.

(B) Error for increasing n. Dashed lines indicate a linear growth.

FIG. 4.2. *Backward error* (2.11) *for computing the BLR LU factorization of a Poisson matrix of order* $n = 4096$ *with the UFC algorithm (Algorithm* 4.1*) and with a global threshold.*

algorithms analyzed here.

More generally, any numerical instabilities coming from the algorithm (such as when no pivoting is employed, leading to a large growth factor) may potentially be hidden behind the low-rank error term, if ε is large enough compared with u. We however wish to emphasize that, with pivoting, the BLR LU factorization is numerically stable even if $\varepsilon \sim u$. Indeed, the error bound (4.15) is not any worse than standard LU factorization (modulo a slightly larger constant c).

4.1.3 *Impact of intermediate recompressions.*

We now discuss the impact on the accuracy of performing intermediate recompressions during the update step, as described and analyzed in Lemma 3.4.

Adapting the proof of Theorem 4.1 to the use of intermediate recompressions is straightforward. It suffices to invoke Lemma 3.4 instead of Lemma 3.3, which changes the expression for R_{ik} in (4.1) to

$$R_{ik} = A_{ik} - \sum_{j=1}^{k-1} \left(\widetilde{L}_{ij} \widetilde{U}_{jk} + H_{ik}^{(j)} \right),$$

which has an extra term $H_{ik}^{(j)}$ satisfying $\|H_{ik}^{(j)}\| \leqslant \varepsilon \beta_{ik}^{(j)}$. For simplicity, let us consider the same choice $\beta_{ik}^{(j)} = \beta_{ik}^{H}$ for all j. The rest of the proof carries over and we obtain $A = \widetilde{L}\widetilde{U} + \Delta A + F + G$, where we now have

$$G_{ik} = \begin{cases} E_{ik}\widetilde{U}_{kk} + H_{ik}, & i > k, \\ 0, & i = k, \\ \widetilde{L}_{ii}E_{ik} + H_{ik}, & i < k, \end{cases} \qquad \|E_{ik}\| \leqslant \varepsilon \beta_{ik}, \quad \|H_{ik}\| \leqslant (\min(i,k) - 1)\varepsilon \beta_{ik}^{H}.$$

To proceed further we must consider specific choices for the β_{ik} and β_{ik}^{H} parameters, as in section 4.1.1. For a scaled local threshold (4.13) and for $\beta_{ik}^{H} = \varepsilon \|A_{ik}\|$, we have $\|G_{ik}\| \leqslant \min(i,k)\varepsilon\|A_{ik}\|$, whereas for a scaled global threshold (4.14) and $\beta_{ik}^{H} = \varepsilon\|A\|$, we obtain instead $\|G_{ik}\| \leqslant \min(i,k)\varepsilon\|A\|$.

TABLE 4.1. *Expression of ξ_p in Theorems 4.2, 4.3, and 4.5, depending on whether a local or global threshold is used, and on whether intermediate recompressions are performed during the LU factorization.*

	Local threshold	Global threshold
Without recompressions	1	p
With recompressions	p	$p^2/\sqrt{6}$

Tedious but straightforward computations lead to the bound $\|G\| \leqslant \xi_p \varepsilon \|A\|$, where $\xi_p = p$ and $\xi_p = p^2/\sqrt{6}$ for a local and global threshold, respectively. The low-rank error is therefore a factor roughly p times larger when intermediate recompressions are performed than when they are not, for both local and global thresholds.

THEOREM 4.2 Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular matrix partitioned into p^2 blocks of order b. If Algorithm 4.1 runs to completion, it produces BLR LU factors $\widetilde{L}$ and $\widetilde{U}$ of A satisfying

$$A = \widetilde{L}\widetilde{U} + \Delta A, \quad \|\Delta A\| \leqslant \big(\xi_p \varepsilon + \gamma_p\big)\|A\| + \gamma_c \|\widetilde{L}\|\|\widetilde{U}\| + O(u\varepsilon), \tag{4.16}$$

where $c = b + 2r^{3/2} + p$, and where ξ_p is given in Table 4.1.

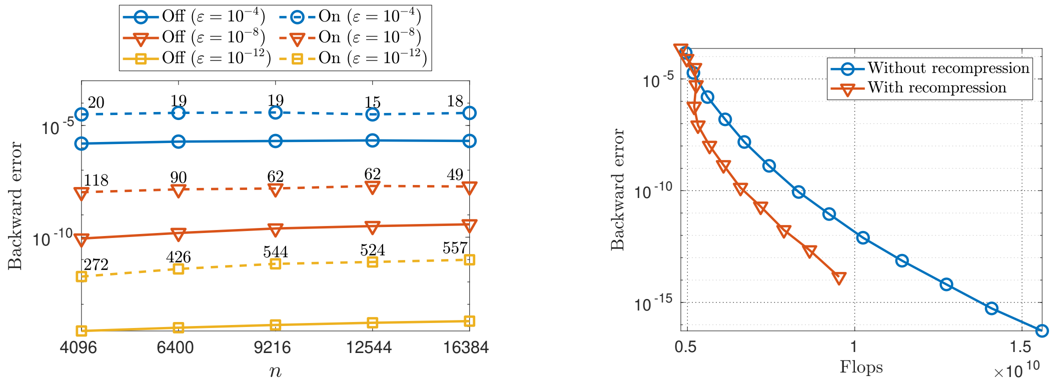
We illustrate the above analysis with some numerical experiments in Figure 4.3, using a scaled global threshold (results with a local threshold are similar and omitted). In Figure 4.3a, we compare the error growth with and without recompressions. Recompression increases the error by a noticeable factor; however, this factor increases relatively slowly with n. The error growth of the factorization with recompression therefore remains contained. To determine whether this increased error pays off in terms of flops, we plot in Figure 4.3b the error as a function of the flop count for the factorization for several values of ε. Clearly, for this Poisson matrix, the strategy using recompressions achieves a better tradeoff. We will show that this remains true for a wide range of matrices in section 5.

We mention that there exist more advanced recompression strategies that have been described in detail in Mary (2017, Chap. 3). Their rounding error analysis is outside our scope, although we expect them to behave comparably to the simple recompression strategy analyzed by Lemma 3.4.

4.2 *BLR matrix LU factorization: UCF algorithm*

In the UFC algorithm (Algorithm 4.1), the compress step is performed after the factor step and thus the latter does not exploit the LR property of the blocks. The UCF algorithm, described in Algorithm 4.2, is based on the idea of performing the compress step earlier, before the factor step, so that the off-diagonal blocks may be factored in LR form, as shown on line 15 and as analyzed in Lemma 3.5. This reduces the number of flops needed for the factor step, which is especially important because this step is asymptotically dominant in the UFC algorithm. The UCF algorithm is thus necessary to achieve an optimal complexity (Amestoy *et al.*, 2017).

However, the UCF algorithm has not yet been widely accepted as the method of choice, and some BLR solvers still use the UFC algorithm by default, such as MUMPS (Amestoy *et al.*, 2019b). There are two reasons for this. The first is that the impact on the accuracy of switching from UFC to UCF was not fully understood and quantified. The next theorem provides an answer to this open question. The second reason is related to numerical pivoting. Well-known pivoting strategies (e.g., partial pivoting, rook pivoting) require access to the entire row and/or column to be factored; however, in the UCF



(A) Backward error (2.11) for increasing n, depending on whether recompressions are "off" or "on". The numbers indicate the ratio between the corresponding errors.

(B) Flops–accuracy tradeoff. Each point corresponds to a different ε threshold.

FIG. 4.3. *Impact of intermediate recompressions on the backward error for a Poisson matrix of order $n = 4096$ with the UFC algorithm (Algorithm 4.1) and with a global threshold.*

Algorithm 4.2 BLR LU factorization: UCF algorithm.

```
1:  {**Input**: a $p \times p$ block matrix $A$. **Output**: its BLR LU factors $\widetilde{L}$ and $\widetilde{U}$.}
2:  for $k = 1$ to $p$ do
3:    UPDATE:
4:      $A_{kk} \leftarrow A_{kk} - \sum_{j=1}^{k-1} \widetilde{L}_{kj} \widetilde{U}_{jk}$.
5:      for $i = k+1$ to $p$ do
6:        $A_{ik} \leftarrow A_{ik} - \sum_{j=1}^{k-1} \widetilde{L}_{ij} \widetilde{U}_{jk}$ and $A_{ki} \leftarrow A_{ki} - \sum_{j=1}^{k-1} \widetilde{L}_{kj} \widetilde{U}_{ji}$.
7:      end for
8:    COMPRESS:
9:      for $i = k+1$ to $p$ do
10:       Compute LR approximations $\widetilde{A}_{ik} \approx A_{ik}$ and $\widetilde{A}_{ki} \approx A_{ki}$.
11:     end for
12:   FACTOR:
13:     Compute the LU factorization $\widetilde{L}_{kk} \widetilde{U}_{kk} = A_{kk}$.
14:     for $i = k+1$ to $p$ do
15:       Solve $\widetilde{L}_{ik} \widetilde{U}_{kk} = \widetilde{A}_{ik}$ for $\widetilde{L}_{ik}$ and $\widetilde{L}_{kk} \widetilde{U}_{ki} = \widetilde{A}_{ki}$ for $\widetilde{U}_{ki}$.
16:     end for
17: end for
```

algorithm, the blocks have already been compressed and so the entries of the original block are no longer available. Strategies estimating these entries based on the entries of the LR blocks have been proposed by Mary (2017) and appear to deliver satisfying results. This is however the object of ongoing research and is outside our scope.

THEOREM 4.3 (BLR LU factorization: UCF algorithm) Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular matrix partitioned into p^2 blocks of order b. If Algorithm 4.2 runs to completion, it produces BLR LU factors of A satisfying

$$A = \widetilde{L}\widetilde{U} + \Delta A, \quad \|\Delta A\| \leqslant \left(\xi_p \varepsilon + \gamma_p\right)\|A\| + \gamma_c \|\widetilde{L}\|\|\widetilde{U}\| + O(u\varepsilon), \tag{4.17}$$

where $c = b + 2r^{3/2} + p$, and where ξ_p is given in Table 4.1.

Proof. In contrast with (4.1) in the proof of Theorem 4.1, the (i,k) block of the L factor is now computed by solving instead

$$L_{ik}\widetilde{U}_{kk} = \widetilde{R}_{ik}, \quad i > k, \tag{4.18}$$

where $\widetilde{R}_{ik}$ is a LR approximation to $\widehat{R}_{ik}$ satisfying $\widetilde{R}_{ik} = \widehat{R}_{ik} + E_{ik}$, with $\|E_{ik}\| \leqslant \varepsilon\beta_{ik}$, and where $\widehat{R}_{ik}$ still satisfies (4.3). (4.18) takes the form of a triangular solve with an LR right-hand side, and thus by (3.17) we have

$$\widetilde{L}_{ik}\widetilde{U}_{kk} = \widetilde{R}_{ik} + \Delta R_{ik}^{(k)}, \quad \|\Delta R_{ik}^{(k)}\| \leqslant \gamma_b \|\widetilde{L}_{ik}\|\|\widetilde{U}_{kk}\|. \tag{4.19}$$

We therefore obtain

$$A_{ik} \circ (J + \Theta_k) - \sum_{j=1}^{k} \widetilde{L}_{ij}\widetilde{U}_{jk} = \sum_{j=1}^{k-1} \Delta R_{ik}^{(j)} - \Delta R_{ik}^{(k)} - E_{ik} = F_{ik} + G_{ik}, \tag{4.20}$$

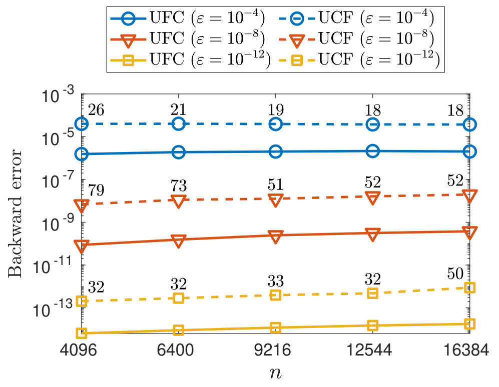
with the same bound (4.6) on $\|F_{ik}\|$ as for the UFC algorithm, but a different bound $\|G_{ik}\| \leqslant \varepsilon\beta_{ik}$ instead of (4.7), without the term $\|\widetilde{U}_{kk}\|$. This concludes the proof for the case $i > k$. The cases $i = k$ and $i < k$ are similar and overall we have $A - \widetilde{L}\widetilde{U} = \Delta A + F + G$, where the blockwise bounds on $\|\Delta A\|$ and $\|F\|$ are the same as those for the UFC algorithm, whereas $\|G_{ik}\| \leqslant \varepsilon\beta_{ik}$ for all i (with $\beta_{kk} = 0$). The bound $\|G\| \leqslant \xi_p \|A\|$ trivially follows with $\xi_p = 1$ or $\xi_p = p$ depending on whether $\beta_{ik} = \|A_{ik}\|$ or $\beta_{ik} = \|A\|$, respectively. □

A notable difference of the UCF algorithm is that, unlike the UFC algorithm, it does not depend on the norm of the diagonal blocks of the LU factors. The UCF algorithm thus avoids the issue of having different compression in the L and U factors, as discussed in section 4.1.1, and hence there is no reason to scale the threshold as suggested for the UFC algorithm (see (4.14)). The main conclusion to draw from Theorem 4.3 is therefore that the UCF algorithm satisfies the same error bound as the UFC one when the latter algorithm uses a scaled threshold (as in Theorem 4.2).

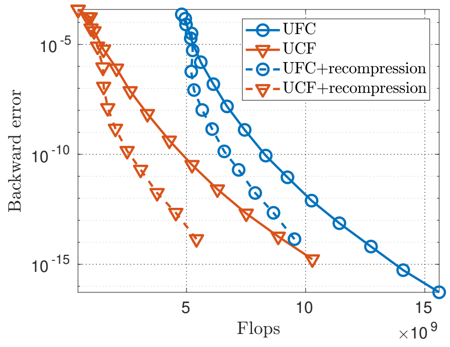
This conclusion is supported by numerical experiments in Figure 4.4. Figure 4.4a shows that the UFC and UCF algorithms yield similar errors on Poisson matrices, regardless of the matrix size n. Indeed, even though the UCF algorithm yields a backward error larger by a noticeable factor, this factor does not increase with n. Therefore, since the UCF algorithm achieves a lower flop count, it achieves a much better tradeoff than the UFC one, and this is illustrated in Figure 4.4b. These observations will be extended to a wider range of matrices in section 5.

Finally, we briefly comment on some other BLR LU factorization variants, for which we have performed similar analyses that we omit for the sake of conciseness.

The FUC algorithm (Mary, 2017) performs the compression step only after the completion of the LU factorization. This variant therefore only reduces the storage, not the flops, for performing the LU



(A) Backward error (2.11) for increasing n (without intermediate recompression). The numbers indicate the ratio between the errors for the UCF and UFC algorithms.



(B) Flops–accuracy tradeoff ($n = 4096$). Each point corresponds to a different ε threshold.

FIG. 4.4. *Comparison between UFC (Algorithm 4.1) and UCF (Algorithm 4.2) BLR LU factorizations for Poisson matrices.*

factorization. Its main advantage is to avoid the low-rank and the floating-point errors accumulating together, replacing the $O(u\varepsilon)$ term in the bounds of Theorems 4.1 and 4.3 by $O(u^2)$. This is however a negligible reduction of the error, and therefore the FUC variant is not competitive with the other variants.

Another widely used algorithm is the CUF variant (see, for example, Amestoy *et al.* (2017) and Pichon *et al.* (2018)), which compresses the entire matrix A and then computes its BLR LU factorization. The CUF algorithm is very similar to the UCF one, only differing in that, at line 6 of Algorithm 4.2, the blocks A_{ik} (and A_{ki}) are already in LR form. Therefore the result of the product of the LR factors $\widetilde{L}_{ij}\widetilde{U}_{jk}$ may be obtained directly as an LR matrix, avoiding the last product in Lemma 3.3: this just affects the constant in the error bound and we therefore conclude that the UCF and CUF algorithms achieve similar error bounds. Note that one particularity of the CUF algorithm is that the use of intermediate recompressions (section 4.1.3) is mandatory to contain the growth of the ranks of the BLR LU factors throughout the factorization.

4.3 *BLR linear systems*

We first analyze the solution of a triangular system $\widetilde{T}x = v$, where $\widetilde{T}$ is a BLR matrix.

THEOREM 4.4 (BLR triangular system) Let $\widetilde{T} \in \mathbb{R}^{n\times n}$ be a triangular BLR matrix partitioned into p^2 LR blocks $\widetilde{T}_{ij} \in \mathbb{R}^{b\times b}$ and let $v \in \mathbb{R}^n$. If the solution to the system $\widetilde{T}x = v$ is computed by solving the triangular system $T_{ii}x_i = v_i - \sum_{j=1}^{i-1}\widetilde{T}_{ij}x_j$ for each block $x_i = x((i-1)b+1 : ib)$, the computed solution $\widehat{x}$ satisfies

$$(\widetilde{T} + \Delta\widetilde{T})\widehat{x} = v + \Delta v, \quad \|\Delta\widetilde{T}\| \leqslant \gamma_c\|\widetilde{T}\|, \quad \|\Delta v\| \leqslant \gamma_p\|v\|, \tag{4.21}$$

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

Proof. Let $w_i^{(j)} = \widetilde{T}_{ij}\widehat{x}_j$, where $\widehat{x}_j$ is the jth block-row of the computed $\widehat{x}$ in the previous $i-1$ steps. By

Lemma 3.1, the computed $\widehat{w}_i^{(j)}$ satisfies

$$\widehat{w}_i^{(j)} = \big(\widetilde{T}_{ij} + F_{ij}\big)\widehat{x}_j, \quad \|F_{ij}\| \leqslant \gamma_d \|\widetilde{T}_{ij}\|,$$

with $d = b + r^{3/2}$. Let $w_i = v_i - \sum_{j=1}^{i-1} w_i^{(j)}$; the computed $\widehat{w}_i$ satisfies

$$\widehat{w}_i = v_i \circ (e + \Delta e_i) - \sum_{j=1}^{i-1} \widehat{w}_i^{(j)} \circ (e + \Delta e_j), \quad |\Delta e_j| \leqslant \gamma_p e,$$

where $e = [1, \dots, 1]^T$ is the vector of ones. By Lemma 2.2, the computed solution $\widehat{x}_i$ to $T_{ii} x_i = \widehat{w}_i$ satisfies

$$\big(T_{ii} + F_{ii}\big)\widehat{x}_i = \widehat{w}_i, \quad \|F_{ii}\| \leqslant \gamma_b \|T_{ii}\|.$$

Therefore, recalling that $\widetilde{T}_{ii} = T_{ii}$, we have

$$\sum_{j=1}^{i} \big(\widetilde{T}_{ij} + \Delta\widetilde{T}_{ij}\big)\widehat{x}_j = v_i \circ (e + \Delta e_i), \tag{4.22}$$

with $\Delta\widetilde{T}_{ij} = \Theta_j \circ \widetilde{T}_{ij} + F_{ij} \circ (J + \Theta_j)$ and thus $\|\Delta\widetilde{T}_{ij}\| \leqslant \gamma_{d+p+O(u)} \|\widetilde{T}_{ij}\|$. Gathering (4.22) over all block-rows i, we obtain $(\widetilde{T} + \Delta\widetilde{T})\widehat{x} = v + \Delta v$, with $\|\Delta v\| \leqslant \gamma_p \|v\|$ and

$$\|\Delta\widetilde{T}\| \leqslant \gamma_{d+p+O(u)} \Big(\sum_{i=1}^{p} \sum_{j=1}^{i} \|\widetilde{T}_{ij}\|^2 \Big)^{1/2} \leqslant \gamma_{d+p} \|\widetilde{T}\| + O(u^2),$$

as required. □

We are ready for our final theorem, which builds upon all our previous analyses to prove the backward stability of the solution to linear systems by BLR LU factorization.

THEOREM 4.5 (BLR linear system) Let $\widetilde{A} \in \mathbb{R}^{n\times n}$ be a $pb \times pb$ BLR matrix and let $v \in \mathbb{R}^n$. If the linear system $\widetilde{A}x = v$ is solved by solving the triangular systems $\widetilde{L}y = v$, $\widetilde{U}x = y$, where $\widetilde{L}$ and $\widetilde{U}$ are the BLR LU factors computed by either Algorithm 4.1 or 4.2, then the computed solution $\widehat{x}$ satisfies

$$\big(A + \Delta A\big)\widehat{x} = v + \Delta v, \tag{4.23}$$

$$\|\Delta A\| \leqslant \big(\xi_p \varepsilon + \gamma_p\big)\|A\| + \gamma_{3c} \|\widetilde{L}\|\|\widetilde{U}\| + O(u\varepsilon), \tag{4.24}$$

$$\|\Delta v\| \leqslant \gamma_p \big(\|v\| + \|\widetilde{L}\|\|\widetilde{U}\|\|\widehat{x}\|\big) + O(u^2), \tag{4.25}$$

where $c = b + 2r^{3/2} + p$, and where ξ_p is given in Table 4.1.

Proof. By Theorems 4.2 and 4.3, the BLR LU factors computed by the UFC algorithm or the UCF algorithm satisfy $A + \Delta A = \widetilde{L}\widetilde{U}$, with

$$\|\Delta A\| \leqslant \big(\xi_p \varepsilon + \gamma_p\big)\|A\| + \gamma_c \|\widetilde{L}\|\|\widetilde{U}\| + O(u\varepsilon).$$

By Theorem 4.4, the computed $\widehat{y}$ satisfies

$$\big(\widetilde{L} + \Delta\widetilde{L}\big)\widehat{y} = v + \Delta v, \quad \|\Delta\widetilde{L}\| \leqslant \gamma_c \|\widetilde{L}\|, \quad \|\Delta v\| \leqslant \gamma_p \|v\|.$$

Similarly the computed $\widehat{x}$ satisfies

$$\big(\widetilde{U}+\Delta\widetilde{U}\big)\widehat{x}=\widehat{y}+\Delta\widehat{y}, \quad \|\Delta\widetilde{U}\| \leqslant \gamma_c\|\widetilde{U}\|, \quad \|\Delta\widehat{y}\| \leqslant \gamma_p\|\widehat{y}\|.$$

We therefore obtain on the one hand

$$\big(\widetilde{L}+\Delta\widetilde{L}\big)\big(\widetilde{U}+\Delta\widetilde{U}\big)\widehat{x}=\big(A+\Delta A+\Delta\widetilde{L}\widetilde{U}+\widetilde{L}\Delta\widetilde{U}+\Delta\widetilde{L}\Delta\widetilde{U}\big)\widehat{x}=\big(A+\Delta A'\big)\widehat{x}$$

and on the other hand

$$\big(\widetilde{L}+\Delta\widetilde{L}\big)\big(\widetilde{U}+\Delta\widetilde{U}\big)\widehat{x}=v+\Delta v+\widetilde{L}\Delta\widehat{y}+\Delta\widetilde{L}\Delta\widehat{y}=v+\Delta v',$$

yielding $\big(A+\Delta A'\big)\widehat{x}=v+\Delta v'$, with

$$\begin{aligned}\|\Delta A'\| &= \|\Delta A+\Delta\widetilde{L}\widetilde{U}+\widetilde{L}\Delta\widetilde{U}+\Delta\widetilde{L}\Delta\widetilde{U}\| \leqslant \big(\xi_p\varepsilon+\gamma_p\big)\|A\|+\gamma_{3c}\|\widetilde{L}\|\|\widetilde{U}\|+O(u\varepsilon),\\ \|\Delta v'\| &= \|\Delta v+\widetilde{L}\Delta\widehat{y}+\Delta\widetilde{L}\Delta\widehat{y}\| \leqslant \gamma_p\big(\|v\|+\|\widetilde{L}\|\|\widetilde{U}\|\|\widehat{x}\|\big)+O(u^2).\end{aligned}$$

□

5. Additional experiments and discussion

In the previous sections we have illustrated our analysis with numerical experiments performed on Poisson matrices (as described in section 2.3). In this final section, we provide some additional experiments to demonstrate that the conclusions drawn in the previous sections extend to many other kind of problems coming from various real-life applications. We use 26 root separators (Schur complements) obtained from sparse matrices from the SuiteSparse collection (Davis & Hu, 2011). The full list is given in Table 1.1 in the supplementary materials.

The main conclusions drawn from our analysis and experiments in the previous sections were the following.

1. As predicted by our analysis, we have observed a tight correlation between the low-rank threshold ε and the backward error. We show that this crucial result remains true for a wide range of matrices in section 5.1.

2. We experimentally determined using a Poisson matrix of relatively small order $n = 4096$ that the use of a global threshold, the UCF algorithm, and intermediate recompressions achieves a better flops–accuracy tradeoff than the use of a local threshold, the UFC algorithm, and no recompressions, respectively. In section 5.2, we first analyze using Poisson matrices how this comparison evolves as n increases. Then, in section 5.3, we extend these conclusions to a wide range of matrices.

5.1 *Impact of ε for a wide range of matrices*

For each matrix A we solve a linear system $Ax = v$ via BLR LU factorization, using the UCF algorithm with a global threshold and intermediate recompressions. We report the backward error (2.11) in Figure 5.1, which shows that for all these matrices there is a good and often even excellent correlation between the threshold ε and the measured backward error.

The main conclusion of our analysis, which is that the backward error is directly determined by ε, is therefore confirmed experimentally for a wide range of matrices.

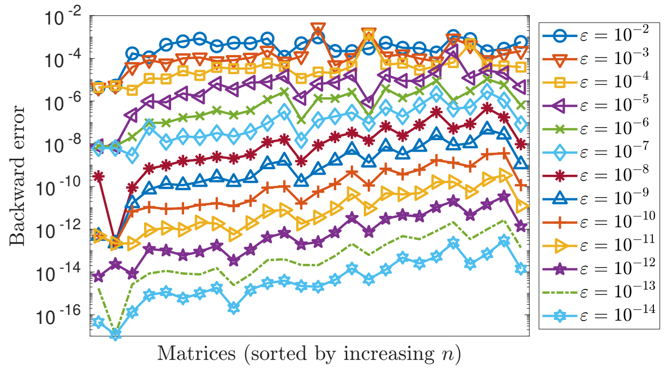


FIG. 5.1. *Backward error* (2.11) *for solving a linear system by BLR LU factorization for* 26 *real-life matrices using the UCF algorithm (Algorithm* 4.2*) with a global threshold and intermediate recompressions.*

5.2 *Flops–accuracy tradeoff for increasing n*

On the one hand, the use of a global threshold and intermediate recompressions both lead to a constant ξ_p larger by about a factor $p = n/b$, as shown in Table 4.1. On the other hand, intermediate recompressions and the UCF algorithm both reduce the asymptotic complexity of the LU factorization (Amestoy *et al.*, 2017). Even a global threshold may provide an asymptotic improvement, because the proportion of blocks of small norm with respect to the norm of the global matrix increases with n.

It is therefore important to investigate whether the strategy achieving the best flops–accuracy tradeoff depends on n. Figure 5.2 compares the tradeoff achieved by two strategies: the first uses the UFC algorithm with a local threshold and without recompressions, whereas the second uses the UCF algorithm with a global threshold and with recompressions. We compare these strategies for two Poisson matrices of order $n = 4096$ and $n = 16384$. Not only is the second strategy the best choice for both matrices, but the gap between the two is larger for $n = 16384$ than for $n = 4096$. Indeed, for five different values of ε (from 10^{-13} to 10^{-3}, indicated by the dashed lines on the figure), we measure the flops required by each strategy and plot the ratio between the two. The figure shows that, except for $\varepsilon = 10^{-3}$, this ratio is larger for the larger matrix. Additional experiments (not shown) on matrices of intermediate order between 4096 and 16384 show that this ratio gradually increases with n. We conclude that the second strategy becomes more and more beneficial with respect to the first strategy as n increases.

This experimental observation may in fact be justified theoretically for some classes of matrices. For instance, for Poisson matrices, the ranks of the blocks are known to be logarithmically dependent on the threshold ε, that is, $r = O(\log 1/\varepsilon)$ (Bebendorf, 2008). The impact of a larger ξ_p on the error can be compensated by simply using a smaller threshold $\varepsilon' = \varepsilon/\xi_p$, which in turn yields a larger rank $r' = O(\log \xi_p/\varepsilon)$. Therefore, as $p = n/b$ increases, compensating for the error increase due to a larger ξ_p only increases the cost of the factorization by logarithmic factors of n. Since a global threshold, intermediate recompressions, and the UCF algorithm all reduce this cost by factors $O(n^{\alpha})$, with $\alpha > 0$, we conclude that the use of these strategies must eventually become beneficial for large enough n.

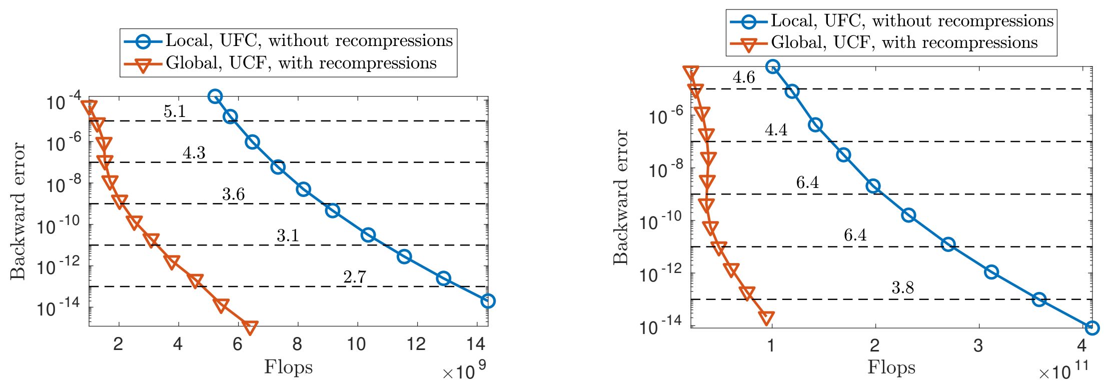


FIG. 5.2. *Flops–accuracy tradeoff for two opposite strategies for the solution of a BLR linear system, using two Poisson matrices of different orders $n = 4096$ (left) and $n = 16384$ (right). The numbers indicate the ratio between the flops required by the two strategies for a given cutoff value of ε (corresponding to the dashed lines).*

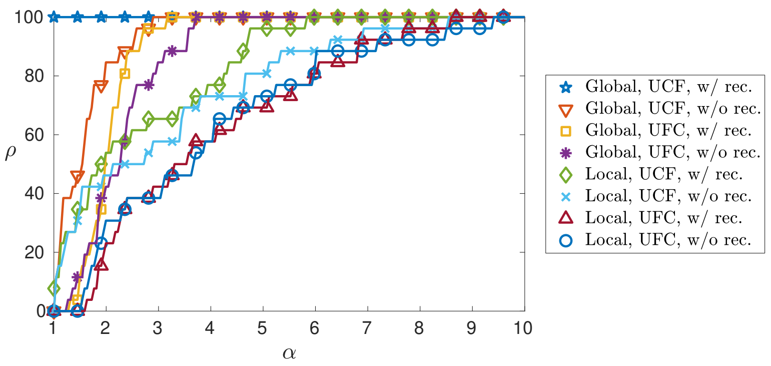


FIG. 5.3. *Performance profile of the eight possible strategies to solve a BLR linear system over 26 real-life matrices. ρ (y-axis) indicates the percentage of matrices for which a given strategy requires less than α (x-axis) times the flops required by the best strategy. For any given matrix, all eight strategies achieve roughly the same backward error.*

5.3 *Flops–accuracy tradeoff for a wide range of matrices*

We finally compare the flops–accuracy tradeoff achieved by global and local thresholds, the UFC and UCF algorithms, and the use of intermediate recompressions on the set of real-life matrices. We compare the eight possible strategies depending on the combination of parameters.

For each matrix, we run each strategy multiple times with slightly different values of ε between 10^{-7} and 10^{-9}. We then select a cutoff value and choose for each strategy the largest ε producing a backward error smaller than this cutoff. This is done to guarantee that all eight strategies achieve roughly the same backward error. We can then measure the number of flops required by each strategy and plot the result as a performance profile in Figure 5.3.

As the figure shows, the strategy using the UCF algorithm with a global threshold and with recompressions achieves the best tradeoff for *all* 26 matrices, thereby confirming our previous observations that this is the best parameter setting. This strategy can reduce the number of flops by a factor up to 10 with no loss of accuracy. The performance of the remaining seven strategies gives some indication of the relative importance of each parameter: using a global threshold has the greatest impact, followed by the UCF algorithm, and finally the intermediate recompressions.

6. Conclusions

We have analyzed the errors introduced in various matrix algorithms by the use of block low-rank (BLR) approximations. Our analysis provides important new theoretical guarantees, as well as some new insights into the numerical behavior of BLR matrix algorithms in floating-point arithmetic. We now gather and summarize our main conclusions.

6.1 *Summary*

We have derived a set of normwise error bounds that share a common point: they are expressed as the sum of two terms, one associated with the low-rank truncation errors, whose magnitude can be controlled via the low-rank threshold ε, and the other associated with the floating-point errors, whose magnitude depends on the unit roundoff u.

Usually, we have $u \ll \varepsilon$, and therefore the error is mainly determined by ε. In particular, we have proved in Theorem 4.5 that BLR linear systems $Ax = v$ can be solved with a backward error proportional to $\xi_p \varepsilon \|A\|$, where ξ_p is a small constant growing at most quadratically with the number of block-rows and block-columns $p = n/b$. Our analysis therefore proves for the first time the backward stability of the solution of BLR linear systems and provides a theoretical justification for the empirical observation that the backward error is closely related to the low-rank threshold. Users can therefore control the numerical behavior of BLR solvers simply by setting ε to the target accuracy.

When $u \ll \varepsilon$, the unit roundoff has only a limited impact. This remark is of particular relevance in the context where BLR solvers are used as preconditioners for iterative methods (Higham & Mary, 2019), for which the low-rank threshold may be set to very large values (such as $\varepsilon = 0.01$ or even 0.1). In this setting, our analysis indicates that the use of low precisions, such as half precision, should be very attractive; see Higham *et al.* (2019) and the references therein for details of half precision and how it can be exploited in standard LU factorization.

We have analyzed several key parameters in the BLR LU factorization and assessed how to choose them to obtain the best possible tradeoff between flops and accuracy. First, we have shown that the use of a global threshold (block A_{ij} is compressed such that $\|A_{ij} - \widetilde{A}_{ij}\| \leqslant \varepsilon \|A\|$) should be preferred to that of a local one ($\|A_{ij} - \widetilde{A}_{ij}\| \leqslant \varepsilon \|A_{ij}\|$). Second, the use of intermediate recompressions in the update step (the so-called LUAR strategy in Amestoy *et al.* (2017)) only impacts the constant in the error bound and is therefore recommended. Finally, we have compared two different factorization variants, the UFC and UCF algorithms, which differ in when the BLR compression is incorporated in the LU algorithm, and we have shown that they yield similar error bounds; the UCF algorithm, which achieves the best complexity, should therefore be preferred.

We have supported all of these conclusions with numerical experiments on a wide range of matrices from various real-life applications.

6.2 *Perspectives*

There exist numerous structured matrix representations other than BLR, such as multilevel (Amestoy *et al.*, 2019a) and hierarchical (Bebendorf, 2008) representations, HSS matrices (Xia *et al.*, 2010), and so on. Our analysis could be extended to these other type of matrices (we note the existing work regarding HSS matrices by Xi & Xia (2016)), and we expect that the resulting analyses would yield similar results.

If the threshold ε is chosen too close to the unit roundoff u, Assumption 2.1 no longer holds and we are unable to accurately detect the numerical rank of the blocks, which dramatically increases the cost of the factorization. This situation is becoming more likely with the growing use of low precision floating-point arithmetic. In this context, recent advances that reduce rounding error accumulation (Blanchard *et al.*, 2020a,b) can help to decrease the threshold of ε at which we are forced to switch to a higher precision arithmetic.

Acknowledgments

We thank Cleve Ashcraft and an anonymous referee for their constructive comments. Some experiments on the larger problems have been run using the crunch machines at the LIP laboratory (ENS Lyon).

References

AMESTOY, P. R., BROSSIER, R., BUTTARI, A., L'EXCELLENT, J.-Y., MARY, T., MÉTIVIER, L., MINIUSSI, A., & OPERTO, S. 2016. Fast 3D frequency-domain full waveform inversion with a parallel Block Low-Rank multifrontal direct solver: application to OBC data from the North Sea. *Geophysics*, **81**(6), R363–R383.

AMESTOY, P. R., BUTTARI, A., L'EXCELLENT, J.-Y., & MARY, T. 2017. On the Complexity of the Block Low-Rank Multifrontal Factorization. *SIAM J. Sci. Comput.*, **39**(4), A1710–A1740.

AMESTOY, PATRICK, ASHCRAFT, CLEVE, BOITEAU, OLIVIER, BUTTARI, ALFREDO, L'EXCELLENT, JEAN-YVES, & WEISBECKER, CLÉMENT. 2015. Improving Multifrontal Methods by Means of Block Low-Rank Representations. *SIAM J. Sci. Comput.*, **37**(3), A1451–A1474.

AMESTOY, PATRICK R., BUTTARI, ALFREDO, L'EXCELLENT, JEAN-YVES, & MARY, THEO. 2019a. Bridging the Gap Between Flat and Hierarchical Low-Rank Matrix Formats: The Multilevel Block Low-Rank Format. *SIAM J. Sci. Comput.*, **41**(3), A1414–A1442.

AMESTOY, PATRICK R., BUTTARI, ALFREDO, L'EXCELLENT, JEAN-YVES, & MARY, THEO. 2019b. Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures. *ACM Trans. Math. Software*, **45**(1), 2:1–2:26.

BEBENDORF, MARIO. 2008. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*. Lecture Notes in Computational Science and Engineering (LNCSE), vol. 63. Springer-Verlag.

BLANCHARD, PIERRE, HIGHAM, NICHOLAS J., & MARY, THEO. 2020a. A Class of Fast and Accurate Summation Algorithms. *SIAM J. Sci. Comput.*, **42**(3), A1541–A1557.

BLANCHARD, PIERRE, HIGHAM, NICHOLAS J., LOPEZ, FLORENT, MARY, THEO, & PRANESH, SRIKARA. 2020b. Mixed Precision Block Fused Multiply-Add: Error Analysis and Application to GPU Tensor Cores. *SIAM J. Sci. Comput.*, **42**(3), C124–C141.

CHARARA, ALI, KEYES, DAVID, & LTAIEF, HATEM. 2018. Tile Low-Rank GEMM Using Batched Operations on GPUs. *Pages 811–825 of:* ALDINUCCI, MARCO, PADOVANI, LUCA, & TORQUATI, MASSIMO (eds), *Euro-Par 2018: Parallel Processing*. Cham: Springer International Publishing.

DAVIS, TIMOTHY A., & HU, YIFAN. 2011. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Software*, **38**(1), 1:1–1:25.

GEORGE, A. 1973. Nested Dissection of a Regular Finite Element Mesh. *SIAM J. Numer. Anal.*, **10**(2), 345–363.

HIGHAM, NICHOLAS J. 1990. Exploiting Fast Matrix Multiplication within the Level 3 BLAS. *ACM Trans. Math. Software*, **16**(4), 352–368.

HIGHAM, NICHOLAS J. 1992. Stability of a Method for Multiplying Complex Matrices with Three Real Matrix Multiplications. *SIAM J. Matrix Anal. Appl.*, **13**(3), 681–687.

HIGHAM, NICHOLAS J. 2002. *Accuracy and Stability of Numerical Algorithms*. Second edn. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

HIGHAM, NICHOLAS J., & MARY, THEO. 2019. A New Preconditioner that Exploits Low-Rank Approximations to Factorization Error. *SIAM J. Sci. Comput.*, **41**(1), A59–A82.

HIGHAM, NICHOLAS J., & PRANESH, SRIKARA. 2019. Simulating Low Precision Floating-Point Arithmetic. *SIAM J. Sci. Comput.*, **41**(5), C585–C602.

HIGHAM, NICHOLAS J., PRANESH, SRIKARA, & ZOUNON, MAWUSSI. 2019. Squeezing a Matrix Into Half Precision, with an Application to Solving Linear Systems. *SIAM J. Sci. Comput.*, **41**(4), A2536–A2551.

IDA, AKIHIRO, NAKASHIMA, HIROSHI, & KAWAI, MASATOSHI. 2018. Parallel Hierarchical Matrices with Block Low-rank Representation on Distributed Memory Computer Systems. *Pages 232–240 of: Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*. HPC Asia 2018. New York, NY, USA: ACM.

JEANNEROD, C.-P., MARY, T., PERNET, C., & ROCHE, D. 2019. Exploiting fast matrix arithmetic in block low-rank factorizations. *SIAM J. Matrix Anal. Appl.* Submitted.

MARY, THÉO. 2017 (Nov.). *Block Low-Rank Multifrontal Solvers: Complexity, Performance, and Scalability*. Ph.D. thesis, Université de Toulouse, Toulouse, France.

PICHON, GRÉGOIRE, DARVE, ERIC, FAVERGE, MATHIEU, RAMET, PIERRE, & ROMAN, JEAN. 2018. Sparse supernodal solver using block low-rank compression: Design, performance and analysis. *Journal of Computational Science*, **27**, 255–270.

RIGAL, J. L., & GACHES, J. 1967. On the Compatibility of a Given Solution With the Data of a Linear System. *J. Assoc. Comput. Mach.*, **14**(3), 543–548.

SHANTSEV, D. V., JAYSAVAL, P., DE LA KETHULLE DE RYHOVE, S., AMESTOY, P. R., BUTTARI, A., L'EXCELLENT, J.-Y., & MARY, T. 2017. Large-scale 3D EM modeling with a Block Low-Rank multifrontal direct solver. *Geophys. J. Int.*, **209**(3), 1558–1571.

XI, Y., & XIA, J. 2016. On the Stability of Some Hierarchical Rank Structured Matrix Algorithms. *SIAM J. Matrix Anal. Appl.*, **37**(3), 1279–1303.

XIA, JIANLIN, CHANDRASEKARAN, SHIVKUMAR, GU, MING, & LI, XIAOYE S. 2010. Fast Algorithms for Hierarchically Semiseparable Matrices. *Numer. Linear Algebra Appl.*, **17**(6), 953–976.