Vol.2022-QS-5 No.6
2022/3/24

Quantum read-only memoryの効率的な記述と，その最適化による量子特異値変換の優位性の見積もり

冬鏡 澪[1,2,a)]　鈴木 泰成[2,3]　徳永 裕己[2]

概要：量子特異値変換 (Quantum singular value transformation, QSVT) は，様々な計算量的優位性がある量子アルゴリズムを統一的に記述できる量子アルゴリズムであり，誤り耐性量子コンピューターで初期に実現できる有用なアルゴリズムとして期待されている．QSVT では Quantum read-only memory (QROM) というモジュールを用いた操作が実行する際の時間的なボトルネックになっていることがわかっている．しかし，QROM の理論的な記述は明らかになっているものの，効率的に QROM の量子回路を効率的に記述し回路を設計する枠組みが存在しなかったため，QROM の設計の最適化や QROM の構築に必要なリソース見積もりといった研究が困難であるという課題があった．本発表では QROM を効率的に設計するために開発したライブラリと，それを用いた最適化やリソース見積もりの結果について報告する．開発した記述方式は既存の量子プログラミング言語やフレームワークに比べ簡潔に QROM を記述できる．また，設計した QROM に ZX-calculus による回路最適化を適用することで，一定の状況下で既存の回路実装に比べて 35%程度の効率改善が可能であることを示した．また，最適化の結果をもとに，量子特異値変換を用いた逆行列の計算について，量子計算で優位性を確認するために必要なリソースの見積もりを行った．

Efficient description and optimization of quantum read-only memory, and resource estimation of computational supremacy with quantum singular value transformation

Abstract: Quantum singular value transformation (QSVT) is a quantum algorithm that can unify several quantum algorithms and is expected as a useful algorithm in the early regime of fault-tolerant quantum computing (FTQC). While the implementation of quantum read-only memory (QROM) is a bottleneck of QSVT, it has been difficult to optimize and estimate the resource of QROM due to its complexity. In this paper, we present a novel software framework that enables a flexible description of complicated modules of FTQC such as a QROM. Then, we describe a QROM module with our framework and perform circuit optimization by ZX-calculus. As a result, we observed a 35% reduction of required T-depth. Based on the result, we also estimate the time and space required for demonstrating the computational supremacy with QSVT.

1. Introduction

近年，従来より高速に情報処理が可能な枠組みとして量子計算が注目を集めている．現実の量子計算機は大きなエラーを持つため，実用的な規模で重要な計算を行うには量子誤り訂正符号で符号化された計算機の上で量子計算を行う誤り耐性量子計算 (Fault Tolerant Quantum Computation, FTQC) が必須となる．例えば，現在の公開鍵暗号方式の一つである RSA 暗号を解読するために必要な規模の素因数分解を行うには FTQC が必要であると考えられている [1], [2]．FTQC では非クリフォードゲートと呼ばれる一部のゲート演算を量子ビットに適用する際には誤り訂正符号の構造に起因して大きなオーバーヘッドが生じるため，1 演算あたりに必要な時間が古典計算機の 1 命令にかかる時間よりも長くなる．このため量子計算機が古典計算機に対して実際に高速に問題を解けるようになるには，上記のオーバーヘッドを解消できる程度に量子計算に優位性のある規模の問題を解かねばならない．従って，初期に量子計算機が古典計算機よりも速く問題を解けること

[1] 東京大学 工学系研究科 物理工学専攻
[2] NTT コンピュータ&データサイエンス研究所
[3] JST さきがけ
[a)] tokami @ qi.t.u-tokyo.ac.jp

を実証するには，小さい規模の問題でも計算量理論的に大きな高速化が得られるようなアルゴリズムを用いることが望ましい．

Quantum Singular Value Transformation（QSVT）[3] は様々な量子アルゴリズムを統一的に記述し多くのタスクを効率的に実行することのできる量子アルゴリズムである．QSVT は，例えば Harrow-Hassidim-Lloyd (HHL) のアルゴリズム [4], [5] で実行可能な線形システムの解を求める演算を，HHL アルゴリズムよりも高速に実行することができる．このため，QSVT は広範な量子アルゴリズムの記述が可能でありながら，小規模な FTQC でも通常の計算機に対する計算量的な優位性が得られると期待されている．QSVT では，Quantum Read-Only Memory（QROM）というモジュールを用いた操作が時間的なボトルネックになっている [6]．このため，FTQC をより早期に実用化するには，QROM を最適化しより効率的に実行する必要がある．与えられた量子回路を最適化する方法としては ZX-calculus[7] などが知られている．しかし，QROM は複雑な構造を持ったモジュールであるため，既存の回路記述手法では簡潔に記述する枠組みが存在しない．このため，QROM を簡潔に記述し，最適化を行い，リソースを見積もることは従来困難であった．

本研究ではまず CNOT や Toffoli ゲートなどを複雑に接続する必要があるモジュールを効率的に記述するための枠組みである Q-Divide を Rust 言語を用いて開発した．これによって QROM を記述し，ZX-calculus をもとに回路最適化を行うフレームワークである PyZX[8] を用いて最適化を行った．さらに，最適化された QROM を用いて QSVT による逆行列計算に必要なリソースを推定した．本研究で提案する量子回路記述ライブラリ Q-Divide は，一般的な方式による記述に比べると簡潔に回路を記述することができる．作成した QROM の記述を用いた ZX-calculus による最適化の結果として 35%程度の効率化が実現した．最適化の結果をもとに，線形システムの解を求める演算について量子計算機が古典計算機よりも問題を速く解くために必要な量子ビット数を見積もり，例えば条件数が $\kappa = 10^3$ の行列については 10^{13} 次元程度のサイズから優位性が現れることを示した．

2. Preliminary

2.1 表面符号による FTQC の実装

大規模な量子計算で信頼性のある計算結果を得るには量子誤り訂正符号を用いて計算的に逐次的に誤りを検出し訂正する必要がある．その中でも，表面符号 [9], [10], [11], [12] は実装が容易で符号として高い性能を持つことから標準的な手法として用いられている．本研究では表面符号を用いた FTQC を前提として計算を行う．**図 1** に示される通り，表面符号は物理量子ビットが二次元平面上に格子状に並んで論理量子ビットを構成しており，それがさらに二次元平面上に並んでいる符号である．表面符号は他の誤り訂正符号に比べると，訂正のために必要な操作がどれも 2 次元空間上で局所的であるために，実装が容易であると考えられている [13]．また，表面符号では近似的なアルゴリズムを用いることで一般的な量子ビットの寿命よりも十分に高速にエラーの推定を行うこともできる [14], [15].

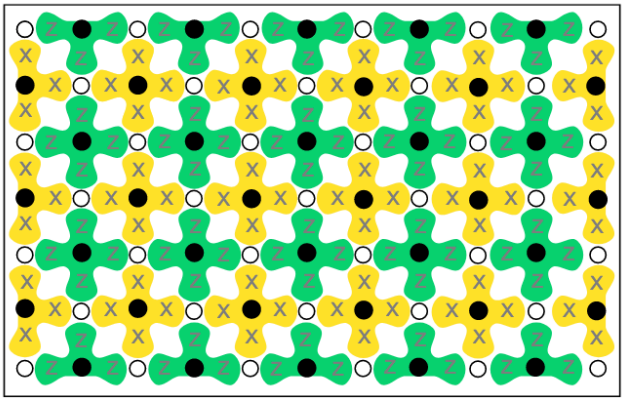

図 1: 表面符号における論理量子ビットの例．黒丸と白丸が物理量子ビットであり，この格子全体で 1 論理量子ビットを構成している．図は [9] より引用.

表面符号で符号化された論理量子ビットに可能な操作は万能量子ゲートセットを構成するため，表面符号で符号化した論理量子ビットでは万能量子計算が可能である [10], [16]. 表面符号において実際にゲート演算を実行するときに，注意しなければならないゲートが T ゲートである．他のゲート（CNOT ゲートや X,Z,H ゲート）に比べて，表面符号では T ゲート演算を実行するのに時間がかかるとされているからである．[9] 表面符号における T ゲートは，魔法状態と呼ばれる状態 $|A\rangle := T|+\rangle = \frac{1}{\sqrt{2}}(e^{i\pi/4}|0\rangle + e^{-i\pi/4}|1\rangle)$ を消費したゲートテレポーテーションにより実現される．ゲートテレポーテーションを行うには，テレポートを行うためにエラー訂正された論理量子ビットの測定値が必要となるため，エラー推定に起因するリアクションタイムと呼ばれるレイテンシが生じ，通常のゲート操作よりも時間を要する．精度良く T ゲートをかけるには高い忠実度の魔法状態を用意する必要があり，これは魔法状態注入と魔法状態蒸留の二つのプロセスで実現できる．この二つのプロセスは多くの論理量子ビットを必要とし，しかも確率的にしか成功しないために成功するまで繰り返さねばならない．従って，魔法状態を生成する操作自体にも多くの時間と量子ビットが必要となる．こうした背景から，FTQC における万能量子計算では，計算中にどのぐらい T ゲートが必要になるかと，魔法状態を生成するために魔法状態の注入と蒸留を行うためだけの回路（以後 Distiller と呼称する）をどのように配置するかがリソース見積もりにおいて支配的な要素となっている．

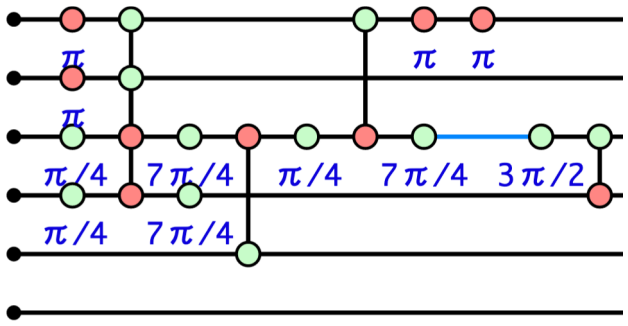

図 2: T-depth が T-count よりも小さくなる例．$\pi/4$ や $7\pi/4$ と下に表示されている部分が T ゲートもしくはその逆操作になっている．この図の回路では T-count は 6 であるが，上から 3 番目と 4 番目の量子ビットに並列に T ゲートをかけることができる部分があるため，T-depth は 4 になる．

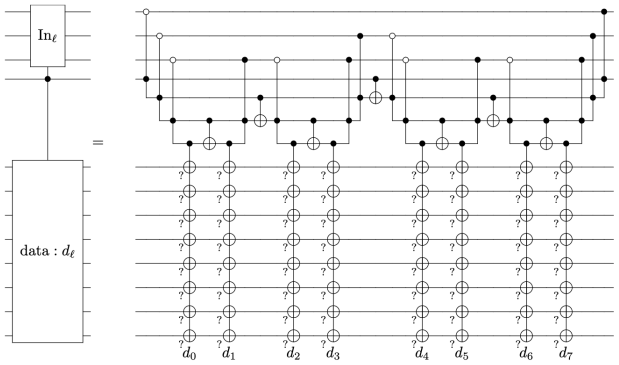

図 3: QROM の基本的な構成．上部が入力に応じて CNOT ゲートの制御量子ビットを反転させ，下部の回路にハードコーディングされたデータを読み出す．図は [6] より引用．

表面符号による FTQC での演算は T ゲートに必要な時間が支配的となるため，回路中に T ゲートがどのように分布しているかの情報は時間を見積もる上で重要である．回路中にある T ゲートの量を見積もる指標として T-count と T-depth の 2 つがある．T-count とは，回路中にいくつの T ゲートが存在するかを数えたものである．一方，T-depth とは，T ゲートを複数の量子ビットに並列に書けることができるときに必要な T ゲートをかける回数である．**図 2** に T-count と T-depth の数え方の例を挙げる．T-depth は全ての T-gate が直列に並んでいるときに T-count と一致するが，一般には T-count 以下の値となる．表面符号において計算を実行する際に Distiller の数が少ない場合は，実行したい T ゲートのスループットに対して，単位時間に供給される魔法状態が常に不足する．この時，FTQC の実行時間は T-count を魔法状態の生成速度で割ったものが良い指標となる．一方，Distiller の数が T ゲートを並列にかけるために充分な数ある場合，T-depth が回路を実行する時間の良い指標となる．

2.2 QSVT とそれを用いた逆行列の計算

QSVT [3] は，2 種類の操作（行列をブロックとして含むユニタリ操作と，行列の基底が張る空間についての回転）を交互に繰り返すことによって，事前に設定した多項式に従って行列の特異値を変換するアルゴリズムである．多項式は奇関数か偶関数であることと，値域 [-1,1] における絶対値の 2 乗が 1 以下であることを満たせば自由に選ぶことができる．そのため，特異値を変換するために用いる多項式を問題によって変えることで様々な量子計算タスクを実行可能であり，量子フーリエ変換や探索，ハミルトニアンの時間発展といったタスクを実行できる．また，多項式に応じて操作なパラメータを事前に計算する必要があるが，その計算は古典的に効率的に計算できることが知られている [17].

QSVT で実行可能なタスクの一つに，与えられた行列の逆行列をベクトルに適用する操作がある．この操作は機械学習やデータ分析などで広く応用が可能なため，FTQC の重要な用途の一つとされている [18], [19]．本研究ではこの逆行列計算に着目しリソース見積もりを行う．なお，逆行列計算は Harrow-Hassidim-Lloyd（HHL）のアルゴリズム [4] でも解けることが知られており，行列の要素の次元を N，行列の条件数（行列の固有値の絶対値の最大値と最小値の比）を κ，計算の精度を ϵ として，$O(\kappa^2 \mathrm{poly}(\log N))/\epsilon)$ で計算できる．一方，QSVT では多項式によって $1/x$ を近似することによって $O(\kappa \log(\kappa/\epsilon) \mathrm{poly}(\log N)))$ で計算できるため，より高速である．[3] 今回は行列を対角化できる形に変形した上で，固有値を多項式に従って変換することで逆行列を計算するタスクについて考察を行った．

QSVT では 2 種類の操作を交互に繰り返すことによって達成されるが，その 1 ステップごとに QROM と呼ばれるような「入力に応じて量子状態を返す」回路が必要になる [6]．QROM の回路の例を**図 3** に示す．QROM では上部の「In$_l$」の左側が入力であり，下部の「data :d_l」の右側が出力である．data :d_l には $|0\rangle$ を入力する．In$_l$ の部分には $|0\rangle, |1\rangle, |2\rangle, ..., |l-1\rangle$ とその重ね合わせを入力することができ，それらの状態は QROM の問い合わせのあとでも変化しない．下部の $d_0, d_1, ..., d_{l-1}$ の部分には，事前に CNOT ゲートを配置することで，上部の入力である $|n\rangle$ $(0, 1, ..., l-1)$ に応じて対応する $|d_n\rangle$ が出力される．

QROM では 3 量子ビット量子ゲートである Toffoli ゲートを多数実行する必要がある．個々の Toffoli ゲートは 4 つの T ゲートを用いて実装することができることが知られており [20]，また QSVT においては QROM に T ゲートの実行が集中している [6].

3. Method

3.1 柔軟に量子回路を記述するライブラリ Q-Divide の構築

QROM に限らず，Toffoli ゲートや CNOT ゲートなどを複雑にかける必要があるアルゴリズムは多い．[1], [4] そのような回路が複雑となる要因として，既存の量子回路の記述の方式 [21] では CNOT ゲートをかけるときに制御量子ビットと被制御量子ビットを同時に指定しなければならないことがあると考えた．そのため，今回は，CNOT のゲートの制御量子ビットと被制御量子ビットを別々に指定することができる量子回路記述ライブラリ（Q-Divide）を Rust 言語で構築した．Rust は高度な抽象化機能と高速な動作とメモリ安全性を両立したプログラミング言語であるため，これを採用した．

開発したライブラリによる CNOT ゲートの記述を Listing 1 に示す．このコードで示されているように，Q-Divide では CNOT ゲートの制御部分と被制御部分を別々に記述し，それを自由に値として受け渡すことができる．コード中の QubitCell は量子ビットオブジェクトへのポインタであり，clone はそのポインタの複製である．これにより，Rust の所有権システムに起因する複雑さを軽減している．制御部分がどの量子ビットの何番目の操作に由来するのかは内部的に自動でカウントされるため，プログラマーが明示的に管理する必要はない．また，制御部分に相当するオブジェクトを使い回すことによって，One-Control-Multi-Target のような記述を行うことも可能である．ただし，制御量子ビットと被制御量子ビットが同じ CNOT ゲートのような実際の量子回路として無効なものを記述しようとすると実行時エラーになる．

Listing 1: CNOT ゲートの記述

```
pub fn cnot(q1: QubitCell, q2: QubitCell) {
    // CNOT gate
    // Declare a control part of a CNOT gate
    let control_from = Qubit::control(q1.
        clone());
    // Declare a target part of a CNOT gate
    let target = Qubit::export(q2.clone());
    // Connect the two parts of a CNOT with
        declared variables.
    target.control_by(&control_from);
}
```

今回我々は，構築したライブラリを用いて**図 4** にあるような QROM（図 3）の制御部分（上部）の再帰構造の基本単位（図 4）をデータ読み出し部分（下部）から分離して Listing 2 に記述した．個の記述では引数で渡された制御部分に相当するオブジェクト（ControlFrom）を使い回すことによって，One-Control-Multi-Target を記述している．関数から被制御部分に相当するオブジェクトを返すことで，別の場所で CNOT の被制御場所を記述できる．

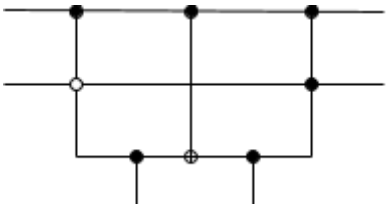

図 4: QROM の基本的な再帰構造の基本単位．制御量子ビットと被制御量子ビットを切り離して記述できるようになると，これを再帰的に組み合わせるだけで QROM の回路の制御部分を記述できる

Listing 2: QROM の一部（図 4）の回路の記述

```
pub fn in_layer(
    q1c: &ControlFrom,
    data: QubitCell,
    output: QubitCell,
) -> (ControlFrom, ControlFrom) {
    Qubit::gate(data.clone(), PrimitiveGate
        ::X);
    toffoli_first_control(q1c, data.clone(),
        output.clone());
    Qubit::gate(data.clone(), PrimitiveGate
        ::X);

    let control_left = Qubit::control(output.
        clone());

    let export_1 = Qubit::export(output.clone
        ());
    export_1.control_by(q1c);

    let control_right = Qubit::control(output
        .clone());

    let export_2 = Qubit::export(output.clone
        ());
    export_2.control_by(q1c);

    Qubit::gate(data.clone(), PrimitiveGate
        ::X);
    toffoli_first_control(q1c, data.clone(),
        output.clone());
    Qubit::gate(data.clone(), PrimitiveGate
        ::X);

    return (control_left, control_right);
}
```

3.2 PyZX による最適化とリソース見積もり

今回記述したライブラリは，量子回路を ZX-calculus の枠組みを用いて最適化する PyZX[8] と呼ばれる回路最適化フレームワークの入出力の形式である JSON 形式の入出力に対応しており，記述した回路を JSON 形式で出力できるほか，PyZX の出力した JSON から T-count および T-depth を計算することができる．今回はライブラリによって記述された回路を出力して ZX-calculus による最適化を行い，最適化前後の T-count および T-depth の変化を観察した．PyZX の最適化は確率的であるため，100 回の最適化を行い，最適化後の T-count および T-depth の平均と標準偏差を計測した．

また，その結果を用いて，量子計算機が古典計算機よりも速く与えられた行列の逆行列をかける操作が実行できるようになるために必要な問題サイズ，およびそのために必要な量子ビット数を計算した．ここでは簡単な概算のために，古典計算機側は秒間 10^{11} の命令ができると推定した．量子計算機では必要な論理量子ビット数と論理ゲートの数から，表面符号で実装することを考慮して回路に必要な量子ビット数や時間を見積もった．

量子計算機に必要な時間 t_N の見積もりは以下のように計算した．まず，T ゲート一つにかかる時間 s を $s = 10^{-5}$ 秒 [6] であるとした．その上で，c を QSVT の 1 ステップあたりに必要な T ゲートの個数の概数（$c = 88\log N$），r を削減率（$r = 0.65$）,ϵ を計算の精度（$\epsilon = 10^{-3}$）とした．次に，必要な QSVT のステップ数 n_L を $n_L = \kappa\log(\kappa/\epsilon)(\log N)^4$ によって計算した．ここで，$(\log N)^4$ は行列の要素の密度についての仮定である．最後に，t_N を $t_N = crsn_L$ で概算した．

必要な物理量子ビット数 q_N は [6] を参考に以下のように計算した．まず，1 ステップあたりに許される論理エラーの確率 p_L を $p_L = 1/10n_L$ とした．次に，表面符号において p_L は格子サイズ d を用いて $p_L = 0.1^{d/2}$ と概算できることから d を求めた．これと，論理量子ビットに必要な数 $n_Q = 4\log N$，さらに，表面符号における魔法状態蒸留や格子手術なども考慮して，$q_N = 10n_Q d^2$ として計算を行った．

4. Result and discussion

4.1 回路の記述の簡素化

QSVT では Method で述べたような QROM が複数個存在している．既存の回路記述方式 [21] によって QROM を記述した場合，行数は同じ程度になるものの，制御部分とデータ読み出し部分を分離できないため，今回開発したライブラリによる記述よりやや複雑になる．今回作ったライブラリは Toffoli,CNOT ゲートによる制御部分を抜き出して簡潔に共通化できるため，より直感的に回路を書くことができる．これにより，記述の再利用性が高まることや，回路記述のデバッグ，テストが容易になることが期待できる．

表 1: 最適化の結果

最適化前の T-depth	最適化後の T-depth	削減率（%）
63	46.62	26.00
127	82.24	35.24
254	144.97	42.93

表 2: 必要な物理量子ビット数の見積もり

κ の値	物理量子ビット数
10^1	214024
10^3	324106
10^5	453563

4.2 回路の最適化の結果

最適化の前後で回路の T-count を評価したところ，T-count の値は ZX-calculus によって改善しないことが分かった．一方, T-depth は回路の規模にも依存するが最大で 35% 程度削減された．最適化による削減率を表 1 に示す．

4.3 量子計算機の優位性が出る状況の見積もり

QSVT を用いた逆行列計算は条件数 κ に対して線形に増えていくため，特に条件数が小さいケースに対して有効である．一方，条件数が小さい行列の逆行列計算は古典計算機でも効率的に解くことができ，共役勾配法を用いると $O(\kappa N)$[22] の計算量で実行できる．QSVT を用いた逆行列計算で必要となる時間と，共役勾配法で必要となる時間を様々な条件数に対して概算し比較したものが**図 5** である．

図 5 より，10^{13} 程度の大きさの行列で量子計算機のほうが速い速度で逆行列をかけられるようになると考えられる．この図をもとに，条件数 κ ごとに古典計算機の速度を超えるために必要な物理量子ビットの数を概算した結果が表 2 である．必要な物理量子ビットは 21 万から 45 万量子ビットであると見積もられる．実際に計算に用いている論理量子ビットの数は 60 量子ビットほどであるが，長いステップの間充分な精度で計算を進めるためにこのように多数の量子ビットが必要になっている．

5. Conclusion and future directions

この研究では，対象を簡潔に記述するライブラリ Q-Divide を開発することで，これまでより柔軟かつ効率的に量子回路を設計することを可能にした．さらに，このライブラリを用いて QROM を設計し，ZX-calculus による最適化を行った．最適化の結果として T-count は減らなかったものの，T-depth を最大で 35%削減することに成功した．さらに，今回見積もった T-gate の数から量子計算機が逆行列演算について古典計算機を超えるのに必要な規模を概

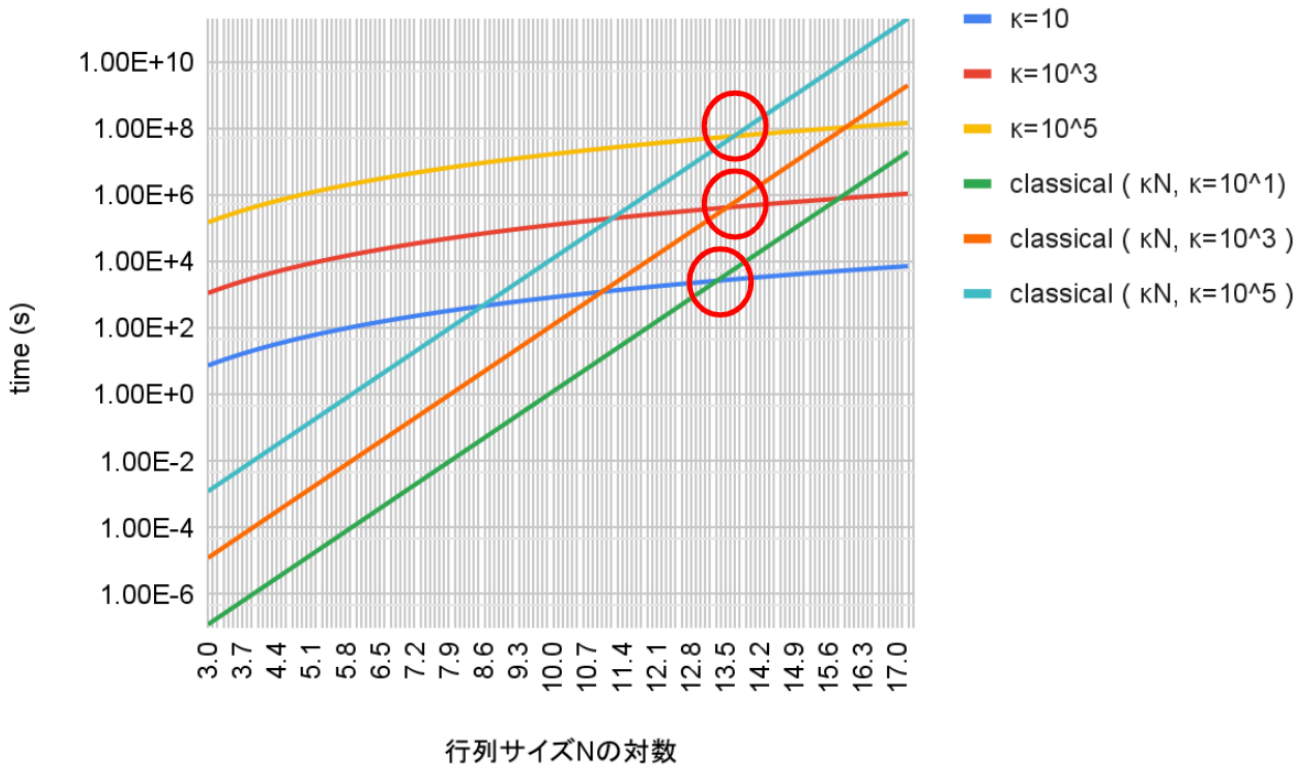

図 5: 古典計算機と量子計算機で与えられた行列の逆行列をかけるときに必要と考えられる時間の見積もり．量子計算機と古典計算機の両方の場合について $\kappa = 10, 10^3, 10^5$ の場合に計算を行い，行列サイズ N の対数に対してかかる時間の見積もりをプロットした．青，赤，黄色の線が量子計算機の場合の時間の見積もりであり，緑，橙，水色の線が古典計算機の時間の見積もりである．赤丸は，それぞれの条件数において量子計算機が古典計算機よりも短時間で問題を解けるようになる地点を示している．

算した．結果として，行列の条件数ごとに必要な問題の規模とその実現に必要な量子計算機のサイズを見積もることができた．

今回開発した Q-Divide のように考察したい対象を簡潔に記述する方式（プログラミング言語やライブラリ）を構築することができれば，より本質的な問題に集中でき，その結果として研究が進展する可能性がある．この方針は QSVT のみならず，Measurement-Based Quantum Computing（MBQC）[23] や誤り訂正符号の設計など，他の問題についても応用することができる可能性がある．

今回，ZX-calculus を用いた最適化によって，構造的に記述されている QROM の T-count がほとんど変化しなかったことは注目に値する．2 つの等価な量子計算プロセスにおいて T ゲートの数や分布がどのように変化しうるのかは研究の余地があると考えられる．一方，T-depth に関しては既存の設計より大幅な減少がみられたことから，QROM にはまだ大きな改善の余地があることが示唆される．今後の研究としては，本件の最適化の結果をもとに体系的な QROM の最適化手法を提案するという課題が考えられる．

今回開発したライブラリ Q-Divide は公開し，QSVT を用いたアルゴリズムやその他の複雑なアルゴリズムの記述と最適化に用いられるようにする予定である．今回行った最適化は最も基本的なものであるため，回路に適切で T-depth をさらに減らせるような最適化の探索を行うこともできると考えられる．

謝辞 本研究は JST さきがけ（助成番号：No. JPMJPR1916），内閣府ムーンショット（助成番号：No. JPMJMS2061）の助成の元で行いました．

参考文献

[1] Gidney, C. and Ekerå, M.: How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, *Quantum*, Vol. 5, p. 433 (2021).
[2] Shor, P. W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM review*, Vol. 41, No. 2, pp. 303–332 (1999).
[3] Martyn, J. M., Rossi, Z. M., Tan, A. K. and Chuang, I. L.: Grand unification of quantum algorithms, *PRX Quantum*, Vol. 2, No. 4, p. 040203 (2021).
[4] Dervovic, D., Herbster, M., Mountney, P., Severini, S., Usher, N. and Wossnig, L.: Quantum linear systems algorithms: a primer, *arXiv preprint arXiv:1802.08227* (2018).
[5] Harrow, A. W., Hassidim, A. and Lloyd, S.: Quantum algorithm for linear systems of equations, *Physical review letters*, Vol. 103, No. 15, p. 150502 (2009).

[6] Babbush, R., Gidney, C., Berry, D. W., Wiebe, N., McClean, J., Paler, A., Fowler, A. and Neven, H.: Encoding electronic spectra in quantum circuits with linear T complexity, *Physical Review X*, Vol. 8, No. 4, p. 041015 (2018).
[7] Kissinger, A. and van de Wetering, J.: Reducing T-count with the ZX-calculus, *arXiv preprint arXiv:1903.10477* (2019).
[8] Kissinger, A. and van de Wetering, J.: Pyzx: Large scale automated diagrammatic reasoning, *arXiv preprint arXiv:1904.04735* (2019).
[9] Fowler, A. G., Mariantoni, M., Martinis, J. M. and Cleland, A. N.: Surface codes: Towards practical large-scale quantum computation, *Physical Review A*, Vol. 86, No. 3, p. 032324 (2012).
[10] Fowler, A. G. and Gidney, C.: Low overhead quantum computation using lattice surgery, *arXiv preprint arXiv:1808.06709* (2018).
[11] Kitaev, A. Y.: Quantum computations: algorithms and error correction, *Russian Mathematical Surveys*, Vol. 52, No. 6, pp. 1191–1249 (1997).
[12] Bravyi, S. B. and Kitaev, A. Y.: Quantum codes on a lattice with boundary, *arXiv preprint quant-ph/9811052* (1998).
[13] Chamberland, C., Noh, K., Arrangoiz-Arriola, P., Campbell, E. T., Hann, C. T., Iverson, J., Putterman, H., Bohdanowicz, T. C., Flammia, S. T., Keller, A., Refael, G., Preskill, J., Jiang, L., Safavi-Naeini, A. H., Painter, O. and Brandão, F. G.: Building a Fault-Tolerant Quantum Computer Using Concatenated Cat Codes, *PRX Quantum*, Vol. 3, No. 1 (online), DOI: 10.1103/prxquantum.3.010329 (2022).
[14] Holmes, A., Jokar, M. R., Pasandi, G., Ding, Y., Pedram, M. and Chong, F. T.: NISQ+: Boosting quantum computing power by approximating quantum error correction, *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, pp. 556–569 (2020).
[15] Ueno, Y., Kondo, M., Tanaka, M., Suzuki, Y. and Tabuchi, Y.: QECOOL: On-Line Quantum Error Correction with a Superconducting Decoder for Surface Code, *arXiv preprint arXiv:2103.07526* (2021).
[16] Horsman, C., Fowler, A. G., Devitt, S. and Van Meter, R.: Surface code quantum computing by lattice surgery, *New Journal of Physics*, Vol. 14, No. 12, p. 123011 (2012).
[17] Low, G. H., Yoder, T. J. and Chuang, I. L.: Methodology of resonant equiangular composite quantum gates, *Physical Review X*, Vol. 6, No. 4, p. 041067 (2016).
[18] Ciliberto, C., Herbster, M., Ialongo, A. D., Pontil, M., Rocchetto, A., Severini, S. and Wossnig, L.: Quantum machine learning: a classical perspective, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 474, No. 2209, p. 20170551 (2018).
[19] Liu, J., Liang, Y. and Ansari, N.: Spark-based large-scale matrix inversion for big data processing, *IEEE Access*, Vol. 4, pp. 2166–2176 (2016).
[20] Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J. A. and Weinfurter, H.: Elementary gates for quantum computation, *Physical review A*, Vol. 52, No. 5, p. 3457 (1995).
[21] Wille, R., Van Meter, R. and Naveh, Y.: IBM' s Qiskit tool chain: Working with and developing for real quantum computers, *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, pp. 1234–1240 (2019).
[22] Shewchuk, J. R. et al.: An introduction to the conjugate gradient method without the agonizing pain (1994).
[23] Briegel, H. J., Browne, D. E., Dür, W., Raussendorf, R. and Van den Nest, M.: Measurement-based quantum computation, *Nature Physics*, Vol. 5, No. 1, pp. 19–26 (2009).