Qiskit Document Tutorials 勉強会
Qiskit Nature
基底状態ソルバー

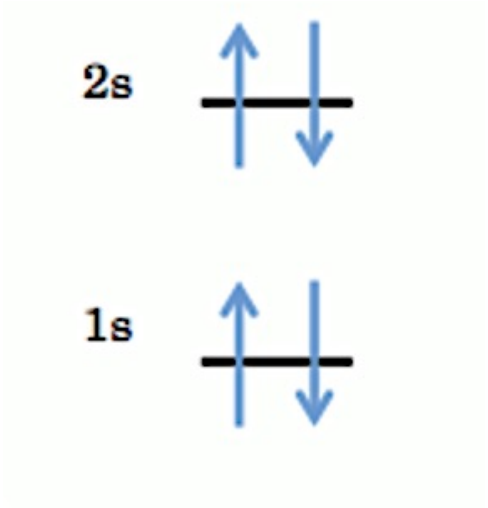
Emi ADACHI

Agenda

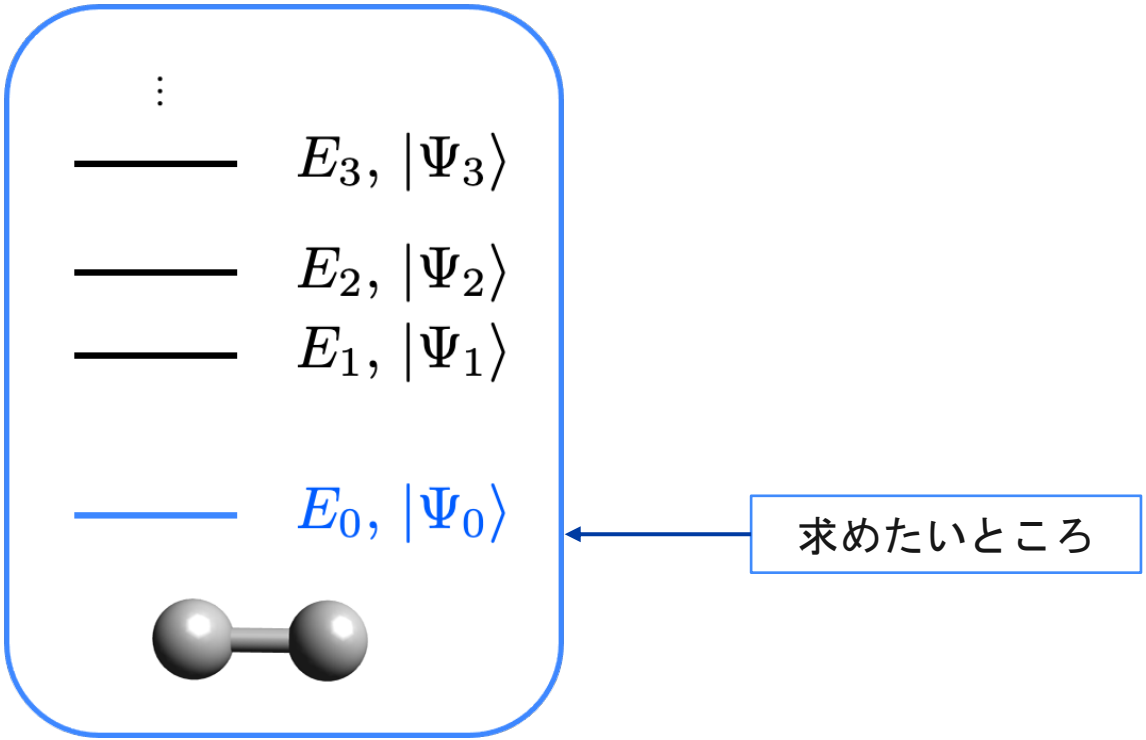
1. 基底状態とは？
2. コードの流れ
3. NumPyを使った場合
4. VQEについて
5. VQEのコードの実行
6. 試行状態・Ansatzについて
7. 最適化コードの実行
8. 参考文献、引用文献一覧

1. 基底状態とは？

構成原理に従い、エネルギー固有値の小さい準位から２個ずつ電子が入った配置



1.1 基底状態のハミルトニアンを求める



2. コードの流れ

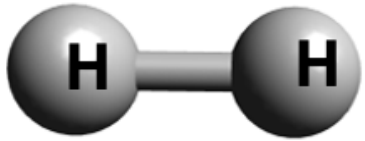
1. 分子構造を設定
2. ハートリーフォック法で近似
3. 第二量子化（フェルミオンで記述）
4. Jordan Wigner変換（パウリ演算子で記述）
5. Jordan Wigner変換の結果の量子ビット数をパリティマッピングで減らす
6. ソルバーとしてNumPy（古典の厳密行列計算）かVQE（量子計算）を選択して解く
   （VQEの場合は事前にAnsatz（UCC,Two Local）を選んでおく）

分子構造設定→ハートリーフォック法で近似→第二量子化→Jordan Wigner変換→パリティマッピング

```
# Aerをインポート
from qiskit import Aer

# UnitsType,Moleculeをインポート
from qiskit_nature.drivers import UnitsType, Molecule

#ElectronictructureDriverType,ElectronicStructureMoleculeDriverをインポート
from qiskit_nature.drivers.second_quantization import ElectronicStructureDriverType, ElectronicStructureMoleculeDriver

# ElectronicStructureProblemをインポート
from qiskit_nature.problems.second_quantization import ElectronicStructureProblem

# 最終的に計算する時に使う量子ビットを減らす：QubitConverterをインポート
from qiskit_nature.converters.second_quantization import QubitConverter

# JordanWigner変換：JordanWignerMapperをインポート
from qiskit_nature.mappers.second_quantization import JordanWignerMapper

# 水素分子の構造を設定
molecule = Molecule(geometry=[['H', [0., 0., 0.]],
                              ['H', [0., 0., 0.735]]],
                    charge=0, multiplicity=1)

# PySCFにより、電子密度を求める
# 基底：STO-3G
driver = ElectronicStructureMoleculeDriver(molecule, basis='sto3g', driver_type=ElectronicStructureDriverType.PYSCF)
es_problem = ElectronicStructureProblem(driver)

qubit_converter = QubitConverter(JordanWignerMapper())
```

2.1 分子構造の設定

- 原子間の距離をオングストローム・Åに設定
- 水素分子の定義をする
  - 位置:原子座標を(x,y,z)軸の座標で設定
  - スピン多重度N:スピン状態に関する縮退度
  - 全電荷:分子全体の総電荷
  - 自由度: 物理系の状態を記述するのに必要な最小限の変数の個数
  - 質量



2.2 Hartree Fock法の概要

- 考え方：

電子間相互作用を近似的に取り扱うことにより、多電子系においても１電子波動関数を定義

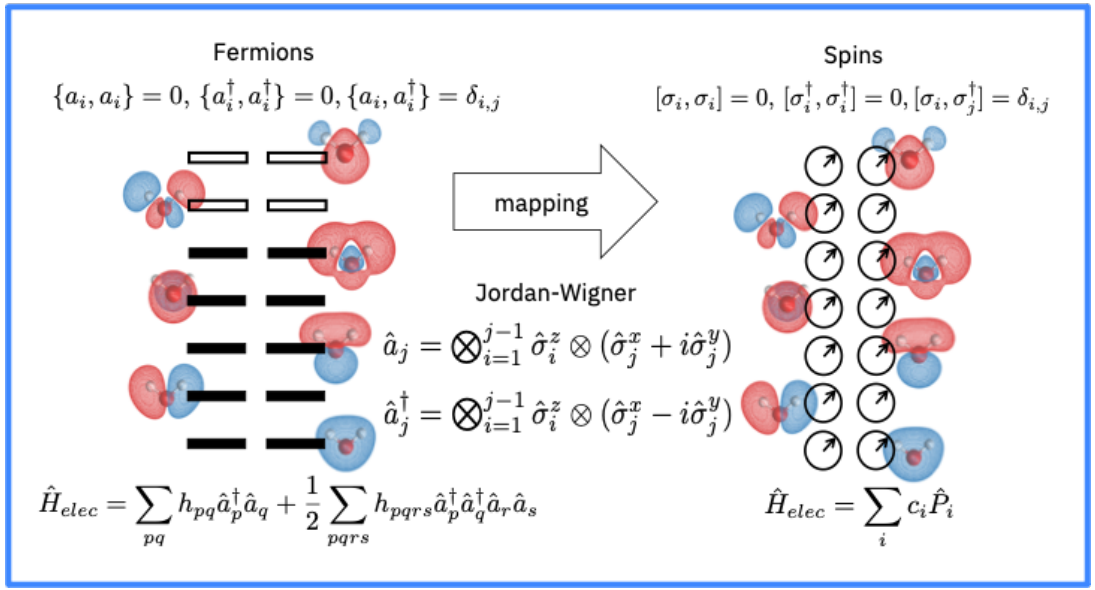
- 表現法：電子基底状態を単一の基底配置に対応するSlater行列式で表す
- Slater行列式：Pauliの原理を満足するために用いる行列式

2.3 第二量子化とは？

- 場の量子化
- 波動関数の反対称性を第二量子化演算子の代数学的な特質に置き換えた理論
- 多電子系を扱う時の力点をN電子波動関数から１電子積分、２電子積分へと移した方法

2.4 Jordan Wigner 変換とは？

Jordan Wigner変換とは、電子の各スピン軌道を量子ビットにマッピングする



2.4.1 Jordan Wigner変換のしくみ

生成・消滅演算子の反交換関係を満足させ、波動関数を反対称化する

$$\hat{H}_{elec} = \sum_{pq} h_{pq} \hat{a}_p^\dagger \hat{a}_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r \hat{a}_s$$

$\hat{a}_p{}^\dagger$:生成演算子 , $\hat{a}_p$:消滅演算子

2.5 パリティマッピングで量子ビットを減らす

```
qubit_converter = QubitConverter(JordanWignerMapper())
```

　Jordan Wigner変換では計算する際、ビットが少なくとも４量子ビット必要
→対称性を考慮することで量子ビットの削減を可能にする
→そうすると、２量子ビットで計算することができる

2.6 ソルバーの種類

- NumPyソルバー(古典)：NumPyで最小の固有値を求める方法、
  古典の厳密行列計算

- 量子変分固有値ソルバー(VQE)：量子力学の変分法アプリケーション
  - 試行状態・Ansatzを選択
    - UCC(Qiskitでは***UCCSD***)
    - TwoLocal
    - 自作

3. NumPyを使って求める方法

1. NumPyを使って求める方法のコードの流れ
2. 分子構造の設定
3. NumPyを使う方法のコード
4. コードの説明

3.1 NumPyを使う場合の流れ

1. 解きたい電子系のハミルトニアンを用意する
2. NumPyソルバー・NumPyEigenSolverを使う
   NumPyソルバー(古典)：NumPyで最小の固有値を求める方法

3.2 ハミルトニアンの用意

```
# Aerをインポート
from qiskit import Aer

# UnitsType,Moleculeをインポート
from qiskit_nature.drivers import UnitsType, Molecule

#ElectronictructureDriverType,ElectronicStructureMoleculeDriverをインポート
from qiskit_nature.drivers.second_quantization import ElectronicStructureDriverType, ElectronicStructureMoleculeDriver

# ElectronicStructureProblemをインポート
from qiskit_nature.problems.second_quantization import ElectronicStructureProblem

# 最終的に計算する時に使う量子ビットを減らす：QubitConverterをインポート
from qiskit_nature.converters.second_quantization import QubitConverter

# JordanWigner変換：JordanWignerMapperをインポート
from qiskit_nature.mappers.second_quantization import JordanWignerMapper

# 水素分子の構造を設定
molecule = Molecule(geometry=[['H', [0., 0., 0.]],
                              ['H', [0., 0., 0.735]]],
                    charge=0, multiplicity=1)

# PySCFにより、電子密度を求める
# 基底：STO-3G
driver = ElectronicStructureMoleculeDriver(molecule, basis='sto3g', driver_type=ElectronicStructureDriverType.PYSCF)
es_problem = ElectronicStructureProblem(driver)

qubit_converter = QubitConverter(JordanWignerMapper())
```

3.3 NumPyを使う方法

```
# 古典NumPyで固有値の最小値を求める
# algorithmsからNumPyMinimumEigensolverをインポート
from qiskit.algorithms import NumPyMinimumEigensolver

# ハミルトニアンの行列を対角化を用いる
numpy_solver = NumPyMinimumEigensolver()

# GroundStateEigensolverを求める

calc = GroundStateEigensolver(qubit_converter, numpy_solver)

#  水素分子
res = calc.solve(es_problem)
print(res)
```

3.4 *NumPyMinimumEigensolver*について

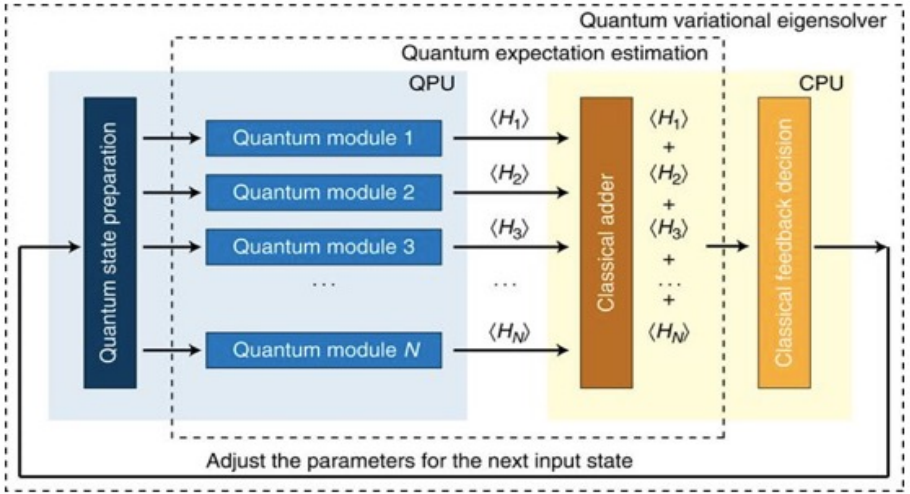
```
CLASS NumPyMinimumEigensolver(filter_criterion=None)
```

- パラメータ: filter_criterion
(Optional[Callable[[Union[List, ndarray], float,Optional[List[float]], bool])
  - 固有値/固有状態をフィルタリングすることができるcallable
callable:指定した引数が呼び出し可能かどうか(関数のように扱えるか)を判定
  - 最小固有値ソルバーは，実現可能な状態のみを探索、実現可能な状態の中で最小の固有値を持つ固有状態を返す
  - callableはfilter(eigenstate, eigenvalue, aux_values)というシグネチャを持つ
  - この値を考慮するかどうかを示すブール値を返す
  - 実現可能な要素がない場合は、結果が空になることもある

4. VQEについて

1. VQEアルゴリズムとは？
2. VQEの流れ
3. Ansatzについて
4. 重要参考資料の紹介

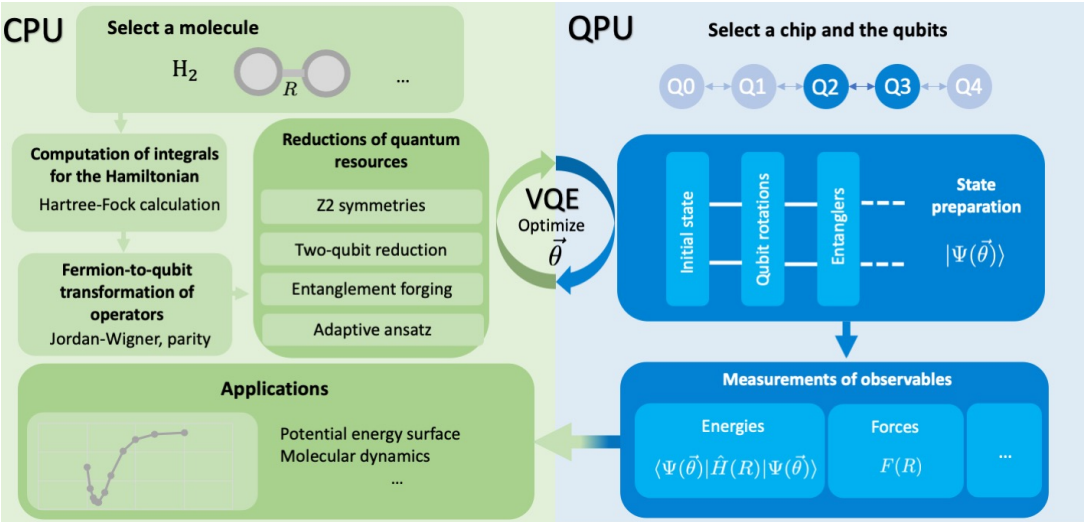
4.1 VQEアルゴリズムとは？



- Variational Quantum Eigensolver
- 変分量子固有値ソルバー
- 量子力学の変分法アプリケーション
- 量子コンピューター上で効率的に記述できる量子状態を用いて、量子多体系の基底状態を求める変分アルゴリズム

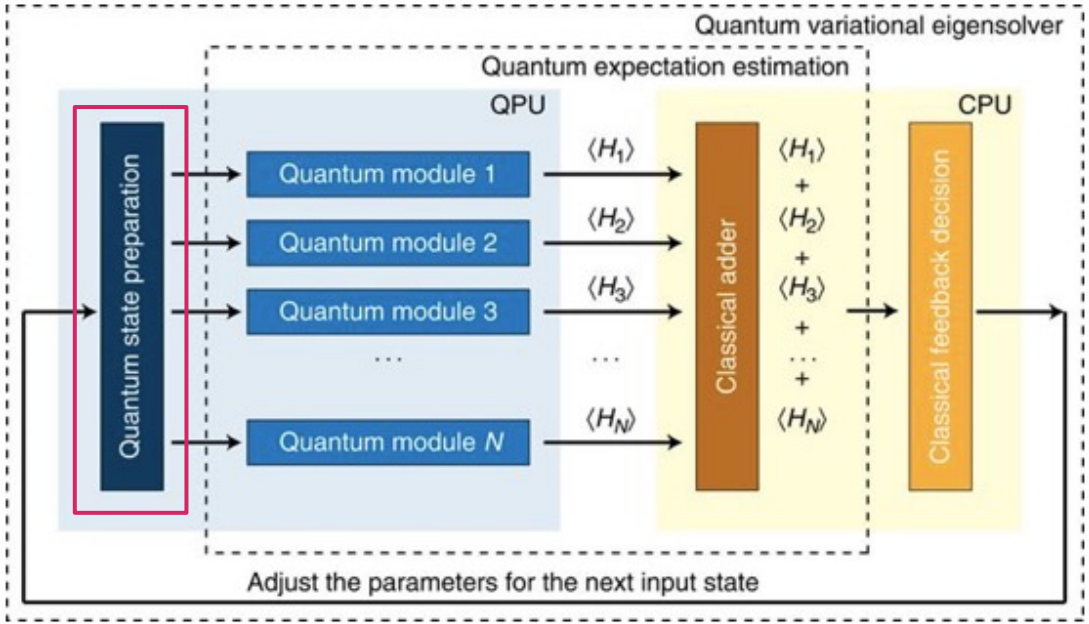
出典：https://arxiv.org/abs/1304.3061

4.2 VQEの流れ



1. 解きたい電子系のハミルトニアンを用意(電子構造と同じ)
2. 初期化された量子ビットをパラメータθで指定される量子回路に入力し、**Ansatz**を作る
3. 測定値からエネルギー期待値を計算する
4. Step2,3を繰り返し、エネルギー期待値を最小化するようにθを最適化

4.3 試行状態・Ansatzとは？



適当な波動関数$|\psi\rangle$ を$|\psi_{min}\rangle$に近似させるため初期推測をすること
→その結果、エネルギー期待値を計算することができる

4.4 重要参考資料：VQEアルゴリズム

参考資料：
Qiskit Textbook : https://qiskit.org/textbook/ja/ch-applications/vqe-molecules.html
Qiskit Tutorial: https://qiskit.org/documentation/tutorials.html#algorithms

出典：Quantum Tokyo You tubeチャンネル

5. VQEのコード実行の流れ

1. ハミルトニアンの用意（電子構造と同じ・共通事項）
2. VQEコードを実行するための準備
3. 試行状態・Ansatzの設定
4. 最適化：エネルギー期待値を最小化

5.1 ハミルトニアンの用意

```
# Aerをインポート
from qiskit import Aer

# UnitsType,Moleculeをインポート
from qiskit_nature.drivers import UnitsType, Molecule

#ElectronictructureDriverType,ElectronicStructureMoleculeDriverをインポート
from qiskit_nature.drivers.second_quantization import ElectronicStructureDriverType, ElectronicStructureMoleculeDriver

# ElectronicStructureProblemをインポート
from qiskit_nature.problems.second_quantization import ElectronicStructureProblem

# 最終的に計算する時に使う量子ビットを減らす：QubitConverterをインポート
from qiskit_nature.converters.second_quantization import QubitConverter

# JordanWigner変換：JordanWignerMapperをインポート
from qiskit_nature.mappers.second_quantization import JordanWignerMapper

# 水素分子の構造を設定
molecule = Molecule(geometry=[['H', [0., 0., 0.]],
                              ['H', [0., 0., 0.735]]],
                    charge=0, multiplicity=1)

# PySCFにより、電子密度を求める
# 基底：STO-3G
driver = ElectronicStructureMoleculeDriver(molecule, basis='sto3g', driver_type=ElectronicStructureDriverType.PYSCF)
es_problem = ElectronicStructureProblem(driver)

qubit_converter = QubitConverter(JordanWignerMapper())
```

5.2 VQEを使うための準備

```
# StatevectorSimulatorをインポート
from qiskit.providers.aer import StatevectorSimulator

# Aerをインポート
from qiskit import Aer

# QuantumInstanceをインポート
from qiskit.utils import QuantumInstance

# VQEUCCFactoryをインポート
from qiskit_nature.algorithms import VQEUCCFactory

# backend：aer_simulator_statevector
quantum_instance = QuantumInstance(backend = Aer.get_backend('aer_simulator_statevector'))
vqe_solver = VQEUCCFactory(quantum_instance)
```

←***Ansatz***の選択
UCC or **TwoLocal**

5.2.1 コードの説明

- ***Aer*** : シミュレーターとノイズモデルにより開発を加速する
- ***StatevectorSimulator*** :理想的な量子回路の状態ベクトルをシミュレーターする
- ***QuantumInstance*** :実行設定を含むQuantumバックエンド

5.2.2 *StatevectorSimulator*について

```
CLASS StatevectorSimulator(configuration=None, properties=None, provider=None,
      **backend_options)
```

- 理想的な量子回路の状態ベクトルをシミュレーターする
- 設定可能なオプション：
  - CPUとGPUのシミュレーション方法
  - 追加の設定可能なオプションをサポート
  - 初期化時に適切な kwargs を使用
    （kwargs: 複数のキーワード引数を辞書として受け取ること）
  - set_options()メソッドを使って設定・更新できる
- ランタイム・オプション
  - run()メソッドでkwargsとして指定
  - 実行時にのみ適用。以前に設定されていたオプションは上書きされる

出典：https://qiskit.org/documentation/stubs/qiskit.providers.aer.StatevectorSimulator.html?highlight=statevectorsimulator#qiskit.providers.aer.StatevectorSimulator

5.2.3 *QuantumInstance*について

QuantumInstance ¶

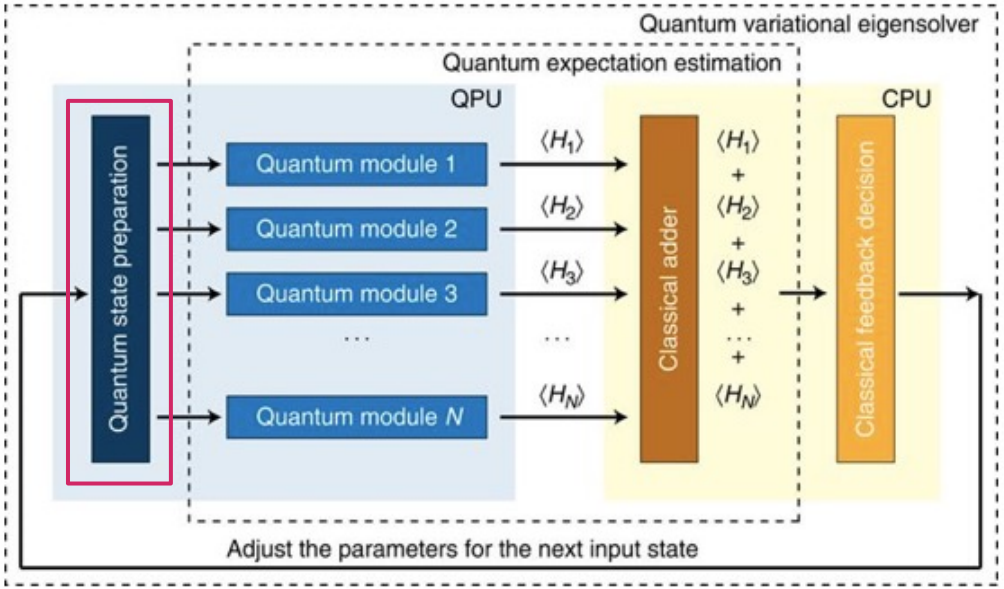
```
CLASS QuantumInstance(backend, shots=None, seed_simulator=None, max_credits=10,
      basis_gates=None, coupling_map=None, initial_layout=None, pass_manager=None,
      seed_transpiler=None, optimization_level=None, backend_options=None, noise_model=None,
      timeout=None, wait=5.0, skip_qobj_validation=True, measurement_error_mitigation_cls=None,
      cals_matrix_refresh_period=30, measurement_error_mitigation_shots=None,
      job_callback=None, mit_pattern=None)
```

- 実行設定を含むQuantumバックエンド
- Qiskit Terraのバックエンドと、回路のトランスパイルと実行のための設定を保持
- Aquaアルゴリズムに提供されると、アルゴリズムはインスタンスを使い必要な回路を実行

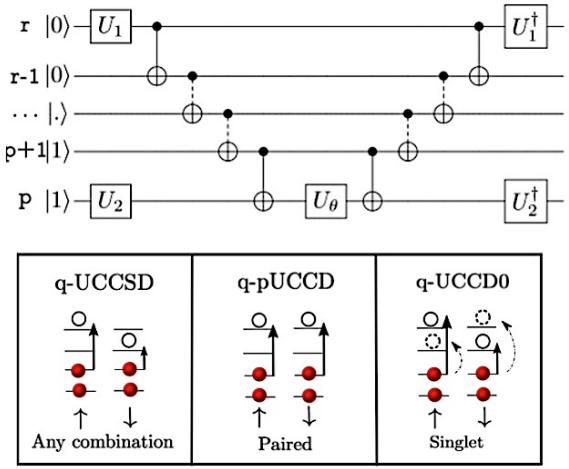
出典：https://qiskit.org/documentation/stubs/qiskit.utils.QuantumInstance.html?highlight=quantuminstance#qiskit.utils.QuantumInstance

6. Ansatzコードの実行について

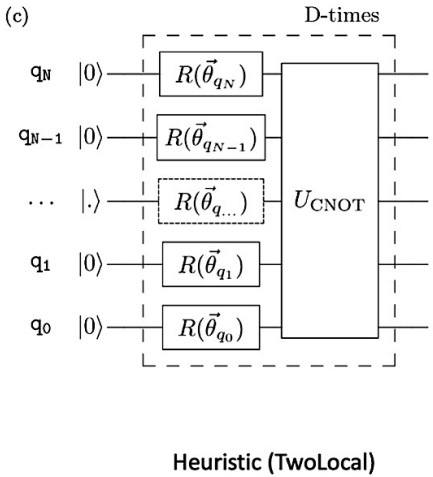
1. Ansatzの種類
2. UCCについて
3. UCCを選択する場合
4. TwoLocalについて
5. TwoLocalコードの実行



6.1 Ansatzの種類



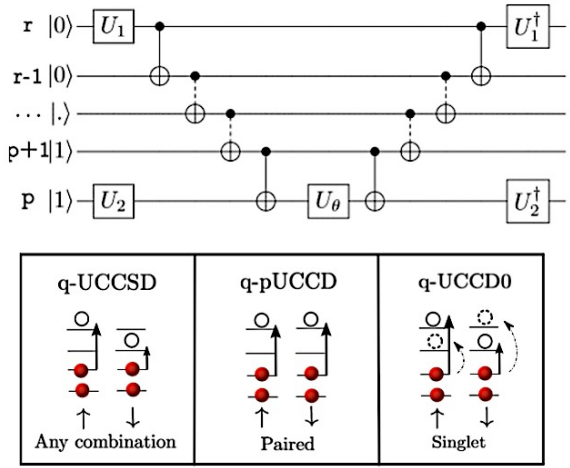
UCC



Heuristic (TwoLocal)

- UCC:物理的な発想によるもので、量子回路に電子の励起を大まかにマップ
- TwoLocal : 回転層とエンタングルメント層を交互に配置したパラメータ化された回路
- 自作でもOK

6.2 UCCについて(Qiskit：UCCSD)



- 物理的な発想によるもので、量子回路に電子の励起を大まかにマップ
- 量子ユニタリー結合クラスターシングル・ダブル試行状態:すべての可能な一電子励起と二電子励起まで考慮
- ペアダブルq-pUCCD (PUCCD) とシングレットのq-UCCD0 (SUCCD)は、これらの励起の一部のみを考慮
- 解離プロファイルに良い結果をもたらすことが証明
- q-pUCCD は、一励起状態を持たず、第二励起状がペア

6.3 UCCを選択する場合

```
# StatevectorSimulatorをインポート
from qiskit.providers.aer import StatevectorSimulator

# Aerをインポート
from qiskit import Aer

# QuantumInstanceをインポート
from qiskit.utils import QuantumInstance

# VQEUCCFactoryをインポート
from qiskit_nature.algorithms import VQEUCCFactory

# backend：aer_simulator_statevector
quantum_instance = QuantumInstance(backend = Aer.get_backend('aer_simulator_statevector'))
vqe_solver = VQEUCCFactory(quantum_instance)
```

6.3.1 *VQEUCCFactory*について

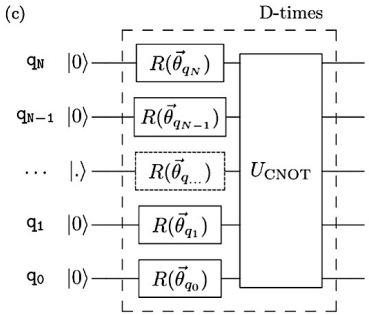
VQEUCCFactory ¶

```
CLASS  VQEUCCFactory(quantum_instance, optimizer=None, initial_point=None, gradient=None,
       expectation=None, include_custom=False, ansatz=None, initial_state=None)
```

- UCCSD Ansatz波動関数を用い、VQE最小固有値ソルバーを構築するためのFactory Method
- UCCSD : 量子ユニタリー結合クラスターシングル・ダブル(q-UCCSD)試行状態

➢ Factory Medthod
オブジェクト作成時に、作成するオブジェクトのクラスをサブクラスに選ばせる

6.4 *TwoLocal*について



Heuristic (TwoLocal)

- 回路の深さ(Depth)を短くするために発明
- Rゲートがパラメーター化された1量子ビットの回転
- 2量子ビットゲートによるU_{CNOT}のエンタングラーがある
- 特定のD回、独立なパラメーターを使って同じブロックを繰り返した後、基底状態に達するという考え

6.5 *TwoLocal*を設定する場合

```
# 試行状態：Ansatz

# VQEをインポート
from qiskit.algorithms import VQE

# TwoLocalをインポート
from qiskit.circuit.library import TwoLocal

tl_circuit = TwoLocal(rotation_blocks = ['h', 'rx'], entanglement_blocks = 'cz',
                      entanglement='full', reps=2, parameter_prefix = 'y')

another_solver = VQE(ansatz = tl_circuit,
                     quantum_instance = QuantumInstance(Aer.get_backend('aer_simulator_statevector')))
```

回転層

エンタングルメント層

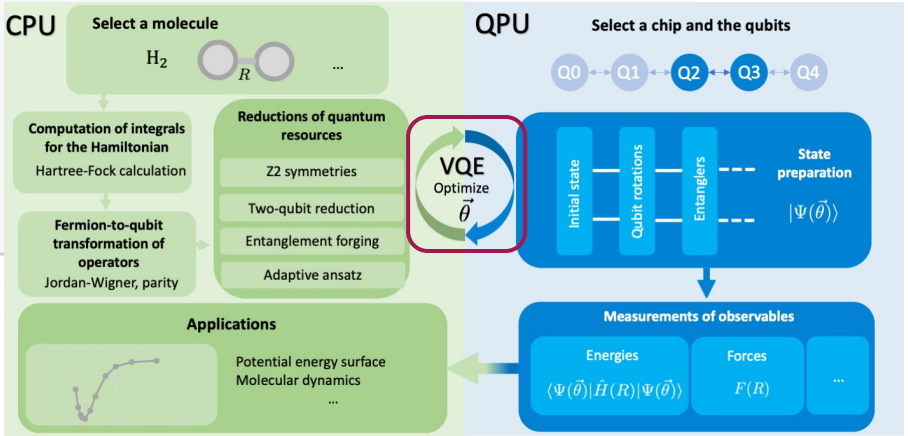
6.5.1 *TwoLocal*について

```
CLASS TwoLocal(num_qubits=None, rotation_blocks=None, entanglement_blocks=None,
      entanglement='full', reps=3, skip_unentangled_qubits=False,
      skip_final_rotation_layer=False, parameter_prefix='θ', insert_barriers=False,
      initial_state=None, name='TwoLocal')
```

- 回転層とエンタングルメント層を交互に配置したパラメータ化された回路
- 回転層:すべての量子ビットに適用される1量子ビットゲート
- エンタングルメント層:2つの量子ビットゲートを用いて、エンタングルメントを用いて設定された戦略に従って、量子ビットを絡める
- 文字列(ry,cx)やゲートタイプ(RYGate,CXgate)、QuantumCircuitで指定できる

7. 最適化

```
# VQEソルバーを使った場合
# GroundStateEigensolverをインポート
from qiskit_nature.algorithms import GroundStateEigensolver


calc = GroundStateEigensolver(qubit_converter, vqe_solver)


res = calc.solve(es_problem)

print(res)
```



7.1 *GroundStateEigensolver*について

```
CLASS GroundStateEigensolver(transformation, solver)
```

- 最小の固有値ソルバーを用いた基底状態の計算

- パラメータ
  - transformation (Transformation)

量子ビット演算子変換

  - solver (Union[MinimumEigensolver, MinimumEigensolverFactory])

最小固有値ソルバーまたはMESFactoryオブジェクト
例：VQEUCCSDFactory

参考文献、引用文献

- 大阪大学・藤井研究室・量子コンピューティング講義
- 安藤耕司研究室資料
- ザボ、オストランド『新しい量子化学』
- 「量子化学」のことが一冊でまるごとわかる
- 量子化学　基礎から応用まで
- すぐできる量子化学計算・ビギナーズマニュアル
- アトキンス物理化学
- 現代物理化学
- 量子コンピューティング・基本アルゴリズムから量子機械学習まで
- Qiskit Nature Tutorials
- Qiskit Textbook
- You tube: Quantum Tokyo
- Medium 記事 "Introducing Qiskit Nature"
- IBM Quantum Challenge 2021
- Qiskit Global Summer School Chemistry
- Head First Design Patterns
- https://arxiv.org/abs/1304.3061

Thank you