SIAM J. MATRIX ANAL. APPL. © 2013 Society for Industrial and Applied Mathematics
Vol. 34, No. 3, pp. 1401–1429

LU FACTORIZATION WITH PANEL RANK REVEALING PIVOTING AND ITS COMMUNICATION AVOIDING VERSION[*]

AMAL KHABOU[†], JAMES W. DEMMEL[‡], LAURA GRIGORI[§], AND MING GU[¶]

Abstract. We present block LU factorization with panel rank revealing pivoting (block LU_PRRP), a decomposition algorithm based on strong rank revealing QR panel factorization. Block LU_PRRP is more stable than Gaussian elimination with partial pivoting (GEPP), with a theoretical upper bound of the growth factor of $(1+\tau b)^{(n/b)-1}$, where b is the size of the panel used during the block factorization, τ is a parameter of the strong rank revealing QR factorization, n is the number of columns of the matrix, and for simplicity we assume that n is a multiple of b. We also assume throughout the paper that $2 \leq b \ll n$. For example, if the size of the panel is $b = 64$, and $\tau = 2$, then $(1+2b)^{(n/b)-1} = (1.079)^{n-64} \ll 2^{n-1}$, where 2^{n-1} is the upper bound of the growth factor of GEPP. Our extensive numerical experiments show that the new factorization scheme is as numerically stable as GEPP in practice, but it is more resistant to pathological cases. The block LU_PRRP factorization does only $O(n^2 b)$ additional floating point operations compared to GEPP. We also present block CALU_PRRP, a version of block LU_PRRP that minimizes communication and is based on tournament pivoting, with the selection of the pivots at each step of the tournament being performed via strong rank revealing QR factorization. Block CALU_PRRP is more stable than CALU, the communication avoiding version of GEPP, with a theoretical upper bound of the growth factor of $(1+\tau b)^{\frac{n}{b}(H+1)-1}$, where H is the height of the reduction tree used during tournament pivoting. The upper bound of the growth factor of CALU is $2^{n(H+1)-1}$. Block CALU_PRRP is also more stable in practice and is resistant to pathological cases on which GEPP and CALU fail.

Key words. LU factorization, pivoting, growth factor, numerical stability, communication avoiding, strong rank revealing QR factorization

AMS subject classifications. 65F05, 68W10

DOI. 10.1137/120863691

1. Introduction.

The LU factorization is an important operation in numerical linear algebra since it is widely used for solving linear systems of equations or computing the determinant of a matrix, or as a building block of other operations. It consists of the decomposition of a matrix A into the product $A = \Pi LU$, where L is a lower triangular matrix, U is an upper triangular matrix, and Π a permutation matrix. The performance of the LU decomposition is critical for many applications, and it has received significant attention over the years. Recently, large efforts have been

[*]Received by the editors January 25, 2012; accepted for publication (in revised form) by N. J. Higham July 5, 2013; published electronically September 26, 2013.
http://www.siam.org/journals/simax/34-3/86369.html

[†]Laboratoire de Recherche en Informatique, Université Paris-Sud 11 and INRIA Saclay-Ile de France, Orsay, France. Current address: School of Mathematics, The University of Manchester, Manchester M13 9PL, England (amal.khabou@manchester.ac.uk).

[‡]Computer Science Division and Mathematics Department, University of California at Berkeley, Berkeley, CA 94720-1776 (demmel@cs.berkeley.edu). This author was supported in part by Microsoft (award 024263) and Intel (award 024894) funding and by matching funding by U.C. Discovery (award DIG07-10227). Additional support comes from Par Lab affiliates National Instruments, Nokia, NVIDIA, Oracle, and Samsung, and DOE grants DE-SC0003959, DE-SC0004938, and DE-AC02-05CH11231.

[§]Applied Mathematics, INRIA, Rocquencourt, Paris 75005, France (laura.grigori@inria.fr). This author was supported in part by the French National Research Agency (ANR) through COSINUS program, project PETALH ANR-10-COSI-013.

[¶]Mathematics Department, University of California at Berkeley, Berkeley, CA 94720-1776 (mgu@math.berkeley.edu). This author's work was supported in part by NSF award CCF-0830764 and by the DOE Oce of Advanced Scientific Computing Research under contract number DE-AC02-05CH11231.

invested in optimizing this linear algebra kernel, in terms of both numerical stability and performance on emerging parallel architectures.

The LU decomposition can be computed using Gaussian elimination with partial pivoting, a very stable operation in practice, except for pathological cases such as the Wilkinson matrix [22, 17], the Foster matrix [8], or the Wright matrix [24]. Many papers [21, 19, 20] discuss the stability of the Gaussian elimination, and it is known [17, 10, 9] that the pivoting strategy used, such as complete pivoting, partial pivoting, or rook pivoting, has an important impact on the numerical stability of this method, which depends on a quantity referred to as the growth factor. However, in terms of performance, these pivoting strategies represent a limitation, since they require asympotically more communication than established lower bounds on communication indicate is necessary [4, 1].

Technological trends show that computing floating point operations is becoming exponentially faster than moving data from the memory where they are stored to the place where the computation occurs. Due to this, the communication becomes in many cases a dominant factor of the runtime of an algorithm, that leads to a loss of its efficiency. This is a problem for both a sequential algorithm, where data needs to be moved between different levels of the memory hierarchy, and a parallel algorithm, where data needs to be communicated between processors. This challenging problem has prompted research on algorithms that reduce the communication to a minimum, while being numerically as stable as classic algorithms, and without increasing significantly the number of floating point operations performed [4, 11]. We refer to these algorithms as communication avoiding. One of the first such algorithms is the communication avoiding LU factorization (CALU) [11, 12]. This algorithm is optimal in terms of communication, that is, it performs at most polylogarithmic factors more than the theoretical lower bounds on communication require [4, 1]. Thus, it brings considerable improvements to the performance of the LU factorization compared with the classic routines that perform the LU decomposition such as the PDGETRF routine of ScaLAPACK [2], thanks to a novel pivoting strategy referred to as tournament pivoting. It was shown that CALU is faster in practice than the corresponding routine PDGETRF implemented in libraries as ScaLAPACK or vendor libraries, on both distributed [11] and shared memory computers [6]. While in practice CALU is as stable as Gaussian elimination with partial pivoting (GEPP), in theory the upper bound of its growth factor is worse than that obtained with GEPP. One of our goals is to design an algorithm that minimizes communication and that has a smaller upper bound of its growth factor than CALU.

In the first part of this paper we present the block LU_PRRP factorization, a novel LU decomposition algorithm based on what we call panel rank revealing pivoting (PRRP). The block LU_PRRP factorization is based on an algorithm that computes the LU decomposition as follows. At each step of the block factorization, a block of columns (panel) is factored by computing the strong rank revealing QR (RRQR) factorization [13] of its transpose. The permutation returned by the panel rank revealing factorization is applied to the rows of the input matrix, and the block L factor of the panel is computed based on the R factor of the strong RRQR factorization. Then the trailing matrix is updated. The factors obtained from this decomposition can be stored in place, and so block LU_PRRP has the same memory requirements as standard LU and can easily replace it in any application.

We show that block LU_PRRP is more stable than GEPP. Its growth factor is bounded by $(1 + \tau b)^{(n/b)-1}$, where b is the size of the panel used during the block factorization, τ is a parameter of the strong rank revealing QR factorization, n is the number of columns of the matrix, and for simplicity we assume that n is a multiple of b. We also assume throughout the paper that $2 \leq b \ll n$. The previous bound is smaller than 2^{n-1}, the upper bound of the growth factor for GEPP. For example, if the size of the panel is $b = 64$ and $\tau = 2$, then $(1 + 2b)^{(n/b)-1} = (1.079)^{n-64} \ll 2^{n-1}$. In terms of cost, block LU_PRRP performs only $O(n^2 b)$ more floating point operations than GEPP. In addition, our extensive numerical experiments on random matrices and on a set of special matrices show that the block LU_PRRP factorization is very stable in practice and leads to modest growth factors, smaller than those obtained with GEPP. It also easily solves pathological cases, such as the Wilkinson matrix and the Foster matrix, on which GEPP fails. While the Wilkinson matrix is a matrix constructed such that GEPP has an exponential growth factor, the Foster matrix [9] arises from a real application.

In the second part of the paper we introduce the block CALU_PRRP factorization, the communication avoiding version of block LU_PRRP. It is based on tournament pivoting, a strategy introduced in [12] in the context of CALU, a communication avoiding version of GEPP. With tournament pivoting, the panel factorization is performed in two steps. The first step selects b pivot rows from the entire panel at a minimum communication cost. For this, sets of b candidate rows are selected from blocks of the panel, which are then combined together through a reduction-like procedure, until a set of b pivot rows are chosen. Block CALU_PRRP uses the strong RRQR factorization to select b rows at each step of the reduction operation, while CALU is based on GEPP. In the second step of the panel factorization, the pivot rows are permuted to the diagonal positions, and the QR factorization with no pivoting of the transpose of the panel is computed. Then the algorithm proceeds as the block LU_PRRP factorization. Note that the usage of the strong RRQR factorization ensures that bounds are respected locally at each step of the reduction operation, but it does not ensure that the growth factor is bounded globally as in block LU_PRRP.

This algorithm has two significant advantages over other classic factorization algorithms. First, it minimizes communication up to some polylogarithmic factors, and hence it will be more efficient than block LU_PRRP and GEPP on architectures where communication is expensive. Second, it is more stable than CALU. Theoretically, the upper bound of the growth factor in exact arithmetic of block CALU_PRRP is smaller than that of CALU for a reduction tree with a same height.

Extensive experimental results show that block CALU_PRRP is as stable as block LU_PRRP, GEPP, and CALU on random matrices and a set of special matrices. Its growth factor is slightly smaller than that of CALU. In addition, it is also stable for matrices on which GEPP fails.

We also discuss a different version of block LU_PRRP that minimizes communication but can be less stable than block CALU_PRRP, our method of choice for reducing communication. In this different version, the panel factorization is performed only once, during which its off-diagonal blocks are annihilated using a reduce-like operation, with the strong RRQR factorization being the operator used at each step of the reduction. Independently of the shape of the reduction tree, the upper bound of the growth factor of this method is the same as that of block LU_PRRP. This is because at every step of the algorithm, a row of the current trailing matrix is updated only once. We refer to the version based on a binary reduction tree as block parallel LU_PRRP, and to the version based on a flat tree as block pairwise LU_PRRP.

The remainder of the paper is organized as follows. Section 2 presents the algebra of the block LU_PRRP factorization, discusses its stability, and compares it with that of GEPP. It also presents experimental results showing that block LU_PRRP is more stable than GEPP in terms of worst case growth factor. Section 3 presents the algebra of block CALU_PRRP, a communication avoiding version of block LU_PRRP. It describes similarities between block CALU_PRRP and block LU_PRRP, and it discusses its stability. The communication optimality of block CALU_PRRP is shown in section 4, where we also compare its performance model with that of the CALU algorithm. Section 5 discusses two alternative algorithms that can also reduce communication but can be less stable in practice. Section 6 concludes and presents our future work.

2. Block LU_PRRP factorization.

In this section we introduce the block LU_PRRP factorization, a block LU decomposition based on PRRP. It is based on an algorithm that factors at each step a block of columns (a panel), and then it updates the trailing matrix. The main difference between block LU_PRRP and GEPP resides in the panel factorization. In GEPP the panel factorization is computed using LU with partial pivoting, while in block LU_PRRP it is computed by performing a strong RRQR factorization of its transpose. This leads to a different selection of pivot rows, and the obtained R factor is used to compute the block L factor of the panel. In exact arithmetic, block LU_PRRP performs a block LU decomposition [3] with a different pivoting scheme, which aims at improving the numerical stability of the factorization by bounding more efficiently the growth of the elements. We also discuss the numerical stability of block LU_PRRP, and we show that both in theory and in practice, block LU_PRRP is more stable than GEPP.

2.1. The algebra.

Block LU_PRRP factors the input matrix A of size $m \times n$ by traversing blocks of columns of size b. Consider the first step of the factorization, with the matrix A having the following partition:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{2.1}$$

where A_{11} is of size $b \times b$, A_{21} is of size $(m-b) \times b$, A_{12} is of size $b \times (n-b)$, and A_{22} is of size $(m-b) \times (n-b)$. The main idea of the block LU_PRRP factorization is to eliminate the elements below the $b \times b$ diagonal block such that the multipliers used during the update of the trailing matrix are bounded by a given threshold τ. For this, we perform a strong RRQR factorization on the transpose of the first panel of size $m \times b$ to identify a permutation matrix Π, that is, b pivot rows,

$$\begin{aligned} \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}^T \Pi = \begin{bmatrix} \bar{A}_{11} \\ \bar{A}_{21} \end{bmatrix}^T &= Q \begin{bmatrix} R(1:b, 1:b) & R(1:b, b+1:m) \end{bmatrix} \\ &= Q \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}, \end{aligned}$$

where $\bar{A}$ denotes the permuted matrix A. The strong RRQR factorization ensures that the quantity $R_{12}^T (R_{11}^{-1})^T$ is bounded by a given threshold τ in the max norm. The strong RRQR factorization, as described in Algorithm 2 in Appendix A, computes first the QR factorization with column pivoting, followed by additional swaps of the columns of the R factor and updates of the QR factorization, so that $\|R_{12}^T (R_{11}^{-1})^T\| \leq \tau$ in max norm, that is, the norm defined as $\|A\| = \max_{i,j} |a_{ij}|$. After the panel factorization, the transpose of the computed permutation Π_1 is applied to the input matrix A, and then the trailing matrix is updated,

$$\bar{A} = \Pi_1^T A = \begin{bmatrix} I_b & \\ L_{21} & I_{m-b} \end{bmatrix} \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ & \bar{A}_{22}^s \end{bmatrix} = \begin{bmatrix} I_b & \\ L_{21} & I_{m-b} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ & \bar{A}_{22}^s \end{bmatrix}, \tag{2.2}$$

where

$$\bar{A}^s_{22} = \bar{A}_{22} - L_{21}\bar{A}_{12}. \tag{2.3}$$

Note that in exact arithmetic we have $L_{21} = \bar{A}_{21}\bar{A}_{11}^{-1} = R_{12}^T(R_{11}^{-1})^T$. The factorization in (2.2) is equivalent to the factorization

$$\bar{A} = \Pi_1^T A = \begin{bmatrix} I_b & \\ \bar{A}_{21}\bar{A}_{11}^{-1} & I_{m-b} \end{bmatrix} \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ & \bar{A}^s_{22} \end{bmatrix},$$

where the first block row of the U factor is formed by $U_{11} = \bar{A}_{11}$, $U_{12} = \bar{A}_{12}$, and

$$\bar{A}^s_{22} = \bar{A}_{22} - \bar{A}_{21}\bar{A}_{11}^{-1}\bar{A}_{12}, \tag{2.4}$$

and $\bar{A}_{21}\bar{A}_{11}^{-1}$ was computed in a numerically stable way such that it is bounded in max norm by τ.

The factorization then continues recursively on computing the block LU_PRRP factorization of $\bar{A}^s_{22}$. Algorithm 1 shows the computation of the block LU_PRRP factorization $\Pi^T A = LU$ of a matrix A of size $n \times n$ by using panels of size b. The number of floating-point operations performed by Algorithm 1 is

$$\#flops = \frac{2}{3}n^3 + O(n^2 b),$$

which is only $O(n^2 b)$ more floating point operations than GEPP. The detailed counts are presented in Appendix C of [16]. When the QR factorization with column pivoting is sufficient to obtain the desired bound for each panel factorization, and no additional swaps are performed, the total cost is

$$\#flops = \frac{2}{3}n^3 + \frac{3}{2}n^2 b.$$

Block LU_PRRP has the same memory requirements as the standard LU decomposition since the factors can be stored in place, and so it can easily replace it in any application.

Once the block LU_PRRP factorization has been performed, a linear system $Ax = y$ can be solved by computing a forward substitution $Lz = \Pi^T y$, followed by a block back substitution $Ux = z$. At the last step of the substitution, the computation is

$$U_{11}x_1 = z_1 - U_{12}x_2,$$

and similar relations hold at the previous steps. Hence the block back substitution requires either using the QR factorization of the diagonal blocks U_{ii} or computing their GEPP factorization. We will discuss both possibilities in the following error analysis section, while the numerical experiments presented later in the paper use GEPP during the block back substitution. For simplicity, we ignore in the presentation the permutation produced by GEPP, and we write the factorization as $U_{ii} = L_{d_{ii}} U_{d_{ii}}$. We refer to the block diagonal matrix formed by the $L_{d_{ii}}$ factors ($U_{d_{ii}}$ factors) as L_d (U_d, respectively). We note that an LU factorization with L lower unit triangular and U upper triangular can be obtained if the LU factors of the diagonal blocks U_{ii} are used also to update the corresponding trailing matrices. For example, if at the first step of the factorization, the LU factors of $U_{11} = \bar{A}_{11} = L_{d_{11}} U_{d_{11}}$ are used to update the trailing matrix $\bar{A}_{12}$, then the decomposition obtained after the elimination of the first panel of the input matrix A is

(2.5)
$$\bar{A} = \Pi^T A = \begin{bmatrix} I_b & \\ L_{21} & I_{m-b} \end{bmatrix} \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ & \bar{A}_{22}^s \end{bmatrix} = \begin{bmatrix} L_{d_{11}} & \\ L_{21} L_{d_{11}} & I_{m-b} \end{bmatrix} \begin{bmatrix} U_{d_{11}} & L_{d_{11}}^{-1} \bar{A}_{12} \\ & \bar{A}_{22}^s \end{bmatrix}.$$

ALGORITHM 1. BLOCK LU_PRRP FACTORIZATION OF A MATRIX A OF SIZE $n \times n$ USING A PANEL OF SIZE b.

```
1: Compute the strong RRQR factorization $(A(1:n,1:b))^T \Pi_1 = Q_1 R_1$.
2: $L_{21} = (R_1(1:b,1:b)^{-1} R_1(1:b,b+1:n))^T$.
3: Pivot by applying the permutation matrix $\Pi_1^T$ on the entire matrix, $A = \Pi_1^T A$.
4: $U_{11} = A(1:b,1:b)$ , $U_{12} = A(1:b,b+1:n)$.
5: Update the trailing matrix,
6:    $A(b+1:n,b+1:n) = A(b+1:n,b+1:n) - L_{21} U_{12}$.
7: Compute recursively the block LU_PRRP factorization of $A(b+1:n,b+1:n)$.
```

2.2. Numerical stability. The stability of an LU decomposition depends on the growth factor. In his backward error analysis [22], Wilkinson proved that the computed solution $\hat{x}$ of the linear system $Ax = y$, where A is of size $n \times n$, obtained by GEPP or complete pivoting satisfies

$$(A + \Delta A)\hat{x} = y, \qquad \|\Delta A\|_\infty \le c(n) g(n) u \|A\|_\infty.$$

Here, $c(n)$ is a cubic polynomial, u is the machine precision, and $g(n)$ is the growth factor defined by

$$g(n) = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|} \ge 1,$$

where $a_{ij}^{(k)}$ denotes the entry in position (i, j) obtained after k steps of elimination. Thus the growth factor measures the growth of the elements during the elimination. The LU factorization is backward stable if $g(n)$ is small. Lemma 9.6 of [18] (section 9.3) states a more general result, showing that the LU factorization without pivoting of A is backward stable if the growth factor is small. Wilkinson [22] showed that for partial pivoting, the growth factor $g(n) \le 2^{n-1}$, and this bound is attainable. He also showed that for complete pivoting, the upper bound satisfies $g(n) \le n^{1/2}(2.3^{1/2} \dots n^{1/(n-1)})^{1/2} \sim cn^{1/2} n^{1/4 \log n}$. In practice the growth factors are much smaller than the upper bounds. These error analyses are performed for LU factorizations that produce a lower triangular matrix L and an upper triangular matrix U. The error analysis of a block LU factorization that produces a block lower triangular matrix L and a block upper triangular matrix U is given in [3]. Our stability analysis of block LU_PRRP is based on the stability analysis of block LU from [3]. We use in the following the max norm.

2.2.1. Growth factor in exact arithmetic. We derive an upper bound in exact arithmetic of the growth factor for block LU_PRRP factorization. We use the

Table 2.1

Upper bounds of the growth factor $g_{blockLU\_PRRP}(n, b, \tau)$ obtained from factoring a matrix of size $m \times n$ using block LU_PRRP with different panel sizes and $\tau = 2$.

b	8	16	32	64	128
$g(n, b, 2)$	$(1.42)^{n-8}$	$(1.24)^{n-16}$	$(1.14)^{n-32}$	$(1.08)^{n-64}$	$(1.04)^{n-128}$

same notation as in the previous section and we assume without loss of generality that the permutation matrix is the identity. It is easy to see that the growth factor obtained after the elimination of the first panel is bounded by $1 + \tau b$. At the kth step of the block factorization, the active matrix $A^{(k)}$ is of size $(m-(k-1)b)\times(n-(k-1)b)$, and the decomposition performed at this step can be written as

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix} = \begin{bmatrix} I_b & \\ L_{21}^{(k)} & I_{m-(k+1)b} \end{bmatrix} \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ & A_{22}^{(k)s} \end{bmatrix}.$$

The active matrix at the $(k+1)$st step is $A_{22}^{(k+1)} = A_{22}^{(k)s} = A_{22}^{(k)} - L_{21}^{(k)} A_{12}^{(k)}$. Then $\max_{i,j} |a_{i,j}^{(k+1)}| \le \max_{i,j} |a_{i,j}^{(k)}|(1 + \tau b)$ with $\max_{i,j} |L_{21}^{(k)}(i, j)| \le \tau$, and we have

$$g^{(k+1)} \le g^{(k)}(1 + \tau b). \tag{2.6}$$

Induction on (2.6) leads to a growth factor of block LU_PRRP performed on the n/b panels of the matrix A that satisfies

$$g_{blockLU\_PRRP}(n, b, \tau) \le (1 + \tau b)^{(n/b)-1}, \tag{2.7}$$

where for simplicity we assume that n is a multiple of b. The improvement of the upper bound of the growth factor of block LU_PRRP with respect to GEPP is illustrated in Table 2.1, where the panel size varies from 8 to 128, and the parameter τ is equal to 2. For $b \ge 64$, the worst case growth factor becomes arbitrarily smaller than for GEPP.

However, if the block backward substitution uses GEPP factorizations of the diagonal $b \times b$ blocks or an LU factorization is computed as in (2.5), then each additional GEPP factorization leads to a growth factor bounded by 2^{b-1}. The growth factor of $U_{d_{n/b,n/b}}$, the U factor of the last $b \times b$ diagonal block, is the product of the growth factor of block LU_PRRP and the growth factor generated by GEPP, and it is upper bounded by $(1 + \tau b)^{(n/b)-1} \cdot 2^{b-1}$. The upper bound of the growth factor of the previous diagonal blocks is smaller than the one of the last diagonal block. In summary, by using the max norm, we have

$$\|L\| \le \tau, \|U\| \le g_{blockLU\_PRRP}(n, b, \tau)\|A\|,$$
$$\|L_d\| \le 1, \|U_d\| \le g_{blockLU\_PRRP}(n, b, \tau) \cdot 2^{b-1}\|A\|.$$

The optimal block b that minimizes element growth in $\|U_d\|$ is a rather complicated function of n and τ. A simple but somewhat suboptimal choice is $b = \sqrt{n}$, which leads to

$$g_{blockLU\_PRRP}(n, b, \tau) \cdot 2^{b-1} \le \left(1 + \tau\sqrt{n}\right)^{\sqrt{n}-1} 2^{\sqrt{n}-1} = e^{\sqrt{n}\log(\tau\sqrt{n})+O(\sqrt{n})}.$$

Not surprisingly, this upper bound is much better than 2^{n-1} but much worse than the element growth upper bound for complete pivoting.

2.2.2. Error analysis. We consider the block LU_PRRP factorization given in Algorithm 1 that decomposes the matrix A of size $n \times n$ as $\Pi^T A = LU$, where L is a block lower triangular matrix and U is a block upper triangular matrix. The first step of the decomposition is given in (2.2). Our rounding error analysis of block LU_PRRP uses the same model of floating point arithmetic as in [18, 3] and follows closely the analysis of block LU provided in [3]. The difference between Algorithm 1 and block LU from [3] lies in the computation of the matrix L_{21}. In block LU, this matrix is computed by performing the GEPP factorization of A_{11} and then by solving $L_{21}A_{11} = A_{21}$, while in block LU_PRRP it is computed from the strong rank revealing factorization of $[A_{11}; A_{21}]^T$.

In the following formulas, $c_i(n), i = 1, \ldots, 4$ denote constants depending polynomially on n, and we will ignore their detailed expressions. The analysis assumes that the computed matrix $\hat{L}_{21}$ from the factorization displayed in (2.2) satisfies the relation

$$\hat{L}_{21}\bar{A}_{11} = \bar{A}_{21} + E_{21}, \quad \|E_{21}\| \leq c_1(n)u\|\hat{L}_{21}\|\|\bar{A}_{11}\| + O(u^2). \tag{2.8}$$

This assumption is satisfied by block LU [3] (the variant referred to as Implementation 1 in [3]). Since the strong RRQR computations in Algorithm 1 are numerically stable, Algorithm 1 satsifies this assumption as well. For the error analysis of the computed solution $\hat{x}$ to $Ax = y$, it is assumed in addition that for each $b \times b$ square diagonal block U_{ii}, the computed solution $\hat{x}_i$ to $U_{ii}x_i = b_i$ satisfies

$$(U_{ii} + \Delta U_{ii})\hat{x}_i = b_i, \quad \|\Delta U_{ii}\| \leq c_2(b)u\|U_{ii}\| + O(u^2). \tag{2.9}$$

Again, this assumption is satisfied in our case when the QR factorization is used during the block back substitution. We recall in the following the error analysis result of the block LU factorization from [3], which applies as well to block LU_PRRP factorization from Algorithm 1.

THEOREM 2.1 (Demmel, Higham, Schreiber [3]). *Let $\hat{L}$ and $\hat{U}$ be the computed block factors of a matrix A of size $n \times n$ obtained from Algorithm BLU in* [3]. *Under assumptions* (2.8) *(with $\bar{A}_{11} = A_{11}$ and $\bar{A}_{21} = A_{21}$) and* (2.9), *the factorization and the computed solution $\hat{x}$ to $Ax = y$ satisfy*

$$\hat{L}\hat{U} = A + E, \quad \|E\| \leq c_3(n)u\left(\|A\| + \|\hat{L}\|\|\hat{U}\|\right) + O(u^2), \tag{2.10}$$

$$(A + \Delta A)\hat{x} = y, \quad \|\Delta A\| \leq c_4(n)u\left(\|A\| + \|\hat{L}\|\|\hat{U}\|\right) + O(u^2). \tag{2.11}$$

This result indicates that element growth is the only factor controlling the numerical stability of our algorithm. We note that if the GEPP factorization of the diagonal blocks is used during the block back substitution, assumption (2.9) is true if the additional growth factor generated by the GEPP factorization of the diagonal blocks is included in $c_2(b)$, which is propagated later in $c_4(n)$.

2.3. Experimental results. We measure the stability of the block LU_PRRP factorization experimentally on a large set of test matrices by using the growth factor, the normwise backward stability, and the componentwise backward stability. The tests are performed in MATLAB. We use a collection of matrices that includes ran-

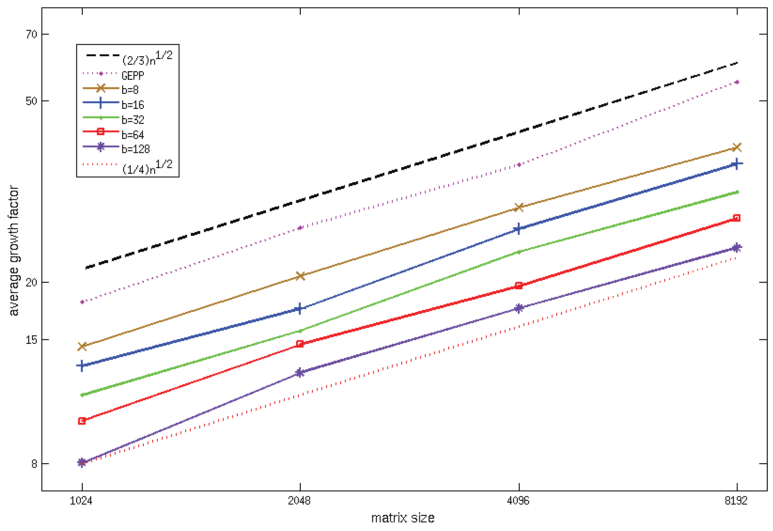

FIG. 2.1. *Growth factor $g_{blockLU\_PRRP}$ of the block LU_PRRP factorization of random matrices.*

dom matrices whose elements follow a normal distribution, the same set of special matrices used in the context of CALU (Table A.6 in [12]), and several pathological matrices on which GEPP fails because of large growth factors. For random matrices, in MATLAB notation, the test matrix is $A = \mathsf{randn}(n, n)$, and the right-hand side is $b = \mathsf{randn}(n, 1)$. The size of the matrix is chosen such that n is a power of 2, that is, $n = 2^k$, and the sample size is 10 if $k < 13$ and 3 if $k \geq 13$. The set of special matrices includes ill-conditioned matrices as well as sparse matrices. The pathological matrices considered are the Wilkinson matrix and two matrices arising from practical applications, presented by Foster [8] and Wright [24], for which the growth factor of GEPP grows exponentially. We recall that the Wilkinson matrix was constructed to attain the upper bound of the growth factor of GEPP [22, 17]. In the tests, in most of the cases, the panel factorization is performed by using the QR with column pivoting factorization instead of the strong RRQR factorization. This is because in practice $R_{12}^T(R_{11}^{-1})^T$ is already well bounded after performing the RRQR factorization with column pivoting. ($\|R_{12}^T(R_{11}^{-1})^T\|_{\max}$ is rarely bigger than 3 for random matrices.) Hence no additional swaps are needed to ensure that the elements are well bounded. However, for ill-conditionned special matrices (condition number $\geq 10^{14}$), to get small growth factors, we perform the panel factorization by using the strong RRQR factorization. In fact, for these cases, QR with column pivoting does not ensure a small bound for the max norm of $R_{12}^T(R_{11}^{-1})^T$.

We also test a generalized Wilkinson matrix.

The MATLAB code of such a matrix is detailed in Appendix F of the technical report [16].

The experimental results show that the block LU_PRRP factorization is very stable. Figure 2.1 displays the growth factor of block LU_PRRP for random matrices (randn) of size varying from 1024 to 8192 and for sizes of the panel varying from 8 to 128. We observe that the smaller the size of the panel is, the bigger the element growth is. In fact, for a smaller size of the panel, the number of panels and the number of updates on the trailing matrix is bigger, and this leads to a larger growth factor. But for all panel sizes, the growth factor of block LU_PRRP is smaller than

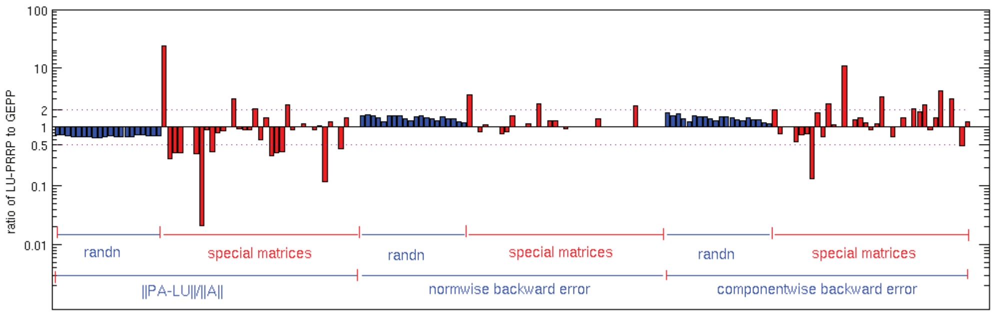

FIG. 2.2. *A summary of all our experimental data, showing the ratio between max(block LU_PRRP's backward error, machine epsilon) and max(GEPP's backward error, machine epsilon) for all the test matrices in our set. Each vertical bar represents such a ratio for one test matrix. Bars above* $10^0 = 1$ *mean that block LU_PRRP's backward error is larger, and bars below* 1 *mean that GEPP's backward error is larger.*

the growth factor of GEPP. For example, for a random matrix of size 4096 and a panel of size 64, the growth factor is only about 19, which is smaller than the growth factor obtained by GEPP, and as expected, much smaller than the theoretical upper bound of $(1.078)^{4032}$.

Table 6.1 in Appendix B presents more detailed results showing the stability of the block LU_PRRP factorization for a set of special matrices (as given in Table A.6 of [12]). There, we include different metrics, such as the norm of the factors, the value of their maximum element, and the backward error of the LU factorization. We evaluate the normwise backward stability by computing an accuracy test as performed in the high-performance Linpack (HPL) benchmark [7], and denoted as HPL3,

$$\text{HPL3} = ||Ax - b||_\infty / (\epsilon ||A||_\infty ||x||_\infty * N).$$

In HPL, the method is considered to be accurate if the value of HPL3 is smaller than 16. More generally, the value should be of order $O(1)$. For the block LU_PRRP factorization HPL3 is at most 1.60×10^{-2}. We also display the normwise backward error, using the 1-norm,

$$\eta := \frac{||r||}{||A||\ ||\hat{x}|| + ||b||}, \tag{2.12}$$

and the componentwise backward error

$$w := \max_i \frac{|r_i|}{(|A|\ |\hat{x}| + |b|)_i}, \tag{2.13}$$

where $\hat{x}$ is a computed solution to $Ax = b$, and $r = b - A\hat{x}$ is the computed residual. Our tests residuals are computed with double-working precision. We note that the results for random matrices are given in the technical report [16]. We recall here that in all our experiments we use the GEPP factorization of the diagonal blocks during the block back substitution.

Figure 2.2 summarizes all our stability results for block LU_PRRP. This figure displays the ratio of the maximum between the backward error and machine epsilon of block LU_PRRP versus GEPP. The backward error is measured using three metrics,

Table 2.2
Stability of the block LU_PRRP factorization of a Wilkinson matrix on which GEPP fails.

n	b	g_W	$\|\|U\|\|_1$	$\|\|U^{-1}\|\|_1$	$\|\|L\|\|_1$	$\|\|L^{-1}\|\|_1$	$\frac{\|\|PA-LU\|\|_F}{\|\|A\|\|_F}$
2048	128	1	1.02e+03	6.09e+00	1	1.95e+00	4.25e-20
	64	1	1.02e+03	6.09e+00	1	1.95e+00	5.29e-20
	32	1	1.02e+03	6.09e+00	1	1.95e+00	8.63e-20
	16	1	1.02e+03	6.09e+00	1	1.95e+00	1.13e-19
	8	1	1.02e+03	6.09e+00	1	1.95e+00	1.57e-19

Table 2.3
Stability of the block LU_PRRP factorization of a practical matrix (Foster) on which GEPP fails.

n	b	g_W	$\|\|U\|\|_1$	$\|\|U^{-1}\|\|_1$	$\|\|L\|\|_1$	$\|\|L^{-1}\|\|_1$	$\frac{\|\|PA-LU\|\|_F}{\|\|A\|\|_F}$
2048	128	2.66	1.28e+03	1.87e+00	1.92e+03	1.92e+03	4.67e-16
	64	2.66	1.19e+03	1.87e+00	1.98e+03	1.79e+03	2.64e-16
	32	2.66	4.33e+01	1.87e+00	2.01e+03	3.30e+01	2.83e-16
	16	2.66	1.35e+03	1.87e+00	2.03e+03	2.03e+00	2.38e-16
	8	2.66	1.35e+03	1.87e+00	2.04e+03	2.02e+00	5.36e-17

the relative error $\|PA - LU\|/\|A\|$, the normwise backward error η, and the componentwise backward error w of block LU_PRRP versus GEPP, and the machine epsilon. We take the maximum of the computed error with epsilon since smaller values are mostly roundoff error, and so taking ratios can lead to extreme values with little reliability [5]. Results for all the matrices in our test set are presented, that is, 20 random matrices for which detailed results are presented in [16], and 37 special matrices for which results are presented in Table 6.1. This figure shows that for random matrices, almost all ratios are between 0.5 and 2. For special matrices, there are few outliers, up to 23.71 for the backward error ratio of the special matrix *hadamard* and down to 2.12×10^{-2} for the backward error ratio of the special matrix *moler*.

We consider now pathological matrices on which GEPP fails. Table 2.2 presents results for the linear solver using the block LU_PRRP factorization for a Wilkinson matrix [23] of size 2048 with a size of the panel varying from 8 to 128. The growth factor is 1 and the relative error $\frac{||PA-LU||}{||A||}$ is on the order of 10^{-19}.

For the Foster matrix, it was shown that the growth factor of GEPP can be as large as $(\frac{2}{3})(2^{n-1} - 1)$, which is close to the maximum theoretical growth factor of GEPP of 2^{n-1}. Table 2.3 presents results for the linear solver using the block LU_PRRP factorization for a Foster matrix of size 2048 with a size of the panel varying from 8 to 128. (We choose the same parameters of the Foster matrix as the ones which give exponential growth factor for GEPP.) According to the obtained results, block LU_PRRP gives a modest growth factor of 2.66 for this practical matrix, while GEPP has a growth factor of 10^{18} for the same parameters.

For matrices arising from the two-point boundary value problems described by Wright, the results of the linear solver based on the block LU_PRRP factorization are presented in Table 2.4. Again, block LU_PRRP gives minimum possible pivot growth 1 for this practical matrix, compared to the GEPP method which leads to a growth factor of 10^{95} using the same parameters.

All the previous tests show that the block LU_PRRP factorization is very stable for random, and for more special, matrices, and it also gives modest growth factor for pathological matrices on which GEPP fails. We note that we were not able to

Table 2.4
Stability of the block LU_PRRP factorization on a practical matrix (Wright) on which GEPP fails.

n	b	g_W	$\|\|U\|\|_1$	$\|\|U^{-1}\|\|_1$	$\|\|L\|\|_1$	$\|\|L^{-1}\|\|_1$	$\frac{\|\|PA-LU\|\|_F}{\|\|A\|\|_F}$
2048	128	1	3.25e+00	8.00e+00	2.00e+00	2.00e+00	4.08e-17
	64	1	3.25e+00	8.00e+00	2.00e+00	2.00e+00	4.08e-17
	32	1	3.25e+00	8.00e+00	2.05e+00	2.07e+00	6.65e-17
	16	1	3.25e+00	8.00e+00	2.32e+00	2.44e+00	1.04e-16
	8	1	3.40e+00	8.00e+00	2.62e+00	3.65e+00	1.26e-16

find matrices for which block LU_PRRP attains the upper bound of $(1+\tau b)^{(n/b)-1}$ for the growth factor.

3. Communication avoiding block LU_PRRP.

In this section we present a version of the block LU_PRRP algorithm that minimizes communication, and so it will be more efficient than block LU_PRRP and GEPP on architectures where communication is expensive. We show in this section that this algorithm is more stable than CALU, an existing communication avoiding algorithm for computing the LU factorization [12].

3.1. Matrix algebra.

Block CALU_PRRP uses tournament pivoting, a strategy introduced in [11] that allows one to minimize communication. As in block LU_PRRP, at each step the factorization of the current panel is computed, and then the trailing matrix is updated. However, in block CALU_PRRP the panel factorization is performed in two steps. The first step, which is a preprocessing step, uses a reduction operation to identify b pivot rows with a minimum amount of communication. The strong RRQR factorization is the operator used at each node of the reduction tree to select a new set of b candidate pivot rows from the candidate pivot rows selected at previous stages of the reduction. The final b pivot rows are permuted into the first positions, and then the QR factorization with no pivoting of the transpose of the entire panel is computed.

In the following we illustrate tournament pivoting on the first panel, with the input matrix A partitioned as in (2.1). We consider that the first panel is partitioned into $P = 4$ blocks of rows,

$$A(:,1:b) = \begin{bmatrix} A_{00} \\ A_{10} \\ A_{20} \\ A_{30} \end{bmatrix}.$$

The preprocessing step uses a binary reduction tree in our example, and we number the levels of the binary reduction tree starting from 0. At the leaves of the reduction tree, a set of b candidate rows are selected from each block of rows A_{i0} by performing the strong RRQR factorization on the transpose of each block A_{i0}. This gives the following decomposition,

$$\begin{bmatrix} A_{00}^T\Pi_{00} \\ A_{10}^T\Pi_{10} \\ A_{20}^T\Pi_{20} \\ A_{30}^T\Pi_{30} \end{bmatrix} = \begin{bmatrix} Q_{00}R_{00} \\ Q_{10}R_{10} \\ Q_{20}R_{20} \\ Q_{30}R_{30} \end{bmatrix},$$

which can be written as

$$A(:,1:b)^T\bar{\Pi}_0 = A(:,1:b)^T \begin{bmatrix} \Pi_{00} & & & \\ & \Pi_{10} & & \\ & & \Pi_{20} & \\ & & & \Pi_{30} \end{bmatrix} = \begin{bmatrix} Q_{00}R_{00} \\ Q_{10}R_{10} \\ Q_{20}R_{20} \\ Q_{30}R_{30} \end{bmatrix},$$

where $\bar{\Pi}_0$ is a $m \times m$ permutation matrix with diagonal blocks of size $(m/P) \times (m/P)$, Q_{i0} is an orthogonal matrix of size $b \times b$, and each factor R_{i0} is a $b \times (m/P)$ upper triangular matrix.

At the next level of the binary reduction tree, two matrices A_{01} and A_{11} are formed by combining together two sets of candidate rows,

$$A_{01} = \begin{bmatrix} (\Pi_{00}^T A_{00})(1:b,1:b) \\ (\Pi_{10}^T A_{10})(1:b,1:b) \end{bmatrix},$$
$$A_{11} = \begin{bmatrix} (\Pi_{20}^T A_{20})(1:b,1:b) \\ (\Pi_{30}^T A_{30})(1:b,1:b) \end{bmatrix}.$$

Two new sets of candidate rows are identified by performing the strong RRQR factorization of each matrix A_{01} and A_{11},

$$A_{01}^T\Pi_{01} = Q_{01}R_{01},$$
$$A_{11}^T\Pi_{11} = Q_{11}R_{11},$$

where Π_{10}, Π_{11} are permutation matrices of size $2b \times 2b$, Q_{01}, Q_{11} are orthogonal matrices of size $b \times b$, and R_{01}, R_{11} are upper triangular factors of size $b \times 2b$.

The final b pivot rows are obtained by performing one last strong RRQR factorization on the transpose of the $b \times 2b$ matrix

$$A_{02} = \begin{bmatrix} (\Pi_{01}^T A_{01})(1:b,1:b) \\ (\Pi_{11}^T A_{11})(1:b,1:b) \end{bmatrix},$$

that is,

$$A_{02}^T\Pi_{02} = Q_{02}R_{02},$$

where Π_{02} is a permutation matrix of size $2b \times 2b$, Q_{02} is an orthogonal matrix of size $b \times b$, and R_{02} is an upper triangular matrix of size $b \times 2b$. This operation is performed at the root of the binary reduction tree, and this ends the first step of the panel factorization. In the second step, the final pivot rows identified by tournament pivoting are permuted to the first positions of A,

$$\bar{A} = \bar{\Pi}^T A = \bar{\Pi}_2^T\bar{\Pi}_1^T\bar{\Pi}_0^T A,$$

where the matrices $\bar{\Pi}_i$ are obtained by extending the permutation matrices to the dimension $m \times m$, that is,

$$\bar{\Pi}_1 = \begin{bmatrix} \bar{\Pi}_{01} & \\ & \bar{\Pi}_{11} \end{bmatrix}$$

with $\bar{\Pi}_{i1}$ for $i = 0, 1$ formed as

$$\bar{\Pi}_{i1} = \begin{bmatrix} \Pi_{i1}(1:b,1:b) & & \Pi_{i1}(1:b,b+1:2b) & \\ & I_{\frac{m}{P}-b} & & \\ \Pi_{i1}(b+1:2b,1:b) & & \Pi_{i1}(b+1:2b,b+1:2b) & \\ & & & I_{\frac{m}{P}-b} \end{bmatrix}$$

and

$$\bar{\Pi}_2 = \begin{bmatrix} \Pi_{02}(1:b,1:b) & & \Pi_{02}(1:b,b+1:2b) & \\ & I_{2\frac{m}{P}-b} & & \\ \Pi_{02}(b+1:2b,1:b) & & \Pi_{02}(b+1:2b,b+1:2b) & \\ & & & I_{2\frac{m}{P}-b} \end{bmatrix}.$$

Once the pivot rows are in the diagonal positions, the QR factorization with no pivoting is performed on the transpose of the first panel,

$$\bar{A}^T(1:b,:) = QR = Q \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}.$$

This factorization is used to update the trailing matrix, and the elimination of the first panel leads to the following decomposition (we use the same notation as in section 2):

$$\bar{A} = \begin{bmatrix} I_b & \\ \bar{A}_{21}\bar{A}_{11}^{-1} & I_{m-b} \end{bmatrix} \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ & \bar{A}_{22}^s \end{bmatrix},$$

where

$$\bar{A}_{22}^s = \bar{A}_{22} - \bar{A}_{21}\bar{A}_{11}^{-1}\bar{A}_{12}.$$

The block CALU_PRRP factorization continues the same procedure on the trailing matrix $\bar{A}_{22}^s$. As the block LU_PRRP factorization, the block CALU_PRRP factorization computes a block LU factorization of the input matrix A. A linear system is solved by using a block forward substitution and a block backward substitution, which will use the factorizations of the diagonal blocks of A. Note that the factors L and U obtained by the block CALU_PRRP factorization are different from the factors obtained by the block LU_PRRP factorization. The two algorithms use different pivot rows, and in particular the factor L of block CALU_PRRP is no longer bounded by a given threshold τ as in block LU_PRRP. This leads to a different worst case growth factor for block CALU_PRRP, which we will discuss in the following section.

The following figure displays the binary tree based tournament pivoting performed on the first panel using an arrow notation (as in [12]). The function $f(A_{ij})$ computes a strong RRQR of the matrix A_{ij}^T to select a set of b candidate rows. At each node of the reduction tree, two sets of b candidate rows are merged together and form a matrix A_{ij}, the function f is applied to A_{ij}, and another set of b candidate pivot rows is selected. While in this section we focused on binary trees, tournament pivoting can use any reduction tree, and this allows the algorithm to adapt on different architectures. Later in the paper we will also consider a flat reduction tree,

$$\begin{tikzcd}
A_{00} \arrow[r] & f(A_{00}) \arrow[dr] & & \\
 & & f(A_{01}) \arrow[ddr] & \\
A_{10} \arrow[r] & f(A_{10}) \arrow[ur] & & \\
 & & & f(A_{02}). \\
A_{20} \arrow[r] & f(A_{20}) \arrow[dr] & & \\
 & & f(A_{11}) \arrow[uur] & \\
A_{30} \arrow[r] & f(A_{30}) \arrow[ur] & &
\end{tikzcd}$$

3.2. Numerical stability of block CALU_PRRP. In this section we discuss the stability of block CALU_PRRP and we identify similarities with block LU_PRRP which hold in exact arithmetic. We also discuss the growth factor of block CALU_PRRP in exact arithmetic, and we show that its upper bound depends on the height of the reduction tree. For the same reduction tree, this upper bound is smaller than that obtained with CALU.

To address the numerical stability of block CALU_PRRP, we show that in exact arithmetic, performing block CALU_PRRP on a matrix A is equivalent to performing block LU_PRRP on a larger matrix A_{LU_PRRP}, which is formed by blocks of A (sometimes slightly perturbed) and blocks of zeros. This reasoning is also used in [12] to show the same equivalence between CALU and GEPP. While this similarity is explained in detail in [12], here we focus only on the first step of the block CALU_PRRP factorization. We explain the construction of the larger matrix A_{LU_PRRP} to expose the equivalence between the first step of the block CALU_PRRP factorization of A and the block LU_PRRP factorization of A_{LU_PRRP}.

Consider a nonsingular matrix A of size $m \times n$ and the first step of its block CALU_PRRP factorization using a general reduction tree of height H. Tournament pivoting selects b candidate rows at each node of the reduction tree by using the strong RRQR factorization. Each such factorization leads to an L factor which is bounded locally by a given threshold τ. However, this bound is not guaranteed globally. When the factorization of the first panel is computed using the b pivot rows selected by tournament pivoting, the L factor will not be bounded by τ. This results in a larger growth factor than the one obtained with the block LU_PRRP factorization. Recall that in block LU_PRRP, the strong RRQR factorization is performed on the transpose of the entire panel, and so every entry of the obtained lower triangular factor L is bounded in absolute value by τ.

However, we show now that in exact arithmetic, the growth factor obtained after the first step of the block CALU_PRRP factorization is bounded by $(1 + \tau b)^{H+1}$. Consider a row j, and let $A^s(j, b+1 : n)$ be the updated row obtained after the first step of elimination of block CALU_PRRP. Suppose that row j of A is a candidate row at level $k-1$ of the reduction tree, and so it participates in the strong RRQR factorization computed at a node s_k at level k of the reduction tree, but it is not selected as a candidate row by this factorization. We refer to the matrix formed by the candidate rows at node s_k as $\bar{A}_k$. Hence, row j is not used to form the matrix $\bar{A}_k$. Similarly, for every node i on the path from node s_k to the root of the reduction tree of height H, we refer to the matrix formed by the candidate rows selected by strong RRQR as $\bar{A}_i$. Note that in practice it can happen that one of the blocks of the panel is singular, while the entire panel is nonsingular. In this case strong RRQR will select less than b linearly independent rows that will be passed along the reduction tree. However, for simplicity, we assume in the following that the matrices $\bar{A}_i$ are nonsingular. For a more general solution, the reader can consult [12].

Let Π be the permutation returned by the tournament pivoting strategy performed on the first panel, that is, the permutation that puts the matrix $\bar{A}_H$ on diagonal. The following equation is satisfied:

$$\begin{pmatrix} \bar{A}_H & \bar{\bar{A}}_H \\ A(j,1:b) & A(j,b+1:n) \end{pmatrix} = \begin{pmatrix} I_b & \\ A(j,1:b)\bar{A}_H^{-1} & 1 \end{pmatrix} \cdot \begin{pmatrix} \bar{A}_H & \bar{\bar{A}}_H \\ & A^s(j,b+1:n) \end{pmatrix}, \tag{3.1}$$

where

$$\begin{aligned} \bar{A}_H &= (\Pi A)(1:b,1:b), \\ \bar{\bar{A}}_H &= (\Pi A)(1:b,b+1:n). \end{aligned}$$

The updated row $A^s(j,b+1:n)$ can be also obtained by performing block LU_PRRP on a larger matrix A_{LU_PRRP} of dimension $((H-k+1)b+1) \times ((H-k+1)b+1)$,

$$\begin{aligned} A_{LU\_PRRP} &= \begin{pmatrix} \bar{A}_H & & & & & \bar{\bar{A}}_H \\ \bar{A}_{H-1} & \bar{A}_{H-1} & & & & \\ & \bar{A}_{H-2} & \bar{A}_{H-2} & & & \\ & & \ddots & \ddots & & \\ & & & \bar{A}_k & \bar{A}_k & \\ & & & & (-1)^{H-k}A(j,1:b) & A(j,b+1:n) \end{pmatrix} \\ &= \begin{pmatrix} I_b & & & & & \\ \bar{A}_{H-1}\bar{A}_H^{-1} & I_b & & & & \\ & \bar{A}_{H-2}\bar{A}_{H-1}^{-1} & I_b & & & \\ & & \ddots & \ddots & & \\ & & & \bar{A}_k\bar{A}_{k+1}^{-1} & I_b & \\ & & & & (-1)^{H-k}A(j,1:b)\bar{A}_k^{-1} & 1 \end{pmatrix} \\ &\quad \cdot \begin{pmatrix} \bar{A}_H & & & & & \bar{\bar{A}}_H \\ & \bar{A}_{H-1} & & & & \bar{\bar{A}}_{H-1} \\ & & \bar{A}_{H-2} & & & \bar{\bar{A}}_{H-2} \\ & & & \ddots & & \vdots \\ & & & & \bar{A}_k & \bar{\bar{A}}_k \\ & & & & & A^s(j,b+1:n) \end{pmatrix}, \end{aligned} \tag{3.2}$$

where

$$\bar{\bar{A}}_{H-i} = \begin{cases} \bar{\bar{A}}_H & \text{if } i = 0, \\ -\bar{A}_{H-i}\bar{A}_{H-i+1}^{-1}\bar{\bar{A}}_{H-i+1} & \text{if } 0 < i \le H-k. \end{cases} \tag{3.3}$$

Equation (3.2) can be easily verified in exact arithmetic, since

$$\begin{aligned} A^s(j,b+1:n) &= A(j,b+1:n) - (-1)^{H-k}A(j,1:b)\bar{A}_k^{-1}(-1)^{H-k}\bar{\bar{A}}_k \\ &= A(j,b+1:n) - A(j,1:b)\bar{A}_k^{-1}\bar{A}_k\bar{A}_{k+1}^{-1}\dots\bar{A}_{H-2}\bar{A}_{H-1}^{-1}\bar{A}_{H-1}\bar{A}_H^{-1}\bar{\bar{A}}_H \\ &= A(j,b+1:n) - A(j,1:b)\bar{A}_H^{-1}\bar{\bar{A}}_H. \end{aligned}$$

Equations (3.1) and (3.2) show that in exact arithmetic, the Schur complement obtained after each step of performing the block CALU_PRRP factorization of a matrix A is equivalent to the Schur complement obtained after performing the LU_PRRP

TABLE 3.1

Bounds for the growth factor obtained from factoring a matrix of size $m \times (b+1)$ and a matrix of size $m \times n$ using block CALU_PRRP, block LU_PRRP, CALU, and GEPP. Block CALU_PRRP and CALU use a reduction tree of height H, and the bounds hold in exact arithmetic. The strong RRQR used in block LU_PRRP and block CALU_PRRP is based on a threshold τ.

	Matrix of size $m \times n$			
	CALU	Block CALU_PRRP	GEPP	Block LU_PRRP
Upper bound	$2^{n(H+1)-1}$	$(1+\tau b)^{\frac{n}{b}(H+1)-1}$	2^{n-1}	$(1+\tau b)^{\frac{n}{b}-1}$

factorization of a larger matrix A_{LU_PRRP}, formed by blocks of A (sometimes slightly perturbed) and blocks of zeros. More generally, this implies that the entire block CALU_PRRP factorization of A is equivalent to the block LU_PRRP factorization of a larger and very sparse matrix, formed by blocks of A and blocks of zeros. (We omit the proofs here, since they are similar to the proofs presented in [12].)

Equation (3.2) is used to derive the upper bound of the growth factor of block CALU_PRRP from the upper bound of the growth factor of block LU_PRRP. The elimination of each row of the first panel using block CALU_PRRP can be obtained by performing block LU_PRRP on a matrix of maximum dimension $m \times b(H+1)$. Hence the upper bound of the growth factor obtained after one step of block CALU_PRRP is $(1+\tau b)^{H+1}$. In exact arithmetic, this leads to an upper bound of $(1+\tau b)^{\frac{n}{b}(H+1)-1}$ for a matrix of size $m \times n$.

Table 3.1 summarizes the bounds of the growth factor of block CALU_PRRP derived in this section, and also recalls the bounds of block LU_PRRP, GEPP, and CALU, the communication avoiding version of GEPP. For block LU_PRRP and block CALU_PRRP, the values displayed correspond to the growth factor of the block L and U factors. As discussed in section 2 already, block LU_PRRP is more stable than GEPP in terms of worst case growth factor. From Table 3.1, it can be seen that for a reduction tree of a same height, block CALU_PRRP is more stable than CALU.

In the following we show that in exact arithmetic, block CALU_PRRP can have a smaller worst case growth factor than GEPP. Consider a parallel version of block CALU_PRRP based on a binary reduction tree of height $H = \log P$, where P is the number of processors. The upper bound of the growth factor becomes $(1+\tau b)^{(n/b)(logP+1)-1}$, which is smaller than $2^{n(logP+1)-1}$, the upper bound of the growth factor of CALU. For example, if the threshold is $\tau = 2$, the panel size is $b = 64$, and the number of processors is $P = 128 = 2^7$, then the growth factor of block CALU_PRRP is $\approx (1.8)^n$. This quantity is much smaller than 2^{7n}, the upper bound of CALU, and even smaller than the worst case growth factor of GEPP of 2^{n-1}. In general, the upper bound of block CALU_PRRP can be smaller than the one of GEPP, if the different parameters τ, H, and b are chosen such that the condition

$$H \leq \frac{b}{(\log b + \log \tau)} \tag{3.4}$$

is satisfied. For a binary tree of height $H = \log P$, it corresponds to $\log P \leq b/(\log b + \log \tau)$. This condition can be satisfied in practice by choosing b and τ appropriately for a given number of processors P. For example, when $P \leq 512$, $b = 64$, and $\tau = 2$, the condition (3.4) is satisfied, and the worst case growth factor of block CALU_PRRP is smaller than the one of GEPP. However, for a sequential version of block CALU_PRRP using a flat tree of height $H = n/b$, the condition to be satisfied becomes $n \leq b^2/(\log b + \log \tau)$, which is more restrictive. In practice, the

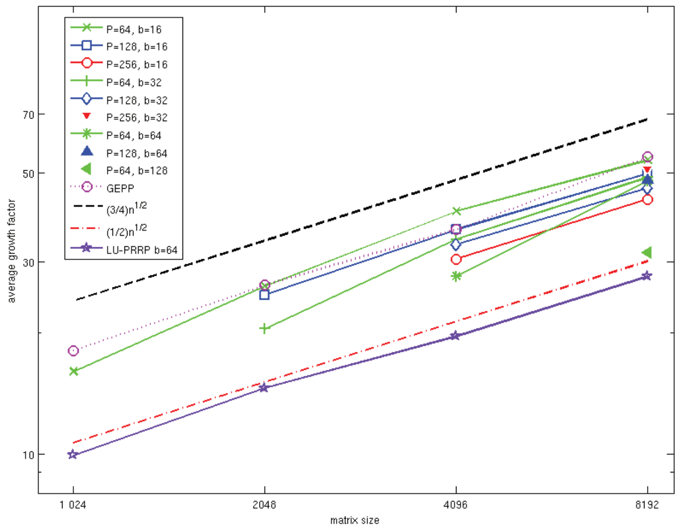

FIG. 3.1. *Growth factor $g_{blockLU\_PRRP}$ of binary tree based block CALU_PRRP for random matrices.*

size of b is chosen depending on the size of the memory, and it might be the case that it will not satisfy the condition in (3.4).

3.3. Experimental results. In this section we present experimental results and show that block CALU_PRRP is stable in practice and compare them with those obtained from CALU and GEPP in [12]. We present results for both the binary tree scheme and the flat tree scheme. As in section 2, we perform our tests on matrices whose elements follow a normal distribution. To measure the stability of block CALU_PRRP, we discuss several metrics, that concern the LU decomposition and the linear solver using it, such as the growth factor and normwise and componentwise backward errors. We also perform tests on several special matrices including sparse matrices.

Figure 3.1 displays the values of the growth factor $g_{blockLU\_PRRP}$ for random matrices of the binary tree based block CALU_PRRP with different block sizes b and different number of processors P. As explained in section 3.1, the block size determines the size of the panel, while the number of processors determines the number of block rows in which the panel is partitioned. This corresponds to the number of leaves of the binary tree. We observe that the growth factor of binary tree based block CALU_PRRP is in most of the cases better than that of GEPP. The curves of the growth factor lie between $\frac{1}{2}n^{1/2}$ and $\frac{3}{4}n^{1/2}$ in our tests on random matrices. These results show that binary tree based block CALU_PRRP is stable and the growth factor values obtained for the different layouts are better than those obtained with binary tree based CALU. Figure 3.1 includes also the growth factor of the block LU_PRRP method with a panel of size $b = 64$. We note that these results are better than those of binary tree based block CALU_PRRP.

Figure 3.2 displays the values of the growth factor g_W for flat tree based block CALU_PRRP with a block size b varying from 8 to 128. The growth factor g_W is decreasing with increasing the panel size b. We note that the curves of the growth factor lie between $\frac{1}{4}n^{1/2}$ and $\frac{3}{4}n^{1/2}$ in our tests on random matrices. We also note that

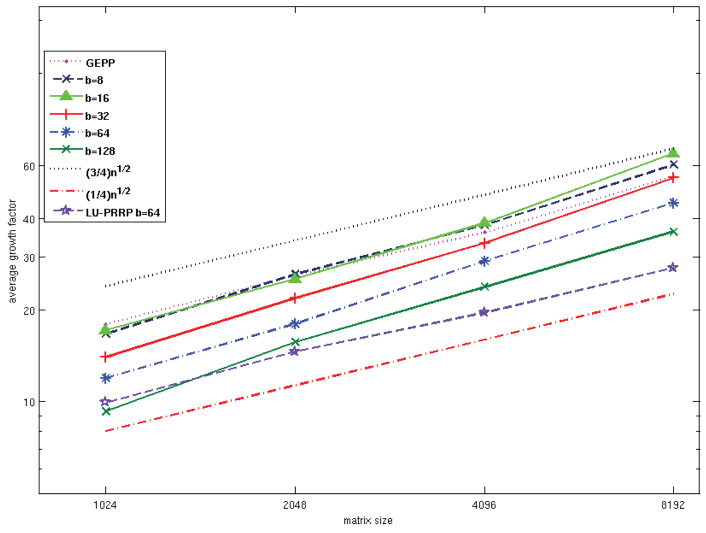

FIG. 3.2. *Growth factor g_W of flat tree based block CALU_PRRP for random matrices.*

the results obtained with the block LU_PRRP method with a panel of size $b = 64$ are better than those of flat tree based block CALU_PRRP. The growth factors of both binary tree based and flat tree based block CALU_PRRP have similar (sometimes better) behavior compared to the growth factors of GEPP.

Table 6.2 in Appendix B presents results for the linear solver using binary tree based block CALU_PRRP, together with binary tree based CALU and GEPP for comparison. Results for the flat tree based block CALU_PRRP are not included here, but can be found in the technical report [16]. We note that for the binary tree based block CALU_PRRP, when $m/P = b$, the algorithm uses only $P1 = m/(b+1)$ processors, since to perform a strong RRQR on a given block, the number of its rows should be at least the number of its columns plus one. Table 6.2 includes some other metrics, such as the norm of the factors, the norm of the inverse of the factors, their conditioning, the value of their maximum element, and the backward error of the LU factorization.

Figure 3.3 summarizes all our stability results for the block CALU_PRRP factorization based on both binary tree and flat tree schemes, which is analogous to Figure 2.2. Results for all the matrices in our test set are presented, that is, 25 random matrices for binary tree base block CALU_PRRP, 20 random matrices for flat tree based block CALU_PRRP, and 37 special matrices. We note that all these results are detailed in [16]. As can be seen, nearly all ratios are between 0.5 and 2.5 for random matrices. However, there are a few outliers, for example, the relative error ratio has values between 24.2 for the special matrix *hadamard* and 5.8×10^{-3} for the special matrix *moler*.

For the set of pathological matrices considered in section 2 on which GEPP fails, our numerical experiments results are the following. For the Wilkinson matrix, both CALU and block CALU_PRRP based on flat and binary tree give modest element growth. For the generalized Wilkinson matrix, the Foster matrix, and the Wright matrix, CALU fails with both flat tree and binary tree reduction schemes. However, both flat tree based and binary tree based block CALU_PRRP are stable for these matrices.

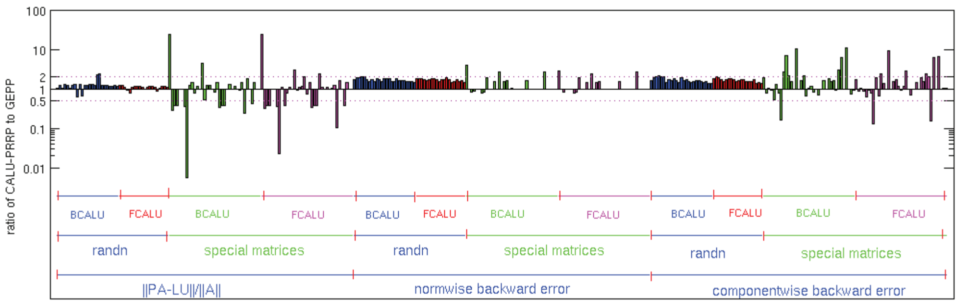

FIG. 3.3. *A summary of all our experimental data, showing the ratio of max(block CALU_PRRP's backward error, machine epsilon) to max(GEPP's backward error, machine epsilon) for all the test matrices in our set. Each vertical bar represents such a ratio for one test matrix.*

Through the results detailed in this section, in Appendix B, and in [16], we conclude that binary tree based and flat tree based block CALU_PRRP is very stable for both random matrices and more special ones, including dense and sparse matrices.

4. Lower bounds on communication. In this section we focus on the parallel block CALU_PRRP algorithm based on a binary reduction tree, and we show that it minimizes the communication between different processors of a parallel computer. For this, we use known lower bounds on the communication performed during the LU factorization of a dense matrix of size $n \times n$, which are

$$\# \text{ words\_moved} = \Omega\left(\frac{n^3}{\sqrt{M}}\right), \tag{4.1}$$

$$\# \text{ messages} = \Omega\left(\frac{n^3}{M^{\frac{3}{2}}}\right), \tag{4.2}$$

where # *words_moved* refers to the volume of communication, # *messages* refers to the number of messages exchanged, and M refers to the size of the memory (the fast memory in the case of a sequential algorithm, or the memory per processor in the case of a parallel algorithm). These lower bounds were first introduced for dense matrix multiplication [14, 15], generalized later to LU factorization [4], and then to almost all direct linear algebra [1]. Note that these lower bounds apply to algorithms based on orthogonal transformations under certain conditions [1]. However, this is not relevant to our case, since block CALU_PRRP uses orthogonal transformations that need communication only to select pivot rows, while the update of the trailing matrix is still performed as in the classic LU factorization algorithm. We note that at the end of the preprocessing step, each processor working on the panel has the final set of b pivot rows. Thus all these processors perform in parallel a QR factorization without pivoting on the transpose of the final $b \times b$ block. After this phase each processor below the diagonal has the R_{11} factor, computes his chunk of R_{12}, and finally computes his block of L_{21} factor as detailed previously. Therefore, the QR factorization applied to the transpose of the panel does not imply any communication. Hence, the lower bounds from (4.1) and (4.2) are valid for block CALU_PRRP.

We estimate now the cost of computing in parallel the block CALU_PRRP factorization of a matrix A of size $m \times n$. The matrix is distributed on a grid of $P = P_r \times P_c$ processors using a two-dimensional (2D) block cyclic layout. We use the following performance model. Let γ be the cost of performing a floating point operation, and let

TABLE 4.1

Performance estimation of parallel (binary tree based) block CALU_PRRP, parallel CALU, and PDGETRF routine when factoring an $m \times n$ matrix, $m \geq n$. The input matrix is distributed using a 2D block cyclic layout on a $P_r \times P_c$ grid of processors. Some lower order terms are omitted.

	Parallel block CALU_PRRP
# messages	$\frac{3n}{b}\log_2 P_r + \frac{2n}{b}\log_2 P_c$
# words	$\left(nb + \frac{3}{2}\frac{n^2}{P_c}\right)\log_2 P_r + \frac{1}{P_r}\left(mn - \frac{n^2}{2}\right)\log_2 P_c$
# flops	$\frac{1}{P}\left(mn^2 - \frac{n^3}{3}\right) + \frac{2}{P_r}\left(2mn - n^2\right)b + \frac{2nb^2}{3}(5\log_2 P_r - 1)$
	Parallel CALU
# messages	$\frac{3n}{b}\log_2 P_r + \frac{3n}{b}\log_2 P_c$
# words	$\left(nb + \frac{3n^2}{2P_c}\right)\log_2 P_r + \frac{1}{P_r}\left(mn - \frac{n^2}{2}\right)\log_2 P_c$
# flops	$\frac{1}{P}\left(mn^2 - \frac{n^3}{3}\right) + \frac{1}{P_r}\left(2mn - n^2\right)b + \frac{n^2 b}{2P_c} + \frac{nb^2}{3}(5\log_2 P_r - 1)$
	PDGETRF
# messages	$2n\left(1 + \frac{2}{b}\right)\log_2 P_r + \frac{3n}{b}\log_2 P_c$
# words	$\left(\frac{nb}{2} + \frac{3n^2}{2P_c}\right)\log_2 P_r + \log_2 P_c \frac{1}{P_r}\left(mn - \frac{n^2}{2}\right)$
# flops	$\frac{1}{P}\left(mn^2 - \frac{n^3}{3}\right) + \frac{1}{P_r}\left(mn - \frac{n^2}{2}\right)b + \frac{n^2 b}{2P_c}$

$\alpha + \beta w$ be the cost of sending a message of size w words, where α is the latency cost and β is the inverse of the bandwidth. Then, the total running time of an algorithm is estimated to be

$$\alpha \cdot (\#\text{ messages}) + \beta \cdot (\#\text{ words\_moved}) + \gamma \cdot (\#\text{ flops}),$$

where #messages, #words_moved, and #flops are counted along the critical path of the algorithm.

Table 4.1 displays the performance of parallel block CALU_PRRP. (A detailed estimation of the counts is presented in [16, Appendix D].) It also recalls the performance of two existing algorithms, the PDGETRF routine from ScaLAPACK which implements GEPP, and the CALU factorization. All three algorithms have the same volume of communication, since it is known that PDGETRF already minimizes the volume of communication. However, the number of messages of both block CALU_PRRP and CALU is smaller by a factor of the order of b than the number of messages of PDGETRF. This improvement is achieved thanks to tournament pivoting. In fact, partial pivoting, as used in the routine PDGETRF, leads to an $O(n \log P)$ number of messages, and because of this, GEPP cannot minimize the number of messages.

Compared to CALU, block CALU_PRRP sends a small factor of less messages (depending on P_r and P_c). If we consider the additional GEPP of the diagonal blocks, then the entire LU factorization performs $\frac{1}{P_r}\left(2mn - n^2\right)b + \frac{nb^2}{3}(5\log_2 P_r + 1)$ more flops than CALU (which represents a lower order term). This is because block CALU_PRRP uses the strong RRQR factorization at every node of the reduction tree of every panel factorization, while CALU uses GEPP.

We show now that block CALU_PRRP is optimal in terms of communication. We choose optimal values of the parameters P_r, P_c, and b, as used in CAQR [4] and CALU [12], that is,

$$P_r = \sqrt{\frac{mP}{n}}\ ,\ P_c = \sqrt{\frac{nP}{m}}\ \text{ and } b = \frac{1}{4}\log^{-2}\left(\sqrt{\frac{mP}{n}}\right) \cdot \sqrt{\frac{mn}{P}} = \log^{-2}\left(\frac{mP}{n}\right) \cdot \sqrt{\frac{mn}{P}}.$$

Table 4.2
Performance estimation of parallel (binary tree based) block CALU_PRRP and CALU with an optimal layout. The matrix factored is of size $n \times n$. Some lower order terms are omitted.

	Parallel block CALU_PRRP with optimal layout	Lower bound
# messages	$\frac{5}{2}\sqrt{P}\log^3 P$	$\Omega(\sqrt{P})$
# words	$\frac{n^2}{\sqrt{P}}\left(\frac{1}{2}\log^{-1}P + \log P\right)$	$\Omega(\frac{n^2}{\sqrt{P}})$
# flops	$\frac{1}{P}\frac{2n^3}{3} + \frac{2n^3}{P\log^2 P} + \frac{5n^3}{3P\log^3 P}$	$\frac{1}{P}\frac{2n^3}{3}$
	Parallel CALU with optimal layout	Lower bound
# messages	$3\sqrt{P}\log^3 P$	$\Omega(\sqrt{P})$
# words	$\frac{n^2}{\sqrt{P}}\left(\frac{1}{2}\log^{-1}P + \log P\right)$	$\Omega(\frac{n^2}{\sqrt{P}})$
# flops	$\frac{1}{P}\frac{2n^3}{3} + \frac{3n^3}{2P\log^2 P} + \frac{5n^3}{6P\log^3 P}$	$\frac{1}{P}\frac{2n^3}{3}$

For a square matrix of size $n \times n$, the optimal parameters are,

$$P_r = \sqrt{P}\ ,\ P_c = \sqrt{P}\ \text{ and } b = \frac{1}{4}\log^{-2}\left(\sqrt{P}\right)\cdot\frac{n}{\sqrt{P}} = \log^{-2}(P)\cdot\frac{n}{\sqrt{P}}.$$

Table 4.2 presents the performance estimation of parallel block CALU_PRRP and parallel CALU when using the optimal layout. It also recalls the lower bounds on communication from (4.1) and (4.2) when the size of the memory per processor is on the order of n^2/P. Both block CALU_PRRP and CALU attain the lower bounds on the number of words and on the number of messages, modulo polylogarithmic factors. Note that the optimal layout allows one to reduce communication, while keeping the number of extra floating point operations performed due to tournament pivoting as a lower order term. While in this section we focused on minimizing communication between the processors of a parallel computer, it is straightforward to show that the usage of a flat tree during tournament pivoting allows block CALU_PRRP to minimize communication between different levels of the memory hierarchy in the sequential case.

5. Less stable factorizations that can also minimize communication.

In this section, we briefly present two alternative algorithms that are based on panel strong RRQR pivoting and that are conceived such that they can minimize communication. But we will see that they can be unstable in practice. These algorithms are also based on block algorithms that factor the input matrix by traversing panels of size b. The main difference between them and block CALU_PRRP is the panel factorization, which is performed only once in the alternative algorithms.

We present first a parallel alternative algorithm, which we refer to as block parallel LU_PRRP. At each step of the block factorization, the panel is partitioned into P block-rows $[A_0; A_1; \dots; A_{P-1}]$. The blocks below the diagonal $b\times b$ block of the current panel are eliminated by performing a binary tree of strong RRQR factorizations. At the leaves of the tree, the elements below the diagonal block of each block A_i are eliminated using strong RRQR. The elimination of each such block row is followed by the update of the corresponding block row of the trailing matrix.

The algorithm continues by performing the strong RRQR factorization of pairs of $b \times b$ blocks stacked atop one another, until all the blocks below the diagonal block are eliminated and the corresponding trailing matrices are updated. The algebra of the block parallel LU_PRRP algorithm is detailed in Appendix E of the technical report [16], while in Figure 5.1 we illustrate one step of the factorization by using an arrow

$$\begin{tikzcd}
g(A_{00}) \arrow[dr] & & \\
& g(A_{01}) \arrow[ddr] & \\
g(A_{10}) \arrow[ur] & & \\
& & g(A_{02}) \\
g(A_{20}) \arrow[dr] & & \\
& g(A_{11}) \arrow[uur] & \\
g(A_{30}) \arrow[ur] & &
\end{tikzcd}$$

FIG. 5.1. *Block parallel LU_PRRP.*

$$\begin{tikzcd}
A_{00} \arrow[r] & g(A_{00}) \arrow[r] & g(A_{01}) \arrow[r] & g(A_{02}) \arrow[r] & g(A_{03}) \\
A_{10} \arrow[urr] & & & & \\
A_{20} \arrow[uurrr] & & & & \\
A_{30} \arrow[uuurrrr] & & & &
\end{tikzcd}$$

FIG. 5.2. *Block pairwise LU_PRRP.*

notation, where the function $g(A_{ij})$ computes a strong RRQR on the matrix A_{ij}^T and updates the trailing matrix in the same step.

A sequential version of the algorithm is based on the usage of a flat tree, and we refer to this algorithm as block pairwise LU_PRRP. Using the arrow notation, Figure 5.2 illustrates the elimination of one panel.

The block parallel LU_PRRP and the block pairwise LU_PRRP algorithms have similarities with the block parallel pivoting and the block pairwise pivoting algorithms. These latter two algorithms were shown to be potentially unstable in [12]. There is a main difference between all these alternative algorithms and algorithms that compute a classic LU factorization as GEPP, block LU_PRRP, and their communication avoiding variants. The alternative algorithms compute a factorization in the form of a product of lower triangular factors and an upper triangular factor. And the elimination of each column leads to a rank update of the trailing matrix larger than one. It is thought in [21] that the rank-1 update property of algorithms that compute an LU factorization inhibits potential element growth during the factorization, while a large rank update might lead to an unstable factorization.

Note, however, that at each step of the factorization, block parallel and block pairwise LU_PRRP use at each level of the reduction tree original rows of the active matrix. Block parallel pivoting and block pairwise pivoting algorithms use U factors previously computed to achieve the factorization, and this could potentially lead to a faster propagation of ill-conditioning.

The upper bound of the growth factor of both block parallel and block pairwise LU_PRRP is $(1+\tau b)^{\frac{n}{b}-1}$, since for every panel factorization, a row is updated only once. Hence, they have the same bounds as the block LU_PRRP factorization, and smaller than that of the block CALU_PRRP factorization. Despite this, they are less stable than the block CALU_PRRP factorization. Figures 5.3 and 5.4 display the growth factor of block parallel LU_PRRP and block pairwise LU_PRRP for matrices following a normal distribution. In Figure 5.3, the number of processors P on which each panel is partitioned is varying from 16 to 32, and the block size b is varying from 2 to 16. The matrix size varies from 64 to 2048, but we have observed the same behavior for matrices of size up to 8192. When the number of processors P is equal to 1, the block parallel LU_PRRP corresponds to the block LU_PRRP factorization. The results show that there are values of P and b for which this method can be very unstable. For the sizes of matrices tested, when b is chosen such that the blocks at the leaves of the reduction tree have more than $2b$ rows, the number of processors

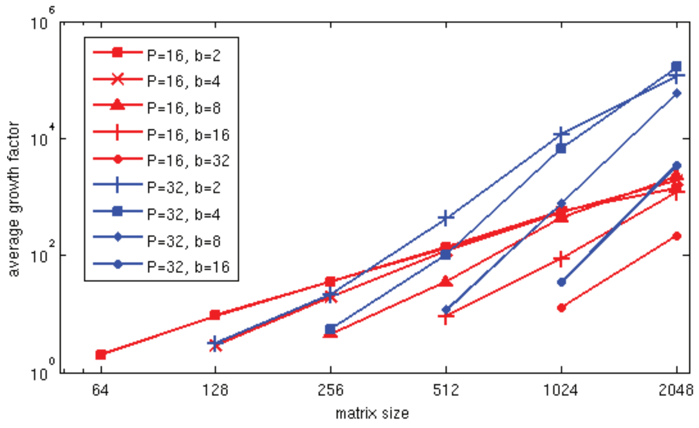

FIG. 5.3. *Growth factor of block parallel LU_PRRP for varying block size b and number of processors P.*

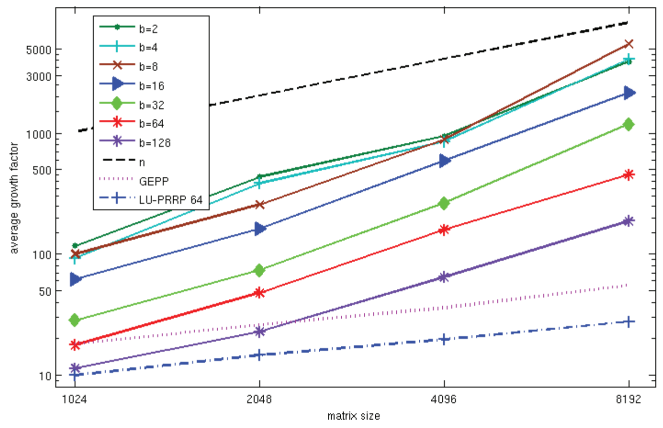

FIG. 5.4. *Growth factor of block pairwise LU_PRRP for varying matrix size and varying block size b.*

P has an important impact, the growth factor increases with increasing P, and the method is unstable.

In Figure 5.4, the matrix size varies from 1024 to 8192. For a given matrix size, the growth factor increases with decreasing the size of the panel b, as one could expect. We note that the growth factor of block pairwise LU_PRRP is larger than that obtained with the block CALU_PRRP factorization based on a flat tree scheme. But it stays smaller than the size of the matrix n for different panel sizes. Hence, this method is more stable than block parallel LU_PRRP. Further investigation is required to conclude on the stability of these methods.

6. Conclusions.

This paper introduces block LU_PRRP, a block LU factorization algorithm based on PRRP. This algorithm is more stable than GEPP in terms of worst case growth factor. It is also very stable in practice for various classes of matrices, including random matrices and a set of more special matrices. Its communication avoiding version, block CALU_PRRP, is also more stable in terms of worst case growth factor (computed in exact arithmetic) than CALU, the communication avoiding version of GEPP. More importantly, there are cases of interest for which the upper bound of the growth factor of block CALU_PRRP is smaller than that of GEPP for several cases of interest. Extensive experiments show that block CALU_PRRP is very stable in practice and leads to results of the same order of magnitude as GEPP, sometimes even better.

Our future work focuses on two main directions. The first direction investigates the design of a communication avoiding algorithm that has smaller bounds on the growth factor than that of GEPP in general. The second direction focuses on estimating the performance of block CALU_PRRP on parallel machines based on multicore processors, and comparing it with the performance of CALU.

Appendix A. We briefly describe strong RRQR introduced by Gu and Eisenstat in [13]. This factorization will be used in our new LU decomposition algorithm, which aims to obtain an upper bound of the growth factor smaller than GEPP (see section 2.) Consider a given threshold $\tau > 1$ and an $h \times p$ matrix B with $p > h$; a strong RRQR factorization on a matrix B gives (with an empty $(2,2)$ block)

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

where $\|R_{11}^{-1}R_{12}\|_{\max} \leq \tau$ with $\| \,.\, \|_{\max}$ being the biggest entry of a given matrix in absolute value. This factorization can be computed by a classical QR factorization with column pivoting followed by a limited number of additional swaps and QR updates if necessary.

The while loop in Algorithm 2 interchanges any pairs of columns that can increase $|det(R_{11})|$ by at least a factor τ. At most $O(\log_\tau n)$ such interchanges are necessary before Algorithm 2 finds a strong RRQR factorization. The QR factorization of $B^T\Pi$ can be computed numerically via efficient and numerically stable QR updating procedures. See [13] for details.

Algorithm 2. Strong RRQR.

```
1: Compute $B^T\Pi = QR$ using the classical RRQR with column pivoting
2: while there exist i and j such that $|(R_{11}^{-1}R_{12})_{ij}| > \tau$ do
3:     Set $\Pi = \Pi\Pi_{ij}$ and compute the QR factorization of $R\,\Pi_{ij}$ (QR updates)
4: end while
Ensure: $B^T\Pi = QR$ with $\|R_{11}^{-1}R_{12}\|_{\max} \leq \tau$
```

Appendix B. We present experimental results for the block LU_PRRP factorization and the binary tree based block CALU_PRRP. We show results obtained for the LU decomposition and the linear solver. Tables 6.1 and 6.3 display the results obtained for the special matrices presented in Table A.6 in [12]. They show the growth factor, the norm of the factor L and U and their inverses, and the relative error of the decomposition. The size of the tested matrices is n = 4096. For block LU_PRRP the size of the panel is $b = 8$. For binary tree based block CALU_PRRP we use $P = 64$ and $b = 8$, and this means that the size of the matrices used at the leaves of the reduction tree is 64×8.

Table 6.2 presents results for the linear solver using binary tree based block CALU_PRRP, together with binary tree based CALU and GEPP for comparison.

The tables are presented in the following order:

- Table 6.1. Stability of the LU decomposition for block LU_PRRP on special matrices.
- Table 6.2. Stability of the linear solver using binary tree based block CALU_PRRP, binary tree based CALU, and GEPP.
- Table 6.3. Stability of the LU decomposition for binary tree based block CALU_PRRP on special matrices.

TABLE 6.1
Stability of the LU decomposition for block LU_PRRP on special matrices ($\tau = 2$).

	Matrix	cond(A,2)	g_W	$\|\|L\|\|_1$	$\|\|L^{-1}\|\|_1$	$\|\|U\|\|_1$	$\|\|U^{-1}\|\|_1$	$\frac{\|\|PA-LU\|\|_F}{\|\|A\|\|_F}$	η	w_b	N_{IR}
Well-conditioned	hadamard	1.0E+0	5.1E+02	5.1E+02	2.5E+13	5.6E+04	5.1E+00	4.4E-15	1.6E-16	1.7E-15	2
	house	1.0E+0	7.3E+00	7.6E+02	7.5E+02	3.7E+02	5.0E+01	5.3E-16	3.7E-17	4.2E-15	2
	parter	4.8E+0	1.5E+00	4.7E+01	3.7E+00	1.4E+01	3.6E+01	8.5E-16	3.8E-16	3.6E-15	2
	ris	4.8E+0	1.5E+00	4.7E+01	1.1E+01	7.3E+00	7.2E+01	8.5E-16	3.8E-16	2.9E-15	2
	kms	9.1E+0	1.0E+00	5.5E+00	6.5E+00	5.9E+00	1.0E+01	2.0E-16	4.6E-17	2.1E-16	0
	toeppen	1.0E+1	1.3E+00	2.3E+00	3.4E+00	3.6E+01	1.0E+00	1.3E-16	8.4E-17	2.1E-15	1
	condex	1.0E+2	1.0E+00	1.9E+00	5.5E+00	5.9E+02	1.2E+00	6.5E-16	5.3E-16	5.0E-15	1
	moler	1.9E+2	1.0E+00	2.7E+03	6.2E+00	8.9E+03	2.9E+00	1.3E-15	9.1E-19	5.0E-18	0
	circul	3.7E+2	1.1E+02	1.0E+03	7.4E+17	7.6E+04	1.4E+01	2.9E-14	8.0E-16	5.3E-15	2
	randcorr	1.4E+3	1.0E+00	3.1E+01	3.8E+01	3.5E+01	7.7E+03	5.8E-16	4.5E-17	3.8E-16	1
	poisson	1.7E+3	1.0E+00	1.9E+00	2.0E+01	7.3E+00	2.0E+01	1.6E-16	8.3E-17	1.6E-15	1
	hankel	2.9E+3	6.4E+01	1.0E+03	2.0E+17	5.7E+04	1.7E+01	2.7E-14	8.1E-16	5.4E-15	2
	jordbloc	5.2E+3	1.0E+00	1.0E+00	1.0E+00	2.0E+00	4.0E+03	0.0E+00	1.9E-17	6.5E-17	0
	compan	7.5E+3	1.0E+00	2.8E+00	4.2E+01	6.6E+02	3.7E+00	8.1E-16	1.5E-16	9.0E-13	1
	pei	1.0E+4	1.0E+00	5.3E+02	7.8E+00	1.0E+01	6.4E+00	5.9E-15	6.6E-18	3.1E-17	0
	randcolu	1.5E+4	3.2E+01	1.0E+03	1.3E+17	8.8E+02	2.3E+04	2.7E-14	6.5E-16	4.1E-15	2
	sprandn	1.6E+4	4.6E+00	7.7E+02	4.1E+17	8.1E+03	2.3E+03	2.3E-14	2.8-15	4.0E-14	2
	riemann	1.9E+4	1.0E+00	4.0E+03	9.1E+00	5.6E+04	1.4E+02	1.3E-15	7.8E-16	5.1E-15	2
	compar	1.8E+6	1.8E+01	1.0E+03	2.0E+17	3.2E+04	1.6E+03	1.5E-14	4.6E-16	3.4E-15	1
	tridiag	6.8E+6	1.0E+00	1.9E+00	2.0E+02	4.0E+00	1.6E+04	3.0E-16	3.0E-17	3.1E-16	1
	chebspec	1.3E+7	1.0E+00	5.2E+01	1.6E+24	9.7E+06	1.7E+00	6.0E-16	1.2E-18	1.5E-15	2
	lehmer	1.8E+7	1.0E+00	1.5E+03	9.5E+00	8.90E+00	8.1E+03	5.8E-16	1.0E-17	6.2E-17	0
	toeppd	2.1E+7	1.0E+00	4.2E+01	8.7E+01	6.8E+04	4.0E+01	6.1E-16	2.5E-17	1.3E-16	0
	minij	2.7E+7	1.0E+00	4.0E+03	9.0E+00	1.8E+04	4.0E+00	4.6E-18	5.4E-19	3.5E-18	0
	randsvd	6.7E+7	5.0E+00	1.0E+03	2.0E+17	5.2E+00	2.3E+09	3.7E-15	1.3E-16	6.9E-16	2
	forsythe	6.7E+7	1.0E+00	1.0E+00	1.0E+00	1.0E+00	6.7E+07	0.0E+00	0.0E+00	0.0E+00	0
Ill-conditioned	fiedler	2.5E+10	1.0E+00	8.3E+02	2.7E+03	5.5E+01	5.2E+06	2.6E-16	8.1E-17	1.6E-15	1
	dorr	7.4E+10	1.0E+00	2.0E+00	5.4E+01	6.7E+05	2.5E+05	1.8E-16	4.0E-17	1.9E-15	1
	demmel	1.0E+14	1.3E+00	1.3E+02	3.3E+16	5.0E+15	2.7E+01	2.6E-15	3.0E-21	1.7E-09	2
	chebvand	3.8E+19	1.9E+02	1.4E+03	1.0E+18	6.9E+04	1.8E+18	3.9E-14	2.4E-17	2.1E-16	0
	invhess	4.1E+19	1.8E+00	4.0E+03	9.0E+00	4.7E+03	5.3E+74	3.2E-15	3.4E-17	2.7E-16	1
	prolate	1.4E+20	9.9E+00	1.5E+03	3.5E+18	1.4E+03	6.7E+19	1.5E-14	5.0E-16	8.0E-15	1
	frank	1.7E+20	1.0E+00	1.9E+00	1.9E+00	4.1E+06	1.5E+17	1.7E-17	3.9E-20	2.8E-17	0
	cauchy	5.5E+21	1.0E+00	1.8E+02	5.4E+05	3.9E+07	5.2E+16	9.4E-16	9.3E-19	1.2E-14	1
	hilb	8.0E+21	1.0E+00	3.0E+03	7.7E+17	2.4E+00	5.4E+23	4.5E-16	6.2E-19	2.4E-17	0
	lotkin	5.4E+22	1.0E+00	2.7E+03	2.2E+18	2.4E+00	5.6E+21	5.3E-17	1.4E-18	7.7E-17	0
	kahan	1.1E+28	1.0E+00	1.0E+00	1.0E+00	5.3E+00	7.6E+52	0E+00	9.5E-18	4.5E-15	1

TABLE 6.2
Stability of the linear solver using binary tree based block CALU_PRRP, binary tree based CALU, and GEPP.

n	P	b	η	w_b	N_{IR}	HPL3
Binary tree based block CALU_PRRP						
8192	256	32	7.5E-15	4.4E-14	2	4.4E-03
		16	6.7E-15	4.1E-14	2	4.6E-03
	128	64	7.6E-15	4.7E-14	2	4.9E-03
		32	7.5E-15	4.9E-14	2	4.9E-03
		16	7.3E-15	5.1E-14	2	5.7E-03
	64	128	7.6E-15	5.0E-14	2	5.2E-03
		64	7.9E-15	5.3E-14	2	5.9E-03
		32	7.8E-15	5.0E-14	2	5.0E-03
		16	6.7E-15	5.0E-14	2	5.7E-03
4096	256	16	3.5E-15	2.2E-14	2	5.1E-03
	128	32	3.8E-15	2.3E-14	2	5.1E-03
		16	3.6E-15	2.2E-14	1.6	5.1E-03
	64	64	4.0E-15	2.3E-14	2	4.9E-03
		32	3.9E-15	2.4E-14	2	5.7E-03
		16	3.8E-15	2.4E-14	1.6	5.2E-03
2048	128	16	1.8E-15	1.0E-14	2	4.8E-03
	64	32	1.8E-15	1.2E-14	2	5.6E-03
		16	1.9E-15	1.2E-14	1.8	5.4E-03
1024	64	16	1.0E-15	6.3E-15	1.3	6.1E-03
Binary tree based CALU						
8192	256	32	6.2E-15	4.1E-14	2	4.5E-03
		16	5.8E-15	3.9E-14	2	4.1E-03
	128	64	6.1E-15	4.2E-14	2	4.6E-03
		32	6.3E-15	4.0E-14	2	4.4E-03
		16	5.8E-15	4.0E-14	2	4.3E-03
	64	128	5.8E-15	3.6E-14	2	3.9E-03
		64	6.2E-15	4.3E-14	2	4.4E-03
		32	6.3E-15	4.1E-14	2	4.5E-03
		16	6.0E-15	4.1E-14	2	4.2E-03
4096	256	16	3.1E-15	2.1E-14	1.7	4.4E-03
	128	32	3.2E-15	2.3E-14	2	5.1E-03
		16	3.1E-15	1.8E-14	2	4.0E-03
	64	64	3.2E-15	2.1E-14	1.7	4.6E-03
		32	3.2E-15	2.2E-14	1.3	4.7E-03
		16	3.1E-15	2.0E-14	2	4.3E-03
2048	128	16	1.7E-15	1.1E-14	1.8	5.1E-03
	64	32	1.7E-15	1.0E-14	1.6	4.6E-03
		16	1.6E-15	1.1E-14	1.8	4.9E-03
1024	64	16	8.7E-16	5.2E-15	1.6	4.7E-03
GEPP						
8192	-		3.9E-15	2.6E-14	1.6	2.8E-03
4096	-		2.1E-15	1.4E-14	1.6	2.9E-03
2048	-		1.1E-15	7.4E-15	2	3.4E-03
1024	-		6.6E-16	4.0E-15	2	3.7E-03

TABLE 6.3
Stability of the LU decomposition for binary tree based block CALU_PRRP on special matrices.

	Matrix	cond(A,2)	g_W	$\|\|L\|\|_1$	$\|\|L^{-1}\|\|_1$	$\|\|U\|\|_1$	$\|\|U^{-1}\|\|_1$	$\frac{\|\|PA-LU\|\|_F}{\|\|A\|\|_F}$	η	w_b	N_{IR}
Well-conditioned	hadamard	1.0E+0	5.1E+02	5.1E+02	1.3E+02	4.9E+04	6.5E+00	5.3E-15	1.2E-15	8.7E-15	2
	house	1.0E+0	7.4E+00	8.3E+02	3.8E+02	3.7E+02	5.0E+01	5.8E-16	4.2E-17	4.8E-15	3
	parter	4.8E+0	1.5E+00	4.7E+01	3.7E+00	1.4E+01	3.6E+01	8.6E-16	6.9E-16	4.4E-15	3
	ris	4.8E+0	1.5E+00	4.7E+01	3.7E+00	7.3E+00	7.2E+01	8.5E-16	6.2E-16	4.4E-15	2
	kms	9.1E+0	1.0E+00	5.1E+00	3.4E+00	4.3E+00	9.3E+00	1.1E-16	6.3E-17	3.6E-16	1
	toeppen	1.0E+1	1.3E+00	2.3E+00	3.4E+00	3.6E+01	1.0E+00	1.0E-16	8.2E-17	4.0E-15	1
	condex	1.0E+2	1.0E+00	1.9E+00	5.5E+00	5.9E+02	1.2E+00	6.5E-16	7.4E-16	5.2E-15	2
	moler	1.9E+2	1.0E+00	4.0E+03	9.0E+00	2.2E+04	3.8E+15	1.0E-18	2.1E-20	2.7E-16	1
	circul	3.7E+2	1.5E+02	1.6E+03	1.4E+03	8.4E+04	2.1E+01	5.2E-14	4.1E-15	3.2E-14	2
	randcorr	1.4E+3	1.4E+02	5.4E+02	4.4E+03	2.9E+03	6.1E+03	2.2E-15	1.3E-16	5.7E-15	2
	poisson	1.7E+3	1.0E+00	1.9E+00	2.2E+01	7.3E+00	2.0E+01	1.7E-16	9.3E-17	1.5E-15	1
	hankel	2.9E+3	6.1E+01	1.84E+03	1.5E+03	6.5E+04	8.0E+01	4.9E-14	3.9E-15	2.4E-14	2
	jordbloc	5.2E+3	1.0E+00	1.0E+00	1.0E+00	2.0E+00	4.0E+03	0.0E+00	1.9E-17	7.67E-17	0
	compan	7.5E+3	2.0E+00	2.9E+00	6.8E+02	6.7E+02	9.4E+00	9.8E-16	6.1E-16	6.3E-12	1
	pei	1.0E+4	1.3E+00	5.5E+02	7.9E+00	1.6E+01	2.9E+00	3.6E-16	2.2E-17	5.0E-17	0
	randcolu	1.5E+4	4.6E+01	1.7E+03	1.4E+03	1.0E+03	1.1E+04	4.8E-14	3.5E-15	2.2E-14	2
	sprandn	1.6E+4	6.6E+00	1.1E+03	1.6E+03	9.6E+03	1.4E+03	4.1E-14	1.3E-14	2.0E-13	(1)
	riemann	1.9E+4	1.0E+00	4.0E+03	1.8E+03	7.6E+06	1.4E+02	1.7E-16	1.2E-16	1.1E-15	1
	compar	1.8E+6	9.0E+02	2.0E+03	1.3E+05	9.0E+05	6.5E+02	1.9E-14	1.2E-15	8.9E-15	1
	tridiag	6.8E+6	1.0E+00	1.9E+00	1.8E+02	4.0E+00	1.6E+04	3.2E-16	2.8E-17	2.6E-16	0
	chebspec	1.3E+7	1.0E+00	5.2E+01	5.6E+01	9.7E+06	1.7E+00	6.0E-16	7.8E-19	1.3E-15	1
	lehmer	1.8E+7	1.0E+00	1.5E+03	8.9E+00	8.9E+00	8.1E+03	5.7E-16	9.92E-18	5.7E-17	0
	toeppd	2.1E+7	1.0E+00	4.3E+01	9.1E+02	6.8E+04	1.8E+01	5.8E-16	2.7E-17	1.7E-16	0
	minij	2.7E+7	1.0E+00	4.0E+03	9.0E+00	1.8E+04	4.0E+00	6.4E-19	6.9E-19	6.1E-18	0
	randsvd	6.7E+7	5.4E+00	1.7E+03	1.4E+03	6.4E+00	3.2E+09	7.2E-15	5.5E-16	3.3E-15	2
	forsythe	6.7E+7	1.0E+00	1.0E+00	1.0E+00	1.0E+00	6.7E+07	0.0E+00	0.0E+00	0.0E+00	0
Ill-conditioned	fiedler	2.5E+10	1.0E+00	9.0E+02	1.2E+01	5.5E+01	1.4E+07	2.5E-16	7.9E-17	1.7E-15	1
	dorr	7.4E+10	1.0E+00	2.0E+00	4.0E+01	6.7E+05	2.5E+05	1.6E-16	3.3E-17	3.4E-15	1
	demmel	1.0E+14	1.5E+00	1.2E+02	5.3E+02	5.5E+15	2.9E+01	3.3E-15	9.0E-21	7.9E-09	2
	chebvand	3.8E+19	3.2E+02	2.2E+03	3.3E+03	8.9E+04	1.6E+20	7.5E-14	3.7E-17	2.3E-16	0
	invhess	4.1E+19	1.7E+00	4.0E+03	9.0E+00	4.5E+03	1.4E+100	2.9E-15	1.5E-16	6.9E-14	1
	prolate	1.4E+20	1.2E+01	2.1E+03	4.7E+03	2.1E+03	1.6E+21	2.8E-14	1.2E-15	3.9E-14	1
	frank	1.7E+20	1.0E+00	1.9E+00	1.9E+00	4.1E+06	2.3E+16	1.6E-17	2.6E-20	6.8E-17	0
	cauchy	5.5E+21	1.0E+00	3.2E+02	4.1E+02	5.4E+07	4.7E+18	6.1E-16	6.7E-19	5.7E-14	2
	hilb	8.0E+21	1.0E+00	2.9E+03	1.4E+03	2.4E+00	2.3E+22	3.2E-16	6.3E-19	2.2E-17	0
	lotkin	5.4E+22	1.0E+00	3.0E+03	1.4E+03	2.4E+00	3.7E+22	6.6E-17	5.2E-18	1.75E-15	1
	kahan	1.1E+28	1.0E+00	1.0E+00	1.0E+00	5.3E+00	7.6E+52	0.0E+00	1.2E-17	3.9E-16	1

Acknowledgments. The authors thank the anonymous reviewers and the editor for their very useful comments that helped us improve the paper. The authors would like to thank Julien Langou for useful discussions on the growth factor of block LU_PRRP factorization and its relation with an optimal block size.

REFERENCES

[1] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, *Minimizing communication in linear algebra*, SIMAX, 32 (2011), pp. 866–890.

[2] J. Choi, J. Dongarra, L. S. Ostrouchov, A. P. Petitet, D. W. Walker, and R. C. Whaley, *The Design and Implementation of the ScaLAPACK LU, QR and Cholesky Factorization Routines*, Scientific Programming, 5 (1996), pp. 173–184.

[3] J. W. Demmel, N. J. Higham, and R. Schreiber, *Block LU factorization*, Numer. Linear Algebra Appl., 2 (1995), pp. 173–190.

[4] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM J. Sci. Comput., 34 (2012), pp. 206–239.

[5] N. J. Dingle and N. J. Higham, *Reducing the Influence of Tiny Normwise Relative Errors on Performance Profiles*, Technical report 2011.90, School of Mathematics, The University of Manchester, 2011.

[6] S. Donfack, L. Grigori, and A. K. Gupta, *Adapting communication-avoiding LU and QR factorizations to multicore architectures*, in Proceedings of the IEEE International Symposium on Parallel & Distributed Processing, IEEE, 2010, pp. 1–10.

[7] J. Dongarra, P. Luszczek, and A. Petitet, *The LINPACK Benchmark: Past, Present and Future*, Concurrency: Practice and Experience, 15 (2003), pp. 803–820.

[8] L. V. Foster, *Gaussian elimination with partial pivoting can fail in practice*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1354–1362.

[9] L. V. Foster, *The growth factor and efficiency of Gaussian elimination with rook pivoting*, J. Comput. Appl. Math., 86 (1997), pp. 177–194.

[10] N. I. M. Gould, *On growth in Gaussian elimination with complete pivoting*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 354–361.

[11] L. Grigori, J. Demmel, and H. Xiang, *Communication avoiding Gaussian elimination*, in Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, IEEE/ACM, 2008, p. 29.

[12] L. Grigori, J. Demmel, and H. Xiang, *CALU: A communication optimal LU factorization algorithm*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1317–1350.

[13] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank reaviling QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–896.

[14] J.-W. Hong and H. T. Kung, *I/O complexity: The Red-Blue Pebble Game*, in STOC '81: Proceedings of the 13th Annual ACM Symposium on Theory of Computing, ACM, New York, 1981, pp. 326–333.

[15] D. Irony, S. Toledo, and A. Tiskin, *Communication lower bounds for distributed-memory matrix multiplication*, J. Parallel Distrib. Comput., 64 (2004), pp. 1017–1026.

[16] A. Khabou, J. Demmel, L. Grigori, and M. Gu, *LU Factorization with Panel Rank Revealing Pivoting and its Communication Avoiding Version*, Technical report UCB/EECS-2012-15, University of California, Berkeley, Berekley, CA,2012.

[17] N. J. Higham and D. J. Higham, *Large Growth Factors in Gaussian Elimination with Pivoting*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 155–164.

[18] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[19] R. D. Skeel, *Iterative refinement implies numerical stability for Gaussian elimination*, Math. Comp., 35 (1980), pp. 817–831.

[20] G. W. Stewart, *Gauss, statistics, and Gaussian elimination*, J. Comput. Graph. Statist., 4 (1995).

[21] L. N. Trefethen and R. S. Schreiber, *Average-case stability of Gaussian elimination*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 335–360.

[22] J. H. Wilkinson, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281–330.

[23] J. H. Wilkinson, *The Algebric Eigenvalue Problem*, Oxford University Press, Oxford, 1985.

[24] S. J. Wright, *A collection of problems for which Gaussian elimination with partial pivoting is unstable*, SIAM J. Sci. Stat. Comput., 14 (1993), pp. 231–238.