大規模分散並列環境におけるコレスキーQR 型アルゴリズムによる縦長行列の列ピボット付き QR 分解の性能評価

深谷 猛

北海道大学 情報基盤センター

１．はじめに

対象の行列を都合のよい行列の積に分解する行列分解は、様々な科学技術計算において利用されている基盤技術である。本稿では、基本的な行列分解の一つである QR 分解に関して、特に、縦長（Tall and Skinny）の行列に対する列ピボット（Column Pivoting）を伴う QR 分解、通称「列ピボット付き QR 分解（QRCP：QR factorization with Column Pivoting)」[1]を扱う。QRCP に対する数値計算アルゴリズムとしては、列ピボット付きハウスホルダーQR 分解[2,3]が広く知られている。これに対して、最近、著者らは、通常（列ピボットがない場合）の縦長行列に対する QR 分解におけるコレスキーQR 型アルゴリズム[4]の有効性に注目し、QRCP に対するコレスキーQR 型のアルゴリズムの研究開発を行っている。本稿では、2023 年の 1 月に東京大学情報基盤センターのスーパーコンピュータシステム「Wisteria/BDEC-01 (Odyssey)」および「Oakbridge-CX」を利用して実施した大規模 HPC チャレンジにおいて、著者らが開発しているコレスキーQR 型の QRCP アルゴリズムの大規模分散並列環境における性能を評価した結果を、アルゴリズムの概要とともに報告する。

２．列ピボット QR 分解（QRCP）の概要

本稿で取り扱う QRCP の概要を述べる。QRCP では、与えられた行列Aに対して

$$AP = QR = (Q_1 \quad Q_2)\begin{pmatrix} R_{11} & R_{12} \\ O & R_{22} \end{pmatrix}$$

の分解を行う。ただし、Aは$m \times n$行列（$m \geq n$）で、本稿では特に縦長（$m \gg n$）の場合を考える。また、Aの数値ランクを$r\ (\leq n)$とする。具体的には、Aの特異値を$\sigma_1 > \sigma_2 > \cdots > \sigma_n$としたときに、$\sigma_r$と$\sigma_{r+1}$の間に大きな差があり、同時に、$\sigma_i/\sigma_1\ (i = r+1, \ldots, n)$が非常に小さい（例：倍精度の場合で$O(10^{-16})$）場合を考える。一方、$P$は$n \times n$置換行列、$Q$は$m \times n$列直交行列（$Q^{\top}Q = I_n$を満たす）、$R$は$n \times n$上三角行列である。なお、$I_n$は$n$次単位行列である。

次に QRCP の特徴であるPの選び方について述べる。kを整数（$1 \leq k \leq n$）として、Q_1を$m \times k$、R_{11}を$k \times k$とする。このとき、$\kappa_2(R_{11})$ができるだけ小さく（良条件）、かつ、$\|R_{22}\|_2$もできるだけ小さくなるように、P, kを選ぶ。ここで$\kappa_2(\cdot)$は行列の 2 ノルム条件数である。なお、理想的には、$k = r, \kappa_2(R_{11}) \simeq \sigma_1/\sigma_r$である。

QRCP は行列のランク情報を考慮した QR 分解（RRQR：Rank Revealing QR 分解）の一種であり、計算コストは特異値分解より一般的に小さい。そのため、様々な科学技術計算の中で使用され、行列の低ランク近似、行列のランク推定、線形従属に近いベクトル群に対する正規直交基底の計算、などの応用例がある[3,5,6]。

QRCP を計算する代表的な数値計算アルゴリズムとしては、列ピボット付きハウスホルダーQR

分解（HQR-CP：Householder QR with Column Pivoting）のアルゴリズムが知られている。HQR-CP では、通常のハウスホルダーQR 分解の手順（ハウスホルダー変換によりAの列を左から順番に上三角化する）において、毎回、すでに選択された列ベクトルが張る空間の直交補空間におけるベクトルの成分が最大である列ベクトルを探索し、貪欲法的に選択することでピボットを決定する。RRQR や HQR-CP の詳細については、例えば、文献[7]などを参照されたい。

３．コレスキーQR 型アルゴリズム

本節では、（列ピボットがない）通常の QR 分解に対するコレスキーQR 型アルゴリズムの概要を述べる。コレスキーQR 型アルゴリズムの基本（以下、CholQR）は、1）$A^{\top}A \to W$、2）$W \to R^{\top}R$（Wのコレスキー分解）、3）$AR^{-1} \to Q$であり（文献[8]の Theorem 5.2.3 参照）、Triangular Orthogonalization 型の QR 分解アルゴリズム[9]の一種である。縦長行列に対しては、計算コストの主要部は手順の 1 と 3 になり、どちらも Level-3 の BLAS ルーチンで実行可能である。そのため、現在の計算機環境では（Intel MKL のような高性能な BLAS ライブラリの恩恵もあって）高い実行性能を得ることができる。また、分散並列計算の場合、（Aに関して列方向の 1 次元ブロック分散を仮定すれば）手順 1 の途中で 1 度だけ集団通信（MPI_Allreduce）が必要で、所謂、通信回避（CA：Communication Avoiding）型アルゴリズム[10]となる。さらに、（分散並列版も含めて）アルゴリズムの実装が非常にシンプルであるという、実用において大きな特徴も持つ。

このように、CholQR は、性能（実行時間）の面で大きな利点を持つが、一方で、計算精度の面で致命的な弱点がある。具体的には、$\kappa_2(A)$の二乗に比例して、Qの直交性が悪化し、かつ、$\kappa_2(A)$が一定以上（倍精度の場合で$\kappa_2(A) \gtrsim 10^8$）の場合にアルゴリズムが破綻する。そのため、CholQR 単体では、計算は高速でも、実用可能な場面は非常に限られる[11]。

上述の計算精度面の課題に対して、まず、CholQR を 2 回繰り返すことでQの直交性が十分な精度（ハウスホルダーQR 等と同程度）まで改善することが報告されている[11,12]。これは、再直交化を行うことと解釈可能であり、アルゴリズムはコレスキーQR2 と呼ばれる。ただし、$\kappa_2(A)$が一定以上の場合にアルゴリズムが破綻する問題は残っている。

悪条件（$\kappa_2(A)$が一定以上）の行列に対しては、CholQR の手順 2 で、$W' \coloneqq W + sI_n$として、W'のコレスキー分解を行い、得られたRを用いて、（QR 分解にはならないが）Aの条件数を下げることが可能であることが示されている。ここで、sはシフトと呼ばれ、十分に小さく、かつW'のコレスキー分解が数値的に実行可能となる（破綻しない）ように設定する。また、この手順は、シフト付きコレスキーQR[4]と呼ばれ、これを前処理として用いた上で、前述のコレスキーQR2 を行うことで、悪条件の行列に対しても十分な精度の QR 分解を得ることが可能となる。これらの詳細については、文献[4]などを参照されたい。

上述した、CholQR の改良版では、演算回数や分散並列計算時の通信回数・通信データ量が、CholQR の 2 倍または 3 倍と増加する。一方で、大部分の演算は Level-3 BLAS で実行可能、通信回数が行列サイズに依存しない（$O(n)$ではなく$O(1)$）、既存ライブラリを容易に利用可能（実装がシンプル）、という CholQR が持つ利点は維持している。そして、実際の計算機上で性能評価を行うと、依然として、ハウスホルダーQR をはじめとする他のアルゴリズムよりも高速となる場合が多いことが分かっている[13,14]。この事実は、現在の計算機環では、演算量以外の部分が計算速度に与える影響が極めて大きい、ということを改めて示しており、高速なアルゴリズムを開発する際の重要な指針となる。

４．QRCP に対するコレスキーQR 型アルゴリズムの概要

前節で紹介したように、通常の QR 分解に対するコレスキーQR 型アルゴリズムが持つ現在の計算機環境との親和性は注目に値するものであり、QRCP に対して同様の特徴を持ったアルゴリズム、すなわち、コレスキーQR 型 QRCP アルゴリズムの開発を目指すのは自然な流れである。

数学的（丸め誤差がない場合）には、CholQR の手順 2 で行うコレスキー分解を完全ピボット（Complete Pivoting）付きコレスキー分解：$P^{\top}WP \rightarrow R^{\top}R$ [15]に置き換えることで、QRCP を計算できる。しかし、数値計算において丸め誤差の影響を無視することはできず、単純にこの手順を実行した場合、誤ったピボットの選択とコレスキー分解の破綻（通常の QR 分解の場合と同じ問題）が起きる。

以下、我々が開発している QRCP に対するコレスキーQR 型アルゴリズムの要点を述べる。なお、アルゴリズムの詳細については、現在、論文を執筆中であるため、本稿で詳細を述べず、論文（もしくはそのプレプリント版）に委ねる。

上述のように、Aのグラム行列Wに対して完全ピボット付きコレスキー分解を行うと、丸め誤差の影響を受ける。ただし、数値実験を行った結果から、完全ピボット付きコレスキー分解の計算の途中までは丸め誤差の影響が小さく、正しいピボットが得られることが確認できた。そこで、完全ピボット付きコレスキー分解の計算における特定部分の値の大きさをチェックし、これがある条件を満たした場合に、以降のピボットが信頼できないとして、計算と途中で止めることを考える。そして、それまでに得られたピボット（置換行列に相当）と上三角行列を用いて、CholQR の手順 3 を行う。以上の手順を繰り返すことで、正しいピボットを段階的に得ることができ、（再直交化に似た構造があるため）Qの直交性も十分な精度となる。以降、本アルゴリズムを Ite-CholQR-CP（Iterative CholeskyQR with Column Pivoting）と呼ぶ。

Ite-CholQR-CP と CholQR との主要な違いは、CholQR の手順 2 のコレスキー分解の変更であり、手順 1 と 3 に違いはない。つまり、目標としている「CholQR の高性能計算への親和性の維持」は達成されている。また、名前が示すように、反復型のアルゴリズムであり、反復回数は完全ピボット付きコレスキー分解を停止する際の条件に依存する。計算量などは、CholQR のそれを反復回数倍にしたものとなるが、適切な条件を設定することで、高々数回（4 回もしくは 5 回程度）となることを見込んでいる。よって、前節の最後で述べたように、演算回数等が数倍に増えたとしても、計算機環境との親和性を有することで、既存アルゴリズムとの計算時間の比較において、十分なポテンシャルを持っていることが期待できる。

５．性能評価

５．１．実装の概要

Ite-CholQR-CP の分散並列実装の概略を示す。Aのデータは列方向の一次元ブロック分散を採用する。具体的には、

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_p \\ \vdots \\ A_P \end{pmatrix}$$

と分割した際に、p番目のプロセスがA_pを保持する分散方法である（P：プロセス数）。また、Qについても同様の分散を採用する。一方、PとRは全てのプロセスが冗長にデータを保持する実

装とする。

$A^{\mathsf{T}}A \to W$の計算は、まず、各プロセスが独立に$W_p \to A_p^{\mathsf{T}}A_p$を計算（DGEMM を利用）し、次に、MPI_Allreduce を使い、すべてのプロセスが$W = \sum_{p=1}^{P} W_p$を得る。部分的な完全ピボット付きコレスキー分解は各プロセスが独立に実行する。その後、得られた置換行列に従って、自分が保持するA_pの列の交換を行う（列の交換のためプロセス間通信は不要）。最後に、$(AP)R^{-1} \to Q$の計算（DTRSM を利用）とRの更新の計算（DTRMM を利用）を各プロセスが独立に行う。以下、この手順を必要な回数だけ反復する。

今回の性能評価では、Fortran90 と倍精度を用いて上記のアルゴリズムを実装した。上述の通り、密行列の基本演算は BLAS、プロセス間通信は MPI を用いた。なお、$W_p \to A_p^{\mathsf{T}}A_p$の計算は、対称性から DSYRK ルーチンを利用する方が演算量は少なくなるが、A_pが縦長の場合には DGEMM の方が高速であることが多く、今回の環境でもその傾向が確認されたために DGEMM を使った。

５．２．テスト問題

数値実験で使用するテスト行列の生成方法を簡単に紹介する。行列生成時の入力として、$m, n\ (\leq m), r\ (\leq n), \sigma\ (0 < \sigma < 1)$を与えると、

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

とした上で、$\Sigma \coloneqq \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$として、$A \coloneqq U\Sigma V$と生成する。ただし、$U, V$はそれぞれ$m \times n, n \times n$のランダムに生成した直交行列である。なお、テスト行列の生成も倍精度で行った。

５．３．評価環境と設定

以下で報告する性能評価は、東京大学情報基盤センターが運用しているスーパーコンピュータシステム「Wisteria/BDEC-01 (Odyssey)」および「Oakbridge-CX」を用いて実施した。以下、前者を BDEC-O、後者を OBCX と呼ぶ。これらのシステムの構成については、東京大学情報基盤センターが提供している公式情報[16,17]に委ねる。一方、今回の性能評価における設定については表 1 の通りである。なお、表 1 に示した設定は、これらのシステムの講習会の資料等を参考にした。

表 1：性能評価実施時の設定。

略称	BDEC-O	OBCX
システム名	Wisteria/BDEC-01 (Odyssey)	Oakbridge-CX
コンパイラ	Fujitsu mpifrtpx (ver. 4.8.1)	Intel mpiifort (ver. 19.1.3.304)
オプション	-Kfast -Kopenmp -Nfjomplib	-O3 -qopenmp -axCORE-AVX512
BLAS/LAPACK	Fujitsu BLAS/LAPACK (ver. 1.2.36), -SSL2BLAMP	Intel MKL (ver. 2020.0.4), -mkl=parallel
MPI	Fujitsu MPI	Intel MPI
プロセス配置	4 プロセス/ノード	2 プロセス/ノード
スレッド配置	12 スレッド/プロセス	12 スレッド/プロセス
使用ノード数	8 ノード～4096 ノード	8 ノード～1024 ノード

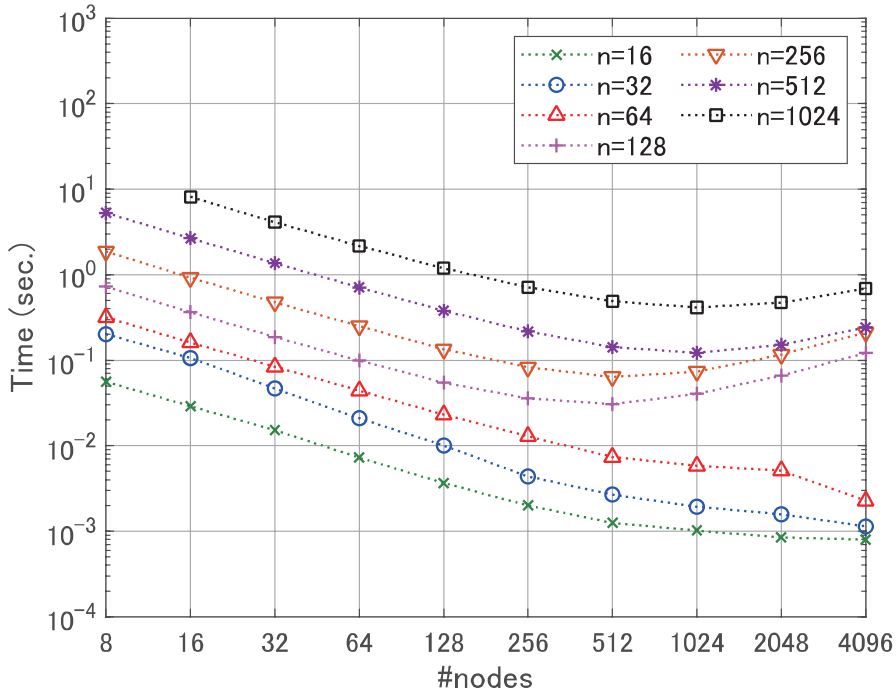
５．４．結果

$m = 16777216\ (= 2^{24})$と行数を固定した上で、$n = 16$から$n = 1024$まで変化させてテスト行列を生成し、プロセス数（ノード数）を変えて、強スケーリングの観点から Ite-CholQR-CP の実行時間を測定した。なお、$r = 0.8n$（四捨五入で整数に丸める）とした。

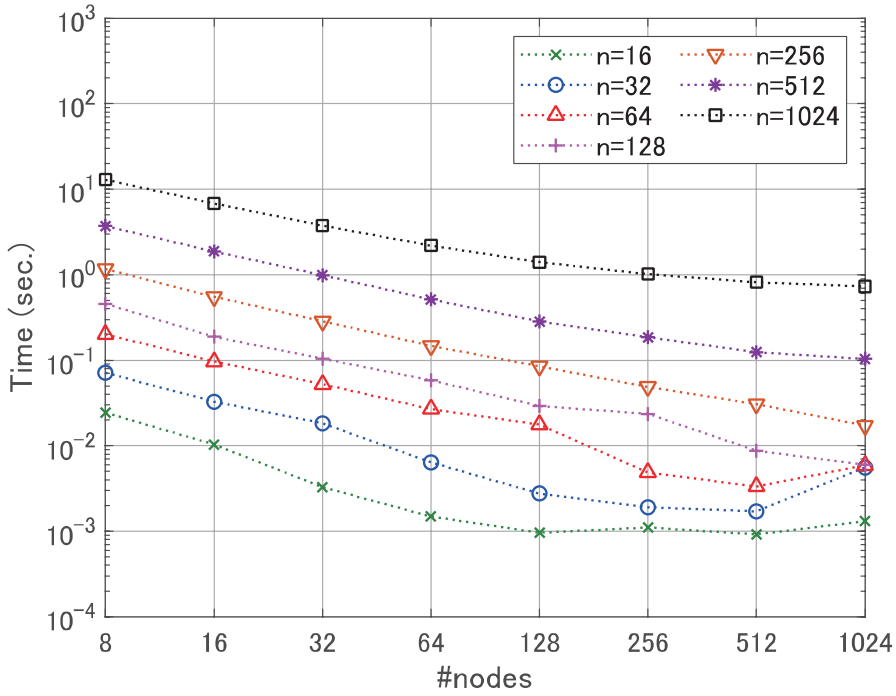
Ite-CholQR-CP の実行時間に関する測定結果を図 1 に示す。BDEC-O（図 1(a)）では、nに関わらず、256 ノードまでは良好に計算時間が短縮していることが分かる。一方、512 ノード以上では、$n \leq 64$については引き続き計算時間が短縮しているが、$n \geq 128$の場合には、計算時間の増加が起きている。nが大きくなるほど、コレスキー分解をはじめとする$n \times n$の行列に関する計算部分の占める割合が増加するため、その影響があると思われる。さらに、プロセス当たりの行数が、プロセス数に反比例して小さくなるため、上記の影響が増加し、ノード数が多い場合に全体の実行時間の増加という形で現れていると推測できる。また、別の理由として、MPI の集団通信の性能（実行バンド幅）がノード数やデータサイズに応じて異なることで、特定のサイズやノード数で挙動が大きく変わっていることも考えられる。現時点では、詳細は不明のため、今後の課題として原因の調査を行う必要がある。一方、OBCX（図 1(b)）では、概ね良好な計算時間の短縮が得られていることが確認できる。nが小さくなるほど、少ないノード数で計算時間の短縮が止まっているのは、並列化による演算時間の削減が、問題サイズが小さい場合ほど早く限界に達しやすい、という強スケーリングで一般的に生じる傾向であり、自然な結果であると考える。

次に、我々が実装した HQR-CP の簡易版との性能比較の結果を報告する。なお、簡易版というのは、教科書等に記載されている HQR-CP のアルゴリズムをそのまま実装すると、実際にはピボット選択を誤る場合が多く、追加の工夫が必要となるからである[18]。なお、追加の工夫に伴い、集団通信のコストが増加するため、以下で示す結果は、HQR-CP の実行時間の下限と Ite-CholQR-CP の実行時間を比較したものであると解釈して構わない。結果は図 2 の通りで、値が 1.0（Baseline）よりも上であれば、Ite-CholQR-CP が HQR-CP よりも高速であることを意味する。

BDEC-O（図 2(a)）では、ノード数が少なくnが大きい場合、および、ノード数が多くnが小さい場合に、Ite-CholQR-CP が高速となっている。一方、OBCX（図 2(b)）では、いずれの条件でも、Ite-CholQR-CP が高速であり、BDEC-O と比べて、高速化率が大きい。以上の結果から、大規模分散並列環境における Ite-CholQR-CP の有効性が十分に確認できたと言える。なお、上述のように、BDEC-O（図 2(a)）と OBCX（図 2(b)）の結果を比べると、傾向が大きく異なっていることが分かる。その理由については、まだ不明であり、今後の調査が必要である。いくつか要因を考えてみると、例えば、HQR-CP は計算の大部分が Level-2 BLAS によるため、両システムのメモリシステムの違いが影響している可能性は高い。また、ネットワークの特性と MPI ライブラリの実装も異なるため、その影響もあり得る。さらに、行列サイズや形状に対する Level-3 BLAS の実行性能の挙動も異なることが考えられ。その影響も十分にあり得る。

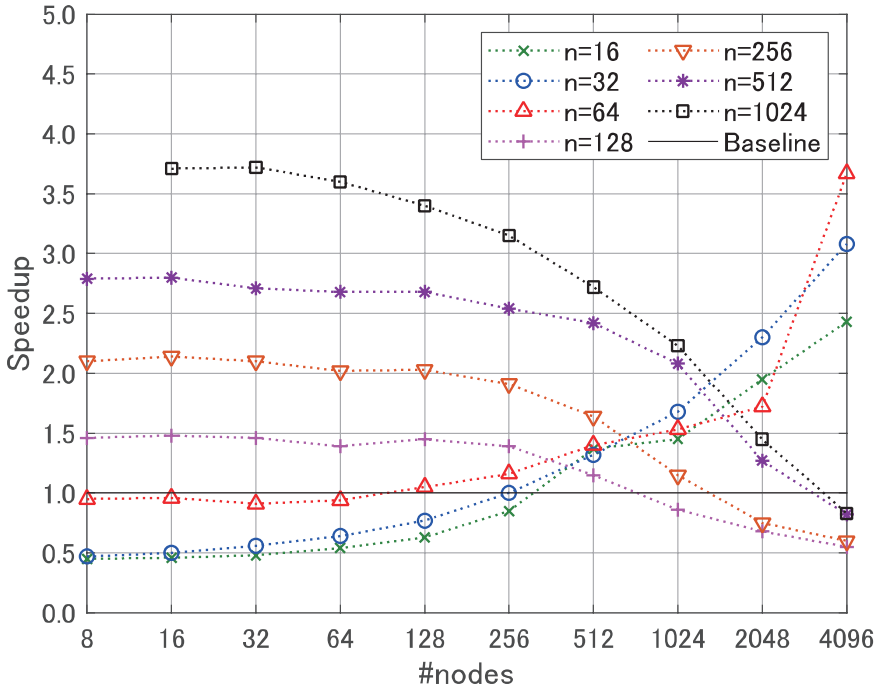


(a) BDEC-O

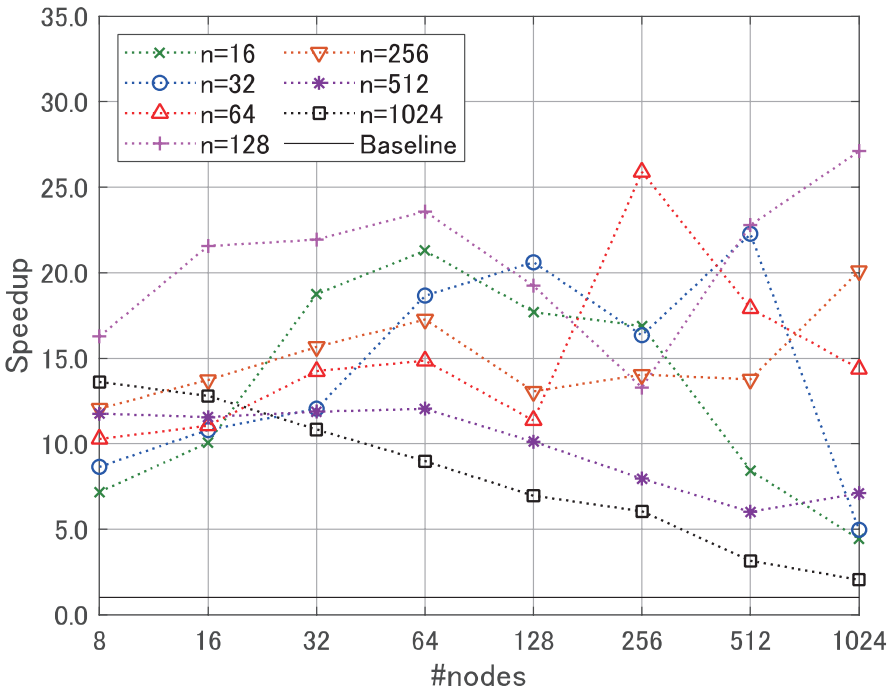


(b) OBCX

図 1：Ite-CholQR-CP の実行時間（強スケーリング、$m = 16777216$）。



(a) BDEC-O



(b) OBCX

図 2：HQR-CP に対する Ite-CholQR-CP の速度向上（強スケーリング、$m = 16777216$）。

６．おわりに

本稿では、我々が開発しているコレスキーQR 型の QRCP アルゴリズム（Ite-CholQR-CP）に関して、BDEC-O および OBCX 上で性能評価を行った結果を報告した。Ite-CholQR-CP は、計算の大半が Level-3 BLAS で実行可能かつ通信回避型アルゴリズムである、といったコレスキーQR 型アルゴリズムが持つ高性能計算への親和性を継承しており、本稿で示した性能評価の結果からも、その効果が確認できた。また、本稿では紹介していないが、マルチコア CPU 環境における有効性も確認できており、計算精度の面でも従来法と同程度となることが数値実験を通して検証できている。以上のことから、Ite-CholQR-CP は QRCP に対する有効な計算手法になり得ることが期待される。

一方、非縦長の行列に対する通信回避型の QRCP アルゴリズム[19]が提案されており、これとの使い分け（アルゴリズム選択）の検討が今後の課題の一つである。また、近年、Randomized アルゴリズムと呼ばれるアルゴリズムが活発に研究されており、QRCP に対する Randomized アルゴリズム[20]との比較は重要な課題である。最後に、縦長行列の QRCP は、通常の QR 分解を行った後に、得られた上三角行列に対して QRCP を行う手法が有効であることが分かっている[21]。したがって、この手法と Ite-CholQR-CP の性能を比較することは、Ite-CholQR-CP の有効性をより明確にするために不可欠である。

謝　辞

大規模 HPC チャレンジの実施に関してお世話になりました、東京大学情報基盤センターの関係者の皆様に深く感謝いたします。本稿は、山本有作 教授（電気通信大学）および中務佑治 准教授（University of Oxford）との共同研究に基づいた内容であり、両氏に感謝いたします。本研究の一部は、JST さきがけ（課題番号：JPMJPR20M8）、JSPS 科研費（課題番号：JP21K11909）、学際大規模情報基盤共同利用・共同研究拠点（JHPCN）、および、革新的ハイパフォーマンス・コンピューティング・インフラ（HPCI）（課題番号：jh230010）の支援を受けています。

参　考　文　献

[1] T. F. Chan, Rank revealing QR factorizations, Linear Algebra Appl., Vol. 88-99 (1987), pp. 67-82.

[2] P. Businger and G. H. Golub, Linear least squares solutions by Householder transformations, Numer. Math., Vol. 7 (1965), pp. 269-276.

[3] G. Golub, Numerical methods for solving linear least squares problems, Numer. Math., Vol. 7 (1965), pp. 206-216.

[4] T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa, Shifted Cholesky QR for Computing the QR Factorization of Ill-Conditioned Matrices, SIAM J. Sci. Comput., Vol. 42 (2020), pp. A477-A503.

[5] S. Chandrasekaran and I. C. F. Ipsen, On Rank-Revealing Factorizations, SIAM J. Matrix Anal. Appl., Vol. 15 (1994), pp. 592-622.

[6] Y. P. Hong and C.-T. Pan, Rank-Revealing QR Factorizations and the Singular Value Decomposition, Math. Comp., Vol. 58 (1992), pp. 213-232.

[7] M. Gu and S. C. Eisenstat, Efficient Algorithms for Computing a Strong Rank-

Revealing QR Factorization, SIAM J. Sci. Comput., Vol. 17 (1996), pp. 848-869.
[8] G. H. Golub and C. F. Van Loan, Matrix Computations, 4th ed., The Johns Hopkins University Press, 2012.
[9] L. N. Trefethen and I. David Bau, Numerical Linear Algebra, SIAM, 1997.
[10] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, Communication-optimal Parallel and Sequential QR and LU Factorizations, SIAM J. Sci. Comput., Vol. 34 (2012), pp. A206-A239.
[11] 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, ScalA'14, pp. 31-38, 2014.
[12] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya, Roundoff Error Analysis of the CholeskyQR2 Algorithm, ETNA, Vol. 44 (2015), pp. 306-326.
[13] 深谷 猛, 縦長行列の QR 分解に対する各種アルゴリズムの比較:Oakforest-PACS 上での性能評価, 東京大学情報基盤センター スーパーコンピューティングニュース, Vol. 22, No. 6 (2020), pp. 28-39.
[14] T. Fukaya, Distributed Parallel Tall-Skinny QR Factorization: Performance Evaluation of Various Algorithms on Various Systems, PDCAT 2022 (LNCS Vol. 13798), pp. 275-287, 2022.
[15] N. J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd ed., SIAM, 2002
[16] 東京大学情報基盤センター, Wisteria/BDEC-01 スーパーコンピュータシステムの紹介, https://www.cc.u-tokyo.ac.jp/supercomputer/wisteria/system.php
[17] 東京大学情報基盤センター, Oakbridge-CX スーパーコンピュータシステムの紹介, https://www.cc.u-tokyo.ac.jp/supercomputer/obcx/system.php
[18] Z. Drmač and Z. Bujanović, On the Failure of Rank-Revealing QR Factorization Software - A Case Study, ACM TMS, Vol. 35 (2008), pp. 12:1-12:28.
[19] J. Demmel, L. Grigori, M. Gu, and H. Xiang, Communication Avoiding Rank Revealing QR Factorization with Column Pivoting, SIAM J. Matrix Anal. Appl., Vol. 36, (2015), pp. 55-89.
[20] J. A. Duersch and M. Gu, Randomized Projection for Rank-Revealing Matrix Factorizations and Low-Rank Approximations, SIAM Rev., Vol. 62 (2020), pp. 661-682.
[21] R. D. da Cunha, D. Becker, and J. C. Patterson, New Parallel (Rank-Revealing) QR Factorization Algorithms, Euro-Par 2002 (LNCS Vol. 2400), pp. 677-686, 2002.