HPC-GENIE: LLMを利用したHPCコード自動生成プロジェクト —概要とケーススタディ—

HPC-GENIE: An Automatic HPC Code Generation Project Using LLMs -- Overview and Case Studies --

片桐 孝洋(Takahiro Katagiri)[1], 林 俊一郎(Shun-ichiro Hayashi) [2], 椋木 大地(Daichi Mukunoki)[1], 星野 哲也(Tetsuya Hoshino)[1], 大島 聡史(Satoshi Ohshima)[3]
[1]名古屋大学情報基盤センター(Information Technology Center, Nagoya University),
[2]名古屋大学大学院情報学研究科(Graduate School of Informatics, Nagoya University)
[3] 九州大学情報基盤研究開発センター(Research Institute for Information Technology, Kyushu University)
e-mail:katagiri.takahiro.m4@f.mail.nagoya-u.ac.jp

1 概要

近年, 生成AI(Generative AI)は自然言語処理, 画像生成, 音声合成, コード生成といった多様な領域において著しい進展を遂げている. 特に大規模言語モデル(Large Language Models, LLM) は膨大な事前学習データを用いて高度な言語生成能力を獲得し, 多様な知識や文脈理解をした出力を実現した[1].この発展には,Transformerの登場とスケーリング則[2], および, 教師あり学習[3]などの技術に支えられている. 生成AIは, GPT-4やClaude, Geminiなどのモデルを中核に, 質問応答, 要約, 翻訳, コード補完, さらには複雑な意思決定支援などの応用が現在, 試みられている.

高性能計算(HPC)分野においてもコード生成AIは, 高度な分野特有の知識が要求される, 並列化や個別アーキテクチャ向けチューニング支援に有望と考えられる. 従来のルールベースやDSL(Domain-Specific Language)では実現できない, 柔軟なコード最適化の支援が期待できる. 特にHPC分野で従来から行われていた, ソフトウェア自動チューニング(Software Auto-tuning, AT)[4]技術を, 生成AIを用いたプロンプトエンジニアリングと融合させることで, 全く新しいHPC基盤の創成が期待されている.

2 HPC-GENIEプロジェクト

HPC-GENIE(High-Performance Computing with GEnerative Neural Intelligence for Execution)プロジェクト(https://www.hpc.itc.nagoya-u.ac.jp/menu/hpc_genie.html)は, 名古屋大学 情報基盤センターおよび大学院情報学研究科の関連者で発足したコード生成 AIを活用したHPCプログラム自動生成プロジェクトである. LLMを活用したプロンプトエンジニアリングとAT技術を融合したAI基盤開発により自動化を行い, HPCソフトウェア開発の生産性を劇的に高めることを目的としている.

従来のAT技術では, 主な技術基盤としてテンプレート生成と探索手法(遺伝的アルゴリズム/焼きなまし法/ベイズ最適化, など)を対象にしていたのに対し, HPC-GENIEではLLMを用いる. また, 探索手法で従来のAT技術では, ヒューリスティクスと性能計測ベースで最適コード探索を行っていたが,HPC-GENIEでは,反復プロンプトによる自己改善により最適コードの探索を目指す, 全く異なるアプローチをとる.

一方, LLMの活用による欠点もある. 特に, 最適化コードの正確性(精度)と再現性にLLMベ

ースの方法は問題がある．従来の AT 技術は計測ベースであり，高い精度でコード生成が可能である．一方で LLM ベースのコード生成は言語モデルの特性から確率的生成になるため，生成されるコード自体にばらつきが生じる．そのため，LLM ベースのコード最適化は再現性/安定性が高くなく，与えるプロンプトに大きく影響する．

3 設計思想

HPC-GENIE では，以下の設計思想で LLM によるコード生成基盤の開発を行う．

- HPC コード生成を実現するためのプロンプトエンジニアリング方式開発
- 複数の LLM が利用できる CLI (Command Line Interface)の開発，及び AT 機能連携
- HPC コード生成を実現するための RAG (Retrieval Augmented Generation) とファインチューニング機能の開発
- HPC コード生成を実現するためのローカル LLM の活用
- コード生成 AI による，精度保証，混合精度演算，説明可能 AI(XAI)，及び AT コードの自動生成

表 1 は従来の LLM によるコード生成 AI 手法と HPE-GENIE の狙いのまとめである。

表 1　従来の LLM によるコード生成 AI 手法と HPC-GENIE での狙い

項目	従来手法	HPC-GENIE の狙い
コード生成方法	汎用 LLM に手動プロンプト入力/回答利用	プロンプトエンジニアリング＋CLI ベースの複数 LLM 選択方式
AT 連携	CLI ツール (OpenTuner, GPTune 等)で手動接続	AT 統合型(コード生成直後に AT 走査)
RAG 機能	実験的に一部 LLM で統合 (LangChain など)	RAG 機構を明示的に内包する
精度保証／混合精度演算制御	演算精度は人手調整，混合精度演算は対応していない	精度保証/混合精度演算を構成要素に明示的統合を計画
説明可能 AI (XAI) 連携	ほぼ非対応 (black-box 出力)	XAI 要素(実行時間/演算精度/電力などの予測の妥当性検証)の統合を計画
ローカル LLM 対応	基本はクラウド LLM(グローバル LLM)利用(ChatGPT, Claude 等)	Swallow LLM 等のローカル LLM を含む (グローバル LLM も選択可)

謝辞　本研究は，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) の支援による(課題番号: jh250015)．本研究は JSPS 科研費 JP23K11126, JP24K02945 の助成を受けたものです．

参考文献

[1] OpenAI et al., GPT-4 Technical Report, arXiv:2303.08774, 2023.
[2] A. Vaswani et al., Attention Is All You Need, Proc. of NIPS2017, 2017.
[3] J. Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Proc. of NAACL-HLT2019, pp.4171-4186, 2019.
[4] T. Katagiri and D. Takahashi, Japanese Auto-tuning Research: Auto-tuning Languages and FFT, Proc. of the IEEE, Vol. 106, Issue 11, pp.2056-2067, 2018.

動的負荷分散プラットフォーム (HALEOS) の評価

Numerical Evaluation Dynamic Load Balancing Platform(HALEOS)

河合 直聡 (Masatoshi Kawai)[1]
[1] 東北大学 サイバーサイエンスセンター (Tohoku University, Cyberscience Center)
e-mail : m.kawai@tohoku.ac.jp

1　概要

HALEOS(Hardware-Adaptive Load-Efficient Optimization System) は MPI/OpenMP で並列化されたアプリケーションを対象に、プロセス毎に異なる負荷に合わせて適したリソースを割り当て、計算時間と消費電力の両方を同時に削減するプラットフォームである。本発表ではこのプラットフォームの格子階層行列法アプリケーションでの性能改善について紹介する。

2　背景

アプリケーションの大規模並列化 (MPI/OpenMP) では、均一な負荷分散が 1 つの課題となる。これは、初期化が完了しないと各プロセスの負荷が分からないケースや、実行中に変化するケースがあるためである。富岳に代表されるような大規模ノード環境では、プロセス数を増やしてこの負荷の不均衡をある程度吸収するアプローチが取れたが、近年のスーパーコンピュータではメニーコア化や GPU の採用による単一ノードの性能が高くなる Fat-Node 化とともに、ノード数の減少が進んでおり、同様のアプローチが取れなくなっている。よって、Fat-Node を有効に使う動的負荷分散手法が必要となっている。

現在、著者を含む研究グループではノード内のコアレベルでの負荷の均一化によるアプリケーションの高速化、省電力化を実現する動的負荷分散プラットフォーム HALEOS(Hardware-Adaptive Load-Efficient Optimization System) を提案、実装している。HALEOS は Dynamic Core Binding(DCB)[1] と UT-Helper[2] という 2 つのコア技術からなり、DCB ではノード毎のコアレベルでの負荷の均一化を行い、UT-Helper では効率的なバックグラウンド処理環境を提供することで、システム全体の高効率化を実現する。本発表では、HALEOS プラットフォームの性能を格子階層行列法アプリケーション [3] を使った性能評価結果について紹介する。

3　HALEOS

HALEOS のコア技術の一つである DCB では、図に示すように、プロセスの負荷の不均衡に合わせて異なる数のコアを割り当て、コアレベルで負荷を均一化する。DCB では、計算時間短縮を

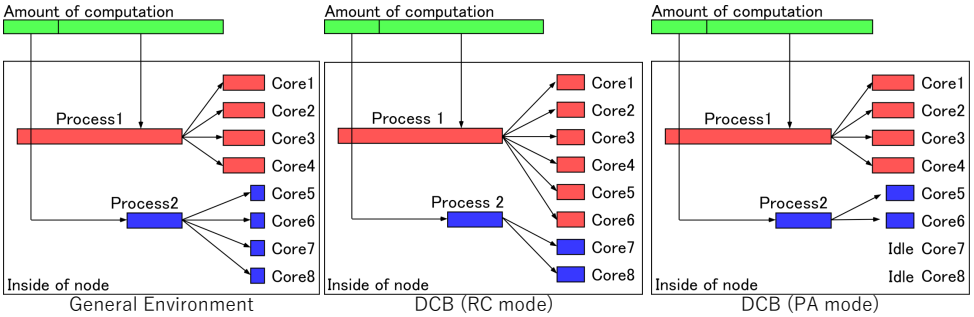

図 1. DCB の概念図

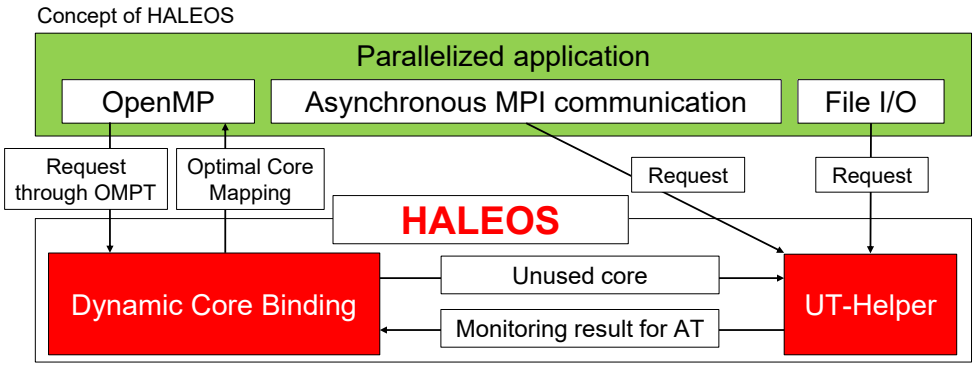

図 2. HALEOS のコンセプト

目的に全コアを割りあてる RC-mode と、消費電力削減を目的に負荷の負荷の小さなプロセスに割り当てるコアを減らす PA-mode を用意している。マルチノード環境では負荷の大きなノードでは RC-mode を、小さなノードでは PA-mode を適用することで、計算時間と消費電力の両方同時削減を実現できる。実際の評価では計算時間を半分に、計算にかかる総消費電力を 1/4 にできる効果を確認している [4]。

さらに UT-Helper では、DCB の動作によって発生した余剰コアにヘルパースレッドを生成し、このスレッドに通信やファイル IO などを担当させることで、主計算部分の性能を劣化させないバックグラウンド処理を実現する。また、ヘルパースレッドで CPU の動作状況を監視し、負荷の不均衡を検出して自動的にコア割り付けを変更する機能を実装予定である。これにより、DCB の動作を完全ブラックボックス化可能となり、OpenMP Tool の利用でユーザーはプログラムコンパイル時に HALEOS プラットフォームライブラリをリンクするだけでその恩恵を受けることができるようになる。

4　まとめ

本研究では、アプリケーションの負荷とシステムの性能を考慮した動的なコア割り付けと、余剰コア上に生成したヘルパースレッドのバックグラウンド処理による、システム利用効率を最大化する HALEOS プラットフォームを紹介した。発表では UT-Helper 部のベンチマーク結果について述べる予定である。

参考文献

[1] Masatoshi Kawai, Akihiro Ida, Toshihiro Hanawa, and Kengo Nakajima. Dynamic core binding for load balancing of applications parallelized with mpi/openmp. In *Computational Science – ICCS 2023*, pp. 378–394, Cham, 2023. Springer Nature Switzerland.

[2] 純工藤, 敏博塙. 余剰コアの活用に向けた実行中プロファイリング手法の検討. Technical report, ハイパフォーマンスコンピューティング研究会, Dec 2020.

[3] Akihiro Ida. Lattice $\mathcal{H}$-matrices on distributed-memory systems. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 389–398, 2018.

[4] Masatoshi Kawai, Akihiro Ida, Toshihiro Hanawa, and Tetsuya Hoshino. Optimize efficiency of utilizing systems by dynamic core binding. HPCAsia '24 Workshops, p. 77–82, New York, NY, USA, 2024. Association for Computing Machinery.

テンソルトレイン分解を導入したKrylov部分空間法の検討と収束特性の解析

A study of Krylov subspace methods with Tensor Train Decomposition and its convergence characteristics

鳥井公平 (Kohei Torii)[1], 小野謙二 (Kenji Ono)[2]
[1] 九州大学 大学院システム情報科学研究院 情報理工学専攻
(Graduate School of Information Science and Electrical Engineering, Kyushu University)
e-mail : torii.kohei.502@s.kyushu-u.ac.jp
[2] 九州大学 情報基盤研究開発センター
(Research Institute for Information Technology, Kyushu University)

1 序論

数値シミュレーションでは，問題の高次元化や空間分割数の増加により，巨大なデータを扱う機会が多くある．特にポアソン方程式のような楕円型偏微分方程式では，大規模疎行列を係数とする連立一次方程式を解く必要がある．係数行列が対称正定値行列の場合，Krylov 部分空間法の一つである CG 法を用いることが多いが，大規模行列-ベクトル積や内積の計算量が次元数に対して指数的に増加する課題がある．そこで，反復計算に低ランク近似を導入した Inexact Krylov 法 [1] が研究されている．また，低ランク近似としてテンソルトレイン（TT）分解は，高い圧縮性能と演算効率から偏微分方程式の数値解法への応用 [2] が進められている．本研究では，CG 法に TT 分解を導入した TT-CG 法を実装し，大規模疎行列の連立一次方程式に対し数値実験を実施した．

2 TT-CG 法

Algorithm 1 は提案する TT-CG 法である．4 つのベクトル x, r, Ap, p に対する TT-SVD の近似精度の閾値を $\delta_x, \delta_r, \delta_{Ap}, \delta_p$ とする．innerprod は TT 分解されたベクトル同士の内積を示す．

Algorithm 1 TT-CG Algorithm

Compute TT-SVD(x_0, δ_x)
Compute TT-SVD($r_0 = b - Ax_0, \delta_r$), Set $p_0 = r_0$
for $k = 0, 1, \ldots$ **do**
  TT-SVD(Ap_k, δ_{Ap})
  $\alpha_k = \text{innerprod}(r_k, r_k)/\text{innerprod}(p_k, Ap_k)$
  TT-SVD($x_{k+1} = x_k + \alpha_k p_k$, δ_x)
  TT-SVD($r_{k+1} = r_k - \alpha_k Ap_k$, δ_r)
  if $|r_{k+1}|/|b| \leq \epsilon$ then **then**
    stop
  end if
  $\beta_k = \text{innerprod}(r_{k+1}, r_{k+1})/\text{innerprod}(r_k, r_k)$
  TT-SVD($p_{k+1} = r_{k+1} + \beta_k p_k$, δ_p)
end for

3　数値実験

2 章で示したように，TT-CG 法は 4 つの近似精度の閾値を設定することで各ベクトルの近似精度を調節する．数値実験では，5 次元ポアソン方程式から生じる連立一次方程式 $\nabla^2 x = b$（格子数 10，$b = 1$，初期解 $x_0 = 0$）を TT-CG 法で解いた．図 1 左は，各閾値の変化と CG 法との数値解の誤差を示している．逐次解の閾値 δ_x と残差ベクトルの閾値 δ_r が数値解に対して与える影響が大きいことが分かった．右図 1 は，行列積 Ap の閾値 δ_{Ap} を変化させたときの残差の収束特性である．δ_{Ap} を大きくすると，収束後半 (残差 10^{-5} 以下) で残差の収束が遅くなることが分かった．

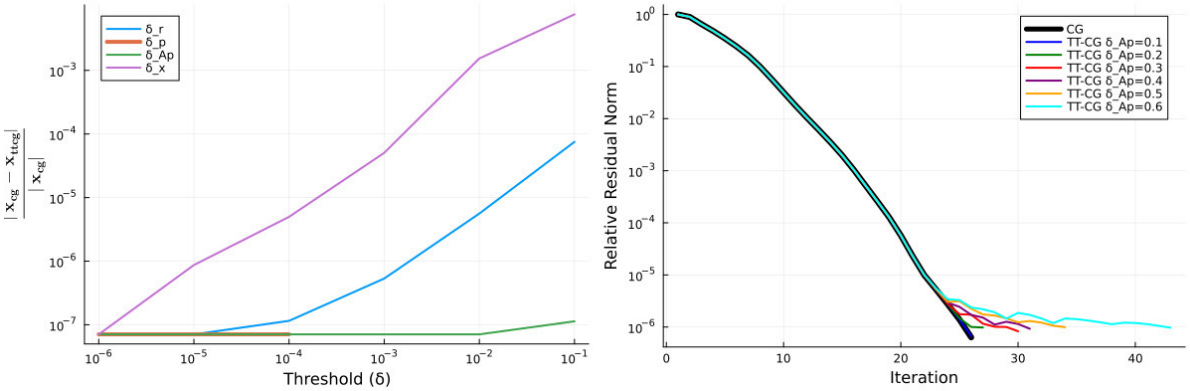

図 1. 近似精度と数値解の誤差（左） δ_{Ap} と残差の収束特性（右.）

表 1 は，δ_r を変化させたときの残差の収束特性と数値解の誤差をまとめたものである．δ_r を大きくすると，反復回数は減少するが，CG 法との誤差は大きくなることが分かった．

表 1. δ_r と残差の収束特性

δ_r	反復回数	誤差
10^{-4}	25	5.05×10^{-5}
10^{-3}	24	5.05×10^{-5}
10^{-2}	22	5.14×10^{-5}
10^{-1}	23	8.25×10^{-5}
0.2	20	1.22×10^{-4}
0.1	21	1.72×10^{-4}

これらの数値実験により，行列積 Ap は低精度化しても数値解に影響を与えにくいが，過度な低精度化は収束性が悪くなることが分かった．また，残差ベクトル r は低精度化すると数値解に影響を与えやすいことが分かった．これは，近似精度と計算コストのバランスをとる上で重要な指標になる．今後の数値実験では，TT-rank の推移や計算コストについて調べ，他の問題に TT-CG 法を使用することで有効性を検証する．

参考文献

[1] Simoncini, Valeria and Szyld, Daniel B., Theory of Inexact Krylov Subspace Methods and Applications to Scientific Computing, SIAM J. on Sci. Comput., 25-2 (2003), 454-477.

[2] Sergey V. Dolgov, et al., Alternating Minimal Energy Methods for Linear Systems in Higher Dimensions, SIAM J. on Sci. Comput., 36-5 (2014), A2248–A2271.

縦長行列の列ピボット付き QR 分解に対するコレスキー QR 型アルゴリズムの実装方法改良の検討

Investigation into improving the implementation of the Cholesky QR-type algorithm for tall-and-skinny QR factorization with column pivoting

深谷 猛 (Takeshi Fukaya)[1]
[1] 北海道大学 (Hokkaido University)
e-mail : fukaya@iic.hokudai.ac.jp

1 はじめに

行列 $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ の数値ランクが r $(\leq n)$，つまり，A の大きい方から i 番目の特異値を σ_i として，σ_r と σ_{r+1} の間に大きなギャップがあり，σ_i/σ_1 $(i = r+1, \ldots, n)$ が十分に小さい場合に，

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

の形に A を分解することを，A に対する列ピボット付き QR 分解（QRCP: QR factorization with Column Pivoting）と呼ぶ．ここで，$P \in \mathbb{R}^{n\times n}$ は置換行列，$Q \in \mathbb{R}^{m\times n}$ は列直交行列，$R \in \mathbb{R}^{n\times n}$ は上三角行列である．また，$Q_1 \in \mathbb{R}^{m\times k}, R_{11} \in \mathbb{R}^{k\times k}$ であり，P と k は，R_{11} の条件数および $\|R_{22}\|_2$ ができるだけ小さくなるように選ぶ．なお，理想的には，$k = r$ で，R_{11} の条件数は σ_1/σ_r 程度である．QRCP は Rank Revealing QR factorization (RRQR) の一種 [1] であり，行列の低ランク近似等の応用を持つ．本研究では，特に A が縦長（$m \gg n$）の場合を想定し，具体的な用途としては，例えば，線形従属の可能性があるベクトル列に対して正規直交基底を計算する場面が挙げられる．

QRCP の数値計算方法としては，ハウスホルダー変換に基づくアルゴリズムが知られており，LAPACK では `DGEQPF` や `DGEQP3`（行列積版の実装）というルーチンが提供されている．一方，縦長行列の（列ピボットがない）通常の QR 分解に対して，コレスキー QR 型アルゴリズムの有効性が示されており，それを踏まえて，著者らは QRCP に対してコレスキー QR 型アルゴリズムを開発した [2]．本発表では，このコレスキー QR 型アルゴリズムに関する実装方法の改良を検討する．なお，本発表では扱わないが，QRCP に対する乱択アルゴリズムの研究も活発に行われている [3].

2 QRCP に対するコレスキー QR 型アルゴリズムの概要

以下ではアルゴリズムの要点のみを述べ，詳細は文献 [2] に委ねる．丸め誤差がない場合，1) $A^\top A = W$, 2) $P^\top W P = R^\top R$, 3) $AR^{-1} = Q$ の手順で QRCP が計算できる．なお，2 は Diagonal pivoting 付きコレスキー分解である．一方，丸め誤差がある場合，2 においてピボット列の誤選択や計算の破綻が生じる．これに対して，2 の計算途中で現れる値を監視して，ある条件を満たした場合，丸め誤差の影響が無視できなくなると判断して計算を止めるアイディアを考案した．その上で，1 から 3 の手順を繰り返すことで反復的にピボット列を選択するアルゴリズムを開発した．

このアルゴリズムでは，A が縦長の場合，1 と 3 の計算が大部分を占めるが，この部分は行列積相当の演算（Level-3 BLAS）で実行可能である．また，分散並列化を行った場合（行方法のブロック分散を仮定），集団通信（例：`Allreduce`）の回数が行列サイズに依存せずに定数回（所謂，通信回避型）である．そのため，近年の計算機環境において高い実行性能を得ることが期待でき，実際に，ハウスホルダー変換に基づくアルゴリズムよりも高速となることが確認されている [2].

3　実装方法改良の指針

今回は以下の二点について実装方法の改良を検討する．

3.1　Diagonal pivoting 付きコレスキー分解

開発したアルゴリズムでは，Diagonal pivoting 付きのコレスキー分解を途中で停止するが，その時点で，

$$P^\top W P = \begin{pmatrix} R_{11}^\top & O \\ R_{12}^\top & I \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ O & I \end{pmatrix} + \begin{pmatrix} O & O \\ O & W'_{22} \end{pmatrix} \tag{2}$$

の形の分解を得る．なお，前節で述べた値の監視は，W'_{22} の対角要素に対して行う．（条件が満たされない場合は，もう一段階計算を進める．）その後，R_{11}, R_{12} を用いて A の更新を行うが，その際，W'_{22} の計算結果は不要となる．この点を踏まえて，Diagonal pivoting 付きコレスキー分解の計算手順を見直し，不要な要素に対する計算量を削減することを試みる．

3.2　グラム行列の計算

開発したアルゴリズムは反復型の構造を持ち，その過程において A の列ベクトルの状況は

$$A \to \begin{pmatrix} A'_* & A_* \end{pmatrix} \to \begin{pmatrix} Q_* & A'_* & A_* \end{pmatrix} \to \begin{pmatrix} Q_* & A'_* \end{pmatrix} \to Q \tag{3}$$

のようになる．ここで，A_* は入力時のまま，A'_* はピボット選択と粗い直交化が行われた状況，Q_* は再直交化により十分な直交性を持つ状況，を表している（この例は 4 反復）．文献 [2] における実装では，簡単のため，各反復で $A^\top A$ のように，行列全体を用いてグラム行列 W を計算していた．しかし，上記の式から分かるように，Q_* である列ベクトルに関して，$Q_*^\top Q_* = I$ が成立する．そこで，この性質を用いて，アルゴリズム中のグラム行列の計算部分の計算量の削減を試みる．

4　おわりに

理論上は，前節で紹介した実装方法の修正を行うことで計算量を削減することが可能であるが，実行効率も変化するため，計算時間における効果は不明瞭である．当日の発表では，実機上での実験結果を紹介し，実装方法変更の効果を議論する．

謝辞　本研究は JSPS 科研費（25K03124）および JHPCN・HPCI（jh250032）の支援を受けています．

参考文献

[1] M. Gu et al., Efficient Algorithms for Computing a Strong Rank-Revealing QR Factorization, SIAM SISC, 17 (1996), 848-–869.

[2] T. Fukaya et al., A Cholesky QR type algorithm for computing tall-skinny QR factorization with column pivoting, IPDPS 2024, 63–75, 2024.

[3] M. Melnichenko et al., CholeskyQR with Randomization and Pivoting for Tall Matrices (CQR-RPT), SIAM SIMAX, 46 (2025), 1701–1734.