

Utility scale quantum computing

量子ユーティリティー授業

量子シミュレーション

2025/8/22

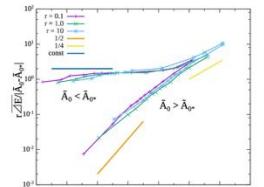
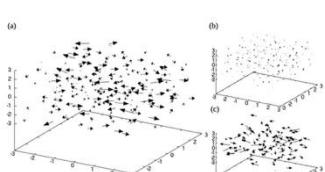
Translated and modified by 野口裕信

Created by Yukio Kawashima

野口 裕信 シニア戦略コンサルタント



IBMコンサルティング事業本部
事業戦略コンサルティング
シニアコンサルタント
IBM Quantum Ambassador



- グローバル総合系コンサルティングファーム2社、大阪大学を経て、2024年11月にIBMへ参画。
- IBMでは現在、事業企画及び研究開発部門への先進テクノロジー全般（量子コンピューティング等）の業務適用構想策定支援や設計開発現場の法規対応活動支援等、戦略コンサルティング業務に広く従事している。IBM量子アンバサダー（2025年クラス）
- 大阪大学では量子コンピューティングの産官学共創活動に従事し、量子ソフトウェアコンソーシアムを運営。40におよぶ参画機関、30大学を対象とした参加人数200名規模の教育プログラム（量子ソフトウェア勉強会）を4年間主催し、大学の共同研究パートナーの新規獲得を含む量子コミュニティの拡大を推進。
また、量子技術イノベーション拠点・产学官連携分科会の委員として、各拠点でノウハウを共有。
- グローバル総合系コンサルティングファームでは、ルール形成戦略コンサルティングサービス部門に所属し、公共（Public Sector）向けに、各国の経済安全保障ルール戦略に則したサイバーセキュリティ調達規則の策定を支援。また、民間（Private Sector）向けに、新規規制に早期かつ正確にアラインできるよう、構想策定から詳細分析・設計や代替製品を含むシステム改修案の策定まで実施し、規制対応を考慮したDX戦略の改定を支援。
- 博士課程在籍時は理論生物学を専攻：テーマ「生命システムにおける形と集団運動」
- 学歴・資格等
 - 一橋大学 経済学部 経済学科 学士課程修了（学士）
 - 東京大学大学院 総合文化研究科 相関基礎科学系 修士課程修了（修士）
 - 東京大学大学院 総合文化研究科 相関基礎科学系 博士課程修了（博士（学術））

本日のアジェンダ

1. 量子シミュレーション(ハミルトニアンシミュレーション、量子ダイナミクス)
2. ハミルトニアン(模型)
3. 量子シミュレーションのためのアルゴリズム
 1. Trotterization
 2. Randomization

量子シミュレーション(ハミルトニアンシミュレーション)

- ・ シュレディンガー方程式を解く

$$i\frac{d}{dt}|\Psi(t)\rangle = \hat{H}|\Psi(t)\rangle$$

↑
ハミルトニアン

←
波動関数

- ・ これを計算することがゴール！

$$|\Psi(t)\rangle = e^{-i\hat{H}t}|\Psi(0)\rangle$$

- ・ できるだけ正確かつ高速に数値的に問題を解く

$$|\Psi(t + \Delta t)\rangle = e^{-i\hat{H}\Delta t}|\Psi(t)\rangle \approx \left(1 - iH\Delta t - \frac{\hat{H}^2\Delta t^2}{2} + \dots\right)|\Psi(t)\rangle$$

↑
微小な時間幅

memo 1

$$\begin{aligned}\frac{d}{dt}x(t) &= ax(t) \\ x &= e^{at}x(0)\end{aligned}$$

memo 2

$$\begin{aligned}x(t + \Delta t) &= e^{a\Delta t}x(t) \\ &\approx (1 + a\Delta t + \frac{(a\Delta t)^2}{2} + \dots)x(t)\end{aligned}$$

なぜ量子コンピュータを使って量子シミュレーションを解くのか？

- 波動関数 $|\Psi(t)\rangle$ の情報をメモリに保管するには
 - 古典コンピュータではシステムサイズの指数倍のメモリを必要する
 - 量子コンピュータではシステムサイズに線形なサイズのメモリで保管できる
- 量子コンピュータの最も重要なアプリケーションと考えられている
 - 量子化学 (材料科学)
 - 高エネルギー物理

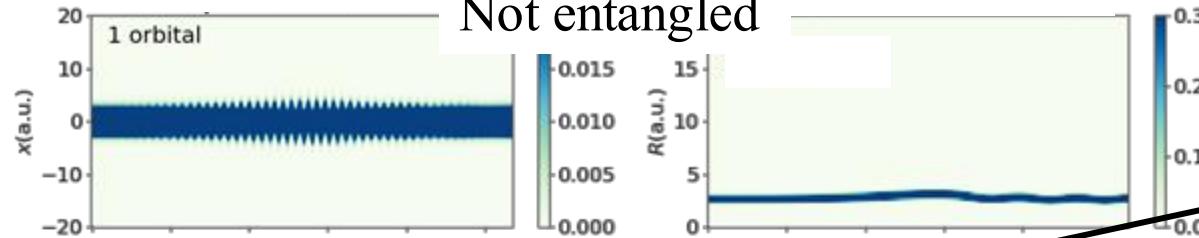
量子シミュレーションでできること: 電子と原子核のダイナミクス

電子密度 $\rho_e(x)$

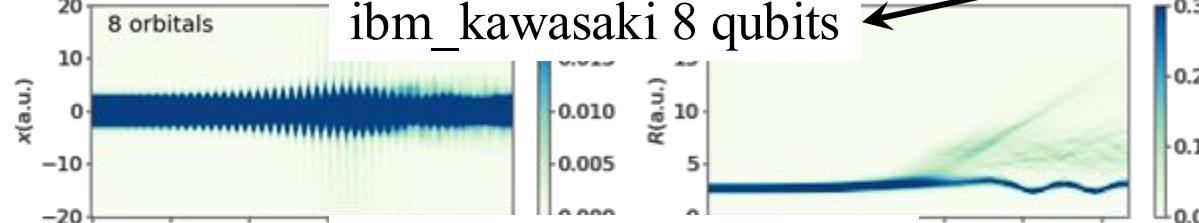
原子核の密度 $\rho_n(R)$

$$\bullet \text{---} \otimes \text{---} \bullet$$

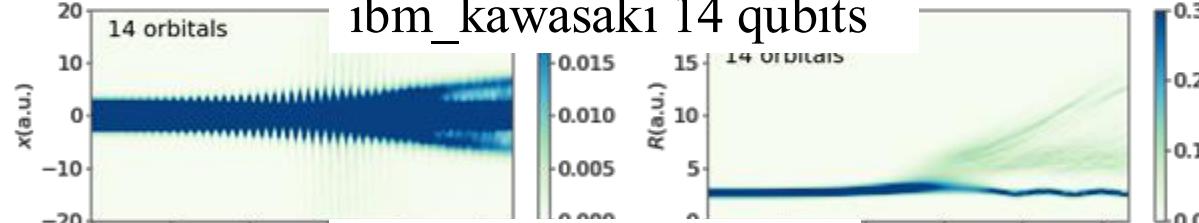
Not entangled



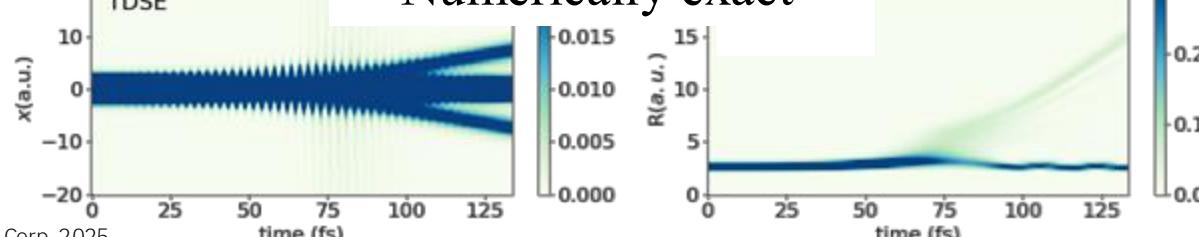
ibm_kawasaki 8 qubits



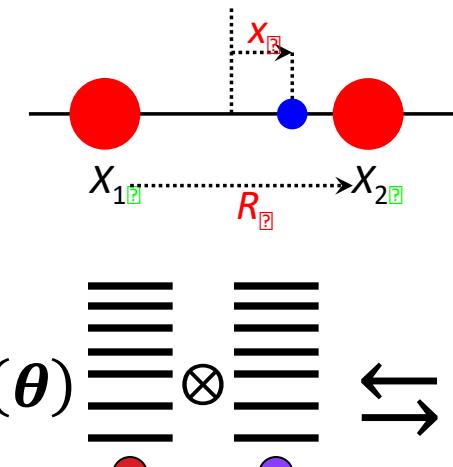
ibm_kawasaki 14 qubits



Numerically exact



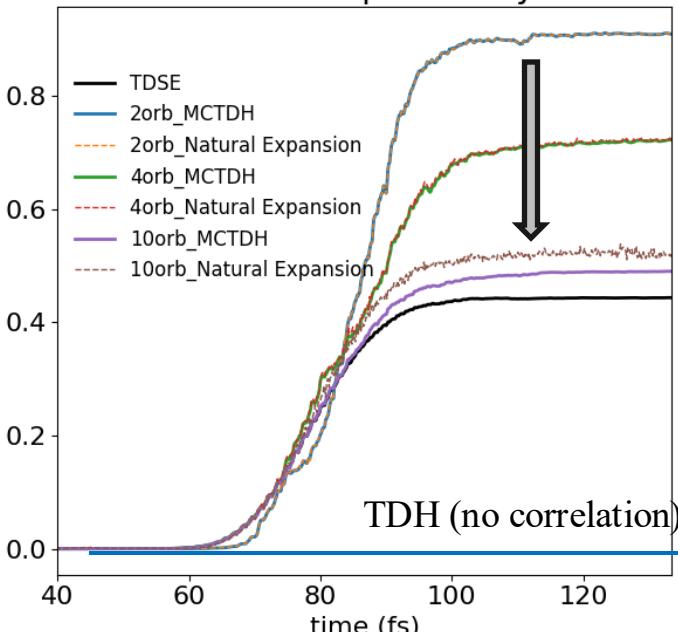
Univ of Tokyo, Professor Sato



強いレーザー場に
拘束された分子

$$U(\theta)$$

Ionization probability

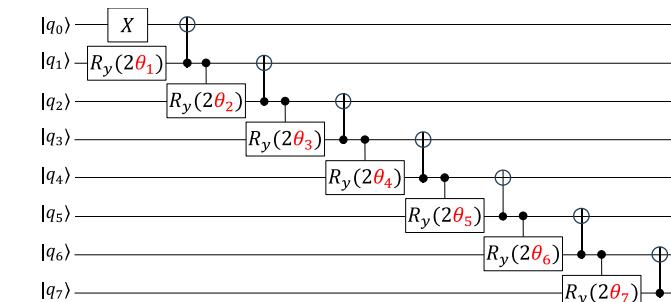


2 qubits

4 qubits

10 qubits

より多く量子
ビットを使えば
より正確に計算
可能



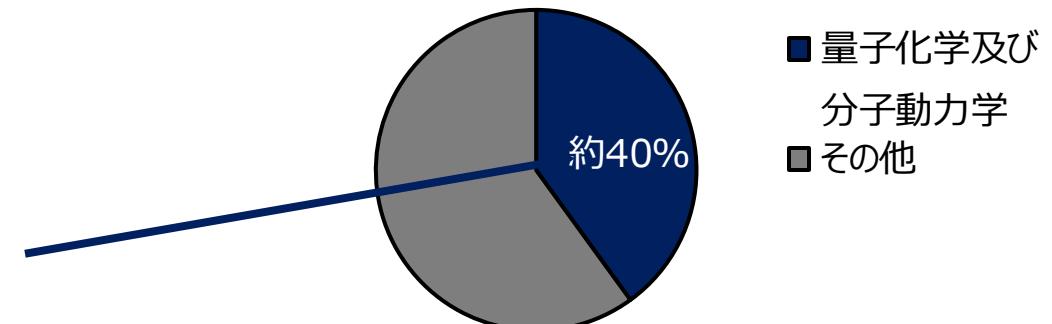
HPC利用からうかがえる量子化学への期待

年間約500億ドルに及ぶHPCの40%程度が、微小な空間・時間スケールの問題を解くために、量子化学及び動力学の計算に利用されており、問題を解くことで切り拓ける事業インパクト創出への期待がうかがえる。

計算科学の対象範囲

分野	量子化学	(古典) 分子動力学	(古典) 粗視化動力学	その他 (流体力学・構造力学等)
典型的な 空間ス ケール	ピコメー トル～ナ ノメート ル	ナノメー トル～マ イクロ メートル	マイクロ メートル ～ミリ メートル	ミリメー トル～
典型的な 時間ス ケール	フェムト 秒～ナノ 秒	ピコ秒～ ミリ秒	ナノ秒～ 秒	マイクロ 秒～

HPC資源利用割合



出所 : Sherrill et al., J. Chem. Phys. 153, 070401
(2020)

2025年のHPC市場規模が約7兆6千億円、
2030年のHPC市場規模が約11兆円

40%が量子化学利用と仮定

2025年のHPCの量子化学計算利用規模は3兆円、
2030年のHPCの量子化学計算利用規模は4兆4千億円

出所 : Mordor Intelligence: ハイパフォーマンスコンピューティング
市場規模・シェア分析 - 成長動向と予測 (2025年～2030年) 、
1米ドル=150円として試算

ハミルトニアンの一般論

- 量子系のハミルトニアンは運動エネルギーとポテンシャルエネルギーを足した全エネルギーを表す演算子

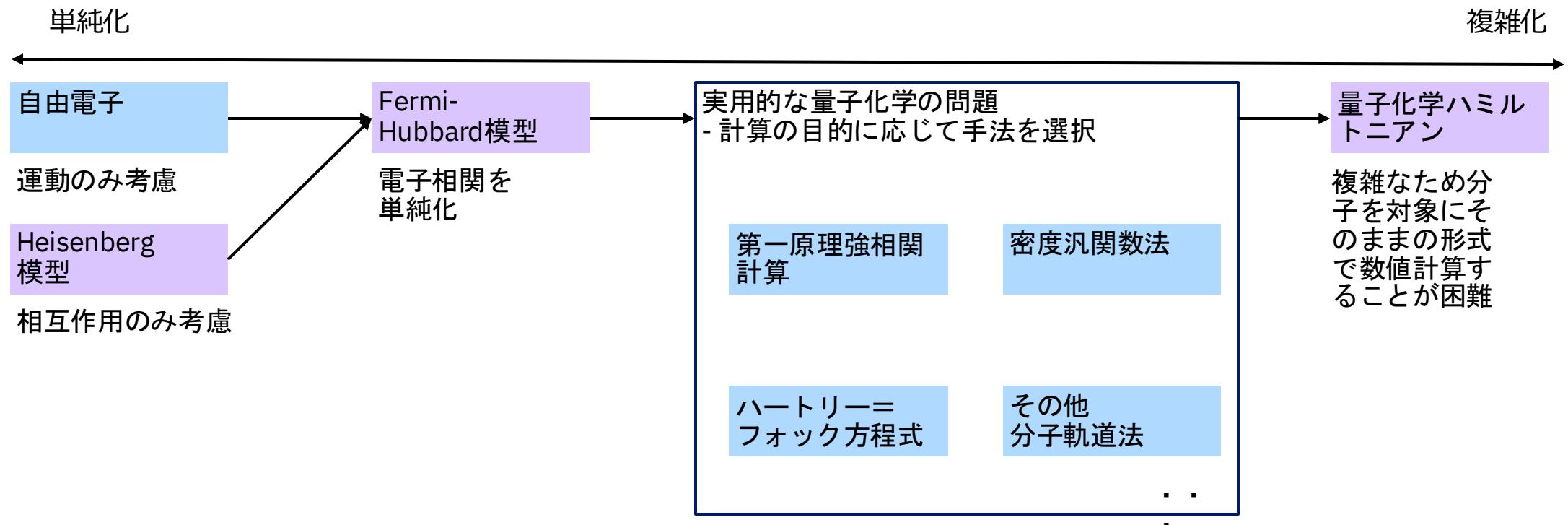
$$\hat{H} = \hat{T} + \hat{V}$$

- ハミルトニアンには時間依存のものと非依存のものとがあるが、今回の発表では非依存のもののみ扱う
- 多くの分野での重要例になっている
 - 量子化学 (材料科学)
 - 凝縮系物理
 - 高エネルギー物理

量子化学の模型（＝ハミルトニアン）と実用問題の関係

古典力学等空間スケールの大きな問題は基礎方程式を起点に実用上重要な問題に取り組むが、微小な空間スケールを対象とする量子化学は計算対象となる電子の数が膨大となるため、単純化した模型を起点に実用上重要な問題に取り組む。

本発表で説明



(スピン) ハミルトニアン

磁性の系を研究するためのスピン系の格子模型

- n -ベクトル模型

- $n=1$: Ising模型

スピン間相互作用 外（磁）場

$$H = - \sum_{\langle i,j \rangle} J \sigma_{Z_i} \sigma_{Z_j} - \sum_i h_i \sigma_{X_i}$$



- $n=2$: XY模型

$$H = - \sum_{\langle i,j \rangle} J \left(\sigma_{X_i} \sigma_{X_j} + \sigma_{Y_i} \sigma_{Y_j} \right) - \sum_i h_i \sigma_{Z_i}$$

- $n=3$: Heisenberg模型

$$H = - \sum_{\langle i,j \rangle} \left(J_X \sigma_{X_i} \sigma_{X_j} + J_Y \sigma_{Y_i} \sigma_{Y_j} + J_Z \sigma_{Z_i} \sigma_{Z_j} \right) - \sum_i h_i \sigma_{Z_i}$$

(フェルミオンの)ハミルトニアン

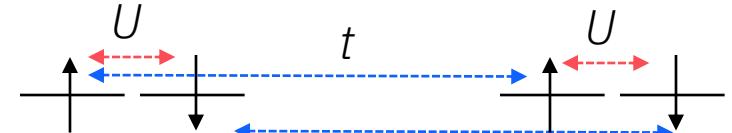
➤ Hubbard模型

電導性と絶縁性を表す系

$$H = -t \sum_{i,\sigma} \left(\hat{c}_{i,\sigma}^\dagger \hat{c}_{i+1,\sigma} + \hat{c}_{i+1,\sigma}^\dagger \hat{c}_{i,\sigma} \right) + U \sum_i \hat{n}_{i,\uparrow} \hat{n}_{i,\downarrow}$$

$$\hat{n}_{i,\sigma} = \hat{c}_{i,\sigma}^\dagger \hat{c}_{i,\sigma}$$

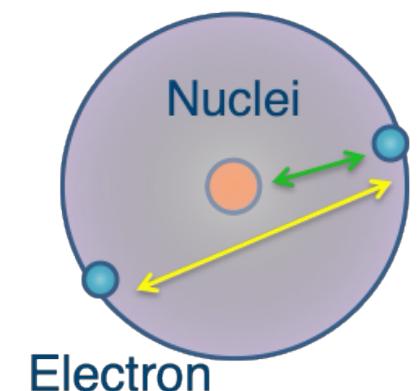
生成演算子 消滅演算子



➤ 量子化学ハミルトニアン

$$\hat{H}_{ele}(\mathbf{r}; \mathbf{R}) = - \sum_i^{N_{ele}} \frac{1}{2} \nabla_i^2 - \sum_A^{N_{nuc}} \sum_i^{N_{ele}} \frac{Z_A}{r_{iA}} + \sum_{i>j}^{N_{ele}} \frac{1}{r_{ij}}$$

電子の運動
エネルギー 電子-原子核
間の引力相互
作用 電子-電子間の
反発相互作用



複雑さ、計算資源

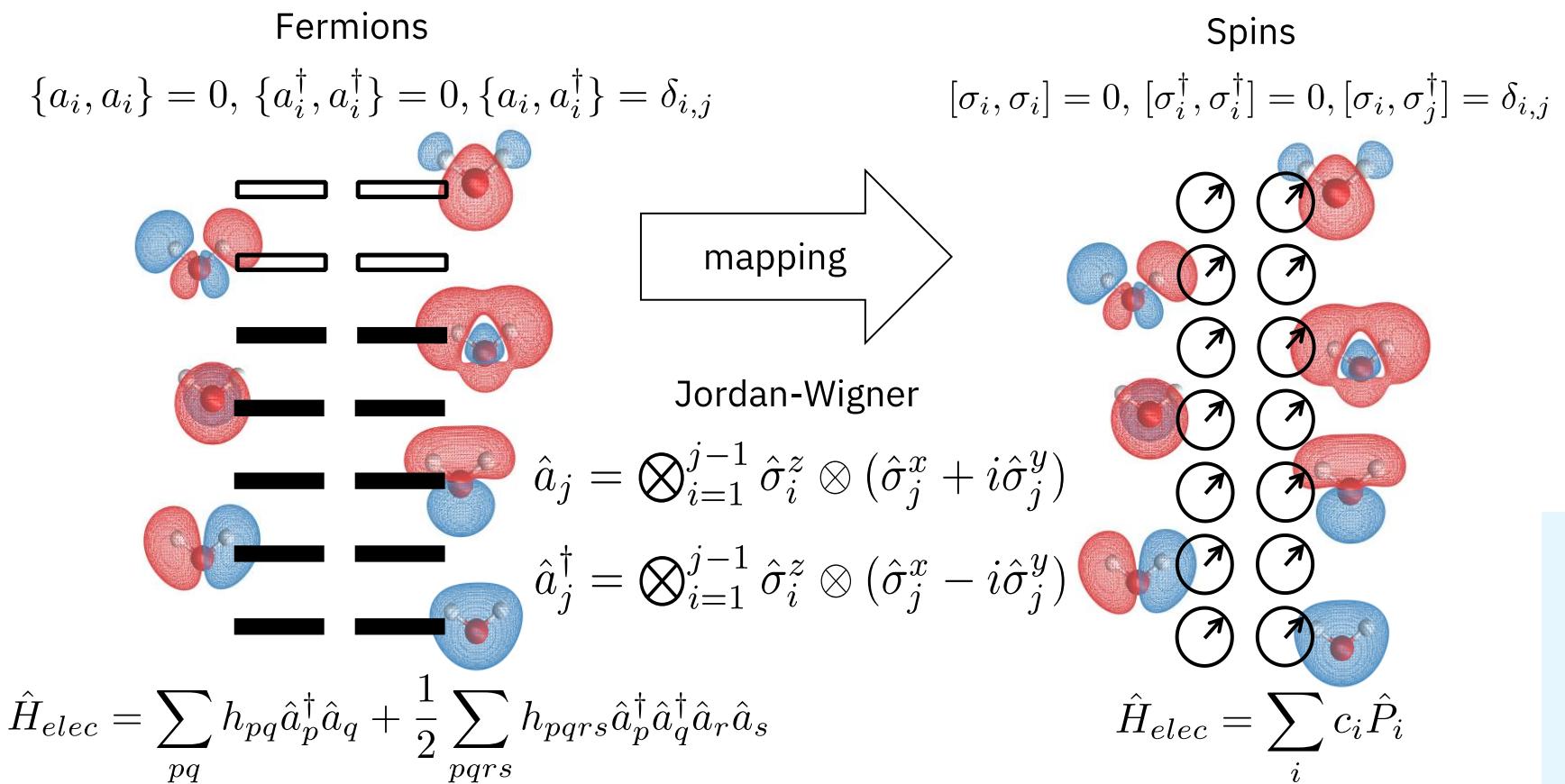
ハミルトニアンにマッピングする

$$H = -t \sum_{i,\sigma} \left(\hat{c}_{i,\sigma}^\dagger \hat{c}_{i+1,\sigma} + \hat{c}_{i+1,\sigma}^\dagger \hat{c}_{i,\sigma} \right) + U \sum_i \hat{n}_{i,\uparrow} \hat{n}_{i,\downarrow}$$

第二量子化されたハミルトニアンを量子ビットにマッピングする:

$$\hat{n}_{i,\sigma} = \hat{c}_{i,\sigma}^\dagger \hat{c}_{i,\sigma}$$

Hubbard量子化学



Jordan-Wignerマッピング（変換）

フェルミオン
ハミルトニアン

$$\hat{H}_M = \sum_{pq} h_{pq} a_p^\dagger a_q + \sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

生成演算子 $a_p^\dagger = \frac{1}{2} (X_p - iY_p) \otimes Z_{p-1} \otimes \cdots \otimes Z_1$

消滅演算子 $a_q = \frac{1}{2} (X_q + iY_q) \otimes Z_{q-1} \otimes \cdots \otimes Z_1$

Jordan-Wigner
マッピング（変換）

memo 略記

$$\sigma_{X_i}, \sigma_{Y_i}, \sigma_{Z_i} = X_i, Y_i, Z_i$$

水素分子の場合、ボンド長=0.735Å、基底関数系をSTO-3G、4スピン軌道で36項になる

$$H_f = -1.26a_0^\dagger a_0 - 0.47a_1^\dagger a_1 - 1.26a_2^\dagger a_2 - 0.47a_3^\dagger a_3$$

$$+ 0.34a_0^\dagger a_0^\dagger a_0 a_0 + 0.33a_0^\dagger a_1^\dagger a_1 a_0 + 0.34a_0^\dagger a_2^\dagger a_2 a_0 + 0.33a_0^\dagger a_3^\dagger a_3 a_0 + \cdots$$

$$+ 0.09a_0^\dagger a_2^\dagger a_3 a_1 + \cdots$$

1体項のマッピング（変換）

右の関係式を使う $X_i^2 = Y_i^2 = Z_i^2 = I,$
 $X_i Y_i = -Y_i X_i = iZ_i,$
 $Y_i Z_i = -Z_i Y_i = iX_i,$
 $Z_i X_i = -X_i Z_i = iY_i$

計算例

$$\begin{aligned} a_3^\dagger a_3 &= \frac{1}{2} (X_3 - iY_3) \otimes Z_2 Z_1 Z_0 \times \frac{1}{2} (X_3 + iY_3) \otimes Z_2 Z_1 Z_0 \\ &= \frac{1}{4} (X_3 Z_2 Z_1 Z_0 - iY_3 Z_2 Z_1 Z_0) \times (X_3 Z_2 Z_1 Z_0 + iY_3 Z_2 Z_1 Z_0) = \frac{1}{4} (I + I + iX_3 Y_3 - iY_3 X_3) \\ &= \frac{1}{2} (I - Z_3) \end{aligned}$$

memo Jordan–Wignerマッピング（変換）

生成演算子 $a_p^\dagger = \frac{1}{2} (X_p - iY_p) \otimes Z_{p-1} \otimes \cdots \otimes Z_1$

消滅演算子 $a_q = \frac{1}{2} (X_q + iY_q) \otimes Z_{q-1} \otimes \cdots \otimes Z_1$

2体項のマッピング（変換）

右の関係式を使う

$$\begin{aligned} X_i^2 &= Y_i^2 = Z_i^2 = I, \\ X_i Y_i &= -Y_i X_i = i Z_i, \\ Y_i Z_i &= -Z_i Y_i = i X_i, \\ Z_i X_i &= -X_i Z_i = i Y_i \end{aligned}$$

計算例

$$\begin{aligned} a_0^\dagger | a_2^\dagger | a_3 | a_1 &= \frac{1}{2} (X_0 - iY_0) \times \frac{1}{2} (X_2 - iY_2) \otimes Z_1 Z_0 \times \frac{1}{2} (X_3 + iY_3) \otimes Z_2 Z_1 Z_0 \times \frac{1}{2} (X_1 + iY_1) \otimes Z_0 \\ &= \frac{1}{16} (X_3 + iY_3) \otimes (X_2 - iY_2) Z_2 \otimes (X_1 + iY_1) Z_1 Z_1 \otimes (X_0 - iY_0) Z_0 Z_0 Z_0 && \text{同じインデックスでまとめる} \\ &= \frac{1}{16} (\boxed{X_3} + iY_3) \otimes (\boxed{-iY_2} + X_2) \otimes (\boxed{X_1} + iY_1) \otimes (\boxed{-iY_0} + X_0) && \text{同じインデックス内で計算} \\ &= \frac{1}{16} [\boxed{-X_3 Y_2 X_1 Y_0} - iX_3 Y_2 Y_1 Y_0 - iY_3 Y_2 X_1 Y_0 + Y_3 Y_2 Y_1 Y_0 - iX_3 X_2 X_1 Y_0 + X_3 X_2 Y_1 Y_0 + Y_3 X_2 X_1 Y_0 + iY_3 X_2 Y_1 Y_0 \\ &\quad - iX_3 Y_2 X_1 X_0 + X_3 Y_2 Y_1 X_0 + Y_3 Y_2 X_1 X_0 + iY_3 Y_2 Y_1 X_0 + X_3 X_2 X_1 X_0 + iX_3 X_2 Y_1 X_0 + iY_3 X_2 X_1 X_0 - Y_3 X_2 Y_1 X_0] \end{aligned}$$

memo Jordan–Wignerマッピング（変換）

生成演算子 $a_p^\dagger = \frac{1}{2} (X_p - iY_p) \otimes Z_{p-1} \otimes \cdots \otimes Z_1$

消滅演算子 $a_q = \frac{1}{2} (X_q + iY_q) \otimes Z_{q-1} \otimes \cdots \otimes Z_1$

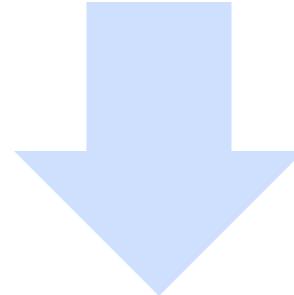
ハミルトニアンへのマッピング

フェルミオンのハミルトニアン

$$H_f = -1.26a_0^\dagger a_0 - 0.47a_1^\dagger a_1 - 1.26a_2^\dagger a_2 - 0.47a_3^\dagger a_3$$

$$+ 0.34a_0^\dagger a_0^\dagger a_0 a_0 + 0.33a_0^\dagger a_1^\dagger a_1 a_0 + 0.34a_0^\dagger a_2^\dagger a_2 a_0 + 0.33a_0^\dagger a_3^\dagger a_3 a_0 + \dots$$

$$+ 0.09a_0^\dagger a_2^\dagger a_3 a_1 + \dots$$



スピノンのハミルトニアン

$$H_q = -0.81 + 0.17(Z_0 + Z_2) - 0.23(Z_1 + Z_3) + 0.12(Z_1Z_0 + Z_3Z_2) + 0.17Z_0Z_2 + 0.17Z_1Z_3$$

$$+ 0.17Z_1Z_2 + 0.17Z_0Z_3 + 0.05(Y_3Y_2Y_1Y_0 + X_3X_2X_1X_0 + Y_3Y_2X_1X_0 + X_3X_2Y_1Y_0)$$

memo 1体・2体の変換公式

$$h_{pq}a_p^\dagger a_q = \frac{1}{4}h_{pq}(X_p - iY_p) \otimes Z_{p-1} \otimes \dots \otimes Z_{q+1} \otimes (X_q + iY_q)$$

$$h_{pqrs}a_p^\dagger a_q^\dagger a_r a_s = \frac{1}{16}h_{pqrs}(X_p - iY_p) \otimes Z_{p-1} \otimes \dots \otimes Z_{q+1} \otimes (X_q - iY_q) \\ (X_r + iY_r) \otimes Z_{r-1} \otimes \dots \otimes Z_{s+1} \otimes (X_s - iY_s)$$

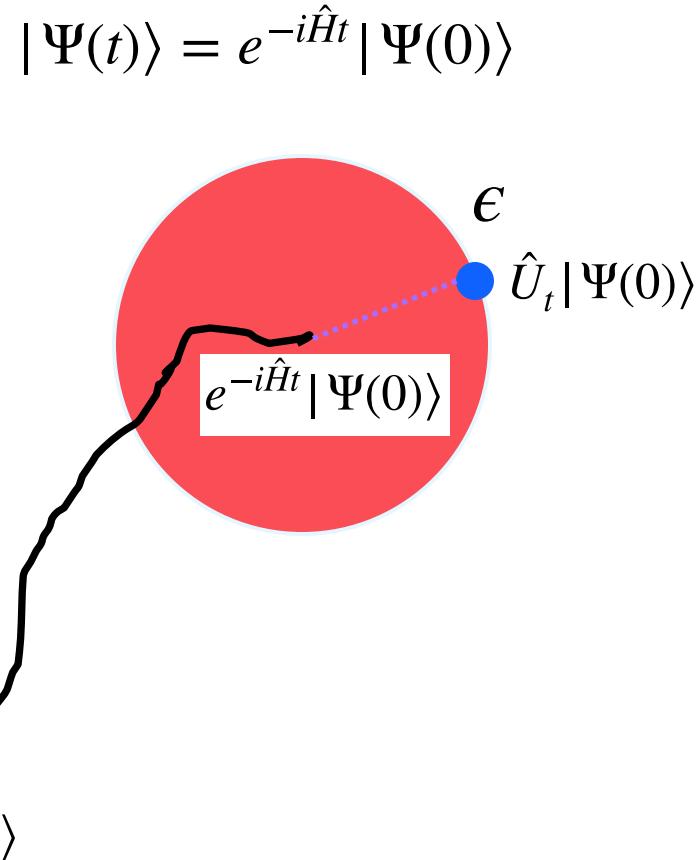
量子シュミレーションのアルゴリズム

ハミルトニアンが既知でも, $e^{-i\hat{H}t}$ の計算方法は自明ではない
正確に計算することは難しい
次の式を満たすUを実装する $\|\hat{U}|\Psi\rangle - e^{-i\hat{H}t}|\Psi\rangle\| \leq \epsilon$

- 演算子のノルム $\hat{A} : V \rightarrow W$
 - i. ある実数 $c > 0$ が存在して任意のベクトル v に対して $\|\hat{A}v\| \leq c\|v\|$
 - ii. 次の式を満たすものが演算子のノルム

$$\|\hat{A}\| := \min\{c \geq 0 : \|\hat{A}v\| \leq c\|v\| \text{ for all } v \in V\}$$

- 性質:
 - $\|\hat{A}\| \geq 0$ and $\|\hat{A}\| = 0$ if and only if $\hat{A} = 0$
 - $\|a\hat{A}\| = |a|\|\hat{A}\|$
 - $\|\hat{A} + \hat{B}\| \leq \|\hat{A}\| + \|\hat{B}\|$



量子シュミレーションのアルゴリズム

ハミルトニアンが既知でも, $e^{-i\hat{H}t}$ の計算方法は自明ではない

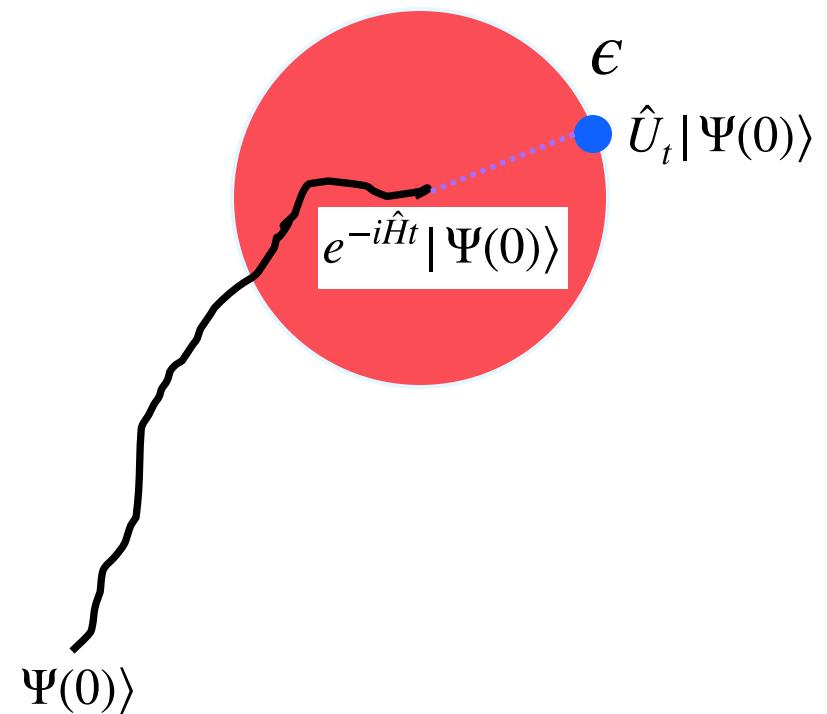
正確に計算することは難しい

次の式を満たすUを実装する

$$\|\hat{U}|\Psi\rangle - e^{-i\hat{H}t}|\Psi\rangle\| \leq \epsilon$$

- 効率的に計算するいくつかの戦略がある
 - 誤差を小さく保つ
 - 回路深さを浅く保つ
- 方法
 - Trotter公式
 - Randomization (QDrift)
 - ”ポストTrotter”
 - ユニタリー演算子の線形組み合わせ
 - Qubitization (量子シグナルプロセッシング)

$$|\Psi(t)\rangle = e^{-i\hat{H}t}|\Psi(0)\rangle$$



Trotterization

仮定：ハミルトニアンが k -local (P は高々“ k ” 個のPauliストリングに作用する)

$$\hat{H} = \sum_{i=1}^L a_i P_i$$

2項からなるハミルトニアンを考える

$$\hat{H} = \hat{H}_1 + \hat{H}_2$$

Lie Product公式：

$$e^{-it(H_1+H_2)} = \lim_{n \rightarrow \infty} \left(e^{-iH_1 \frac{t}{n}} e^{-iH_2 \frac{t}{n}} \right)^n$$

“ n ”を有限に取ると次式が成り立つ

$$e^{-i(\hat{H}_1+\hat{H}_2)\Delta t} = e^{-i\hat{H}_1\Delta t} e^{-i\hat{H}_2\Delta t}$$

ただし H_1 と H_2 が可換な場合に限る（大体非可換...）

memo Baker-Campbell-Hausdorff 公式

$e^X e^Y = e^Z$ を満たす Z は

$$Z = A + B + \frac{1}{2}[A, B] + \frac{1}{12}[A - B, [A, B]] + \dots$$

という形式で書ける。

$e^{X/n} e^{Y/n} = e^Z$ を満たす Z は

$$Z = A/n + B/n + \frac{1}{2n^2}[A, B] + \frac{1}{12n^3}[A - B, [A, B]] + \dots$$

という形式で書ける。

Trotterizationの1次の誤差

$$\hat{U}_{\text{exact}} = e^{-i(\hat{H}_1 + \hat{H}_2)\Delta t}$$

$$\hat{U}_{\text{exact}_2} = \mathbb{I} + (-i\Delta t)(\hat{H}_1 + \hat{H}_2) + \frac{(-i\Delta t)^2}{2} (\hat{H}_1^2 + \hat{H}_2^2 + \hat{H}_1\hat{H}_2 + \hat{H}_2\hat{H}_1)$$

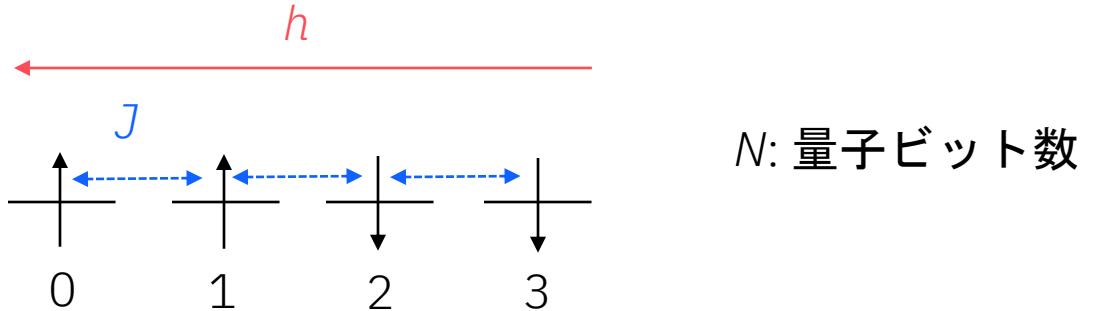
$$\hat{U}_{\text{trotter}} = e^{-i\hat{H}_1\Delta t} e^{-i\hat{H}_2\Delta t}$$

$$\begin{aligned}\hat{U}_{\text{trotter}_2} &= \left[\mathbb{I} + (-i\Delta t)\hat{H}_1 + \frac{(-i\Delta t)^2}{2} (\hat{H}_1^2) \right] \left[\mathbb{I} + (-i\Delta t)\hat{H}_2 + \frac{(-i\Delta t)^2}{2} (\hat{H}_2^2) \right] \\ &\approx \mathbb{I} + (-i\Delta t)(\hat{H}_1 + \hat{H}_2) + \frac{(-i\Delta t)^2}{2} (\hat{H}_1^2 + \hat{H}_2^2 + 2\hat{H}_1\hat{H}_2)\end{aligned}$$

$$\begin{aligned}\|\hat{U}_{\text{exact}_2} - \hat{U}_{\text{trotter}_2}\| &= \left\| \frac{(-i\Delta t)^2}{2} (\hat{H}_2\hat{H}_1 - \hat{H}_1\hat{H}_2) + O(\Delta t^3) \right\| \\ &\leq \frac{1}{2} \|[\hat{H}_2, \hat{H}_1]\| \Delta t^2 + \|O(\Delta t^3)\| \quad (\text{演算子ノルムの三角不等式})\end{aligned}$$

例: 横磁場Ising model

$$H = - \sum_{\langle i,j \rangle}^{N-1} J \sigma_{Z_i} \sigma_{Z_j} - \sum_i^N h_i \sigma_{X_i}$$



$$e^{-i\hat{H}\Delta t} = e^{-i\Delta t(-\sum_{i,j}^N J \sigma_{Z_i} \sigma_{Z_j} - \sum_i^N h_i \sigma_{X_i})} \approx e^{-i\Delta t(-\sum_{i,j}^N J \sigma_{Z_i} \sigma_{Z_j})} e^{-i\Delta t(-\sum_i^N h_i \sigma_{X_i})}$$

$$R_{ZZ}(-2J\Delta t)$$

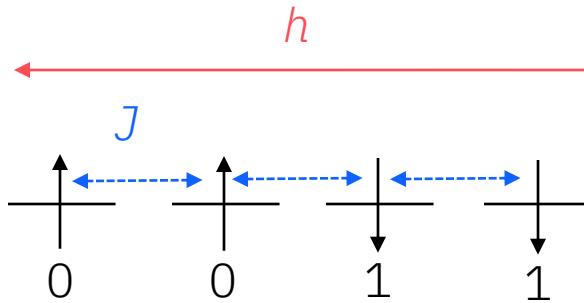
$$R_X(-2h\Delta t)$$

$$R_{ZZ}(\theta) = e^{-i\frac{\theta}{2}\sigma_Z\sigma_Z} = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & e^{i\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & e^{i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} \end{pmatrix}$$

$$R_X(\theta) = e^{-i\frac{\theta}{2}\sigma_X} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

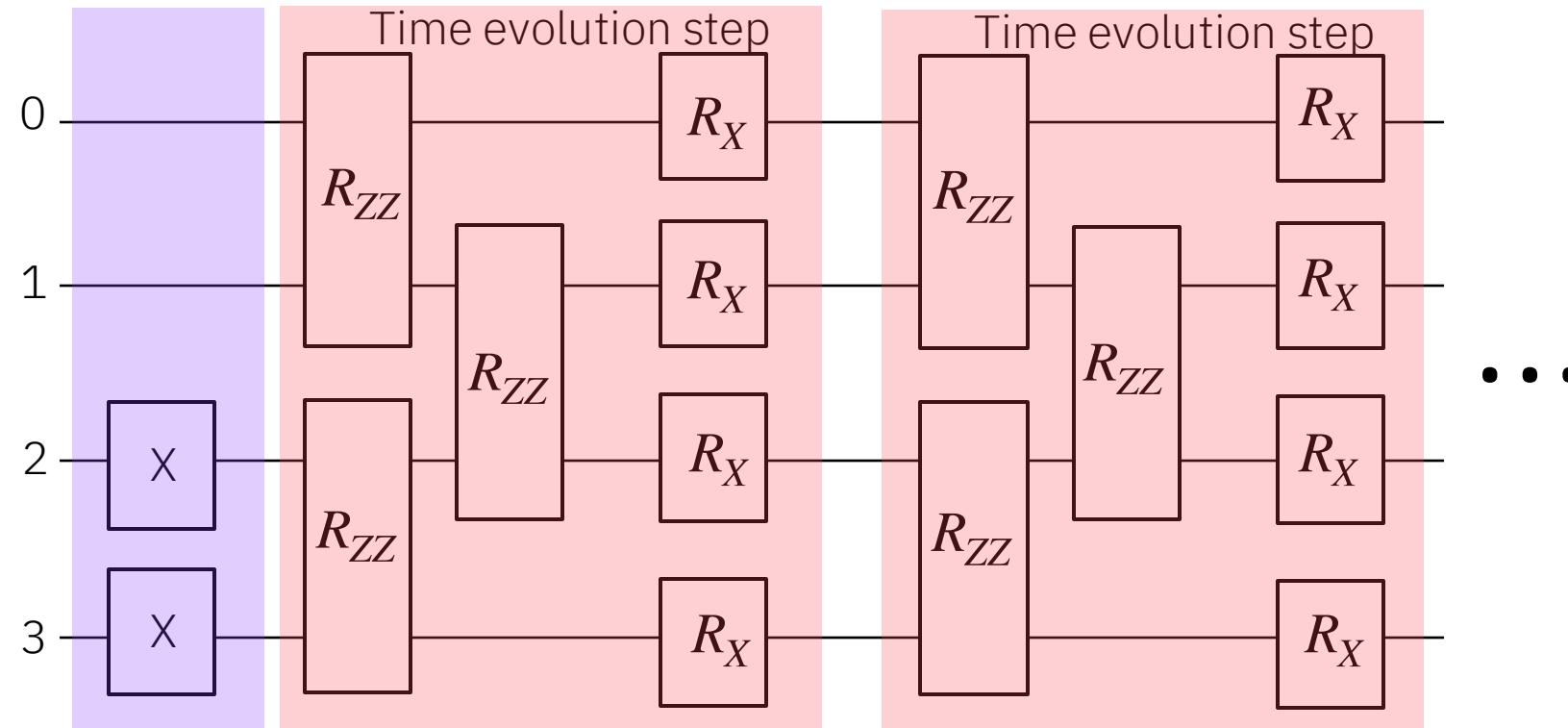
例: 横磁場Ising model

$$H = - \sum_{\langle i,j \rangle}^{N-1} J \sigma_{Z_i} \sigma_{Z_j} - \sum_i^N h_i \sigma_{X_i}$$



0: up spin
1: down spin

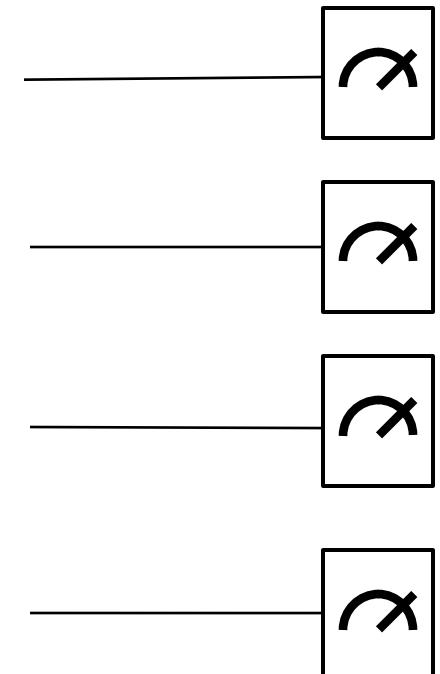
Bit strings
 $|0011\rangle$
 $|1100\rangle$



状態準備

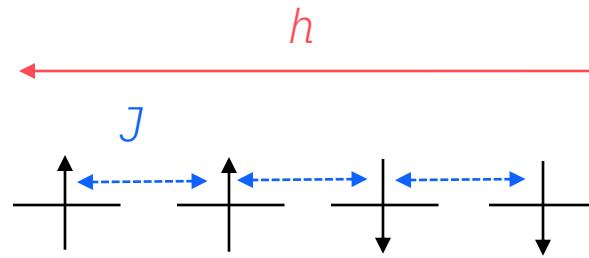
繰り返すと時間tでの波動関数を得る

$$|\Psi(t)\rangle = e^{-i\hat{H}t} |\Psi(0)\rangle$$



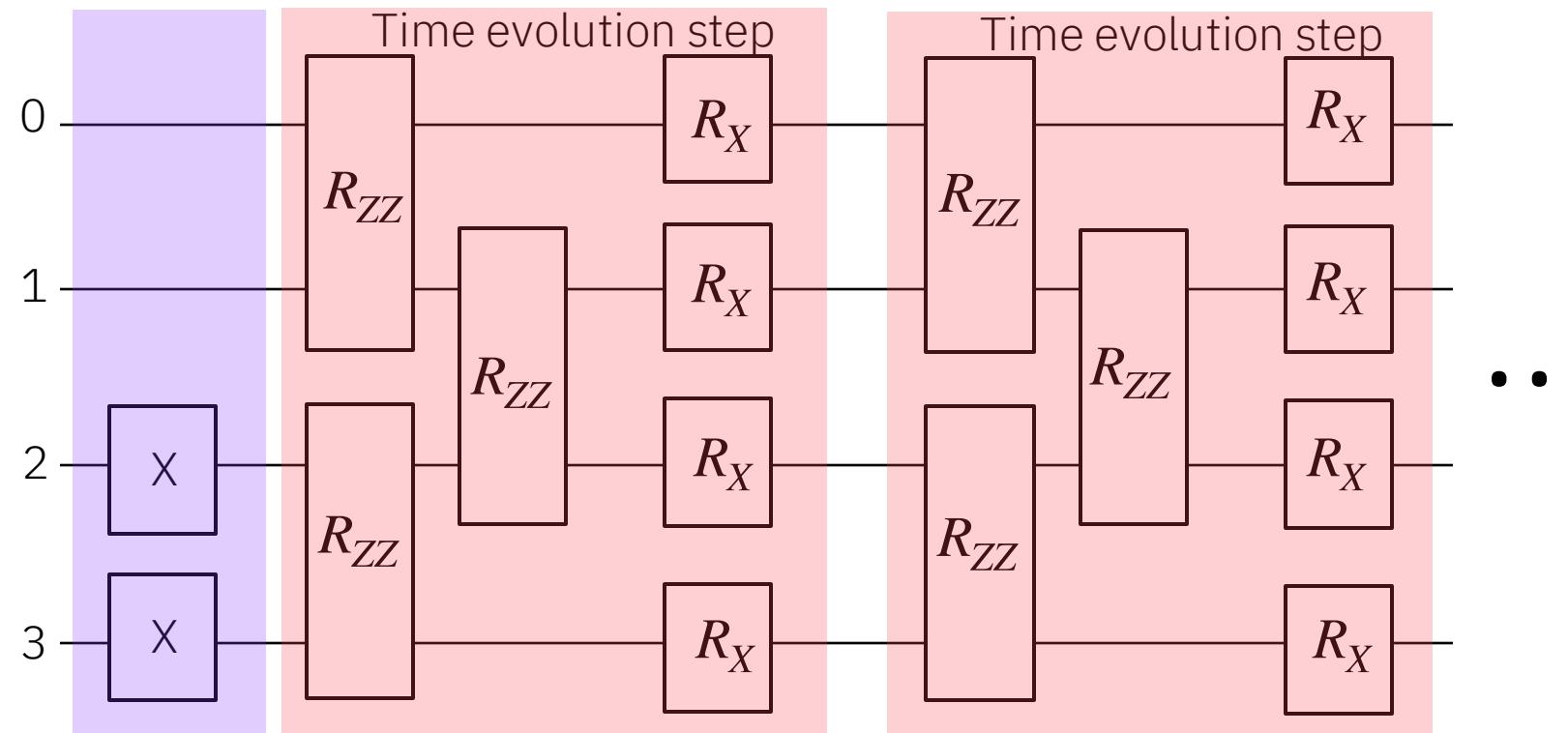
例: 横磁場Ising model

$$H = - \sum_{\langle i,j \rangle}^{N-1} J \sigma_{Z_i} \sigma_{Z_j} - \sum_i^N h_i \sigma_{X_i}$$



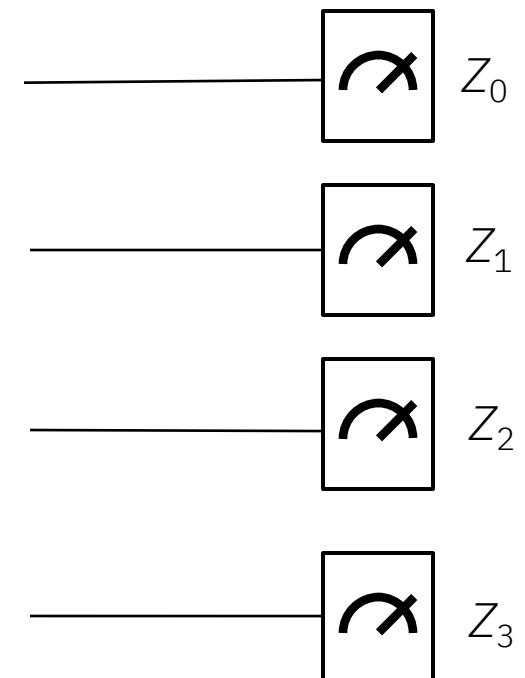
磁化

$$\sum_i^N Z_i / N$$



状態準備

オブザーバブルの期待値を測定する



Suzuki-Trotter公式(2次のオーダー)

ハミルトニアンの一般形式

$$\hat{H} = \sum_{i=1}^L a_i P_i$$

2次のオーダーのSuzuki-Trotter公式

$$U_{ST2} = \prod_{j=1}^L e^{-ia_j P_j \frac{t}{2}} \prod_{j'=L}^1 e^{-ia_{j'} P_{j'} \frac{t}{2}}$$

ハミルトニアンが2項からなる単純な場合

$$\hat{H} = \hat{H}_1 + \hat{H}_2$$

$$\hat{U}_{ST2} = e^{-i\hat{H}_1 \frac{\Delta t}{2}} e^{-i\hat{H}_2 \Delta t} e^{-i\hat{H}_1 \frac{\Delta t}{2}}$$

Suzuki-Trotter公式(2次のオーダー)

$$\hat{U}_{\text{exact}} = e^{-i(\hat{H}_1 + \hat{H}_2)\Delta t}$$

厳密な演算子の3次までの泰勒展開

$$\hat{U}_{\text{exact}_3} = \mathbb{I} + (-i\Delta t)(\hat{H}_1 + \hat{H}_2) + \frac{(-i\Delta t)^2}{2} (\hat{H}_1 + \hat{H}_2)^2 + \frac{(-i\Delta t)^3}{6} (\hat{H}_1 + \hat{H}_2)^3$$

2次のSuzuki-Trotter公式の誤差

$$\|\hat{U}_{\text{exact}_3} - \hat{U}_{\text{ST2}_3}\| \leq \frac{1}{24} \|\hat{H}_2^2 \hat{H}_1 + \hat{H}_1 \hat{H}_2^2 + \hat{H}_1 \hat{H}_2 \hat{H}_1 + \hat{H}_2 \hat{H}_1 \hat{H}_2\| \Delta t^3$$

$$\hat{U}_{\text{ST2}} = e^{-i\hat{H}_1 \frac{\Delta t}{2}} e^{-i\hat{H}_2 \Delta t} e^{-i\hat{H}_1 \frac{\Delta t}{2}}$$

Suzuki-Trotter2次公式を使った演算子の3次の泰勒展開

$$\hat{U}_{\text{ST2}_3} = \left[\mathbb{I} + (-i\Delta t/2)\hat{H}_1 + \frac{(-i\Delta t/2)^2}{2} (\hat{H}_1^2) + \frac{(-i\Delta t/2)^3}{6} (\hat{H}_1^3) \right]$$

$$\left[\mathbb{I} + (-i\Delta t)\hat{H}_2 + \frac{(-i\Delta t)^2}{2} (\hat{H}_2^2) + \frac{(-i\Delta t)^3}{6} (\hat{H}_2^3) \right]$$

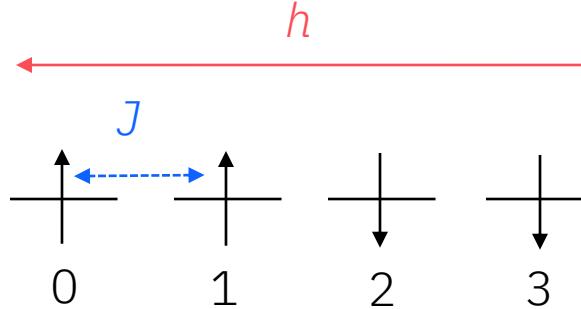
$$\left[\mathbb{I} + (-i\Delta t/2)\hat{H}_1 + \frac{(-i\Delta t/2)^2}{2} (\hat{H}_1^2) + \frac{(-i\Delta t/2)^3}{6} (\hat{H}_1^3) \right]$$

$$\hat{U}_{\text{ST2}_3} \approx \mathbb{I} + (-i\Delta t/2)(\hat{H}_1 + \hat{H}_2) + \frac{(-i\Delta t/2)^2}{2} (\hat{H}_1 + \hat{H}_2)^2$$

$$+ \frac{(-i\Delta t/2)^3}{6} \left[\hat{H}_1^3 + \frac{3}{2}(\hat{H}_1 \hat{H}_2^2 + \hat{H}_2^2 \hat{H}_1 + \hat{H}_1 \hat{H}_2 \hat{H}_1) + \frac{3}{2}(\hat{H}_2 \hat{H}_1^2 + \hat{H}_1^2 \hat{H}_2 + \hat{H}_2^3) \right]$$

例: 橫磁場Ising model

$$H = - \sum_{\langle i,j \rangle}^{N-1} J \sigma_{Z_i} \sigma_{Z_j} - \sum_i^N h_i \sigma_{X_i}$$



$$e^{-i\hat{H}\Delta t} = e^{-i\Delta t(-\sum_{i,j}^N J \sigma_{Z_i} \sigma_{Z_j} - \sum_i^N h_i \sigma_{X_i})}$$

$$\approx e^{-i\frac{\Delta t}{2}(-J \sigma_{Z_0} \sigma_{Z_1})} e^{-i\frac{\Delta t}{2}(-J \sigma_{Z_1} \sigma_{Z_2})} e^{-i\frac{\Delta t}{2}(-J \sigma_{Z_2} \sigma_{Z_3})} e^{-i\frac{\Delta t}{2}(-h \sigma_{X_0})} e^{-i\frac{\Delta t}{2}(-h \sigma_{X_1})} e^{-i\frac{\Delta t}{2}(-h \sigma_{X_2})}$$

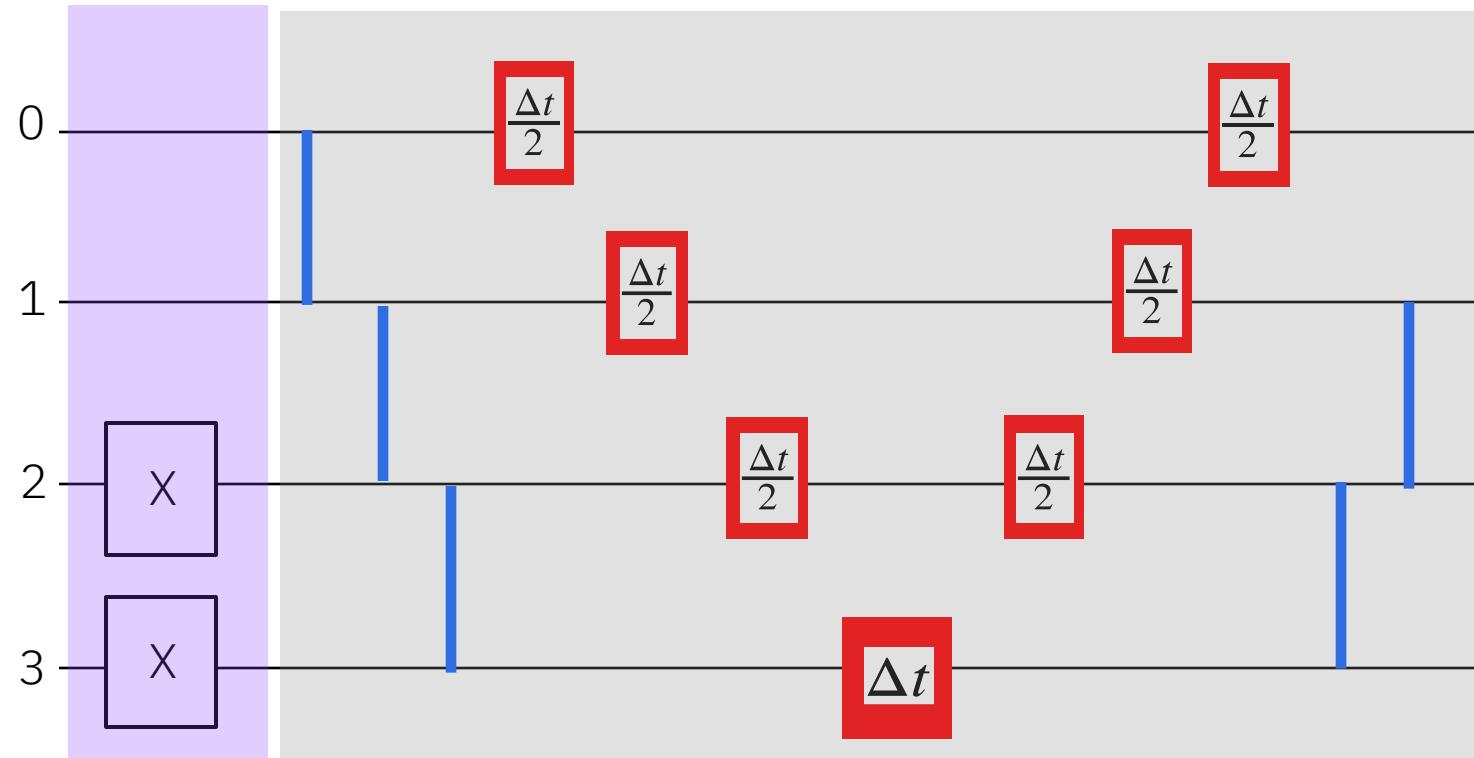
$$e^{-i\Delta t(-h \sigma_{X_3})}$$

$$e^{-i\frac{\Delta t}{2}(-h \sigma_{X_2})} e^{-i\frac{\Delta t}{2}(-h \sigma_{X_1})} e^{-i\frac{\Delta t}{2}(-h \sigma_{X_0})} e^{-i\frac{\Delta t}{2}(-J \sigma_{Z_2} \sigma_{Z_3})} e^{-i\frac{\Delta t}{2}(-J \sigma_{Z_1} \sigma_{Z_2})} e^{-i\frac{\Delta t}{2}(-J \sigma_{Z_0} \sigma_{Z_1})}$$

例: 横磁場Ising model

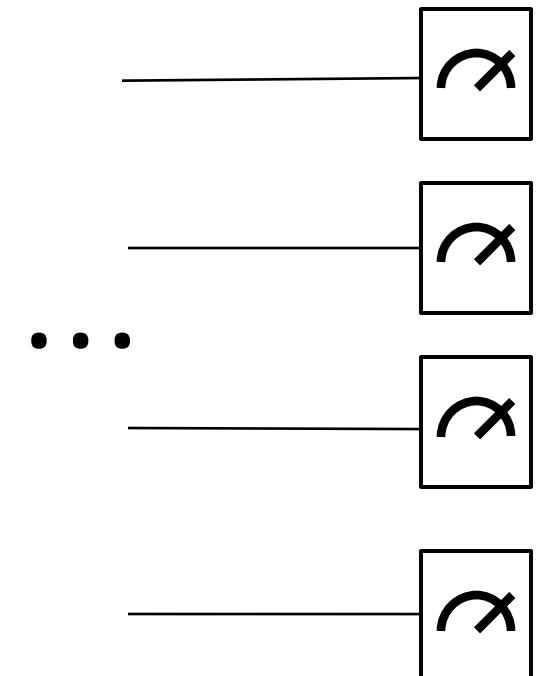
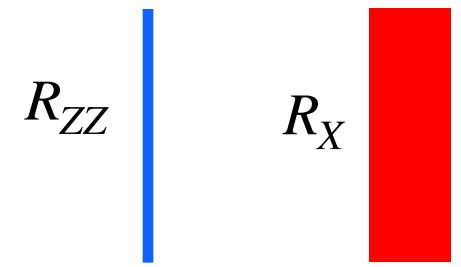
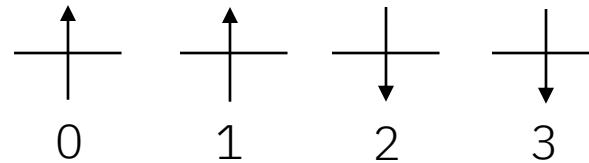
$$H = - \sum_{\langle i,j \rangle}^{N-1} J \sigma_{Z_i} \sigma_{Z_j} - \sum_i^N h_i \sigma_{X_i}$$

時間発展



状態準備

繰り返すと時間tでの波動関数を得る



高次のオーダーを得るためにSuzuki-Trotter公式を再帰的に適用する

2次のオーダーのSuzuki-Trotter公式

$$e^{-itH} \approx \hat{U}_{ST2}(t) = \prod_{j=1}^L e^{-ia_j P_j \frac{t}{2}} \prod_{j'=L}^1 e^{-ia_j P_{j'} \frac{t}{2}}$$

再帰的に用いる場合

$$U_{ST(2k)}(t) = \left[U_{ST(2k-2)}(p_k t) \right]^2 U_{ST(2k-2)}((1 - 4p_k)t) \left[U_{ST(2k-2)}(p_k t) \right]^2$$

$$p_k = 1 / \left(4 - 4^{\frac{1}{2k-1}} \right)$$

例：4次のSuzuki-Trotter公式

$$\hat{U}_{ST4}(t) = \left[\hat{U}_{ST2}(p_2 t) \right]^2 \hat{U}_{ST2}((1 - 4p_2)t) \left[\hat{U}_{ST2}(p_2 t) \right]^2$$

$$p_2 = 1 / \left(4 - 4^{\frac{1}{2*2-1}} \right) = 1 / \left(4 - 4^{\frac{1}{3}} \right) \approx 0.4145$$

$$\hat{U}_{ST4}(\Delta t) = \hat{U}_{ST2}(0.4145\Delta t) \hat{U}_{ST2}(0.4145\Delta t) \hat{U}_{ST2}(-0.6579\Delta t) \hat{U}_{ST2}(0.4145\Delta t) \hat{U}_{ST2}(0.4145\Delta t)$$

Qiskitで簡単に実装可能

①ライブラリのインポート

```
# Import the qiskit library
import numpy as np
import matplotlib.pyplot as plt
import warnings

from qiskit import QuantumCircuit
from qiskit.circuit.library import PauliEvolutionGate
from qiskit.primitives import StatevectorEstimator
from qiskit.quantum_info import Statevector, SparsePauliOp
from qiskit.synthesis import (
    SuzukiTrotter,
    LieTrotter,
)
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2
warnings.filterwarnings("ignore")
```

②ハミルトニアンの設定

```
def get_hamiltonian(nqubits, J, h, alpha):
    # List of Hamiltonian terms as 3-tuples containing
    # (1) the Pauli string,
    # (2) the qubit indices corresponding to the Pauli string,
    # (3) the coefficient.
    ZZ_tuples = [("ZZ", [i, i + 1], -J) for i in range(0, nqubits - 1)]
    Z_tuples = [("Z", [i], -h * np.sin(alpha)) for i in range(0, nqubits)]
    X_tuples = [("X", [i], -h * np.cos(alpha)) for i in range(0, nqubits)]

    # We create the Hamiltonian as a SparsePauliOp, via the method
    # `from_sparse_list`, and multiply by the interaction term.
    hamiltonian = SparsePauliOp.from_sparse_list(
        [ZZ_tuples, *Z_tuples, *X_tuples], num_qubits=nqubits
    )
    return hamiltonian.simplify()
```

n_qubits = 6

```
hamiltonian = get_hamiltonian(nqubits=n_qubits, J=0.2, h=1.2, alpha=np.pi / 8.0)
hamiltonian
```

③初期状態の準備

```
initial_circuit = QuantumCircuit(n_qubits)
initial_circuit.prepare_state("001100")
# Change reps and see the difference when you decompose the circuit
initial_circuit.decompose(reps=1).draw("mpl")
```

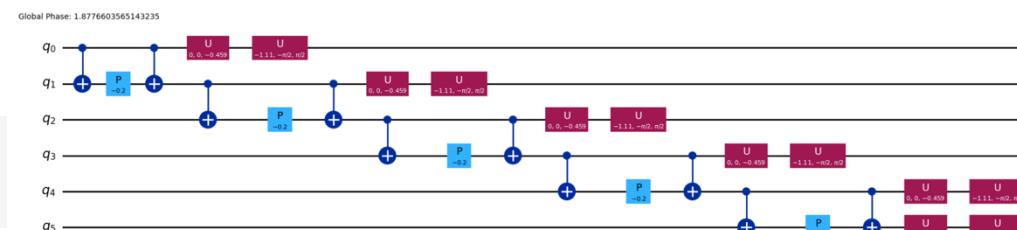
④時間発展回路の設定

```
single_step_evolution_gates_lt = PauliEvolutionGate(
    hamiltonian, dt, synthesis=product_formula_lt
)
single_step_evolution_lt = QuantumCircuit(n_qubits)
single_step_evolution_lt.append(
    single_step_evolution_gates_lt, single_step_evolution_lt.qubits
)

print(
    f"""
Trotter step with Lie-Trotter
-----
Depth: {single_step_evolution_lt.decompose(reps=3).depth()}
Gate count: {len(single_step_evolution_lt.decompose(reps=3))}
Nonlocal gate count: {single_step_evolution_lt.decompose(reps=3).num_nonlocal_gates()}
Gate breakdown: {", ".join(f"({k.upper()}: {v})" for k, v in single_step_evolution_lt.decompose(reps=3).count_ops().items())}
"""
)
single_step_evolution_lt.decompose(reps=3).draw("mpl", fold=-1)
```

Trotter step with Lie-Trotter

```
Depth: 17
Gate count: 27
Nonlocal gate count: 10
Gate breakdown: U: 12, CX: 10, P: 5
```



Notebook: <https://github.com/quantum-tokyo/introduction/blob/main/src/courses/utility-scale-quantum-computing/quantum-simulation-ja.ipynb> (量子位相推定回ご担当の林さんが和訳)

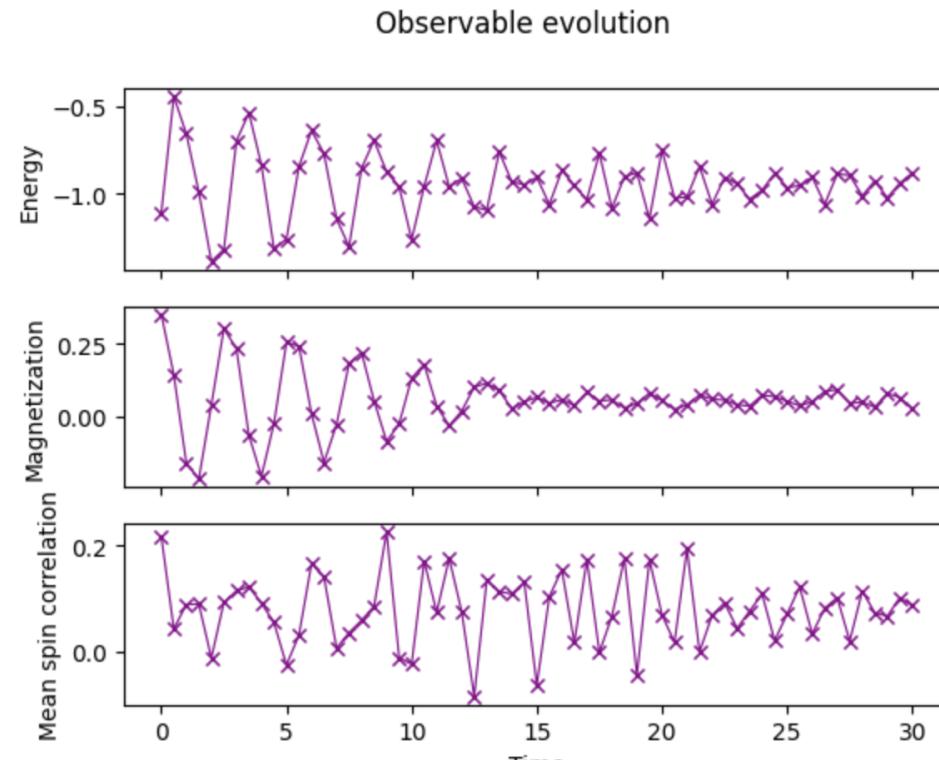
⑤シミュレーションの実行

```
# Initiate the circuit
evolved_state = QuantumCircuit(initial_circuit.num_qubits)
# Start from the initial spin configuration
evolved_state.append(initial_circuit, evolved_state.qubits)
# Initiate Estimator (V2)
estimator = StatevectorEstimator()
# Set number of shots
shots = 10000
# Translate the precision required from the number of shots
precision = np.sqrt(1 / shots)
energy_list = []
mag_list = []
corr_list = []
# Estimate expectation values for t=0.0
job = estimator.run(
    [evolved_state, [hamiltonian, magnetization, correlation]], precision=precision
)
# Get estimated expectation values
evs = job.result[0].data.evs
energy_list.append(evs[0])
mag_list.append(evs[1])
corr_list.append(evs[2])
# Start time evolution
for n in range(num_timesteps):
    # Expand the circuit to describe delta-t
    evolved_state.append(single_step_evolution_gates_lt, evolved_state.qubits)
    # Estimate expectation values at delta-t
    job = estimator.run(
        [evolved_state, [hamiltonian, magnetization, correlation]], precision=precision,
    )
    # Retrieve results (expectation values)
    evs = job.result[0].data.evs
    energy_list.append(evs[0])
    mag_list.append(evs[1])
    corr_list.append(evs[2])
# Transform the list of expectation values (at each time step) to arrays
energy_array = np.array(energy_list)
mag_array = np.array(mag_list)
corr_array = np.array(corr_list)
```

Qiskitで簡単に実装可能

⑥オブザーバブルの時間発展をプロット

```
fig, axes = plt.subplots(3, sharex=True)
times = np.linspace(0, evolution_time, num_timesteps + 1) # includes initial state
axes[0].plot(
    times,
    energy_array,
    label="First order",
    marker="x",
    c="darkmagenta",
    ls="-",
    lw=0.8,
)
axes[1].plot(
    times, mag_array, label="First order", marker="x", c="darkmagenta", ls="-", lw=0.8
)
axes[2].plot(
    times, corr_array, label="First order", marker="x", c="darkmagenta", ls="-", lw=0.8
)
axes[0].set_ylabel("Energy")
axes[1].set_ylabel("Magnetization")
axes[2].set_ylabel("Mean spin correlation")
axes[2].set_xlabel("Time")
fig.suptitle("Observable evolution")
```



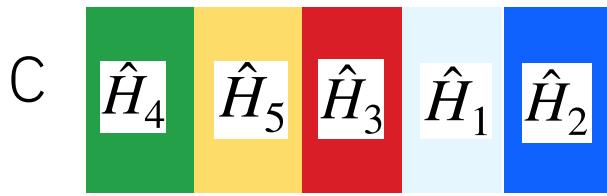
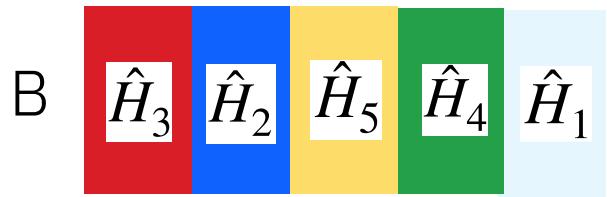
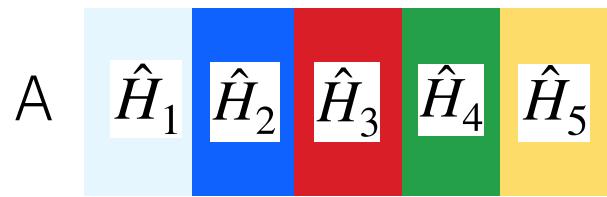
Notebook: <https://github.com/quantum-tokyo/introduction/blob/main/src/courses/utility-scale-quantum-computing/quantum-simulation-ja.ipynb> (量子位相推定回ご担当の林さんが和訳)

Trotterization

- 直観的で実装しやすい
- 必要な量子ビット数が最小 (アンシラ量子ビットが不要)
- エラーに対するゲート深さのスケーリングが最適化されていない
 - 1次オーダーのTrotterizationのスケーリング: $O(t^2/\epsilon)$
 - 2次オーダーのSuzuki-Trotterのスケーリング: $O(t^{1.5}/\epsilon^{0.5})$

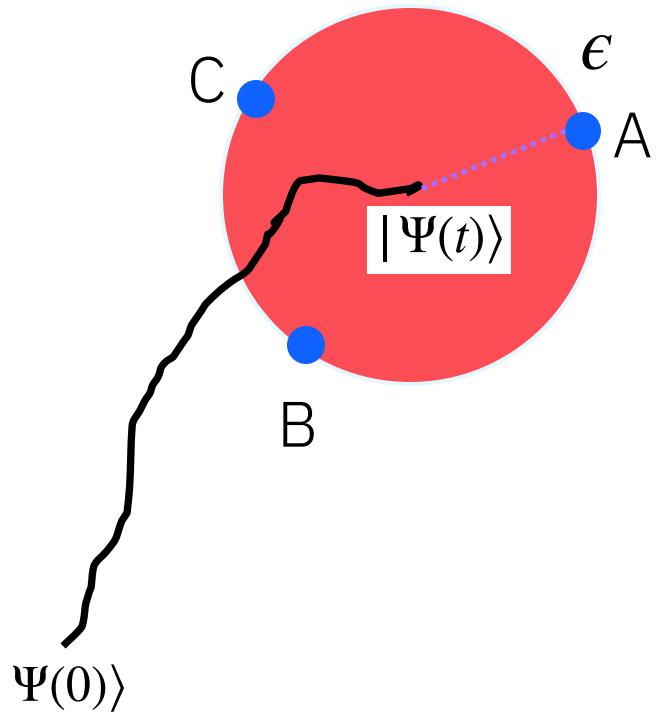
Randomization

Childs, Ostrander, Su, arXiv: 1805.08385



$$\hat{H} = \sum a_j \hat{H}_j \quad \|\hat{H}_j\| = 1$$

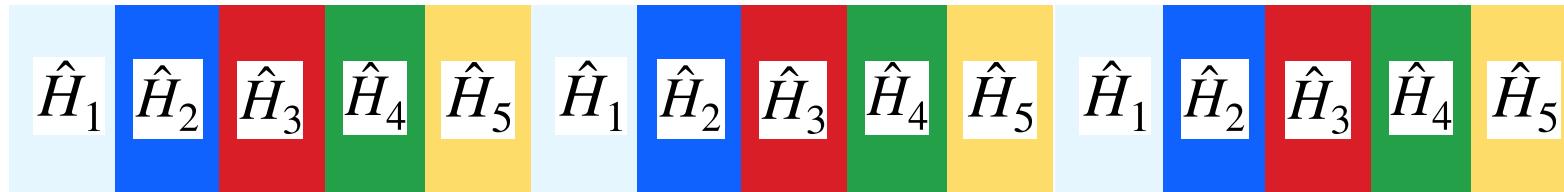
$$e^{-i\hat{H}_1\Delta t} e^{-i\hat{H}_2\Delta t} e^{-i\hat{H}_3\Delta t} e^{-i\hat{H}_4\Delta t} e^{-i\hat{H}_5\Delta t}$$



どの並べかえを用いても同じ誤差でバウンドされる

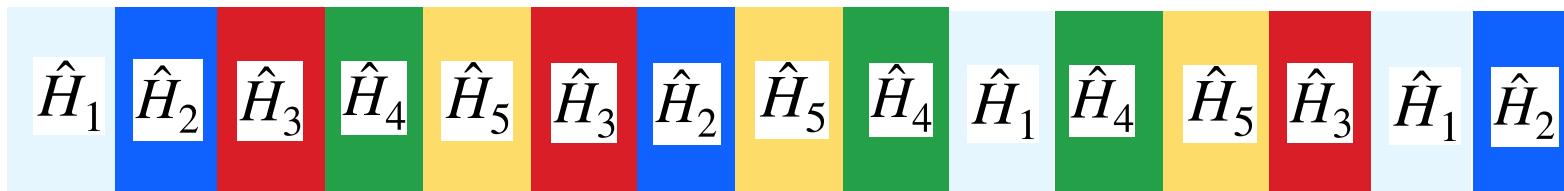
Trotterization vs Randomization

“ふつう”のTrotterization (1次オーダー)



$$N_{\text{gates}} = O\left(\frac{L^4 t^2}{\epsilon}\right)$$

Randomization



$$N_{\text{gates}} = O\left(\frac{L^{2.5} t^{1.5}}{\epsilon^{0.5}}\right)$$

ランダムに並び替えれば誤差をアベレージアウトできるか？

Randomizationのパフォーマンス (決められた誤差範囲内の結果を得るための時間幅)

Childs, Ostrander, Su, arXiv: 1805.08385

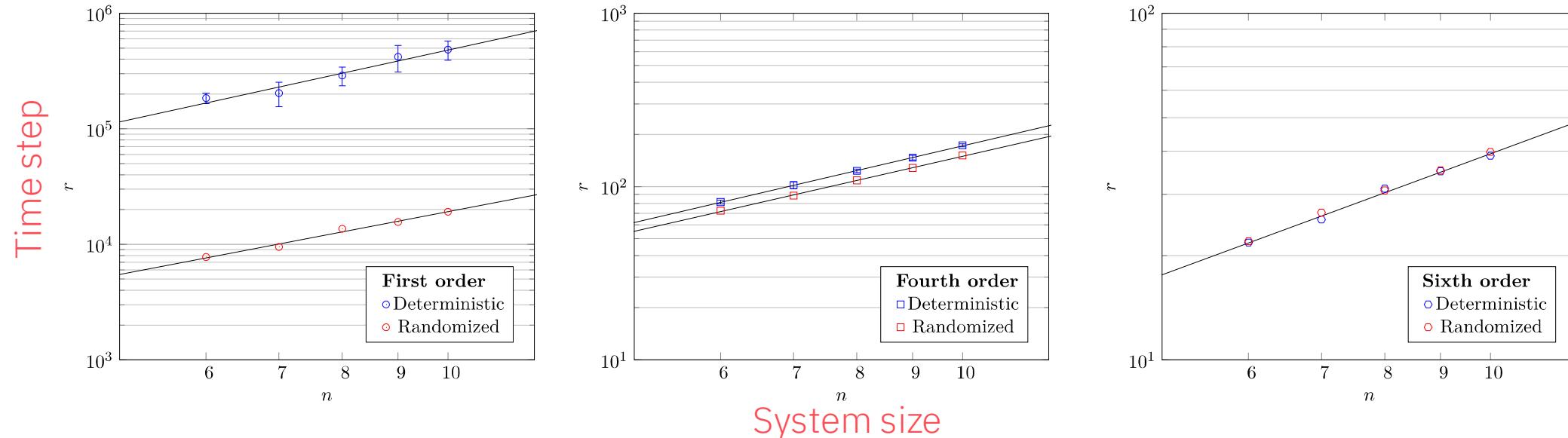


Figure 1: Comparison of the values of r between deterministic and randomized product formulas. Error bars are omitted when they are negligibly small on the plot. Straight lines show power-law fits to the data.

Heisenberg模型を使った計算では低次のオーダーのTrotterizationで

Randomizationは効率的 (near termでは良い)

Randomization

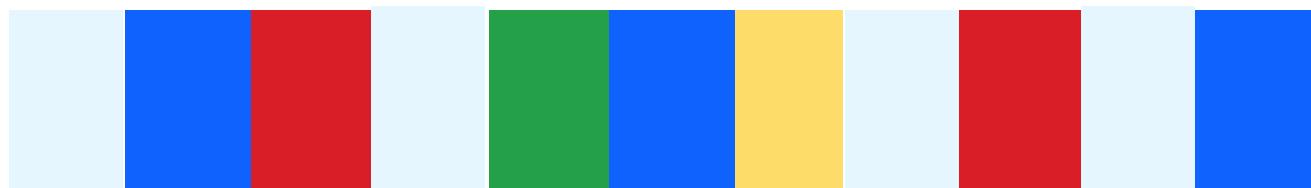


$$\hat{H} = \sum a_j \hat{H}_j \quad a_j \geq 0 \quad \|\hat{H}_j\| = 1$$

もっとエラーをアベレージアウトしたい場合、

ハミルトニアンの項数が大きくなっても適用できるか?

- ・サンプル $e^{-i\lambda\hat{H}_j\Delta t}$
- ・重み付け $p_j = a_j/\lambda$
- $\lambda = \sum_j a_j$



Qdriftのパフォーマンス (決められた誤差範囲内の結果を得るためのゲート数)

Campbell, Phys Rev Lett 123, 070503 (2019)

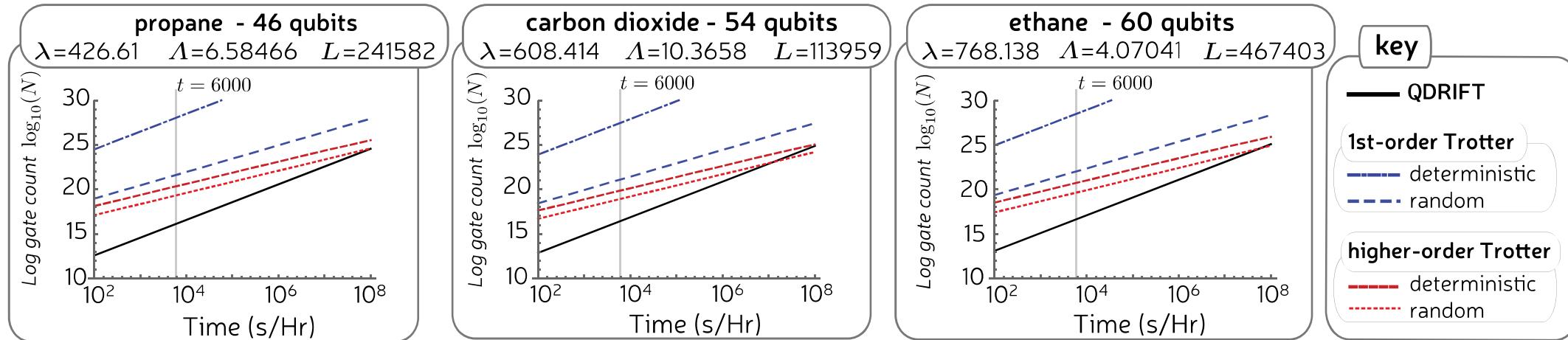


FIG. 2. The number of gates used to implement $U = \exp(iHt)$ for various t and $\epsilon = 10^{-3}$ and three different Hamiltonians (energies in Hartree) corresponding to the electronic structure Hamiltonians of propane (in STO-3G basis), carbon dioxide (in 6-31g basis), and ethane (in 6-31g basis). Since the Hamiltonian contains some very small terms, one can argue that conventional Trotter-Suzuki methods would fare better if they truncate the Hamiltonian by eliminating negligible terms. For this reason, whenever simulating to precision ϵ we also remove from the Hamiltonian the smallest terms with weight summing to ϵ . This makes a fairer comparison, though in practice we found it made no significant difference to performance. For the Suzuki decompositions we choose the best from the first four orders, which is sufficient to find the optimal.

ハミルトニアンの項数が大きいときに有効

$$N_{\text{gates}} = O\left(\frac{2\lambda^2 t^2}{\epsilon}\right)$$

Randomization

- ゲート数のスケーリングがハミルトニアンの項数に寄らない (QDrift)
- ハミルトニアンの項数が大きい量子化学の問題を解く場合にアドバンテージがある

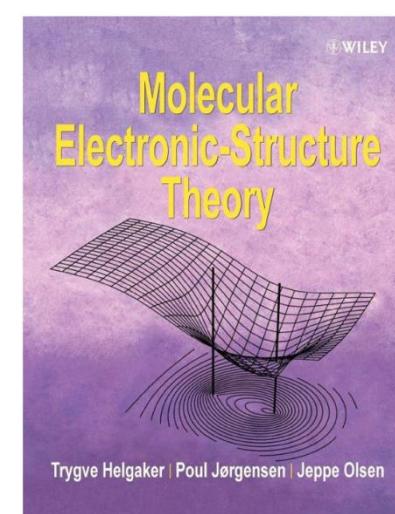
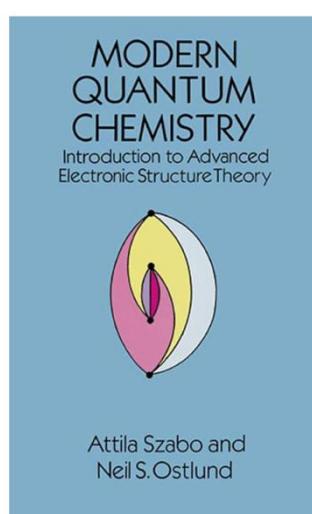
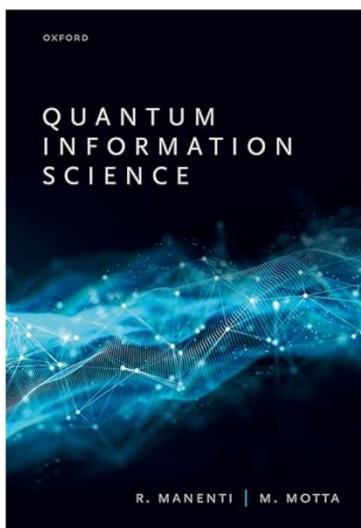
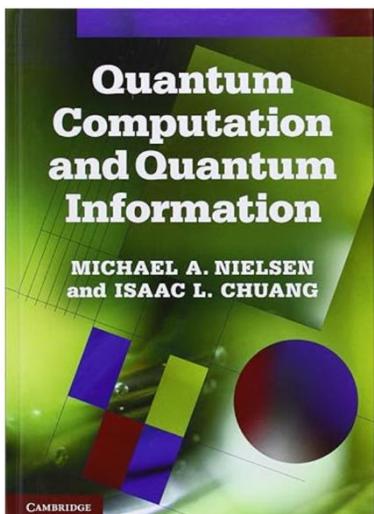
さらに学習を進めるために

量子コンピューティング（量子情報）

- “Quantum Computation and Quantum Information”, Michael A. Nielsen and Isaac L. Chuang, Cambridge University Press.
- “Quantum Information Science”, Riccardo Manenti and Mario Motta, Oxford University Press.

量子化学

- Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory by Attila Szabo, Neil S. Ostlund
- Molecular Electronic-Structure Theory 1st Edition by Trygve Helgaker, Poul Jorgensen, Jeppe Olsen



参考文献

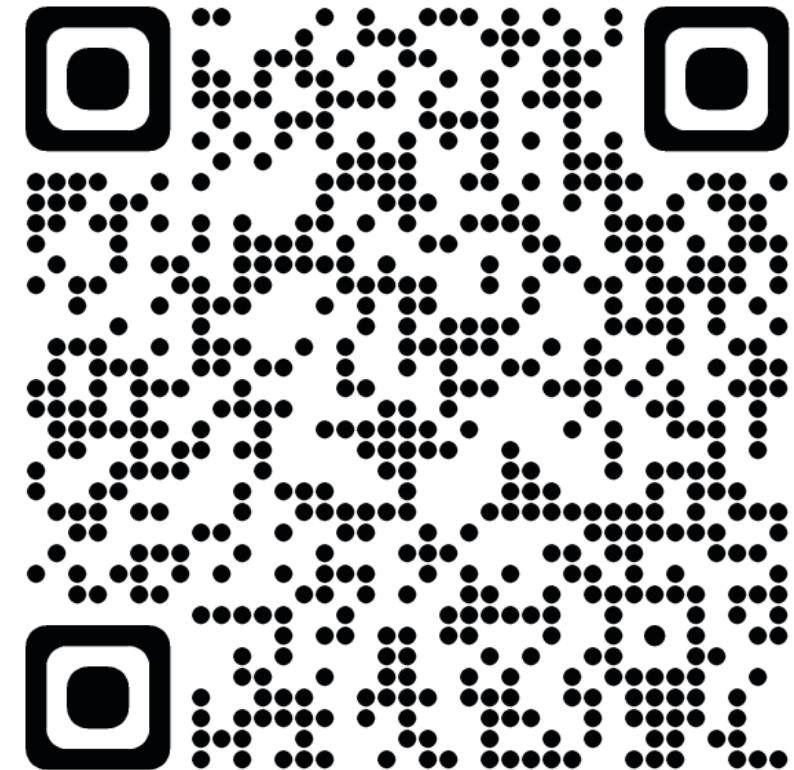
- Slide shared from Professor Sato at the University of Tokyo (スライド. P6)
- Childs et al., Quantum 3, 182 (2019) (スライドP. 33-34)
- Campbell, Phys. Rev. Lett., 123, 070503 (2019) (スライドP. 35-36)

最後に：もっと勉強したい方へ

The screenshot shows a web browser window with the Quantum Tokyo website. The main title is 'ユーティリティー・スケール量子コンピューティング'. Below it is a section titled '概要' containing a detailed description of the course. A numbered list of 14 lessons follows:

1. はじめに
2. [量子ビット・量子ゲート・量子回路](#)
3. [量子テレポーテーション](#)
4. グローバーのアルゴリズム
5. [量子位相推定](#)
6. 量子変分アルゴリズム
7. [量子系のシミュレーション](#)
8. 古典計算によるシミュレーション
9. 量子ハードウェア
10. 量子回路の最適化
11. 量子エラー緩和
12. 量子ユーティリティーの実験 I
13. 量子ユーティリティーの実験 II
14. 量子ユーティリティーの実験 III

The sidebar on the left contains links to other resources like 'Qiskit の始め方', 'IBM Quantum Platform 教材', and 'IBM Research Blog 日本語版'.



<https://quantum-tokyo.github.io/introduction/courses/utility-scale-quantum-computing/overview-ja.html>

IBM Quantum Learning

「Utility-scale quantum computing」の日本語解説

元のコース：<https://quantum.cloud.ibm.com/learning/en/courses/utility-scale-quantum-computing>

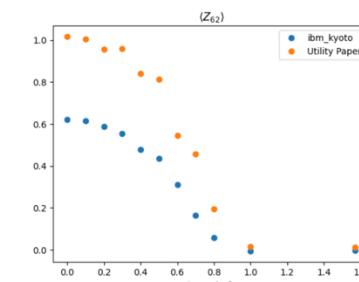
1. はじめに(飛ばします)
2. 量子ビット・量子ゲート・量子回路(7/7(月))
3. 量子テレポーテーション(7/16(水))
4. グローバーのアルゴリズム(7/16(水))
5. 量子位相推定(7/28(月))
6. 量子変分アルゴリズム(7/28(月))
7. 量子系のシミュレーション(8/22(金))
8. 古典計算によるシミュレーション
9. 量子ハードウェア
10. 量子回路の最適化
11. 量子エラー緩和
12. 量子ユーティリティーの実験 I
13. 量子ユーティリティーの実験 II
14. 量子ユーティリティーの実験 III



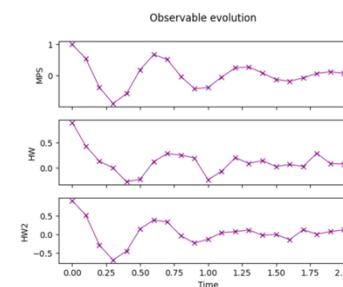
今回の回のJupyter notebookの和訳：

<https://quantum-tokyo.github.io/introduction/courses/utility-scale-quantum-computing/quantum-simulation-ja.html>

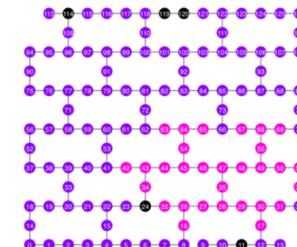
I. Nature paper
(127 qubits x 60 entangling gates)



II. 1D Transverse Ising model
(70 qubits x 80 entangling gates)



III. The largest GHZ state challenge



ご視聴ありがとうございました！

© 2025 International Business Machines Corporation