リソース推定

誤り耐性量子計算での量子AIに向けて

2023/8/3 第3回量子ソフトウェアワークショップ
（株）NTTデータグループ　技術開発本部
加藤 拓己

発表者について – 加藤 拓己

2022年～ NTTデータグループ

- 量子アニーリングや数理最適化の技術導入支援
- ゲート方式量子コンピュータでのリソース推定の研究

それまで: blueqat社

量子コンピューティングライブラリ blueqat

- 誰もが手軽に量子コンピューティングを行えるようにしたい

他、セキュリティソフト開発、プラント制御設計・現地試運転など

「IBM Quantumで学ぶ量子コンピュータ」出版(2021, 秀和システム)
2020年度 未踏ターゲット事業採択

Innovation Center

グローバル6か国に拠点をもち、先進顧客へのアプローチを主眼においた体制で活動を推進。
先進技術を見極め、お客さまとの共創R&Dを通して新たなビジネス創出をめざす

NTT DATA Innovation Center

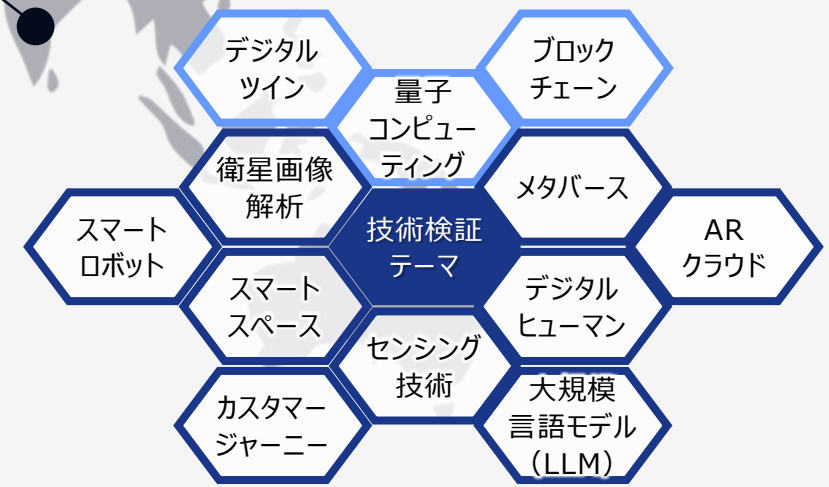
2022年8月～
世界6か国で活動を開始

1 中長期での技術戦略策定

2 中長期視点での先進技術獲得

3 イノベータ顧客との共創

4 グローバルレベルの技術者育成と強化

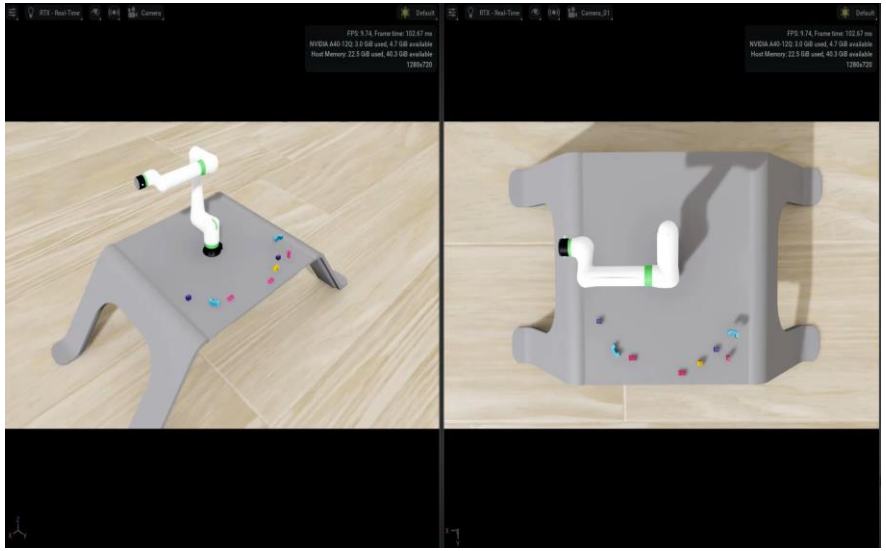


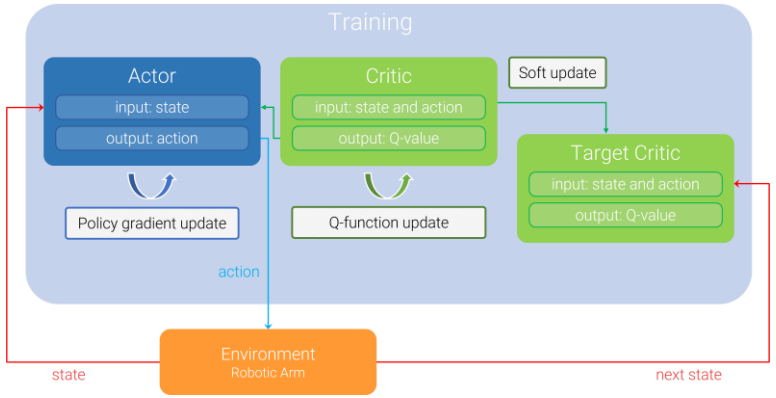
エキスパート100名（リサーチャー、コンサルタント、エンジニア）でスタート。 2025年度末までに300名体制に増強予定。

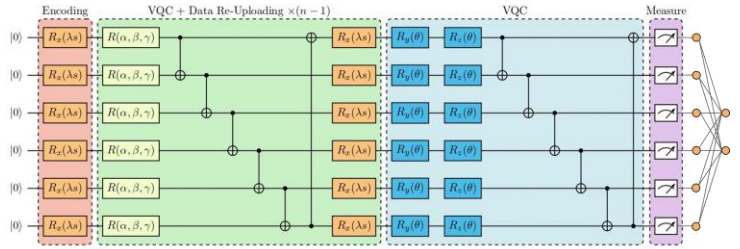
NTT DATA

量子強化学習 + デジタルツイン + ロボットアーム制御

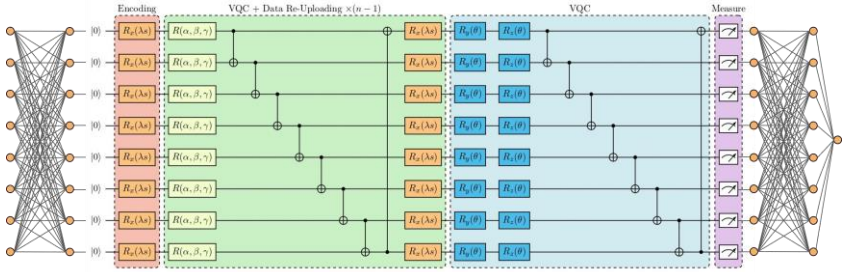
2次元2リンクロボットアームで、量子強化学習モデルを開発した
⇒3次元7リンクにモデルを拡張中
　NVIDIA Omniverseを使ってFANUC社製ロボットアームを制御







Actor量子回路



Critic, target Critic量子回路

Alberto Acuto, Paola Barillà, Ludovico Bozzolo, Matteo Conterno, Mattia Pavese, & Antonio Policicchio. (2022). arXiv:2212.11681

数理最適化によるプリント基板加工の生産効率改善

NTTデータグループ・広島大学・伸光製作所

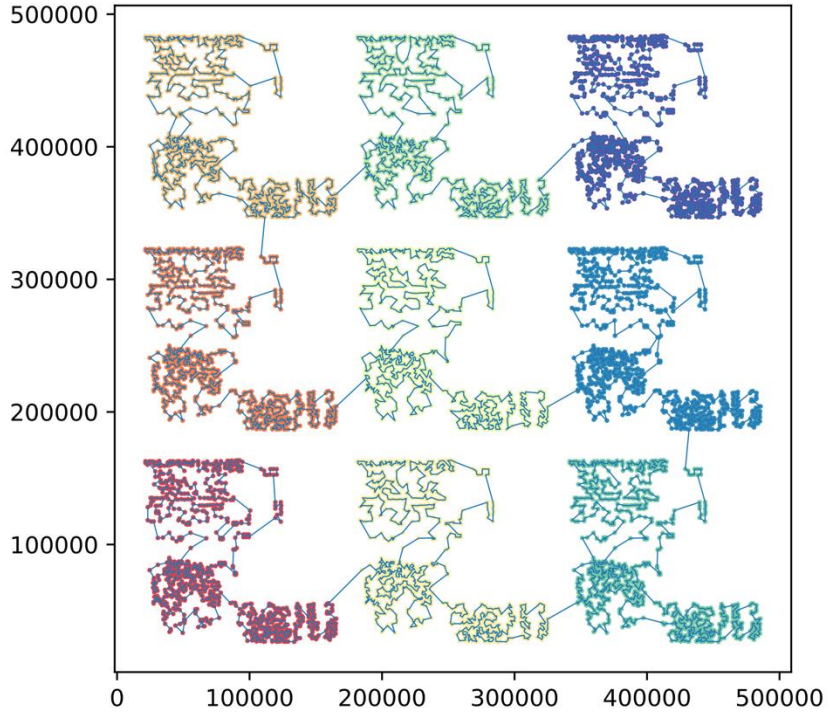
プリント基板の穴あけ工程を効率化したい⇒ドリルの移動時間を短くしたい

- 穴あけの点数：数万点
  ヒューリスティック法を基に、独自に開発
- 製造上の制約：ゾーン制約
  あるゾーンに属する穴を訪問しきってから、次のゾーンに行く
- 加工機の特性を考慮した最適化
  ドリルは、それぞれx軸・y軸方向に動く2つのモーターで移動するため、
  点同士の距離は、ユークリッド距離ではなく、チェビシェフ距離となる

$$d\big((x_1, y_1), (x_2, y_2)\big) = \max(|x_2 - x_1|, |y_2 - y_1|)$$

⇒実業務の特性に合ったアルゴリズムを開発・実装

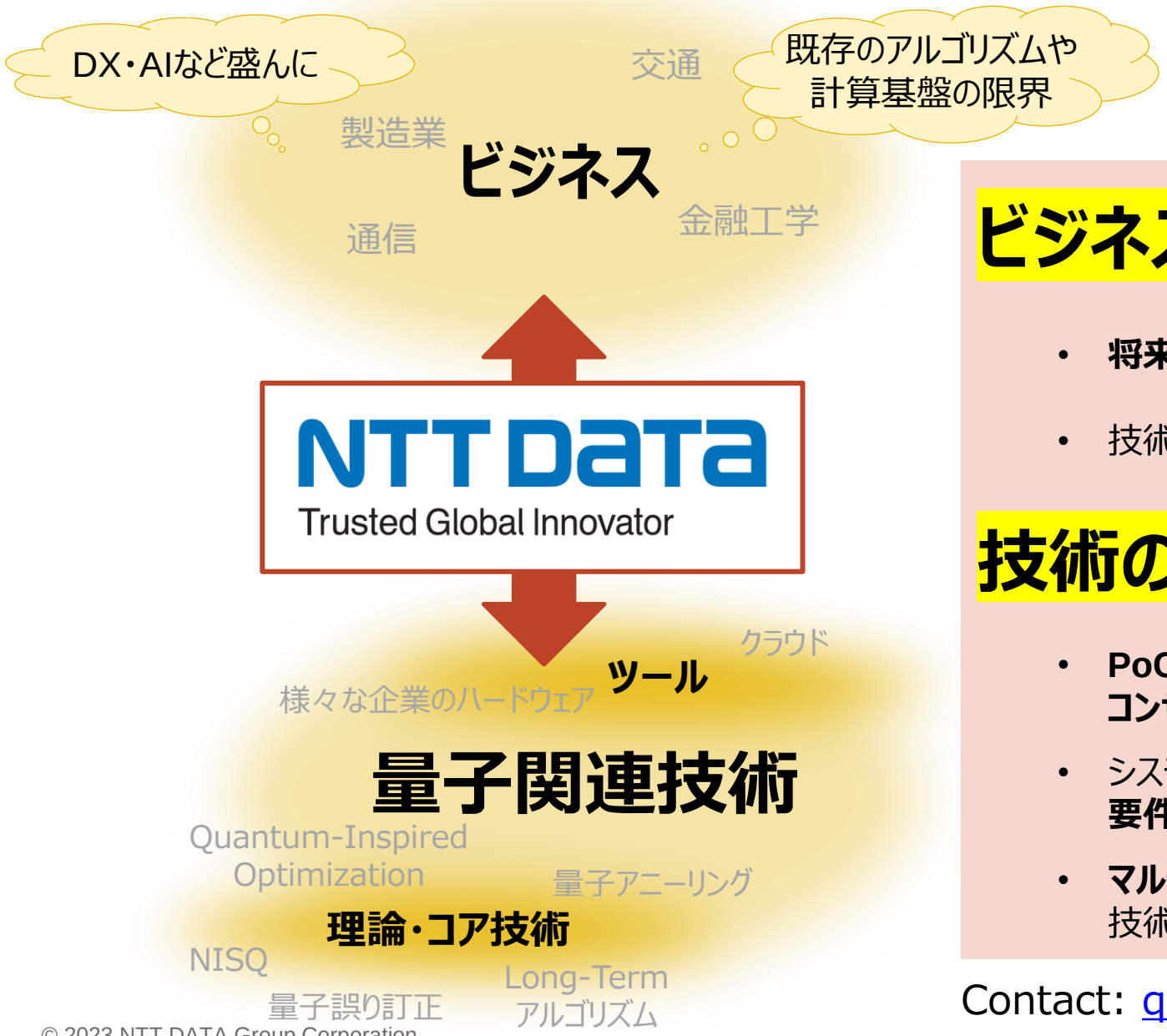
2023年4月より、ソフトウェアを伸光製作所の
製造ラインに導入、効率化・安定運用を実現



↑10,000点の穴の座標（9つのゾーン制約を含む）
に対し、本技術で順序の探索を行った結果

23,544点での検証結果	従来	最適化後
ドリルを移動させる時間	3,142秒（約52分）	2,830秒（約47分）
数値実験結果のチェビシェフ距離	29,371,816 [μm]	14,717,600 [μm]
総加工時間	9,101秒（約152分）	8,788秒（約146分）

NTTデータグループにおける量子技術に対するミッション



ビジネス課題の解決

- **将来的なシステム化**に向けた準備。
- 技術的な用途開拓・課題解決をサポート。

技術の活用の領域の強み

- **PoC（実証実験）の実施支援。**
  コンサルティングや**セミナー**の提供。
- システム開発の実績に基づく、
  要件定義からクラウド等の技術まで、全般に対する深い知識。
- **マルチベンダー。**
  技術・企業を限らず、ビジネス課題ベースでソリューションを提案。

Contact: qcomputer@kits.nttdata.co.jp

量子コンピュータとその種類

	イジングモデル	NISQ	FTQC
用途	主に組合せ最適化	応用に向けた研究が活発	材料・化学計算、金融計算、最適化、暗号解読、…
開発状況	数千量子ビット級のマシンが商業的に提供	数百量子ビット級のマシンが商業的に提供	現状はまだ出来ていない
開発手法	PyQUBOなどのライブラリでQUBO行列を作る	Qiskitなどのライブラリで量子ゲートを並べた量子回路を作る	量子ゲートを手作業で並べる方式では開発が困難

FTQCで解ける問題: 実はたくさんある

物理・化学シミュレーション:
大規模な分子や電子のふるまいのシミュレーションが、効率的に行える

量子モンテカルロ計算:
金融商品の価格付けなどのモンテカルロ計算が、従来手法よりも効率的に解ける

グローバー探索:
総当たり計算などの、解くのが難しかった問題の解が従来手法より効率的に求まる

線形代数・連立方程式:
巨大な疎行列からなる連立方程式の解を効率的に計算でき、リコメンデーション機能や線形回帰、サポートベクタマシン(SVM)などの機械学習への応用が期待できる

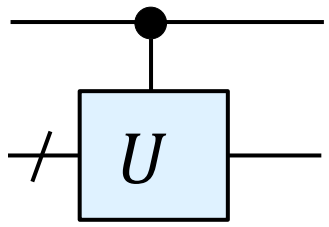
数理最適化:
半正定値計画問題(SDP)への応用が研究されている

巨大な文字列のパターンマッチ:
DNA配列に対する応用も期待できる

素因数分解・離散対数問題:
RSA暗号や楕円曲線暗号などが解読できる (耐量子暗号に将来的に移行)

QSVT – 量子アルゴリズムの「大統一理論」

量子特異値変換(QSVT)という枠組みで、様々な用途の計算が実現可能



ブロックエンコーディング

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix}$$

固有値や特異値を以下のように変換できる

$$U_f = \begin{bmatrix} f(A) & \cdot \\ \cdot & \cdot \end{bmatrix}$$

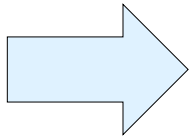
fはチェビシェフ多項式として記述できる多項式

ユニタリ行列ではない行列、正方行列ではない行列も、ブロックエンコーディングを行うことでQSVTを利用できる
⇒大規模な行列計算や機械学習にも応用が期待できる

Martyn, J., Rossi, Z., Tan, A., & Chuang, I. (2021). Grand Unification of Quantum Algorithms. *PRX Quantum, 2, 040203.*

FTQCアルゴリズムのリソース推定

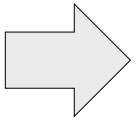
FTQC: 量子優位性が理論的に示されている
⇒先の長い技術であるが、取り組む意義が大きい



優位性を得るために必要なマシンや問題の規模は？
⇒企業の投資戦略策定のため、これらを知りたい

ある特定の問題を特定の量子ソフトウェアで解くために、以下を推定したい

- どれくらいの規模の量子コンピュータが必要か?
- どれくらいの時間が必要か?



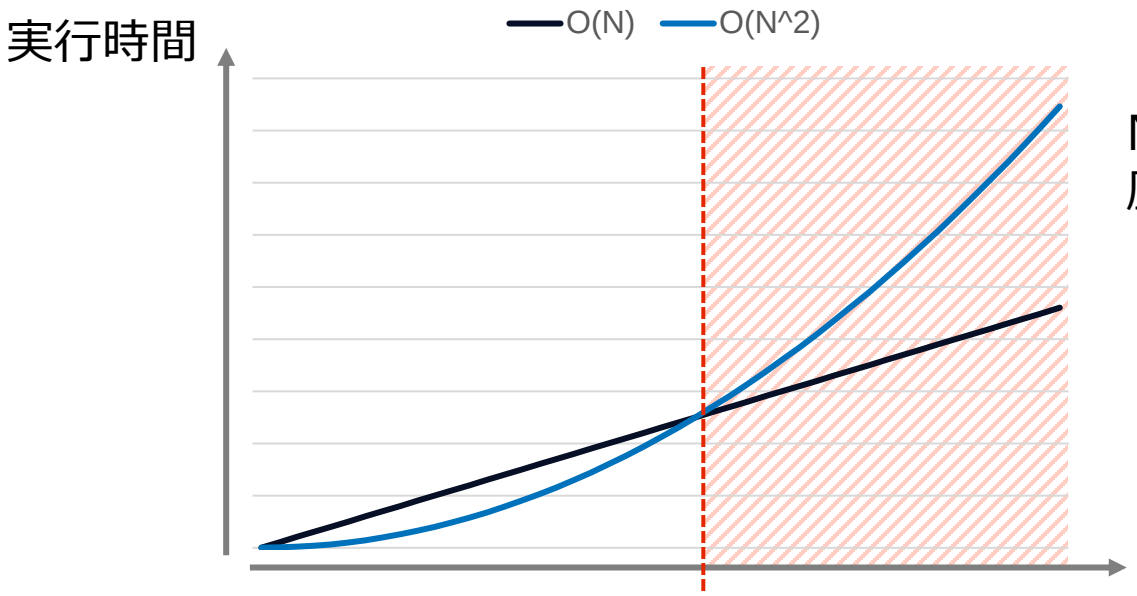
リソース推定

具体的に量子ソフトウェアを書き下し、使用されている量子ゲートの数などから問題を解くために必要な時間を推定する
⇒今回、量子化学計算の問題をターゲットに、量子位相推定アルゴリズムを実装し、リソース推定を行った

どのようにソフトウェアの「速さ」を測るか?

- 計算量のオーダーを用いて測る

⇒「問題の大きさが増えると、実行時間がどのように増えるか」に着目する



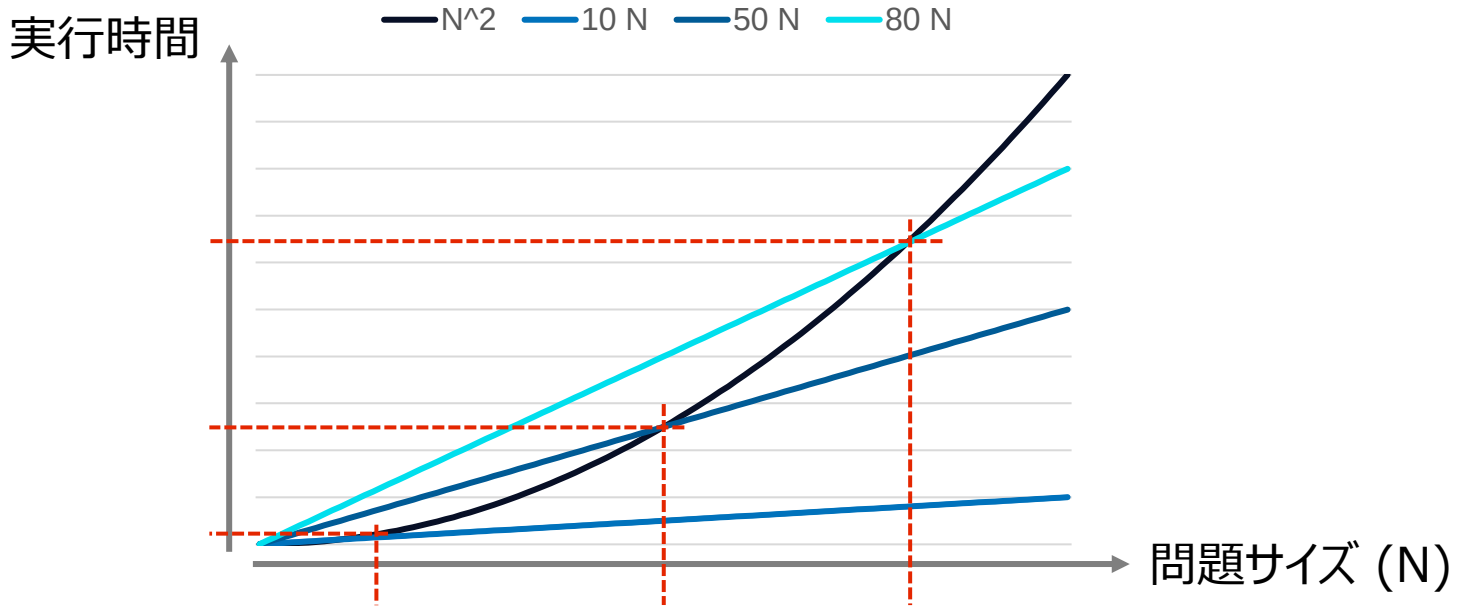
Nが大きくなると、オーダーが小さい方が
圧倒的に計算が速い

FTQC用の量子アルゴリズムは、**古典アルゴリズムと比べて計算量のオーダーが小さいため高速**、という議論がよく行われる

定数倍は無視してもよいか?

- 計算量のオーダーで速さを測るときは、その定数倍は無視する

⇒オーダーだけでは、「どれくらい速くなるか?」「速くなるのは、どの大きさからか?」の議論ができない

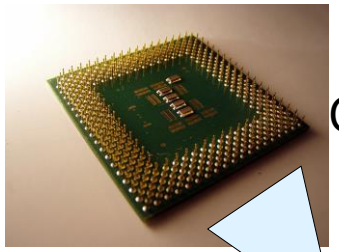


定数項に関わらず、Nが十分に大きくなればオーダーが小さい方が有利である結論には変わりがないが、
「十分に大きく」とはどのくらいかを議論するには、より深く量子ソフトウェアの実装に潜り込む必要がある

古典計算と量子計算で定数項はどれくらい違うか?

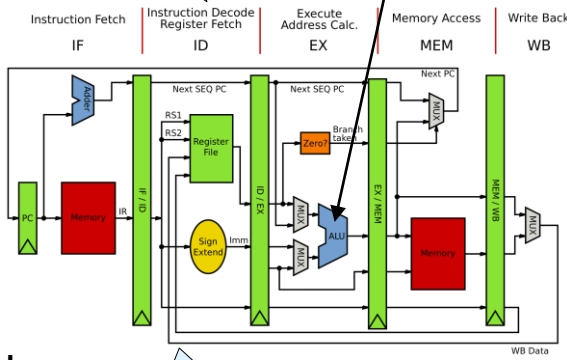
古典コンピュータと量子コンピュータで、加算の速度を比較する

古典: CPUクロック周波数が3GHz～5GHz
64ビット同士の整数加算を10^{-9}秒で計算可能

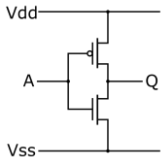


CPU

加算器はここ



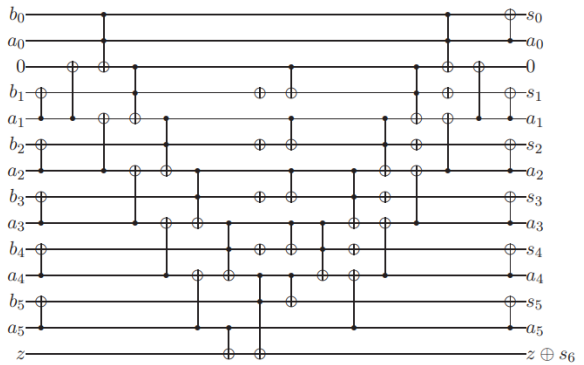
論理ゲート



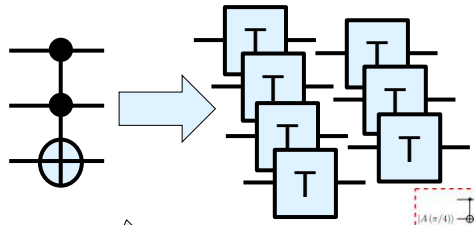
量子: 誤り訂正符号の符号距離を21とおくと、
64ビット同士の整数加算が10^{-1}秒程度の見積もり

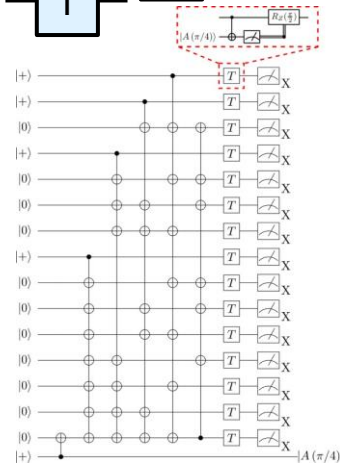
加算回路→回路の深さがビット数に比例

Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, & David Petrie Moulton. (2004). arXiv:quant-ph/0410184



FTQC命令に変換
ToffoliゲートはTゲート7個

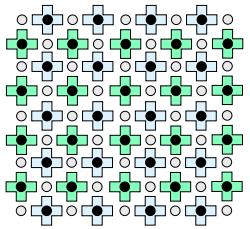




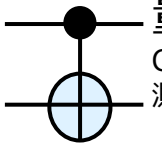
Tゲート操作: 15dサイクル

Trout, C., & Brown, K. (2015). *International Journal of Quantum Chemistry, 115(19), 1296-1304.*

符号距離 d
(誤り訂正の強さ)



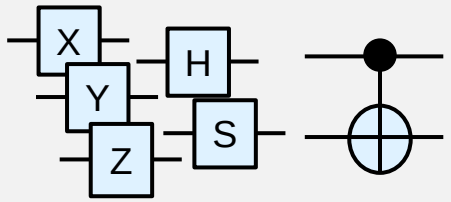
量子誤り訂正符号
1サイクルで深さ4のCNOTと測定操作
→ サイクルあたり1～10 μs



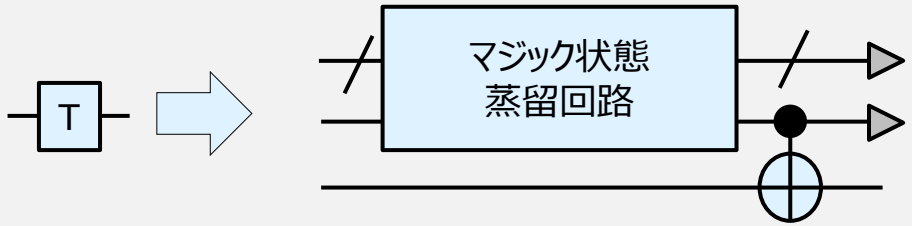
量子ゲート操作・測定操作
CNOT: IBM Quantumで200～500 ns
測定: IBM Quantumで数百～数千ns

NTT DATA

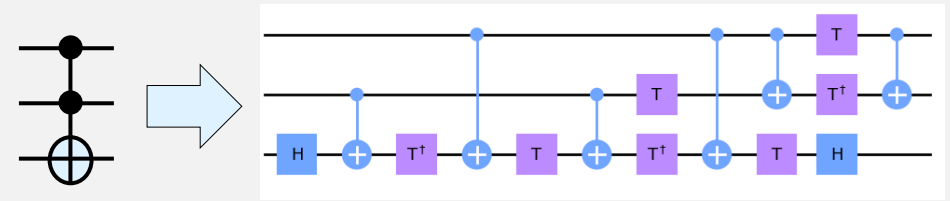
FTQCでの実行時間はTゲートの数で決まる



⇒高速に実行できる



実行には「マジック状態」をひとつ消費する
マジック状態の作成(蒸留)が**実行のボトルネックに**

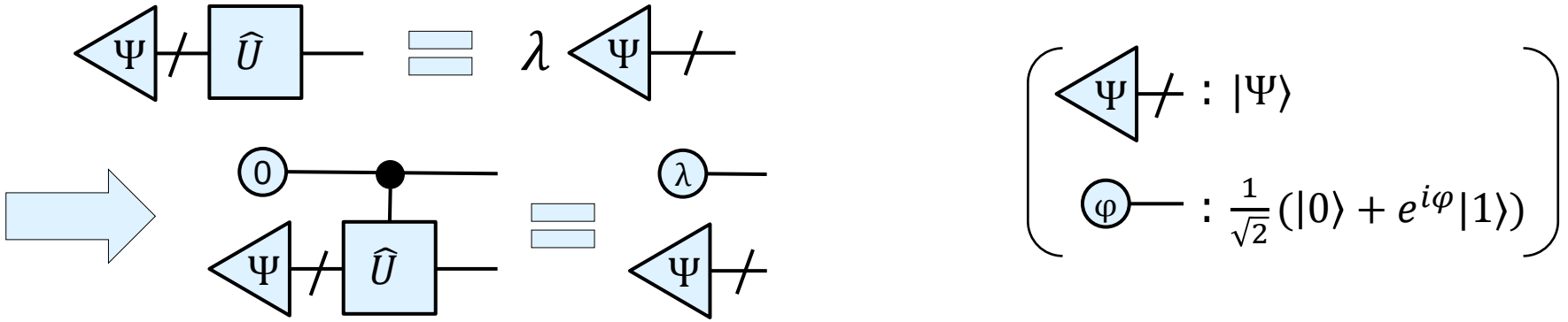


Solovay-Kitaevアルゴリズムで
Tゲート、Hゲートなどに分解

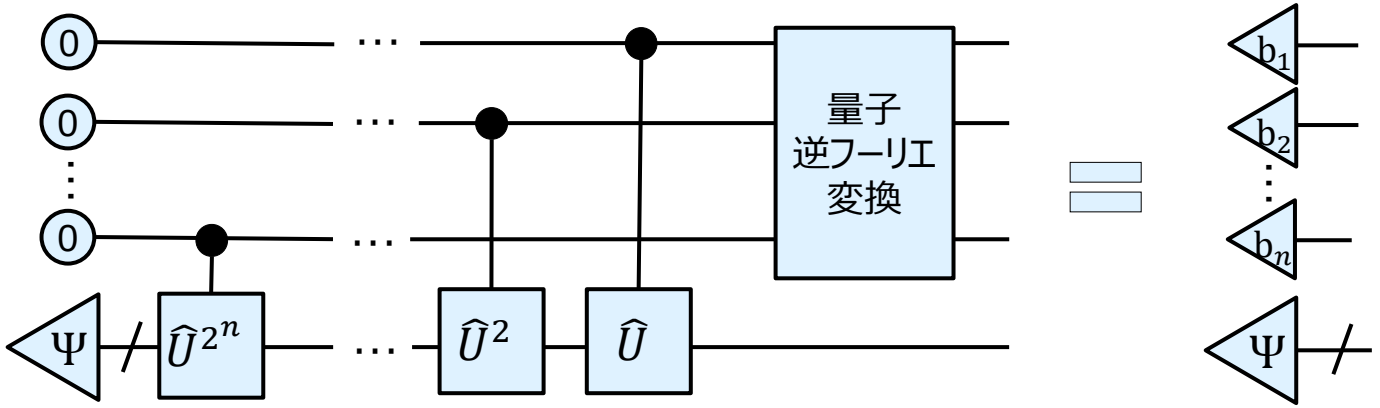
量子ソフトウェアを、Tゲートを使った回路に分解して、Tゲートの数を数えることで、
量子ソフトウェアの実行時間を見積もることができる

量子位相推定

固有方程式$\hat{U}|\Psi\rangle = \lambda|\Psi\rangle$の、固有ベクトル(の近似値)が分かっている場合、
その固有値は、量子位相推定アルゴリズムによって、任意の精度で求めることができる。



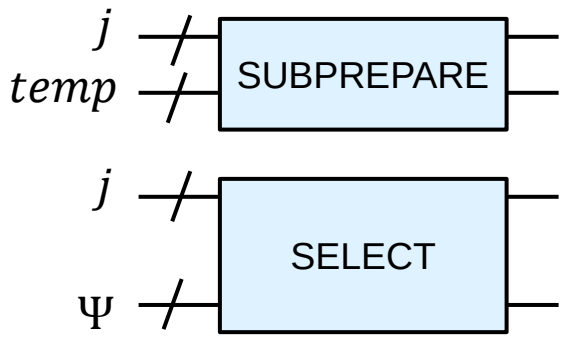
λを$2\pi\ 0.b_1 b_2 \cdots b_n$と2進小数点表記すると$(\lambda = 2\pi \times \sum_{k=1}^{n} 2^{-k} b_k,\ b_k \in \{0,1\}(k = 1..n))$



制御ビットを測定することで、固有値が得られる。

Qubitizationによるハミルトニアンの表現

$\hat{H} = \sum_{j=1}^{d} \alpha_j \ \hat{U}_j$　($\hat{U}_j$はパウリ演算子の積) と書けるとき、以下のような量子回路を作ることができる



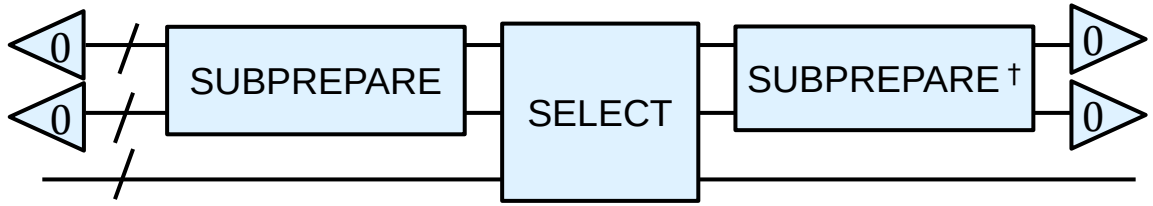
$$\mathrm{SUBPREPARE} = \sum_{j=1}^{d} \sqrt{\frac{\alpha_j}{\|\alpha\|_1}} |j\rangle|temp\rangle\langle 0|\langle temp'|$$

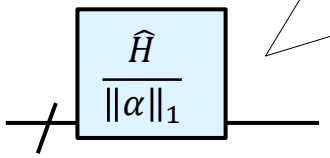
係数の部分を表現

$$\mathrm{SELECT} = \sum_{j=1}^{d} |j\rangle\langle j| \otimes \hat{U}_j$$

演算子の部分を表現

このとき、

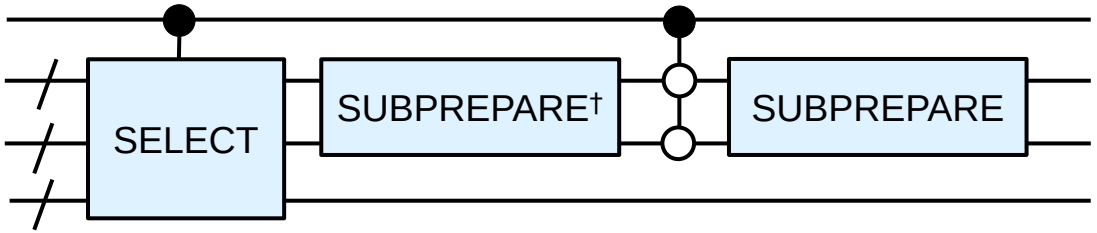


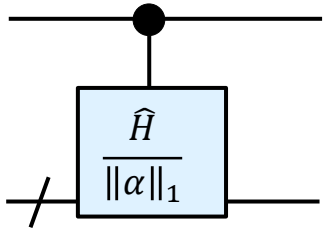


ブロックエンコーディングになっている

$\begin{bmatrix} \hat{H}/\|\alpha\|_1 & \cdot \\ \cdot & \cdot \end{bmatrix}$

さらに、





量子位相推定に使える！

Low, G., & Chuang, I. (2019). *Quantum, 3, 163.*
Babbush, R., Gidney, C., Berry, D., Wiebe, N., McClean, J., Paler, A., Fowler, A., & Neven, H. (2018). *Phys. Rev. X, 8, 041015.*

NTT DATA

量子ソフトウェアの合成とリソース推定

TOML形式で問題を記述

H_2分子

```
[algorithm]
name = "PhaseEstimation"

[[output]]
type = "t_count"

[[output]]
type = "circuit"
file = "lcu_h2_circuit.py"

[algorithm.config]
hamiltonian = "lcu"
lcu = [
  [0.19480867687252212, [["Z", 0]]],
  [0.19480867687252212, [["Z", 0], ["Z", 1]]],
  [-0.29920510757483243, [["Z", 2]]],
  [-0.29920510757483243, [["Z", 1], ["Z", 2], ["Z", 3]]],
  [0.04343266096994482, [["Y", 0], ["Z", 1], ["Y", 2]]],
  [0.04343266096994482, [["X", 0], ["Z", 1], ["X", 2]]],
  [0.04343266096994482, [["X", 0], ["Z", 1], ["X", 2], ["Z", 3]]],
  [0.04343266096994482, [["Y", 0], ["Z", 1], ["Y", 2], ["Z", 3]]],
  [0.17533443216381941, [["Z", 1]]],
  [0.1287656132604534, [["Z", 0], ["Z", 2]]],
  [0.17219827423039824, [["Z", 0], ["Z", 1], ["Z", 2]]],
  [0.17219827423039824, [["Z", 0], ["Z", 1], ["Z", 2], ["Z", 3]]],
  [0.1287656132604534, [["Z", 0], ["Z", 2], ["Z", 3]]],
  [0.18112650599125593, [["Z", 1], ["Z", 3]]]
]

[algorithm.config.accuracy]
iqft_bits = 10
walker_sampling_bits = 10
gridsynth_options = []
```

Tゲートの数を出力

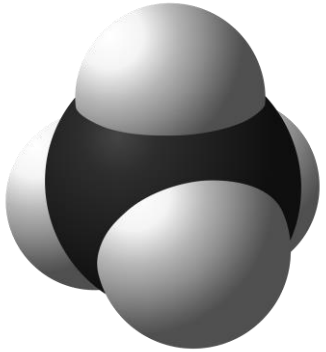
387771 ⟶ 1μs/cycle,
符号距離21で換算
122秒😱

量子回路を出力

```
from qiskit import QuantumRegister, QuantumCircuit
qregs = []
a = QuantumRegister(19, 'a'); qregs.append(a)
b = QuantumRegister(1, 'b'); qregs.append(b)
l = QuantumRegister(4, 'l'); qregs.append(l)
psi = QuantumRegister(4, 'psi'); qregs.append(psi)
qft = QuantumRegister(10, 'qft'); qregs.append(qft)
t_alt = QuantumRegister(4, 't_alt'); qregs.append(t_alt)
t_cmp = QuantumRegister(1, 't_cmp'); qregs.append(t_cmp)
t_dist = QuantumRegister(10, 't_dist'); qregs.append(t_dist)
t_keep = QuantumRegister(10, 't_keep'); qregs.append(t_keep)
qc = QuantumCircuit(*qregs)
qc.h(qft[0])
qc.h(qft[1])
qc.h(qft[2])
qc.h(qft[3])
qc.h(qft[4])
qc.h(qft[5])
qc.h(qft[6])
qc.h(qft[7])
qc.h(qft[8])
qc.h(qft[9])
:
:
```

量子優位性を得るためには、アプリケーションに特化した手法が必要

量子化学計算を素朴に実装しても、速くはならない
⇒量子ソフトウェアを用途に合わせて設計したり、古典で行われている手法を取り入れる必要がある



Methane (CH_4) / 6-31G*　(22軌道)
量子フーリエ変換の桁数: 10ビット
係数の解像度: 10ビット
⇩
Tゲートの個数 = 2,154,931,950
1μs/cycle,符号距離21で換算すると: 188.5時間

22軌道 (Full CIで2^{44}次元) での量子位相推定が現実的な時間の見積もりになったが、まだ改善の余地が大きい

- 電子構造の量子位相推定に特化したSELECT/PREPAREの実装
  - ハミルトニアンの形に着目すると、SELECT/PREPAREのTゲート数のオーダーを下げることができる
- 古典の量子化学計算で行われていた、対称性の利用やFrozen coreなどによるハミルトニアンの項数の削減

解きたい問題に合わせて量子ソフトウェアを設計し、リソース推定すれば、量子優位性のある問題は見つかりそう

量子ソフトウェア合成そのものの困難さ

Tゲートの個数を数えるために、量子ソフトウェアを量子ゲートの形で書く必要があった

$$\hat{H} = \sum_{j=1}^{d} \alpha_j \ \hat{U}_j$$

分子の座標データ

```
H        0.00      0.00      0.00
C       -1.10      0.00      0.00
H       -1.47      0.73      0.73
H       -1.47      0.27     -1.00
H       -1.47     -1.00      0.27
```

PySCF, OpenFermion →

ハミルトニアン

```
[algorithm.config]
hamiltonian = "lcu"
lcu = [[2.6876577579183527, [["Z", 0]]], [16.91173397207865, []], [2.6876577579183527, [["Z", 0], ["Z",
```

変換の過程 (第二量子化、Bravyi-Kitaev変換)で膨大なメモリを使用
スピン軌道数nに対し、O(n^4)～O(n^4 log n)程度だが、省メモリ化を意識した設計はされておらず
スピン軌道数をあまり増やせなかった

非常に長くなるので
ハミルトニアン部分を別ファイルに分けるよう作り替えたい

回路の合成プログラムはRustで作成した

設定 + ハミルトニアン: TOML →

- Tゲートの数を出力
- 量子回路を出力

合成中のゲートを全てメモリに残すのは、軌道数が増えると不可能

$$\mathrm{TCount}(AB) = \mathrm{TCount}(B) + \mathrm{TCount}(A)$$

なので、パーツごとに合成を行い、Tゲート数を数えたらメモリに残さないよう作り直した

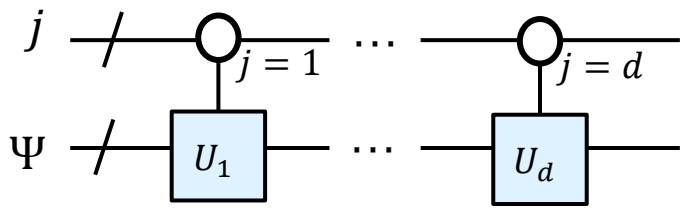
量子回路の出力の場合、回路の合成よりも
出力やI/Oの方が時間がかかった
H_2(2軌道)でQiskit回路が23MB
CH_4は10GB超えても終わらなかったので、
回路の出力を諦めた

CH_4のTゲート数を数えるのにかかった時間：947秒 (releaseビルド)

シミュレーション

リソース推定:　量子ソフトウェアは、大規模でもよく、動作させる必要がない
シミュレーション:　量子ソフトウェアは、小規模なものに限るが古典で動作させられる
⇒リソース推定とシミュレーションの両面からFTQCにアプローチ

$$\mathrm{SELECT} = \sum_{j=1}^{d} |j\rangle\langle j| \otimes \hat{U}_j$$

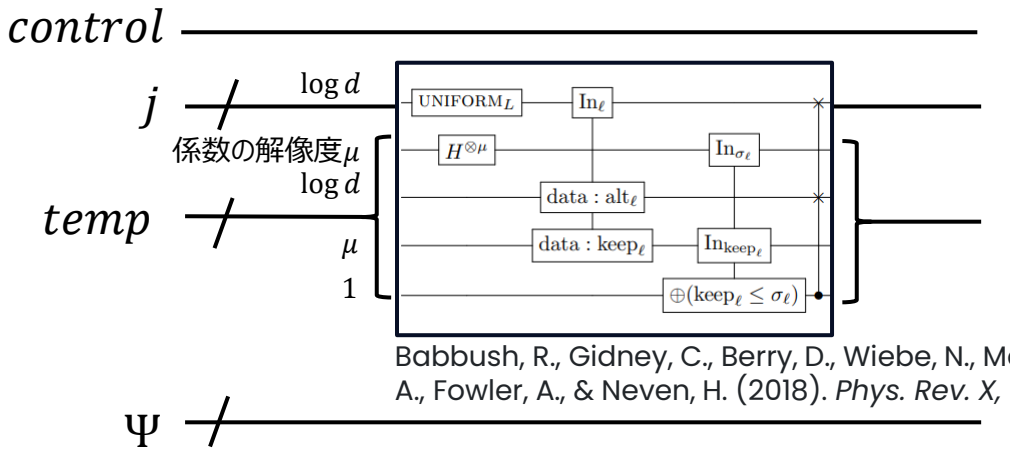


制御ビットが多いと、実機には難しいが、
シミュレーションする上では嬉しい

$$\mathrm{SUBPREPARE} = \sum_{j=1}^{d} \sqrt{\frac{\alpha_j}{\|\alpha\|_1}} |j\rangle|temp\rangle\langle 0|\langle temp'|$$

こういった数式になるユニタリ行列は多数考えられるが、
精度の検証などを考えると、ゲートでの実装と似た形がよい

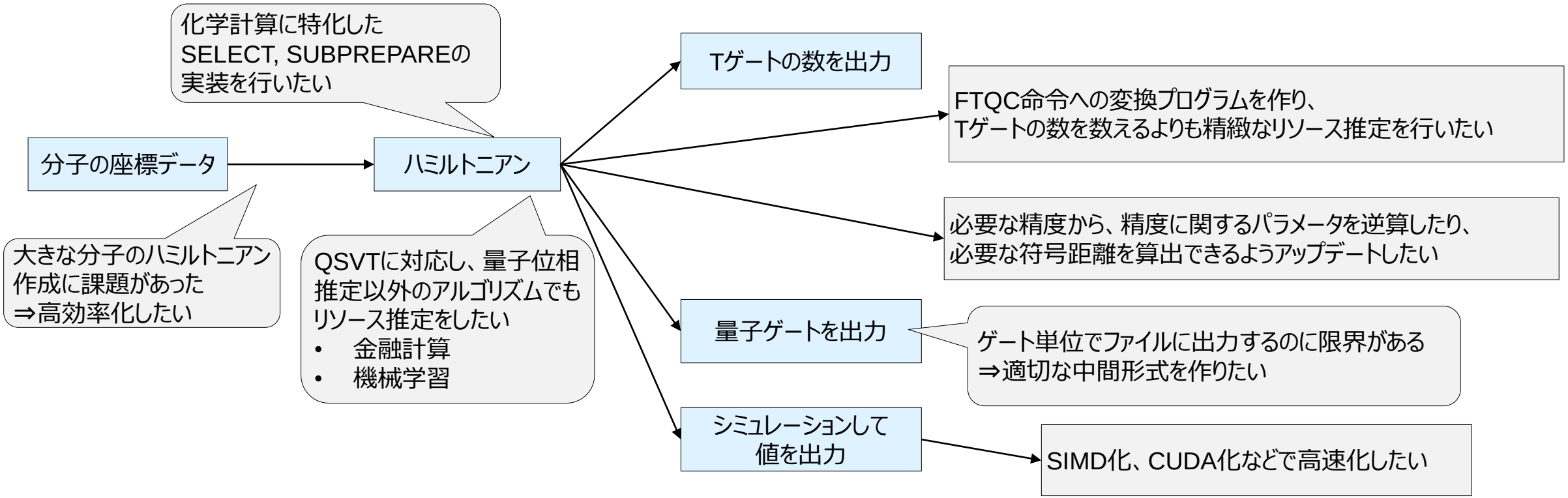
今回参照した実装では、SUBPREPAREに多数のancillaが必要で、シミュレーションする上で課題となる



Babbush, R., Gidney, C., Berry, D., Wiebe, N., McClean, J., Paler, A., Fowler, A., & Neven, H. (2018). *Phys. Rev. X, 8, 041015.*

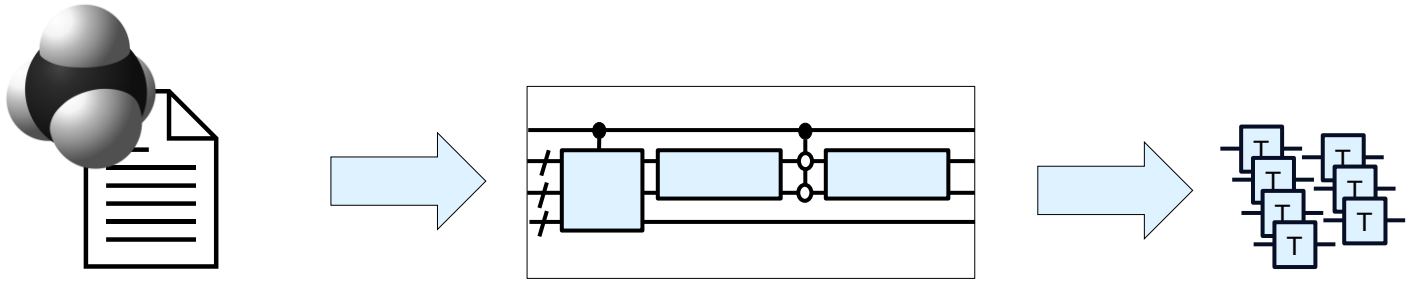
今後の展開

行いたいことはいくらでもあるが、誰にどういった目的で利用してもらえるツールを作るか?
現状: 量子化学の研究者が、FTQCの性能検証のために、Qubitizationでのリソース推定やシミュレーションに利用する想定



まとめ

- FTQCの実行時間を見積もるには、計算量のオーダーだけではなく、リソース推定が必要
- 量子化学計算を題材に、設定ファイルを書くだけで、量子ソフトウェアを合成し、Tゲートの数をカウントするソフトウェアを作成した



設定 ＋ ハミルトニアン　　量子ソフトウェア　　リソース推定

リソース推定を行うビジネス上の意義

FTQCがあれば、どの規模の問題が、どのくらいの速さで解けるのかを知ることが出来る
⇒量子優位性の得られる分野や領域がどこであるかを知ることが出来る
⇒FTQCの応用が期待できる分野で、より戦略的に経営計画、投資計画を立てることが出来る

リソース推定によって量子優位性の根拠を示すことでFTQCの開発・投資を活発化させ、
量子AIの社会実装に向けた取り組みを加速させる

Contact: qcomputer@kits.nttdata.co.jp