A Cholesky QR type algorithm for computing tall-skinny QR factorization with column pivoting

Takeshi Fukaya
Hokkaido University
Sapporo, Hokkaido, Japan
fukaya@iic.hokudai.ac.jp

Yuji Nakatsukasa
University of Oxford
Oxford, UK
nakatsukasa@maths.ox.ac.uk

Yusaku Yamamoto
The University of Electro-Communications
Chofu, Tokyo, Japan
yusaku.yamamoto@uec.ac.jp

***Abstract*—We consider computing the QR factorization with column pivoting (QRCP) for a tall and skinny matrix, which has important applications including low-rank approximation and rank determination. Motivated by recent progresses of Cholesky QR type algorithms for tall-skinny QR factorization (without pivoting), we propose a new Cholesky QR type algorithm for tall-skinny QRCP, which we call Iterative Cholesky QR with Column Pivoting (Ite-CholQR-CP). Through performance evaluation, it is confirmed that Ite-CholQR-CP provides a solution as accurate as that by Householder QR with column pivoting (HQR-CP), which is a widely-used conventional algorithm. In addition, Ite-CholQR-CP outperforms HQR-CP in execution time in single node and distributed parallel computations: up to 45x (single node computation) and 27x (distributed parallel computation) speedup.**

I. Introduction

Numerical algorithms for computing various kinds of matrix factorizations such as LU and QR play an important role in scientific computing. While the peak performance of a computational system is still increasing, its architecture has been getting more and more complicated at the same time. In this situation, it is usually not easy to fully exploit the potential of the system by naively implementing a traditional algorithm, and it is thus vital to develop a new algorithm that is suitable for the architecture of a target system.

In this paper, we consider computing the QR factorization with column pivoting (QRCP), whose details are given in Section II-B. QRCP is among the practical methods to obtain a rank-revealing QR factorization (RRQR) [1] in a broad sense. RRQR has many applications [2]–[4]; for example, the (numerical) rank, range space, and null space of a matrix can be approximately obtained by RRQR.

This work focuses on a special case where a target matrix is tall and skinny, which means that the number of rows is much larger than that of columns. The computation of a tall-skinny QRCP has as application the computation of an orthogonal basis in many numerical methods, e.g., those for solving linear systems and eigenvalue problems. Compared with normal QR (without pivoting), QRCP tends to be more stable when an input matrix (a set of vectors) is close to numerically rank deficient. Another application is the computation of a low-rank approximation of tall-skinny matrix, which often appears in numerical methods related to hierarchical or non-hierarchical matrix approximation techniques such as $\mathcal{H}$ or $\mathcal{H}^2$-matrix [5] and tensor computations such as the tensor train decomposition [6]; these methods require the computation of a low-rank approximation of a tall-skinny matrix many times.

Recently, for the tall-skinny QR factorization without pivoting, the effectiveness of Cholesky QR type algorithms have been reported [7]–[11] together with related theoretical results [12]. Their effectiveness in the development of scientific libraries has also been presented [13]. The structure of Cholesky QR type algorithms is quite suitable for high-performance computing in recent computational systems, which motivates us to aim for developing a new Cholesky QR type algorithm for computing a tall-skinny QR factorization with column pivoting. Mathematically, i.e., if there is no rounding error, it is trivial to construct a Cholesky QR type algorithm for QRCP, however, we cannot ignore the effect of rounding error in practice, which makes the algorithm development non-trivial.

In this research, we first conduct preliminary numerical experiments, whose results guide the algorithm development. The key ideas in our algorithm development are 1) modification of Cholesky factorization with complete pivoting, and 2) algorithm design based on an iterative process in which the correct pivot selections are made step by step. The resulting algorithm, which we call Iterative Cholesky QR with column pivoting (Ite-CholQR-CP), has a similar structure as existing Cholesky QR type algorithms; advantages for high-performance computing are still preserved in Ite-CholQR-CP. Through numerical experiments, the performance of Ite-CholQR-CP is evaluated. It is confirmed that Ite-CholQR-CP provides a QRCP as accurate as that by a conventional algorithm, namely Householder QR with column pivoting (HQR-CP). In addition, Ite-CholQR-CP is faster than HQR-CP in both single node and distributed parallel computations; up to 45 times speedup (single node computation) and 27 times speedup (distributed parallel computation) are obtained.

The rest of the paper is organized as follows: in Section II, we give a brief overview of QRCP including the problem settings. In Section III, we explain an algorithm proposed in this research together with ideas for the algorithm development. In Section IV, we present the results of performance evaluation. After referring to related work in Section V, we conclude in Section VI.

II. QR FACTORIZATION WITH COLUMN PIVOTING

A. Notations

Throughout the paper, we use the following notation:

- I_n: the n-dimensional identity matrix,
- $P_{(i,j)}$: an elementary permutation matrix that swaps the i-th and j-th columns of A by $AP_{(i,j)}$,
- $\kappa_2(A)$: the 2-norm condition number of A, which is defined as $\sigma_{\max}/\sigma_{\min}$, where $\sigma_{\max}, \sigma_{\min}$ are the maximum and minimum singular values of A, respectively,
- $\mathbf{u}$: the unit roundoff, e.g., $\mathbf{u} \simeq 10^{-16}$ in double-precision floating point number,
- MATLAB notation is used for representing a submatrix, e.g., $A(:, i:j)$ represents a submatrix that consists of the columns of A from the i-th to j-th columns.

B. Problem Settings

Let A be an $m \times n$ $(m \geq n)$ matrix with numerical rank r $(\leq n)$. This means that there is a sufficient gap between σ_r and σ_{r+1} and σ_i/σ_1 $(i = r+1, \ldots, n)$ are sufficient small, e.g., $O(\mathbf{u})$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ are the singular values of A.

For this matrix A, we consider computing the factorization

$$AP = QR = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ O & R_{22} \end{pmatrix}, \tag{1}$$

where $P \in \mathbb{R}^{n \times n}$ is a permutation matrix, $Q \in \mathbb{R}^{m \times n}$ is an orthogonal matrix (i.e., $Q^\top Q = I_n$), $R \in \mathbb{R}^{n \times n}$ ia an upper triangular matrix. Let R_{11} be $k \times k$ (and Q_1 be $m \times k$), then the permutation matrix P and the integer k are determined so as to make $\kappa_2(R_{11})$ as small (i.e., well conditioned) as possible and to make $\|R_{22}\|_2$ as small as possible. Ideally, $k = r$ and $\kappa_2(R_{11}) \simeq \sigma_1/\sigma_r$.

The factorization (1) is called the QR factorization with column pivoting (QRCP) for the matrix A, which can be classified into rank revealing QR factorization (RRQR) in a broad sense; for details of RRQR, for example, please see the references [14].

We give other settings in this work. First, we consider the case where A is tall and skinny, i.e., $m \gg n$. Next, we assume that Q is explicitly formed. Finally, we discuss not only single node (shared memory) computation but also distributed parallel (MPI parallel) computation, and the following settings are assumed in the latter case: let

$$A = (A_1^\top \cdots A_p^\top)^\top, \tag{2}$$

and the p-th process has $A_p \in \mathbb{R}^{(m/P) \times n}$ $(p = 1, \ldots, P)$, where P is the number of MPI processes. This is the so-called one-dimensional block row layout. For simplicity, we assume that m can be divided by P. For Q, we assume the same data distribution as that for A, on the other hand, we accept any data distributions for P and R.

Algorithm 1 HQR-CP: Householder QR with Column Pivoting

Input: $A \in \mathbb{R}^{m \times n}$

1: $P := I_n$
2: $c(j) := \|A(:, j)\|_2^2$ $(j = 1, \ldots, n)$
3: **for** $j = 1$ **to** n **do**
4: $\quad p = \arg\max_{j \leq l \leq n} c(l)$
5: $\quad A := AP_{(j,p)}$ // swap columns
6: $\quad c(j) \leftrightarrow c(p)$ // swap values
7: $\quad$ generate a Householder matrix $H_j \in \mathbb{R}^{m \times m}$ from $A(:, j)$
8: $\quad A := H_j A$
9: $\quad R(j, j:n) := A(j, j:n)$
10: $\quad c(l) := c(l) - R(j, l)^2$ $(l = j+1, \ldots, n)$
11: $\quad P := PP_{(j,p)}$
12: **end for**
13: $Q := (H_n \cdots H_1)^{-1} \begin{pmatrix} I_n \\ O \end{pmatrix} = H_1 \cdots H_n \begin{pmatrix} I_n \\ O \end{pmatrix}$

Output: $Q \in \mathbb{R}^{m \times n}$, $R \in \mathbb{R}^{n \times n}$, $P \in \mathbb{R}^{n \times n}$

C. Conventional Algorithm

A conventional algorithm for computing QRCP is the Householder QR algorithm with column pivoting (HQR-CP) [2], [15], which is shown in Algorithm 1. HQR-CP is a kind of greedy algorithm, in which the column with the maximum 2-norm is selected among the remaining columns step by step; the information of 2-norm is stored in the array $c(\cdot)$. It is known that HQR-CP can provide an accurate result in practice, which we regard as the baseline in this research. It is worth noting that it is not necessary to form the explicit Q matrix in some applications; in this case, one can skip the line 13 in Algorithm 1.

For HQR-CP in real and double precision, LAPACK provides the DGEQPF and DGEQP3 routines, and ScaLAPACK provides the PDGEQPF routine. Similar to the case of Householder QR without pivoting, blocking techniques for exploiting Level-3 BLAS routines, e.g., DGEMM, were proposed [16], and they are employed in DGEQP3. However, unlike the case of Householder QR without pivoting, even if blocking techniques are employed, half of the computations in HQR-CP are still done by Level-2 BLAS operations. Another difference is that the BLAS-3 version of parallel routine, i.e., PDGEQP3, has not been officially provided in ScaLAPACK. It is worth mentioning that there is a possibility that a naive implementation of Algorithm 1 fails to correctly select a good pivot [17]; in Algorithm 1, instead of line 10, the explicit recalculation of the values in the array $c(\cdot)$ is sometimes needed during the algorithm.

D. Goal of this research

The goal of this research is to develop an algorithm that computes the QRCP for a tall and skinny matrix, and ideally the developed algorithm

- provides a solution as accurate as that by the conventional HQR-CP algorithms; in addition to standard accuracy

metrics in QR factorization (i.e., orthogonality of Q and residual), we require the algorithm to provide the same pivot selection,
- is faster than DGEQP3 in single node computation and PDGEQPF in distributed parallel computation.

III. Development of an algorithm based on Cholesky QR

A. The Cholesky QR algorithm

The Cholesky QR algorithm, which is shown in Algorithm 2, is a simple algorithm based on triangular orthogonalization [18], and it computes the (thin) QR factorization of A via the Cholesky factorization of the Gram matrix $A^\top A$ [19, Thm. 5.2.3]. When A is tall and skinny, the cost for computing $A^\top A$ (line 1) and AR^{-1} (line 3) is dominant. Since these computations can be done using Level-3 BLAS routines (e.g., DGEMM), high effective performance (i.e., FLOPS) is expected in modern computer systems. In addition, in the case of distributed parallel computation, only one global collective communication (e.g., MPI_Allreduce) is needed for computing $A^\top A$ ($= \sum_{p=1}^{P} A_p^\top A_p$). This means that Cholesky QR can be regarded as Communication-Avoiding (CA) [20], [21], in which $O(1)$ global collective communications are needed instead of $O(n)$ as in Householder QR.

While Cholesky QR has great advantages in high-performance computing, it has serious issues in its numerical aspect. First, the computed Q matrix loses orthogonality, as $\kappa_2(A)$ grows. Second, when $\kappa_2(A) \gtrsim \mathbf{u}^{-1/2}$, the algorithm often breaks down due to the breakdown of the numerical Cholesky factorization of $A^\top A$; the positive definiteness is lost due to rounding errors. In order to remedy these issues, several approaches have been proposed. For example, it was reported that repeating Cholesky QR improves the orthogonality of the computed Q factor [22], and it was shown that the Cholesky QR algorithm with reorthogonalization (CholeskyQR2) can provide a QR factorization as accurate as that by Householder QR when $\kappa_2(A) \lesssim \mathbf{u}^{-1/2}$ [7], [12]. For ill-conditioned ($\kappa_2(A) \gtrsim \mathbf{u}^{-1/2}$) matrices, a preconditioning technique that reduces $\kappa_2(A)$, the so-called shifted Cholesky QR algorithm, was proposed [8], and the shifted CholeskyQR3 algorithm [8], which combines shifted Cholesky QR and CholeskyQR2, can compute an accurate QR factorization for an ill-conditioned matrix with less execution time than Householder QR and TSQR [21]. In addition to the above, other approaches that aim at improving and exploiting Cholesky QR have also been presented [9], [10], [23].

From performance results presented in the above previous works (e.g., [7], [8]), we can confirm a remarkable performance (i.e., an advantage in execution time) for computing a tall-skinny QR factorization, which motivates us to develop a Cholesky QR type algorithm for QRCP of a tall-skinny matrix. What is important in the algorithm development is to preserve the advantages in Cholesky QR; dominant computations are done with Level-3 BLAS routines, and the algorithm is CA.

Algorithm 2 CholQR: Cholesky QR

Input: $A \in \mathbb{R}^{m\times n}$
1: $W := A^\top A$
2: $R := \mathrm{Chol}(W)$
3: $Q := AR^{-1}$
Output: $Q \in \mathbb{R}^{m\times n}$, $R \in \mathbb{R}^{n\times n}$

B. Application of Cholesky QR to QRCP

Mathematically – in other words, if there is no rounding error – it is easy to design a Cholesky QR type algorithm for QRCP; simply using the Cholesky factorization with complete pivoting (Chol-CP) [24]

$$P^\top W P = R^\top R \tag{3}$$

instead of the Cholesky factorization (without pivoting). In the process of Chol-CP, the maximum diagonal element among the remaining submatrix is selected step by step, which corresponds to the selection of pivot columns in the HQR-CP algorithm. Then, we can obtain QRCP as

$$Q := APR^{-1} \;\Leftrightarrow\; AP = QR. \tag{4}$$

However, in practical numerical computations, we cannot avoid the effect of rounding error, and it has to be taken into account when developing a new algorithm.

C. Preliminary experiments for algorithm development

Here, we present the results of preliminary numerical experiments on QRCP using Chol-CP, which guides us in our algorithm development.

First, we examine the pivot selection by Chol-CP in Figure 1 (a); for a test matrix (generated as explained in Section IV-A3 with $m = 10000$, $n = 50$, $r = 40$, $\sigma = 10^{-12}$), we compare the pivot selection between DGEQP3 (HQR-CP) and Chol-CP for $A^\top A$. In Figure 1 (a), we encountered three cases on pivot selection:

- 1st case (1st to 27th pivot): correctly selected (✓),
- 2nd case (28th to 31st): incorrectly selected (×),
- 3rd case (32nd to 40th): not computed (–).

The main reason for the 2nd case is the effect of rounding error, and that for the 3rd case is the breakdown of Chol-CP due to the appearance of a zero or negative diagonal. However, it is an encouraging result that some of the obtained pivot selections (i.e., the 1st case) are correct.

Second, for test matrices with different condition numbers ($m = 10000$, $n = r = 50$, $10^0 \le \kappa_2(A) = 1/\sigma \le 10^{16}$), we apply Chol-CP to $A^\top A$. The results (correct, incorrect, or not computed) are shown in Figure 1 (b), where the y-axis gives the magnitude of $|r_{ii}/r_{11}|$ (r_{ii}: the i-th diagonal element of R computed by DGEQP3). Figure 1 (b) clearly shows the relation between the result of pivot selection and the magnitude of $|r_{ii}/r_{11}|$; if $|r_{ii}/r_{11}|$ is sufficiently larger, it is expected that the i-th pivot is correctly selected.

Finally, we generate 1000 test matrices with varying condition numbers ($\kappa_2(A) = 1/\sigma = 10^{-\gamma}, \gamma \in [1, 16]$, $m = 10000$,

$n = r = 40$) and investigate the distribution of incorrect pivot selections. Figure 1 (c) presents the results with the same y-axis as in Figure 1 (b). From Figure 1 (c), we see that the pivot selections by Chol-CP is unreliable if $|r_{ii}/r_{11}| \lesssim 10^{-6}$; however, we can trust the obtained pivot selections until this condition is satisfied.

From the above numerical results, we can summarize that

- there is a sufficient possibility of exploiting Chol-CP for pivot selections,
- the quantity $|r_{ii}/r_{11}|$ provides an important guide to judge the correctness of pivot selections.

D. Proposed algorithm

1) Modification of Chol-CP: The results presented in the previous subsection indicates the necessity of monitoring the magnitude of the diagonal elements in the process of Chol-CP. Taking this into account, we modify the algorithm Chol-CP to Algorithm 3, which hereafter we call the partial Cholesky factorization with complete pivoting (P-Chol-CP). In this algorithm, we introduce a tolerance, namely ϵ, into the original algorithm of Chol-CP and stop the computation when

$$\frac{W(k,k)}{W(1,1)} < \epsilon^2 \ (\Leftrightarrow \frac{R(k,k)}{R(1,1)} < \epsilon). \tag{5}$$

This algorithm provides an upper triangular matrix R that satisfies

$$\begin{aligned} P^\top W P &= R^\top R + W' \\ &= \begin{pmatrix} R_{11}^\top & O \\ R_{12}^\top & I_{n-n'} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ O & I_{n-n'} \end{pmatrix} + \begin{pmatrix} O & O \\ O & W'_{22} \end{pmatrix}, \end{aligned} \tag{6}$$

where $R_{11} \in \mathbb{R}^{n' \times n'}$ is upper triangular and $R_{12} \in \mathbb{R}^{n' \times (n-n')}$ is rectangular. The stopping criterion (5) means that all of the diagonal elements of $(I_{n-n'} + W'_{22}) \in \mathbb{R}^{(n-n') \times (n-n')}$ are smaller than $(R_{11}(1,1) \cdot \epsilon)^2$. Here, from the observations in the previous subsection, we expect that we can obtain reliable pivot selections from the first n' elements, namely P used for determining R_{11}. It is worth mentioning that similar ideas were proposed for the Cholesky QR algorithm for the normal QR factorization (without CP) [25], [26].

2) Whole algorithm for QRCP: Now, we develop an algorithm for QRCP using P-Chol-CP; the key idea here is to obtain correct pivot selections step by step by using P-Chol-CP in an iterative manner.

Let $A^{(0)} := A$, then

1) compute $W^{(1)} := {A^{(0)}}^\top A^{(0)}$,
2) apply P-Chol-CP to $W^{(1)}$ and obtain $R^{(1)}$, $P^{(1)}$, $n^{(1)}$, where
$$R^{(1)} = \begin{pmatrix} R_{11}^{(1)} & R_{12}^{(1)} \\ O & I_{n-n^{(1)}} \end{pmatrix},$$
3) compute $A^{(1)} := A^{(0)} P^{(1)} {R^{(1)}}^{-1}$.

These are the computations in the first stage in the algorithm. After this, we expect that the first $n^{(1)}$ columns of $A^{(1)}$ are correctly selected by $P^{(1)}$. Let $A^{(1)} = [A_1^{(1)} \ A_2^{(1)}]$, where

Algorithm 3 P-Chol-CP: Partial Cholesky factorization with Complete Pivoting

Input: $W \in \mathbb{R}^{n \times n}$, $\epsilon \in \mathbb{R}$
1: $R := O$, $P := I_n$, $n' := 0$
2: **for** $k = 1$ **to** n **do**
3: $p = \arg\max_{k \le l \le n} W(l,l)$
4: **if** $k > 1$ and $W(p,p) < W(1,1) \cdot \epsilon^2$ **then**
5: **break**
6: **end if**
7: $W := P_{(k,p)}^\top W P_{(k,p)}$ // swap rows and columns
8: $R := R P_{(k,p)}$ // swap columns
9: $R(k,k) := \sqrt{W(k,k)}$
10: $R(k, k+1:n) := W(k, k+1:n)/R(k,k)$
11: $W(k+1:n, k+1:n) := W(k+1:n, k+1:n) - R(k, k+1:n)^\top R(k, k+1:n)$
12: $P := P P_{(k,p)}$, $n' := k$
13: **end for**
14: $R := R + \begin{pmatrix} O & \\ & I_{n-n'} \end{pmatrix}$
Output: $R \in \mathbb{R}^{n \times n}$, $P \in \mathbb{R}^{n \times n}$, $n' \in \mathbb{Z}$

$A_1^{(1)} \in \mathbb{R}^{m \times n^{(1)}}$ and $A_2^{(1)} \in \mathbb{R}^{m \times (n-n^{(1)})}$, then it is worth pointing out that $\kappa_2(A_1^{(1)}) \simeq O(1)$ (i.e., not orthonormal but well conditioned [12]), ${A_1^{(1)}}^\top A_2^{(1)} \simeq O$ (i.e., close to orthogonal), and $\kappa_2(A_2^{(1)}) \ll \kappa_2(A)$.

In the second stage,

1) compute $W^{(2)} := {A^{(1)}}^\top A^{(1)}$ and partition $W^{(2)}$ as
$$W^{(2)} = \begin{pmatrix} W_{11}^{(2)} & W_{12}^{(2)} \\ {W_{12}^{(2)}}^\top & W_{22}^{(2)} \end{pmatrix}, \quad W_{11}^{(2)} \in \mathbb{R}^{n^{(1)} \times n^{(1)}},$$
2) compute $W_{11}^{(2)} = {R_{11}^{(2)}}^\top R_{11}^{(2)}$ (normal Cholesky factorization), $R_{12}^{(2)} := {R_{11}^{(2)}}^{-\top} W_{12}^{(2)}$, and $\tilde{W}_{22}^{(2)} := W_{22}^{(2)} - {R_{12}^{(2)}}^\top R_{12}^{(2)}$,
3) apply P-Chol-CP to $\tilde{W}_{22}^{(2)}$ and obtain $R_{22}^{(2)}$, $\tilde{P}^{(2)}$, $n^{(2)}$, where
$$R_{22}^{(2)} = \begin{pmatrix} R_{22\_11}^{(2)} & R_{22\_12}^{(2)} \\ O & I_{(n-n^{(1)})-n^{(2)}} \end{pmatrix},$$
4) compute $A^{(2)} := A^{(1)} P^{(2)} {R^{(2)}}^{-1}$, where
$$R^{(2)} := \begin{pmatrix} R_{11}^{(2)} & R_{12}^{(2)} \\ O & R_{22}^{(2)} \end{pmatrix}, \quad P^{(2)} := \begin{pmatrix} I_{n^{(1)}} & O \\ O & \tilde{P}^{(2)} \end{pmatrix}.$$

After the above process, we expect to obtain additional $n^{(2)}$ pivot selections correctly; in total, we have $n^{(1)} + n^{(2)}$ correct pivot selections. It is worth noting that

$$A^{(2)} := A^{(1)} P^{(2)} {R^{(2)}}^{-1} = A^{(0)} P^{(1)} {R^{(1)}}^{-1} P^{(2)} {R^{(2)}}^{-1}. \tag{7}$$

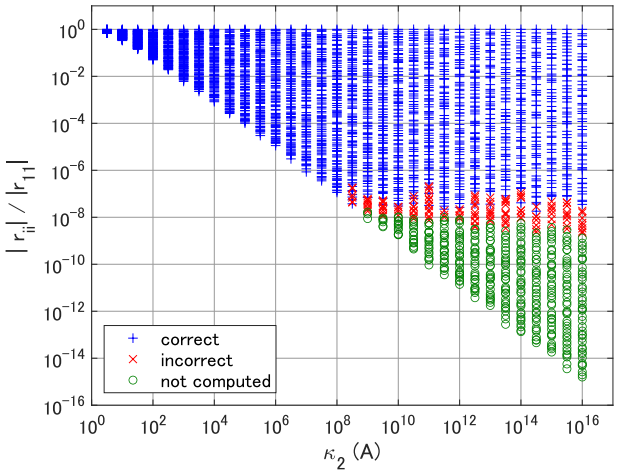
Repeating the above process until $n^{(1)} + \cdots + n^{(l)} = n$, we then have

$$A^{(l)} = A^{(0)} P^{(1)} {R^{(1)}}^{-1} P^{(2)} {R^{(2)}}^{-1} \cdots P^{(l)} {R^{(l)}}^{-1} \tag{8}$$

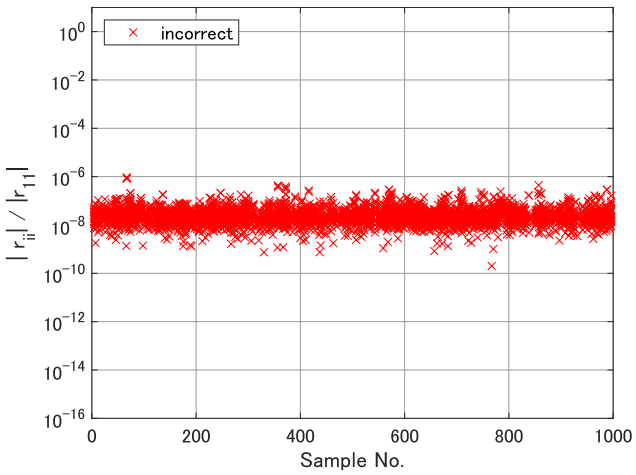
together with all of the correct pivot selections.

Col. Ind.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$\sigma = 10^{-12}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

(a) An example of the correctness of the obtained pivot (✓: correct, ×: incorrect, –: not computed)



(b) Dependency on $\kappa_2(A)$



(c) Distribution of incorrect pivot selections

Fig. 1: Results of preliminary numerical experiments on Chol-CP.

Now, we show that (8) can be written in the form of

$$A^{(l)}R = A^{(0)}P, \tag{9}$$

where R is an upper triangular matrix and P is a permutation matrix. Let $P := P^{(1)} \cdots P^{(l)}$, then we can transform (8) into

$$A^{(l)} = A^{(0)}P({P^{(l)}}^\top \cdots {P^{(2)}}^\top){R^{(1)}}^{-1}P^{(2)}{R^{(2)}}^{-1}\cdots P^{(l)}{R^{(l)}}^{-1} \tag{10}$$

Thus, it remains to show that

$$\begin{aligned}&\left(({P^{(l)}}^\top \cdots {P^{(2)}}^\top){R^{(1)}}^{-1}P^{(2)}{R^{(2)}}^{-1}\cdots P^{(l)}{R^{(l)}}^{-1}\right)^{-1}\\ =&R^{(l)}{P^{(l)}}^\top \cdots R^{(2)}{P^{(2)}}^\top R^{(1)}P^{(2)}\cdots P^{(l)}\end{aligned} \tag{11}$$

is upper triangular.

Let $n'^{(i)} := \sum_{k=1}^{i} n^{(k)}$, recall

$$R^{(i)} = \begin{pmatrix} R_{11}^{(i)} & R_{12}^{(i)} \\ O & I_{n-n'^{(i)}} \end{pmatrix}, \quad R_{11}^{(i)} \in \mathbb{R}^{n'^{(i)} \times n'^{(i)}}, \tag{12}$$

$$P^{(i+1)} = \begin{pmatrix} I_{n'^{(i)}} & O \\ O & \tilde{P}^{(i+1)} \end{pmatrix}, \quad \tilde{P}^{(i+1)} \in \mathbb{R}^{(n-n'^{(i)})\times(n-n'^{(i)})}, \tag{13}$$

then we have

$$\begin{aligned}\bar{R}^{(i)} :=& {P^{(i+1)}}^\top R^{(i)} P^{(i+1)}\\ =& \begin{pmatrix} I_{n'^{(i)}} & O \\ O & {\tilde{P}^{(i+1)}}^\top \end{pmatrix}\begin{pmatrix} R_{11}^{(i)} & R_{12}^{(i)} \\ O & I_{n-n'^{(i)}} \end{pmatrix}\begin{pmatrix} I_{n'^{(i)}} & O \\ O & \tilde{P}^{(i+1)} \end{pmatrix}\\ =& \begin{pmatrix} R_{11}^{(i)} & R_{12}^{(i)}\tilde{P}^{(i+1)} \\ O & I_{n-n'^{(i)}} \end{pmatrix} =: \begin{pmatrix} R_{11}^{(i)} & \bar{R}_{12}^{(i)} \\ O & I_{n-n'^{(i)}} \end{pmatrix},\end{aligned} \tag{14}$$

which means that the structure of $R^{(i)}$ is preserved; only columns in $R_{12}^{(i)}$ are swapped. On the other hand, we have

$$\begin{aligned}R^{(i+1)}\bar{R}^{(i)} &= \begin{pmatrix} R_{11}^{(i+1)} & R_{12}^{(i)} \\ O & I_{n-n'^{(i+1)}} \end{pmatrix}\begin{pmatrix} R_{11}^{(i)} & \bar{R}_{12}^{(i)} \\ O & I_{n-n'^{(i)}} \end{pmatrix}\\ &= \begin{pmatrix} \tilde{R}_{11}^{(i+1)} & \tilde{R}_{12}^{(i)} \\ O & I_{n-n'^{(i+1)}} \end{pmatrix},\end{aligned} \tag{15}$$

in which the structure of $R^{(i+1)}$ is still kept. Using these two facts, we can easily show that (11) is upper triangular, which derives the form of (9).

In the final l-th stage, in which $n'^{(l)} = n$, $R^{(l)}$ is a complete upper triangular matrix; there is no block of I. This means that the computations in this stage is essentially equivalent to the Cholesky QR algorithm except pivoting. By work [12], the obtained matrix, namely $A^{(l)}$, is not necessary orthonormal although $\kappa_2(A^{(l)}) \simeq O(1)$. Thus, it is required to apply reorthogonalization to $A^{(l)}$, which means to apply normal Cholesky QR to $A^{(l)}$, and finally we have an accurate QRCP as

$$A^{(l)}R_{\text{reortho}}^{-1} = Q \Leftrightarrow Q(R_{\text{reortho}}R) = A^{(0)}P. \tag{16}$$

The whole procedure of the proposed algorithm is described in Algorithm 4; we call the algorithm *Iterative Cholesky QR with Column Pivoting (Ite-CholQR-CP)*. Features of the algorithm are summarized below:

- the main (essential) modification from a Cholesky QR type algorithm without pivoting is only in P-Chol-CP; the additional cost here is $O(n^3)$, which is generally much smaller than the $O(mn^2)$ cost for the other parts when $m \gg n$,
- almost all of the computations can be done by using major routines provided in BLAS and LAPACK libraries, which

Algorithm 4 Ite-CholQR-CP: Iterative Cholesky QR with Column Pivoting

Input: $A \in \mathbb{R}^{m \times n}$, pivot tolerance $\epsilon \in \mathbb{R}$

1: $k := 0$, $R := I_n$, $P := I_n$
2: **repeat**
3: $\quad W = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} := A^\top A$, where $W_{11} \in \mathbb{R}^{k \times k}$
4: $\quad W_{11} = R_{11}^\top R_{11}$ // normal Cholesky factorization
5: $\quad R_{12} := R_{11}^{-\top} W_{12}$
6: $\quad W_{22} := W_{22} - R_{12}^\top R_{12}$
7: $\quad [R_{22}, P', k'] :=$ P-Chol-CP(W_{22}, ϵ)
8: $\quad A(:, k+1:n) := A(:, k+1:n)P'$
9: $\quad R_{12} := R_{12}P'$
10: $\quad R' := \begin{pmatrix} R_{11} & R_{12} \\ O & R_{22} \end{pmatrix}$
11: $\quad A := A(R')^{-1}$
12: $\quad R := R'R$
13: $\quad P'' := \begin{pmatrix} I_k & O \\ O & P' \end{pmatrix}$
14: $\quad P := PP'$
15: $\quad k := k + k'$
16: **until** $k = n$
17: $[Q, R'] :=$ CholQR(A) // reorthogonalization
18: $R := R'R$

Output: $Q \in \mathbb{R}^{m \times n}$, $R \in \mathbb{R}^{n \times n}$, $P \in \mathbb{R}^{n \times n}$

enables us to benefit from the optimizations by hardware vendors,

- if a target matrix is tall and skinny (i.e., $m \gg n$), the computations for $A^\top A$ (line 3) and $A(R')^{-1}$ (line 11) are dominant, which can be done with Level-3 BLAS routines as in CholQR,
- in distributed parallel computation, the required communication is the reduction (e.g., MPI_Allreduce) for $A^\top A$ (line 3), whose number does not depend on n as in CholQR; $O(1)$ communications are needed, which means that Ite-CholQR-CP is CA,
- if l iterations are required for satisfying $k = n$ (line 16), both the total computation and communication costs are $(l+1)$ times larger than that in CholQR.

A key point to use Ite-CholQR-CP correctly is set an appropriate tolerance ϵ. Theoretical derivation of an optimal ϵ is one of our future works. In this research, by taking the experimental results shown in Figure 1 (c), we recommend setting $\epsilon \simeq 10^{-5}$. Under this setting, we expect that $l = 4$ (or 3) because $\epsilon^l \lesssim 10^{-16}$ will be required, in which the cost is roughly 5 (or 4) times larger than CholQR. It is worth mentioning that there is room for optimizing the computations in Algorithm 4; we give priority to simplicity in the current procedure shown in Algorithm 4. For example, we have possibilities of making computations in lines 3 to 6 and lines 11 more efficient, which is also a future work.

IV. Performance Evaluation

A. Settings

1) Program implementation: All program code is written in Fortran90 with double-precision floating-point. BLAS and LAPACK routines are used as much as possible; thread parallelized BLAS and LAPACK routines are employed, and manual thread parallelization by OpenMP is not done. In distributed parallel computation, program code is parallelized with MPI routines.

We give a brief description on the implementation of Ite-CholQR-CP. In the program code for single node computation, we use DGEMM for computing $W := A^\top A$; we do not use the symmetric property of W due to the low effective performance of the DSYRK routine in the computational environments used in the evaluation. It is worth mentioning that it will be better to optimize this part depending on a target environment including a BLAS library. P-Chol-CP (Algorithm 3) is implemented using the DSYR routines. Lines 4, 5, and 6 in Algorithm 4 is computed by the DPOTRF, DTRSM, and DSYRK routines, respectively. In our implementation of Ite-CholQR-CP, we use the DTRSM (line 11) and DTRMM (line 12) routines, which is no different from the case of CholQR.

In the program code for distributed parallel computation, we employ the one-dimensional block row layout for A and Q as already explained in Section II-B. The essential difference from the single node computation is issuing the MPI_Allreduce routine in the computation of $W := A^\top A$; each process first computes $W_p := A_p^\top A_p$ $(p = 1, \ldots, P)$, and then by MPI_Allreduce, all processes have W.

In the performance evaluation, we compare Ite-CholQR-CP with HQR-CP. In single node computation, we employ the DGEQP3 routine in LAPACK; for explicitly constructing the Q matrix, we execute the DORGQR routine after DGEQP3. In distributed parallel computation, we simply implement HQR-CP shown in Algorithm 1. This is because ScaLAPACK is not suitable for a tall and skinny matrix [1]. As already mentioned in Section II-C, a naive implementation of Algorithm 1 may fail to find pivots [17]. Thus, in order to provide accurate results, it is sometimes needed to explicitly compute 2-norm of columns, which requires additional communication costs. Considering this fact, we regard the performance of our naive HQR-CP implementation as a rough baseline of HQR-CP in distributed parallel computation. The computation of explicitly forming Q is also an in-house implementation, in which a blocking technique based on the compact WY representation [27] is employed for using DGEMM.

2) Computational environments: Table I lists the specifications of the computational systems used in the evaluation. The settings in each evaluation are shown in Table II, and they are used as follows:

- accuracy: Single node on Grand,

[1] In our understanding, blocking techniques employed in the ScaLAPACK routine works well for a matrix with sufficient number of columns and are not suitable for a tall-skinny matrix, we we use the 1-dimentional process grid.

- execution time in single node computation: Single node on Grand, OBCX, and BDEC-O,
- execution time in distributed parallel computation: Parallel on OBCX and BDEC-O.

These settings are basically determined according to those recommended in the manual and guide of each system.

3) Generation of test matrices: Throughout the experiments, we generate test matrices in the following way: given m, n ($\leq m$), r ($\leq n$), and σ ($0 < \sigma < 1$), set

$$\sigma_i = \begin{cases} \sigma^{\frac{i-1}{r-1}} & (1 \leq i \leq r), \\ 10^{-16} & (r+1 \leq i \leq n), \end{cases} \tag{17}$$

and generate a test matrix as

$$A := U\Sigma V, \tag{18}$$

where $U \in \mathbb{R}^{m\times n}$, $V \in \mathbb{R}^{n\times n}$ are randomly generated orthogonal matrices, and $\Sigma := \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$.

B. Results on accuracy

We compare the accuracy of computed results by Ite-CholQR-CP and HQR-CP; we use the program code of Ite-CholQR-CP in single node computation. Let $m = 10000$, $n = 50$, and $r = 40$, we change σ from 10^{-2} to 10^{-14} and generate test matrices. Then we compute their QRCP by DGEQP3, Ite-CholQR-CP with $\epsilon = 10^{-5}$ and $\epsilon = 0$, which considers only avoiding breakdown. For the computed results, we evaluate the following metrics:

- orthogonality of Q: $\|Q^\top Q - I_n\|_F/\sqrt{n}$,
- residual: $\|A\Pi - QR\|_F/\|A\|_F$,
- condition number of R_{11}: $\kappa_2(R_{11})$,
- 2-norm of R_{22}: $\|R_{22}\|_2$.

We use the information that $r = 40$ when determining R_{11} and R_{22}; how to determine the numerical rank from the computed R factor is not trivial, however, it is outside the scope and we do not discuss it here.

The obtained results are shown in Figure 2. We observe that Ite-CholQR-CP ($\epsilon = 10^{-5}$) provides solutions better than ((a), (b)) or as accurate as ((c), (d)) those by DGEPQ3. In addition, it is also found that when $\kappa_2(A) \gtrsim 10^8$, Ite-CholQR-CP ($\epsilon = 0$) is unstable in terms of $\kappa_2(R_{11})$ and $\|R_{22}\|_2$ ((c), (d)).

Figure 3 illustrates the correctness of the pivot selection in Ite-CholQR-CP with $\epsilon = 10^{-5}$ and $\epsilon = 0$; since we set $r = 40$, selections for the 1st to the 40th columns are essential here. From Figure 3 (a), we can find that regardless of $\kappa_2(A)$ (σ), Ite-CholQR-CP with $\epsilon = 10^{-5}$ correctly selects pivot through its iterative procedure. On the other hand, as shown in Figure 3 (b), in the cases that $\kappa_2(A) > 10^8$ ($\sigma < 10^{-8}$), Ite-CholQR-CP with $\epsilon = 0$ fails to select the correct pivot, which seems to be a main reason for the instability observed in the evaluation of $\kappa_2(R_{11})$ (Figure 2 (c)) and $\|R_{22}\|_2$ (Figure 2 (d)).

From the above results, we expect that Ite-CholQR-CP with an appropriate tolerance ϵ (e.g., 10^{-5}) can compute a QRCP of a tall and skinny matrix as accurate as that by HQR-CP (e.g., DGEQP3). It is worth mentioning that we focus on the equivalence of the pivot selection due to the simplicity in this study, however, there are other metrics for evaluating the accuracy of the factorization.

C. Execution time in single node computation

We next examine the execution time in single node computation. The evaluation settings are as follows:

- $m = 10000$, 50000, and 100000,
- $(n, r) = (16, 13)$, $(32, 26)$, $(64, 51)$, $(128, 102)$, $(256, 205)$, $(512, 410)$, and $(1024, 820)$,
- $\sigma = 10^{-12}$,
- for each test case, we run each method 5 times and evaluate the best results.

Figure 4 provides the speedup ratio of Ite-CholQR-CP ($\epsilon = 10^{-5}$) over DGEQP3 on each system. It is worth noting that the number of iterations in Ite-CholQR-CP is 4 in all cases; in the first 3 iterations, pivot selections are completed (as shown in Figure 3 (a)), and the 4th iteration is done for the reorthogonalization. In Figure 4, we see remarkable speedup of Ite-CholQR-CP for several problem settings; the best result is 45 times speedup for $m = 100000$, $n = 32$ on Grand. On the systems with Intel CPUs (i.e., Grand and OBCX), it is clear that as the shape of a matrix becomes closer to tall and skinny (i.e., larger m and smaller n), the speedup ratio tends to be larger. On the other hand, on BDEC-O, the speedup ratio is large when n is not small excepting $m = 10000$.

To better understand the results in Figure 4, we provide the obtained FLOPS in Figure 5. Here, we calculate the FLOPS value [2] as

$$(\text{FLOPS}) := \frac{4mn^2 - 4n^3/3}{(\text{exe. time})}. \tag{19}$$

As shown in Figure 5, FLOPS of DGEQP3 increases as n becomes larger on Grand and OBCX, on the other hand, FLOPS of Ite-CholQR-CP stagnates or decreases. These behaviors of the FLOPS values are consistent with the speedup ratio shown in Figure 4. On BDEC-O, FLOPS of both DGEQP3 and Ite-CholQR-CP increase as n grows, and we guess that this is due to the difference in BLAS library (i.e., Fujitsu BLAS or Intel MKL). We guess that the difference in memory subsystem (i.e., HBM2 or DDR4) is another reason for the different performance between BDEC-O and the other two systems with Intel CPUs.

In total, from the above results, we can confirm the advantage of Ite-CholQR-CP over DGEQP3 in the computation time of QRCP of a tall and skinny matrix in single node computation. Especially on a system with Intel CPUs, Ite-CholQR-C has a remarkable potential of outperforming the DGEQP3 routines provided in the Intel MKL library if a matrix is sufficiently tall and skinny.

[2] This is a kind of "effective" FLOPS for comparing the performance, and the actual number of floating-point operations in each algorithm is different.

TABLE I: Systems used in the performance evaluation.

Name	Grand (Grand Chariot)	OBCX (Oakbridge-CX)	BDEC-O (Wisteria/BDEC-01, Odyssey)
Site	IIC, Hokkaido Univ.	ITC, The Univ. of Tokyo	ITC, The Univ. of Tokyo
#nodes	1,024	1,368	7,680
Interconnect	Intel Omni-Path (Full-bisection Fat Tree)	Intel Omni-Path (Full-bisection Fat Tree)	Tofu Interconnect D (6D mesh / torus)
Node config.	2 CPUs, 384 GiB DDR4	2 CPUs, 128 GiB DDR4	1 CPU, 32 GiB HBM2
CPU	Intel Xeon Gold 6148 (Skylake, 2.4 GHz, 20 cores)	Intel Xeon Platinum 8280 (Cascade Lake, 2.7 GHz, 28 cores)	Fujitsu A64FX (2.2 GHz, 48 cores, 2 or 4 assistant cores)

TABLE II: Settings in each evaluation.

Name	Grand	OBCX		BDEC-O	
Case	Single node	Single node	Parallel	Single node	Parallel
Compiler	Intel ifort ver. 2021.7.1	Intel ifort ver. 2021.7.1	Intel mpiifort ver. 19.1.3.304	Fujitsu frtpx ver. 4.8.1	Fujitsu mpifrtpx ver. 4.8.1
Options	-O3 -qopenmp -ipo -xCORE-AVX512	-O3 -qopenmp -ipo -axCORE-AVX512	-O3 -qopenmp -axCORE-AVX512	-Kfast -Kopenmp -Nfjomplib	-Kfast -Kopenmp -Nfjomplib
BLAS/ LAPACK	Intel MKL ver. 2022.2 -mkl=parallel	Intel MKL ver. 2022.2 -mkl=parallel	Intel MKL ver. 2020.0.4 -mkl=parallel	Fujitsu BLAS/LAPACK ver. 1.2.36 -SSL2BLAMP	Fujitsu BLAS/LAPACK ver. 1.2.36 -SSL2BLAMP
MPI	Intel MPI	Intel MPI	Intel MPI	Fujitsu MPI	Fujitsu MPI
Assignment	40 threads / process 1 process / node	56 threads / process 1 process / node	28 threads / process 2 processes / node	48 threads / process 1 process / node	12 threads / process 4 processes / node
#nodes	1	1	8 to 1,024	1	8 to 4,096

D. Execution time in distributed parallel computation

Finally, we evaluate the performance in distributed parallel computation. The experimental settings are as follows:

- $m = 16,777,216 (= 2^{24})$,
- $(n, r) = (16, 13)$, $(32, 26)$, $(64, 51)$, $(128, 102)$, $(256, 205)$, $(512, 410)$, and $(1024, 820)$,
- $\sigma = 10^{-12}$,
- $P = 16, 32, \ldots, 1024, 2048$ on OBCX (2 processes per node) and $P = 32, 64, \ldots, 8192, 16384$ on BDEC-O (4 processes per node),
- for each test case, we run each method 5 times and evaluate the best results,
- we evaluate the performance in the strong scaling regime.

Figure 6 gives the results on OBCX, namely the execution time of HQR-CP and Ite-CholQR-CP ($\epsilon = 10^{-5}$), and the speedup ratio of Ite-CholQR-CP over HQR-CP. Those on BDEC-O are provided in Figure 7.

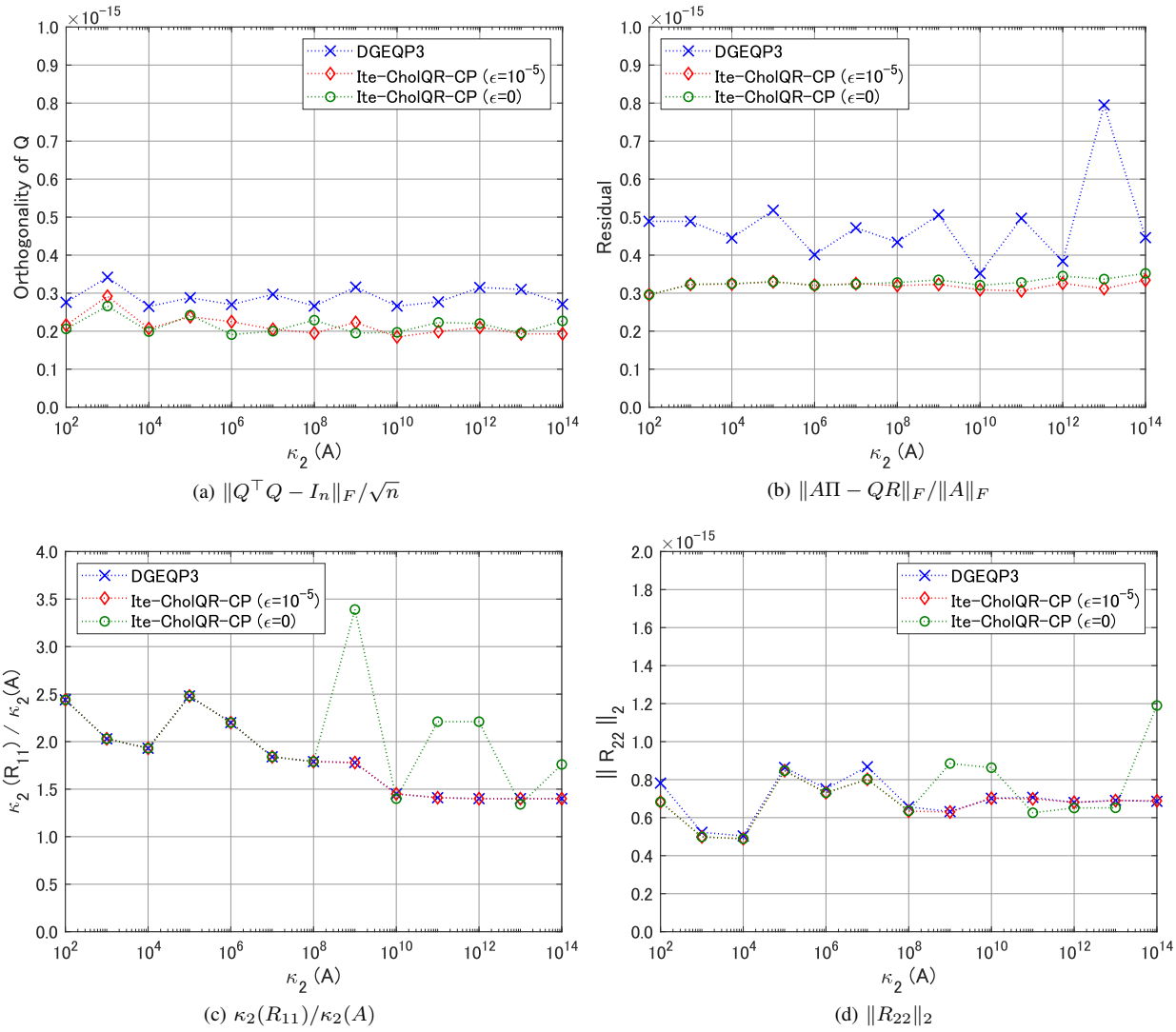
From Figure 6 (c), we can find that Ite-CholQR-CP is faster than HQR-CP in all cases (P and n) and that the best is more than 25 times speedup when $P = 1024$ and $n = 128$. As shown in Figure 6 (b), although the reduction of the execution time by increasing the number of nodes is limited in the case of $n = 16$, 32, and 64, the time itself is still shorter than that of HQR-CP in these cases. Figure 7 (c) shows different performance behaviors from Figure 6 (c); the range of n in which Ite-CholQR-CP is efficient is different between the cases where the number of nodes is small and large. Although Ite-CholQR-CP outperforms HQR-CP in many cases, the obtained speedup ratio is less than that in OBCX.

To better understand the performance of each method, we provide the breakdown of the execution time in Table III together with the percentage of the communication time; we present the cases where the number of nodes is small and large. From Table III, we see that the main reason that Ite-CholQR-CP is faster than HQR-CP lies in the difference of the computational time when the number of nodes is small. These results are consistent with the results shown in Figures 4 (b) and (c); for example, there is no advantage of Ite-CholQR-CP on BDEC-O when $m = 100000$ and $n = 16$ (Figure 4 (c)). On the other hand, from Table III, we can find the differences in the communication cost between Ite-CholQR-CP and HQR-CP when the number of nodes is large; especially in the case of 1024 nodes on OBCX, we can confirm the advantage that Ite-CholQR-CP is CA; the communication cost in Ite-CholQR-CP is much smaller than that in HQR-CP. In order to discuss the communication cost in more detail, we give the behavior of the communication cost on OBCX (1024 nodes) and BDEC-O (4096 nodes) in Figure 8. There is a clear gap in the behavior of the communication cost of Ite-CholQR-CP on BDEC-O; it exists between $n = 64$ and 128. The reasons for this gap is currently not clear, and we guess that it is due to the implementation of the Fujitsu MPI library and/or configuration of the interconnect on BDEC-O.

In conclusion, it can be confirmed that Ite-CholQR-CP is generally faster than HQR-CP in distributed parallel computation. Especially on a system based on Intel CPUs, Ite-CholQR-CP provides more than 25 time speedup over HQR-CP when the number of nodes is sufficiently large, which is due to the feature of CA, i.e., the advantage in the communication cost.

V. Related Work

The main issue in QRCP is how to select pivot correctly and efficiently. In usual, the selection of pivot is step by step,



(a) $\|Q^\top Q - I_n\|_F/\sqrt{n}$ (b) $\|A\Pi - QR\|_F/\|A\|_F$

(c) $\kappa_2(R_{11})/\kappa_2(A)$ (d) $\|R_{22}\|_2$

Fig. 2: Evaluation of the accuracy of the computed results: $m = 10000$, $n = 50$, and $r = 40$.

which makes efficient parallel computation difficult. An early contribution for parallel QRCP is that by Bischof [28], in which an alternative strategy of pivot selection suitable for parallel computation was proposed. Another important study is that provided blocking techniques for using Level-3 BLAS routines [16], as mentioned in Section II-C.

As the size of distributed parallel systems increases, the reduction of the communication cost, i.e., the feature of CA, becomes more important. Under this situation, a CA type RRQR algorithm was proposed [29], in which the so-called tournament pivoting strategy is employed for reducing the communication cost and increasing the efficiency of parallel computation. However, it is reported that the accuracy and stability of QRCP by tournament pivoting are often less than those by the traditional HQR-CP.

An important approach for computing tall-skinny QRCP was presented by Cunha and Patterson [30]. It is shown that one can effectively compute QRCP for a tall and skinny matrix by combining a fast and accurate (normal) QR factorization and the QRCP for the obtained R factor. If one will use TSQR [21] for normal QR factorization, based on the existing performance results [8], [11], we expect that Ite-CholQR-CP will be faster. However, if a Cholesky QR type algorithm (e.g., Shifted CholeskyQR3) will be used, it is not clear which algorithm will be faster. Because the number of iterations in both algorithms depends on the condition number of an input matrix and parameters in each algorithm, general predictions are difficult. It is worth mentioning that our approach is suitable for stopping the computation by monitoring the diagonal of the R factor (i.e., truncation or partial QRCP). In the above approach that combines normal QR and QRCP for R, whole of the normal QR factorization is required. In this situation, our algorithm has a remarkable advantage. It is clear that performance comparisons between the above approach and our approach should be conducted, however, the presented results in this paper show that our approach will be competitive, and it

Col. Ind.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$\sigma = 10^{-3}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	×	✓	✓	✓	×	×	✓
$\sigma = 10^{-6}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×
$\sigma = 10^{-9}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	✓	✓	✓
$\sigma = 10^{-12}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	✓	✓	×	×

1st iteration 2nd iteration 3rd iteration

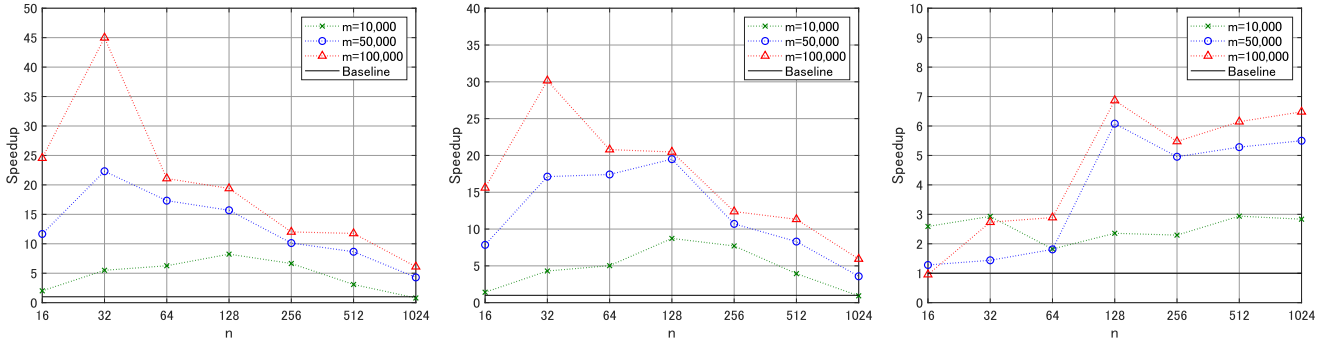
(a) $\epsilon = 10^{-5}$

Col. Ind.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
$\sigma = 10^{-3}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×	✓
$\sigma = 10^{-6}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×	×
$\sigma = 10^{-9}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×	✓	✓	✓	×	×	×	✓	✓	×	×	×	×	×
$\sigma = 10^{-12}$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	✓	×	×	×	×	×

1st iteration 2nd iteration 3rd iteration

(b) $\epsilon = 0$

Fig. 3: Correctness of the pivot selection in Ite-CholQR-CP (✓: correct, ×: incorrect): $m = 10000$, $n = 50$, and $r = 40$.



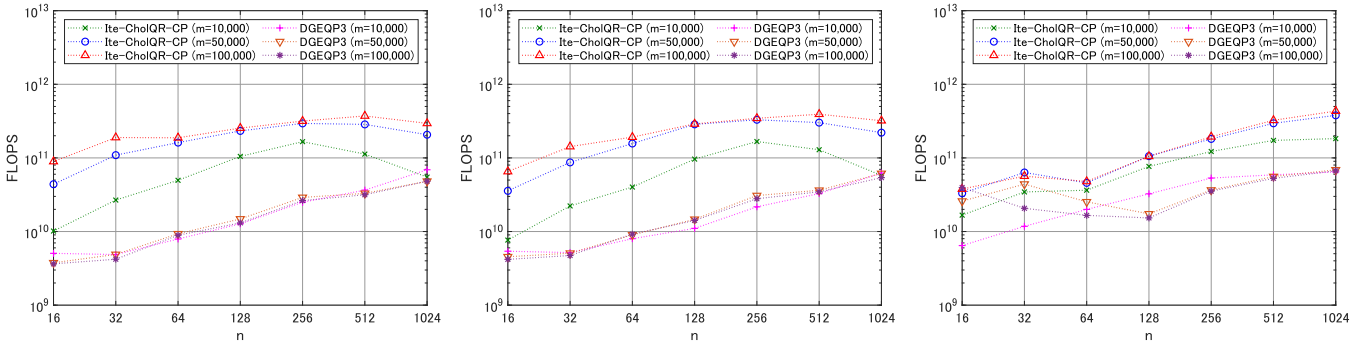
(a) Grand (Skylake) (b) OBCX (Cascade Lake) (c) BDEC-O (A64FX)

Fig. 4: Speedup of Ite-CholQR-CP ($\epsilon = 10^{-5}$) over DGEQP3 on single node.

is worth reporting the idea behind our algorithm, namely, how to extend normal Cholesky QR algorithms to QRCP without losing advantages in high performance computing.

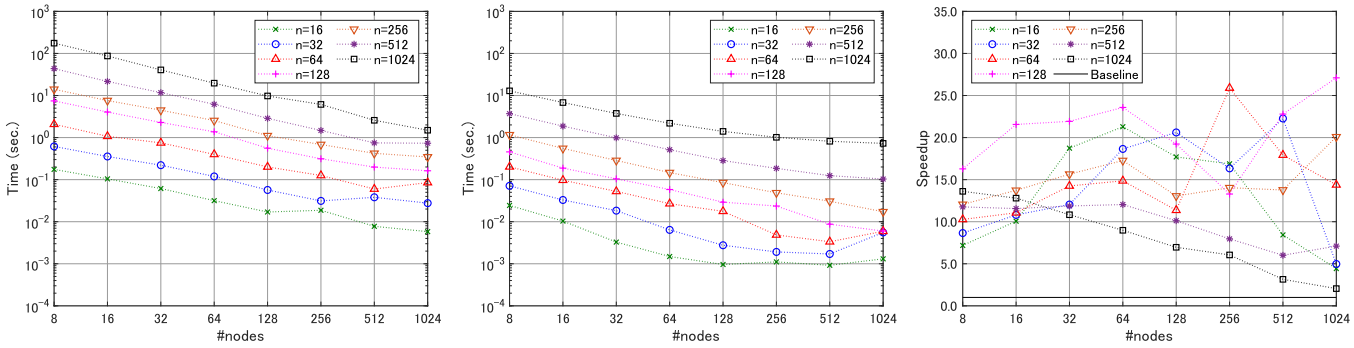
Recently, algorithms based on randomization has attracted much attention in the community of numerical linear algebra [31]. The paper by Duersch and Gu [32] provides a good survey on randomized techniques for rank-revealing matrix factorization including QRCP. The main idea in randomized QRCP algorithms is first obtaining appropriate pivot selections, namely P, via the random sketching with a small cost. Then, by using the information, swap the columns at one time, and apply an algorithm of normal QR factorization to the swapped matrix. Based on this idea, Rokhlin and Tygert proposed a randomized algorithm for overdetermined linear least-squares regression [33]. Although the target problem is not QRCP, the proposed algorithm is essentially equivalent to randomized algorithms for computing QRCP. Furthermore, several algorithms for QRCP based on Householder QR were presented [34]–[36]. An algorithm based on Cholesky QR was also presented [37] and discussed in detail [38].

Many of the above studies [28], [29], [34]–[36] are based on Householder QR and consider the case $m \simeq n$, which is quite different from the problem setting assumed in this work. The papers by Rokhlin and Tygert [33], Balabanov and Grigori [37], and Balabanov [38] consider tall-skinny matrices and are highly relevant to this research. It is clear that comparing the performance of Ite-CholQR-CP with that of an algorithm based on randomization will be important for users with practical applications. However, algorithm development based on randomization is a recent research direction, and algorithms, implementations, and evaluation metrics (especially on accuracy) have not been standardized yet. On the other hand, routines provided in LAPACK (ScaLAPACK) has a long history and has been widely used in various application programs. This is the reason we give a much higher priority to the comparison with HQR-CP than that with recently presented randomized algorithms.



(a) Grand (Skylake) (b) OBCX (Cascade Lake) (c) BDEC-O (A64FX)

Fig. 5: Obtained FLOPS.



(a) HQR-CP (b) Ite-CholQR-CP ($\epsilon = 10^{-5}$) (c) Speedup: Ite-CholQR-CP over HQR-CP

Fig. 6: Performance results in distributed parallel computation on OBCX: $m = 16,777,216$, in the strong scaling regime.

TABLE III: Breakdown of the execution time (in sec.).

(a) OBCX

#nodes	n	HQR-CP comp.	HQR-CP comm.		Ite-CholQR-CP comp.	Ite-CholQR-CP comm.	
8	16	1.7E-01	5.5E-03	(3%)	2.3E-02	1.7E-03	(7%)
	128	7.3E+00	1.4E-01	(2%)	4.0E-01	5.6E-02	(12%)
	1024	1.6E+02	1.3E+01	(7%)	1.2E+01	8.4E-01	(6%)
1024	16	2.3E-03	7.6E-03	(22%)	6.4E-04	6.7E-04	(49%)
	128	9.9E-02	6.3E-02	(61%)	4.0E-03	2.0E-03	(66%)
	1024	7.7E-01	7.3E-01	(52%)	6.8E-01	4.8E-02	(93%)

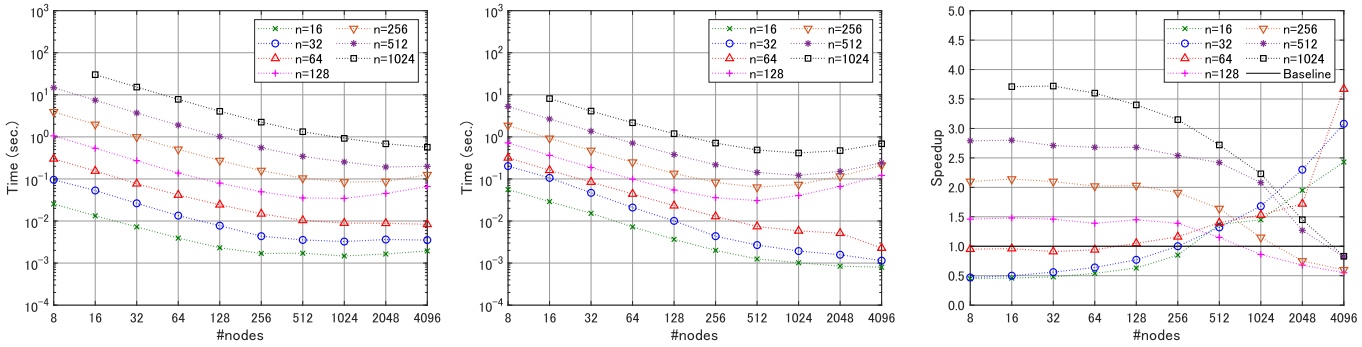
(b) BDEC-O

#nodes	n	HQR-CP comp.	HQR-CP comm.		Ite-CholQR-CP comp.	Ite-CholQR-CP comm.	
16	16	1.2E-02	1.2E-03	(9%)	2.8E-02	1.2E-03	(4%)
	128	5.2E-01	1.4E-02	(3%)	3.6E-01	6.2E-03	(2%)
	1024	3.0E+01	3.9E-01	(1%)	7.9E+00	2.2E-01	(3%)
4096	16	2.3E-04	1.7E-03	(88%)	2.4E-04	5.6E-04	(71%)
	128	3.7E-03	6.3E-02	(94%)	3.0E-03	1.2E-01	(98%)
	1024	2.1E-01	3.6E-01	(63%)	1.8E-01	5.1E-01	(74%)

VI. Conclusion

The main contribution of the paper is to propose a new Cholesky QR type algorithm for computing tall-skinny QRCP, namely Ite-CholQR-CP. The algorithm is based on the ideas of modifying Cholesky factorization with complete pivoting and employing an iterative structure in which pivot selections are made step by step. Ite-CholQR-CP has a similar structure as Cholesky QR type algorithms for normal QR (without pivoting), which preserves the advantages suitable for high-performance computing in recent computational systems. Through the performance evaluation, it is confirmed that Ite-CholQR-CP provides a QRCP as accurate as that by HQR-CP. In addition, in both single node and distributed parallel computations, Ite-CholQR-CP is generally faster than HQR-CP, e.g., DGEPQ3 in LAPACK.

As a next step, a theoretical analysis of Ite-CholQR-CP is needed, which will strengthen the accuracy of solutions computed by Ite-CholQR-CP. It will be also helpful to determine an optimal tolerance ϵ in Ite-CholQR-CP. Another important task is to optimize the algorithm of Ite-CholQR-CP; there is room to reduce the computational cost and to optimize the implementation. To compare Ite-CholQR-CP with randomized algorithms, as mentioned in Section V, is also required; from the viewpoints of both accuracy and execution time, detailed performance tests should be conducted. In addition, as mentioned in Section V, conducting performance comparison with the approach that combines a fast and accurate normal QR factorization and QRCP of the R factor is also impor-



(a) HQR-CP (b) Ite-CholQR-CP ($\epsilon = 10^{-5}$) (c) Speedup

Fig. 7: Performance results in distributed parallel computation on BDEC-O: $m = 16,777,216$, in the strong scaling regime.

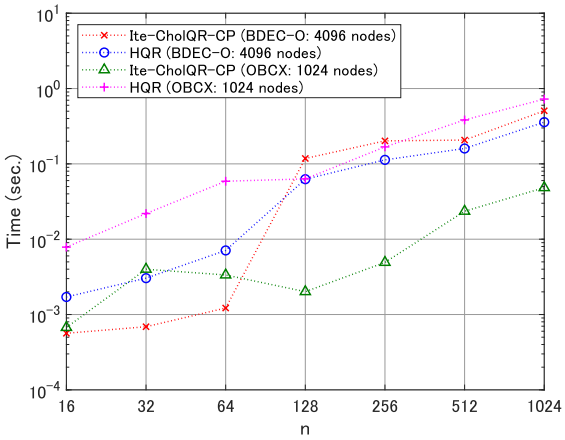


Fig. 8: Behavior of communication time: #nodes is 1024 on OBCX and 4096 on BDEC-O.

tant. Compared with these two approaches, Ite-CholQR-CP, is competitive, however, further detailed performance evaluation will be required for selecting an appropriate algorithm in each application. Finally, it will be of interest to demonstrate the effectiveness of Ite-CholQR-CP in practical applications, which will make the advantages of Ite-CholQR-CP clearer and clarify issues for further performance improvement.

Acknowledgment

The authors thank the anonymous reviewers for their valuable comments. This research was supported by JST, PRESTO (Grant Number: JPMJPR20M8), JSPS KAKENHI (Grant Numbers: JP21K11909, JP23H00462), and "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" and "High Performance Computing Infrastructure" in Japan (Project ID: jh230010). A part of computational resource of Wisteria/BDEC-01 (Odyssey) was awarded by "Large-scale HPC Challenge" Project, Information Technology Center, The University of Tokyo, and the first author thanks the kind support by the stuff in The University of Tokyo.

References

[1] T. F. Chan, "Rank revealing QR factorizations," *Linear Algebra and its Applications*, vol. 88-89, pp. 67–82, 1987.

[2] G. Golub, "Numerical methods for solving linear least squares problems," *Numerische Mathematik*, vol. 7, no. 3, pp. 206–216, 1965.

[3] S. Chandrasekaran and I. C. F. Ipsen, "On Rank-Revealing Factorisations," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 2, pp. 592–622, 1994.

[4] Y. P. Hong and C.-T. Pan, "Rank-Revealing QR Factorizations and the Singular Value Decomposition," *Mathematics of Computation*, vol. 58, no. 197, pp. 213–232, 1992.

[5] Q. Ma, S. Deshmukh, and R. Yokota, "Scalable Linear Time Dense Direct Solver for 3-D Problems without Trailing Sub-Matrix Dependencies," in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2022, pp. 1–12.

[6] H. Al Daas, G. Ballard, P. Cazeaux, E. Hallman, A. Międlar, M. Pasha, T. W. Reid, and A. K. Saibaba, "Randomized Algorithms for Rounding in the Tensor-Train Format," *SIAM Journal on Scientific Computing*, vol. 45, no. 1, pp. A74–A95, 2023.

[7] T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, and Y. Yamamoto, "CholeskyQR2: A simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system," in *Proceedings of the 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, ser. ScalA '14, 2014, pp. 31–38.

[8] T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa, "Shifted Cholesky QR for Computing the QR Factorization of Ill-Conditioned Matrices," *SIAM Journal on Scientific Computing*, vol. 42, no. 1, pp. A477–A503, 2020.

[9] T. Terao, K. Ozaki, and T. Ogita, "LU-Cholesky QR algorithms for thin QR decomposition," *Parallel Computing*, vol. 92, p. 102571, 2020.

[10] I. Yamazaki, S. Tomov, and J. Dongarra, "Mixed-Precision Cholesky QR Factorization and Its Case Studies on Multicore CPU with Multiple GPUs," *SIAM Journal on Scientific Computing*, vol. 37, no. 3, pp. C307–C330, 2015.

[11] T. Fukaya, "Distributed Parallel Tall-Skinny QR Factorization: Performance Evaluation of Various Algorithms on Various Systems," in *Parallel and Distributed Computing, Applications and Technologies*, H. Takizawa, H. Shen, T. Hanawa, J. Hyuk Park, H. Tian, and R. Egawa, Eds., 2023, pp. 275–287.

[12] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya, "Roundoff error analysis of the CholeskyQR2 algorithm," *Electronic Transactions on Numerical Analysis*, vol. 44, pp. 306–326, 2015.

[13] X. Wu and E. Di Napoli, "Advancing the distributed multi-gpu chase library through algorithm optimization and nccl library," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, p. 1688–1696.

[14] M. Gu and S. C. Eisenstat, "Efficient Algorithms for Computing a Strong Rank-Revealing QR Factorization," *SIAM Journal on Scientific Computing*, vol. 17, no. 4, pp. 848–869, 1996.

[15] P. Businger and G. H. Golub, "Linear least squares solutions by householder transformations," *Numerische Mathematik*, vol. 7, no. 3, pp. 269–276, 1965.

[16] G. Quintana-Ortí, X. Sun, and C. H. Bischof, "A BLAS-3 Version of the QR Factorization with Column Pivoting," *SIAM Journal on Scientific Computing*, vol. 19, no. 5, pp. 1486–1494, 1998.

[17] Z. Drmač and Z. Bujanović, "On the Failure of Rank-Revealing QR Factorization Software – A Case Study," *ACM Transactions on Mathematical Software*, vol. 35, no. 2, pp. 12:1–12:28, 2008.

[18] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*. SIAM, 1997.

[19] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. The Johns Hopkins University Press, 2012.

[20] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, "Minimizing Communication in Numerical Linear Algebra," *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 3, pp. 866–901, 2011.

[21] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, "Communication-optimal Parallel and Sequential QR and LU Factorizations," *SIAM Journal on Scientific Computing*, vol. 34, no. 1, pp. A206–A239, 2012.

[22] A. Stathopoulos and K. Wu, "A Block Orthogonalization Procedure with Constant Synchronization Requirements," *SIAM Journal on Scientific Computing*, vol. 23, no. 6, pp. 2165–2182, 2002.

[23] E. Hutter and E. Solomonik, "Communication-Avoiding Cholesky-QR2 for Rectangular Matrices," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019, pp. 89–100.

[24] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Society for Industrial and Applied Mathematics, 2002.

[25] H. D. Nguyen and J. Demmel, "Reproducible Tall-Skinny QR," in *2015 IEEE 22nd Symposium on Computer Arithmetic*, 2015, pp. 152–159.

[26] Z. Xu, J. J. Alonso, and E. Darve, "A numerically stable communication-avoiding s-step GMRES algorithm," 2023, arXiv:2303.08953.

[27] R. Schreiber and C. Van Loan, "A Storage-Efficient WY Representation for Products of Householder Transformations," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, no. 1, pp. 53–57, 1989.

[28] C. H. Bischof, "A Parallel QR Factorization Algorithm with Controlled Local Pivoting," *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 1, pp. 36–57, 1991.

[29] J. W. Demmel, L. Grigori, M. Gu, and H. Xiang, "Communication Avoiding Rank Revealing QR Factorization with Column Pivoting," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 1, pp. 55–89, 2015.

[30] R. D. da Cunha, D. Becker, and J. C. Patterson, "New Parallel (Rank-Revealing) QR Factorization Algorithms," in *Euro-Par 2002 Parallel Processing*, B. Monien and R. Feldmann, Eds. Springer Berlin Heidelberg, 2002, pp. 677–686.

[31] P. Drineas and M. W. Mahoney, "RandNLA: randomized numerical linear algebra," *Communications of the ACM*, vol. 59, no. 6, pp. 80–90, 2016.

[32] J. A. Duersch and M. Gu, "Randomized Projection for Rank-Revealing Matrix Factorizations and Low-Rank Approximations," *SIAM Review*, vol. 62, no. 3, pp. 661–682, 2020.

[33] V. Rokhlin and M. Tygert, "A Fast Randomized Algorithm for Overdetermined Linear Least-Squares Regression," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 36, pp. 13 212–13 217, 2008.

[34] J. A. Duersch and M. Gu, "Randomized QR with Column Pivoting," *SIAM Journal on Scientific Computing*, vol. 39, no. 4, pp. C263–C291, 2017.

[35] P.-G. Martinsson, G. Quintana OrtÍ, N. Heavner, and R. van de Geijn, "Householder QR Factorization With Randomization for Column Pivoting (HQRRP)," *SIAM Journal on Scientific Computing*, vol. 39, no. 2, pp. C96–C115, 2017.

[36] J. Xiao, M. Gu, and J. Langou, "Fast Parallel Randomized QR with Column Pivoting Algorithms for Reliable Low-Rank Matrix Approximations," in *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, Feb. 2017, pp. 233–242.

[37] O. Balabanov and L. Grigori, "Randomized block Gram-Schmidt process for solution of linear systems and eigenvalue problems," 2021, arXiv:2111.14641.

[38] O. Balabanov, "Randomized Cholesky QR factorizations," 2022, arXiv:2210.09953.