日本語訳『Qiskit Textbook』勉強会 第4章 4.1.2

Takehiko Amano
IBM Garage / Tokyo lab.

はじめに – 量子化学について

- 分子の構造や化学反応などを量子力学を利用して解析する分野
- 分子のハミルトニアンを作成してそれを解く。

$$\hat{H} = \sum_i \frac{1}{2m} p_i^2 - \frac{e^2}{4\pi\varepsilon_0} \sum_{Ki} \frac{Z_K}{r_{iK}} + \frac{e^2}{4\pi\epsilon_0} \sum_{i>j} \frac{1}{|r_i - r_j|} + \frac{e^2}{4\pi\epsilon_0} \sum_{K>L} \frac{Z_K Z_L}{|R_K - R_L|}$$

電子の運動量　　核・電子の引力　　電子・電子の反発力　　核間の反発力

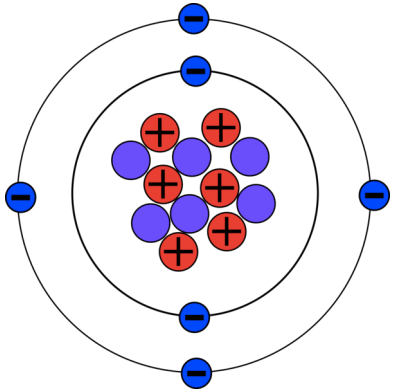
  - ３つの惑星の動きが厳密に解けないのと同様、水素分子すら厳密に解けない（原子は解ける）。
  - 各種近似解法などを利用しておりそれなりの成功 (MO - 分子軌道法, CI -配置間相互作用法など) を収めている。

【ご参考】Fock状態

- 粒子の状態を数で表現する方法。この表現は Fock空間で操作される。
- 状態は、各モードの直積になる。

$$|n_1, n_2, \cdots, n_i, \cdots\rangle \equiv |n_1\rangle|n_2\rangle \cdots |n_i\rangle \cdots$$

- フェルミオンの場合には、n は 0 か 1 になる → 量子ビットと同じ。



粒子数 (N)	フェルミ粒子フォック空間の基底[5]:11
0	$\|0,0,0,\cdots\rangle$
1	$\|1,0,0\cdots\rangle, \|0,1,0,\cdots\rangle, \|0,0,1,\cdots\rangle,\ldots$
2	$\|1,1,0,\cdots\rangle, \|0,1,1,\cdots\rangle, \|0,1,0,1,\cdots\rangle, \|1,0,1,0,\cdots\rangle,\ldots$
...	...

← 量子ビットそのもの

量子コンピューターの量子化学への適用(1/2)

- 分子が大きくなるとすべての電子、核子を古典コンピューターではシミュレーションできなくなる
  - 例：カフェイン($C_8H_{10}N_4O_2$)は 原子核数 = 102 (電子数 102)のように、原子核、電子の総数(N) に応じて**指数関数的**に計算が難しくなる（ハミルトニアン行列は 2^N になる）
  - 原子核は動かないものとする、内殻電子は反応に参加しない(内殻固定)とするなどの方法を取っているのが現状 (Qiskitも量子ビットが足りないのでそのような近似を実施している）。

- 核や電子を量子ビットにマッピングし、ハミルトニアンを量子コンピューターであつかえるように変換することで、効率的な計算ができるのではないかという期待が出てきた。カフェイン程度であれば近い将来シミュレーションできる可能性が高い。

先ほどのハミルトニアンは第二量子化で次に変換

$$H = \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

生成消滅演算子 a , a † はqubit に作用するパウリ行列のテンソル積に変換できる (Jordan Wigner, PARITY(IBM)変換など)

$$a_j^\dagger \leftrightarrow \mathbb{I} \otimes (\sigma_z)_{j-1} \otimes (\sigma^+)_j \otimes (\sigma_x)_{j+1} \otimes \ldots \otimes (\sigma_x)_N$$

https://arxiv.org/pdf/1812.09976.pdf

量子コンピューターの量子化学への適用(2/2)

- |ψ> がハミルトニアンの固有状態の場合、時間発展の結果エネルギー固有値が出てくる

$$U|\Psi\rangle = \exp(-iHt)|\Psi\rangle = \exp(-iEt)|\Psi\rangle$$

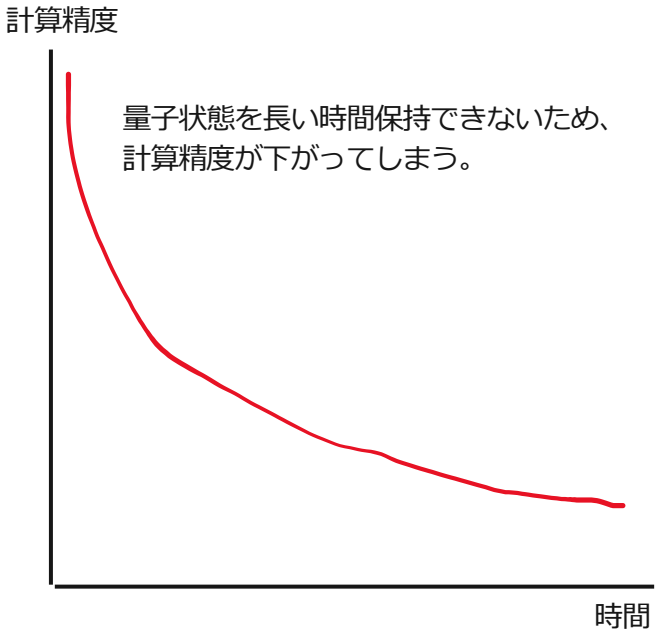
- これを量子コンピューターの回路に実装し、位相推定アルゴリズムを用いて エネルギー E の値を求めることができる。

  - ただし、このままでは絶対位相なので、CNOT 演算適用するなどのテクニックで相対位相にできる。

残念なお知らせ・・

- 前掲の方法 (fullCI という）では、回路の深さが大きすぎてノイズの影響を防げない。

- ただし、ノイズのない量子コンピューターができた場合には、fullCI で量子化学計算を実施することは見込まれている。

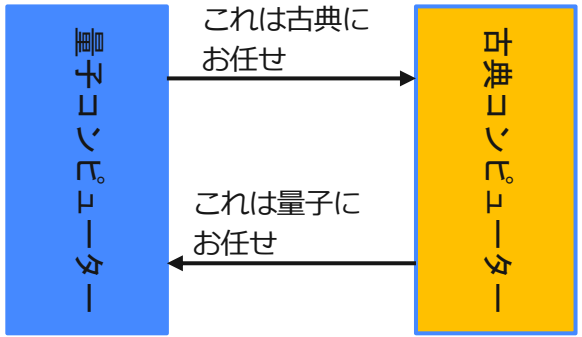


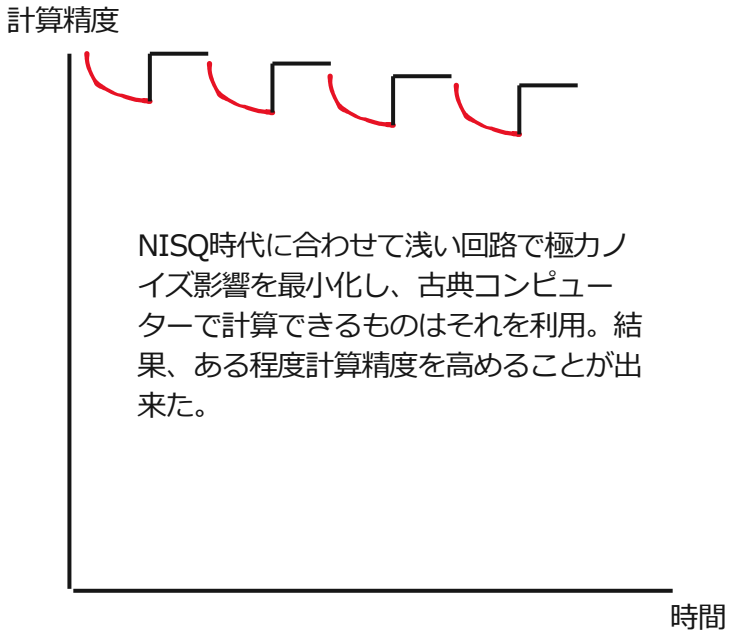
そこで考え出されたテクニックが量子コンピューター+古典コンピュータの組み合わせ

- 量子化学分野では、VQE (Variational Quantum Eigensolver) を使ってかなりの制度のエネルギー推定が可能になった。

  ✓ 量子コンピューターが得意なことは量子コンピューターで実行

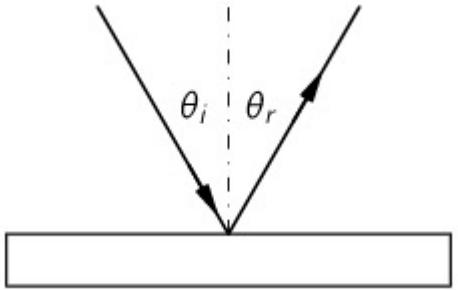
  ✓ 古典コンピューターが得意なことは古典コンピューターで実行





VQE – Variational Quantum Eigensolver

- NISQ 時代に量子コンピューターでもエネルギーなどのハミルトニアン固有値を求めるテクニック。
  - Alberto Peruzzo らによって考案 (2014)
- 元になっているのが「変分原理」(もしくは変分法)と呼ばれる手法。
  - 古典力学や幾何光学では最小作用の原理とも言われている。
  - 計算ではパラメータを変化させてある値が最小（もしくは最大）になるように変化させてみること。
  - VQE では波動関数を変分原理で探し、ハミルトニアンの期待値（即ち、エネルギー）が最小になるようにパラメータを調整する。パラメータの調整部分を古典コンピューターで実施する。



光の場合は、時間が最小になるように、パラメーターを調整すると観測されている光学的経路になる（フェルマーの原理）
上の図では、光の経路は
　　入射角=反射角
が一致した経路になる。

Feyman Lectures on Physics Volume I から抜粋

https://www.feynmanlectures.caltech.edu/I_26.html

量子力学での変分原理(1/2)

- ある演算子 A の固有状態 $|\psi_i\rangle$ があるとします。

$$A|\psi_i\rangle = \lambda_i|\psi_i\rangle$$

- ハミルトニアン H はこの固有状態を使って展開できます。

$$H = \sum_{i=1}^{N} \lambda_i |\psi_i\rangle\langle\psi_i|$$

- ハミルトニアンの期待値は次のように計算できます。

$$\langle H\rangle_\psi \equiv \langle\psi|H|\psi\rangle$$

量子力学での変分原理(2/2)

- さらに展開すると次のようになります。

$$\begin{aligned}\langle H\rangle_\psi = \langle\psi|H|\psi\rangle &= \langle\psi|\left(\sum_{i=1}^{N}\lambda_i|\psi_i\rangle\langle\psi_i|\right)|\psi\rangle \\ &= \sum_{i=1}^{N}\lambda_i\langle\psi|\psi_i\rangle\langle\psi_i|\psi\rangle \\ &= \sum_{i=1}^{N}\lambda_i|\langle\psi_i|\psi\rangle|^2\end{aligned}$$

- 変分原理は、λ_{min} に近くなるように、ψ=ψ(θ) のシータパラメータ(複数) を変動させることです。

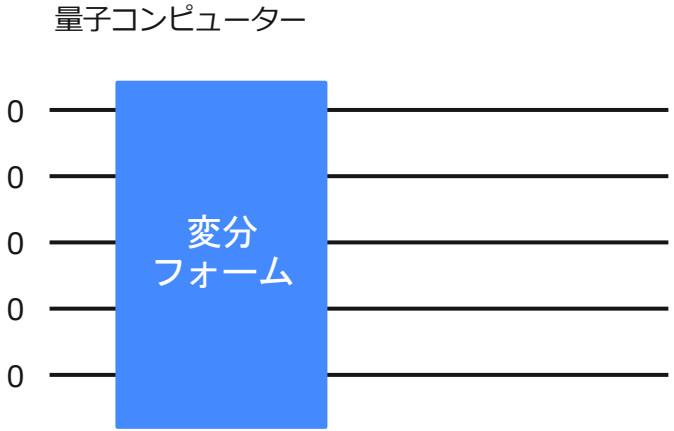
$$\lambda_{min} \leq \langle H\rangle_\psi = \langle\psi|H|\psi\rangle = \sum_{i=1}^{N}\lambda_i|\langle\psi_i|\psi\rangle|^2$$

※ ここまでは量子コンピューターとはあまり関係なく一般的な話です。

VQE ~ 変分原理を量子回路に適用(1/3)

- 変分原理を量子回路に適用するのは極めてシンプルです。

1. 変分フォームと呼ばれる回路を用意します。
2. 初期状態 |000…0> から初めて変分フォーム (Ansatzとも呼ばれる）を適用します。
3. この結果、様々な重ね合わせからなる状態 |ψ> ができます。



$|\psi\rangle$

よく使われる変分フォーム

- Rx
- RxRy
- UCCSD

※ 量子化学への適用の場合には、ゼロから始めるのではなく、Hatree Fock 状態から始めることが多いようです。

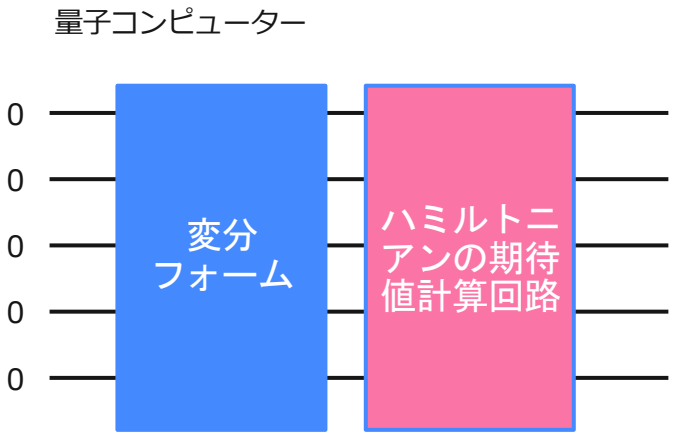
VQE ~ 変分原理を量子回路に適用(2/3)

- 変分原理を量子回路に適用するのは極めてシンプルです。

4. 次に ハミルトニアン の期待値を計算する回路を組み込みます。

   statevestor_similator を利用している場合には、ハミルトニアンの期待値は行列の掛け算で、古典コンピューターで導出できます（Qiskit textbook ではそうしている）

5. 最終的に出てきた結果をまとめます



ハミルトニアンは、 Jordan Wigner, PARITY 変換でパウリ行列 (X, Y, Z) の文字列 (Pauli String) となる。それぞれの行列の期待値を計算し、足し算すればハミルトニアンの期待値が出る。

例： Z の期待値は、以下のように求まる。

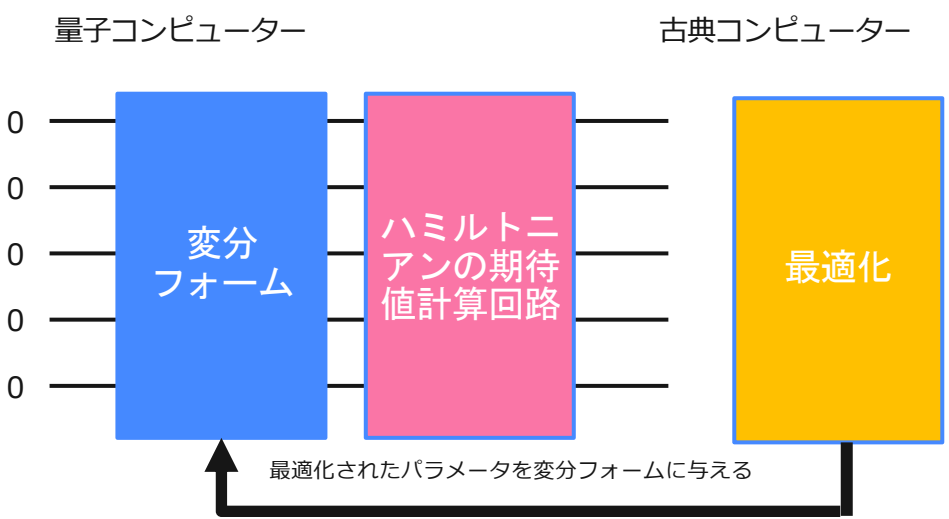
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$\langle\psi|Z|\psi\rangle = |\alpha|^2 - |\beta|^2$$

VQE ~ 変分原理を量子回路に適用(3/3)

- 変分原理を量子回路に適用するのは極めてシンプルです。

6. 出てきた結果から最適化を古典コンピューターを使って実施します。
7. そのパラメータ変動を変分フォームに与え、計算を繰り返します。



よく使われる最適化
- COBYLA
- SLSQP

※ 機械学習でよく使われる勾配降下法は、局所解が出てしまうため VQE ではあまり使われないようです。

【ご参考】最適化について

Qiskit では、パラメータ最適化は scipy の最適化ライブラリを利用しています。

https://docs.scipy.org/doc/scipy/reference/optimize.html#module-scipy.optimize

最適化方法	説明
minimize(method="l-bfgs-b")	境界制約付き記憶制限 BFGS 法
minimize(method="tnc")	truncated Newton 法(Newton-CG 法)
minimize(method="cobyla")	線形近似法
minimize(method="slsqp")	逐次最小 2 乗法

エネルギーを求める問題ではないが、変分法で所望の波動関数を探す簡単な例

- 変分法で、アダマール演算子を適用した状態を探してみる。

厳密解

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

0 の確率が50%
1 の確率が50%

という状態

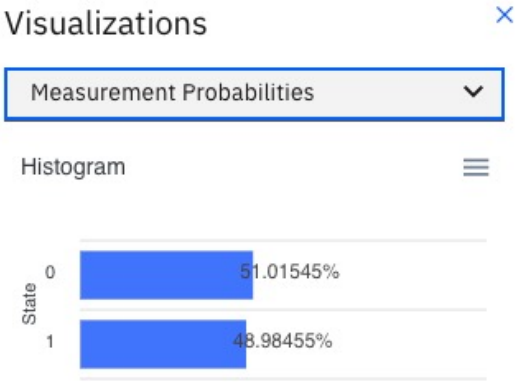
```
from qiskit.aqua.components.optimizers import COBYLA

# Initialize the COBYLA optimizer
optimizer = COBYLA(maxiter=500, tol=0.0001)

# Create the initial parameters (noting that our single qubit variational form has 3
parameters)
params = np.random.rand(3)
ret = optimizer.optimize(num_vars=3, objective_function=objective_function,
initial_point=params)

# Obtain the output distribution using the final parameters
qc = get_var_form(ret[0])
counts = execute(qc, backend, shots=NUM_SHOTS).result().get_counts(qc)
output_distr = get_probability_distribution(counts)

print("Target Distribution:", target_distr)
print("Obtained Distribution:", output_distr)
print("Output Error (Manhattan Distance):", ret[1])
print("Parameters Found:", ret[0])
```

結果

Target Distribution: [0.5 0.5]
Obtained Distribution: [0.5128, 0.4872]
Output Error (Manhattan Distance): 0.0021999999999999797
Parameters Found: [1.55048593 1.44476873 -0.3125408]

IBM Quantum Experience で確認してみる

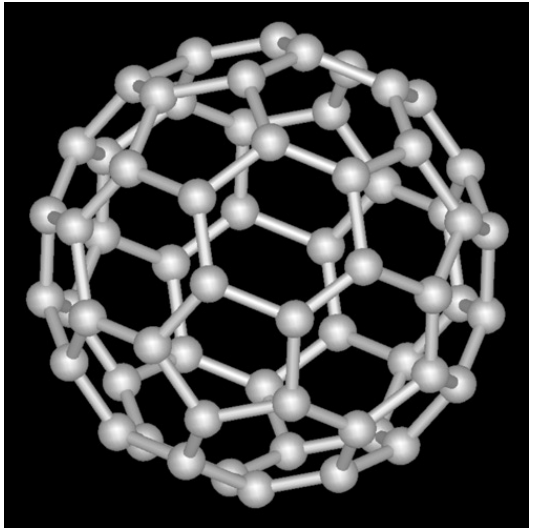
```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[1];
creg c[1];

u3(1.55048593,1.44476873, -0.3125408) q[0];
```



想定の状態に近い。

分子のシミュレーション

- VQE を利用して分子のシミュレーションを実施できます。
  - ✓ 分子の基底エネルギーがわかります（分子の安定性）
- なぜ着目されているのか？
  - ✓ 古典コンピューターで時間がかかっている、もしくはできない計算により、新しい物質の発見（産業利用、創薬などに適用）が期待されています。
  - ✓ 三菱ケミカル様は、リチウムイオン電池の方式について量子コンピューターを利用した研究を実施しています。



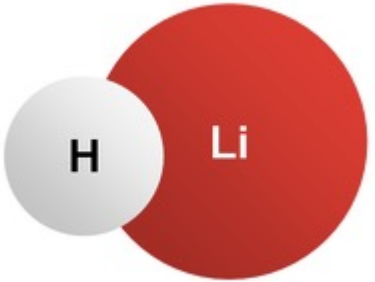
フラーレンは、1970年に予想され、1985年に発見。以降、カーボンナノチューブなど炭素繊維素材産業が劇的に飛躍した。

https://ja.wikipedia.org/wiki/フラーレン

QisKit を利用した分子シミュレーション概要(1/2)

1. PySCF ドライバー等を利用して、分子の構造を設定します。

```
driver = PySCFDriver(atom="Li .0 .0 .0; H .0 .0 " + str(dist),
unit=UnitsType.ANGSTROM, charge=0, spin=0, basis='sto3g')
```



LiH：水素化リチウム

2. ドライバーを実行し、ハミルトニアンの係数を計算します。

```
molecule = driver.run()
ferOp = FermionicOperator(h1=molecule.one_body_integrals,
h2=molecule.two_body_integrals)
```

$$H = \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s$$

ここの計算

QisKit を利用した分子シミュレーション概要(2/2)

3. 変分フォーム、変換方法を指定し、実行する。

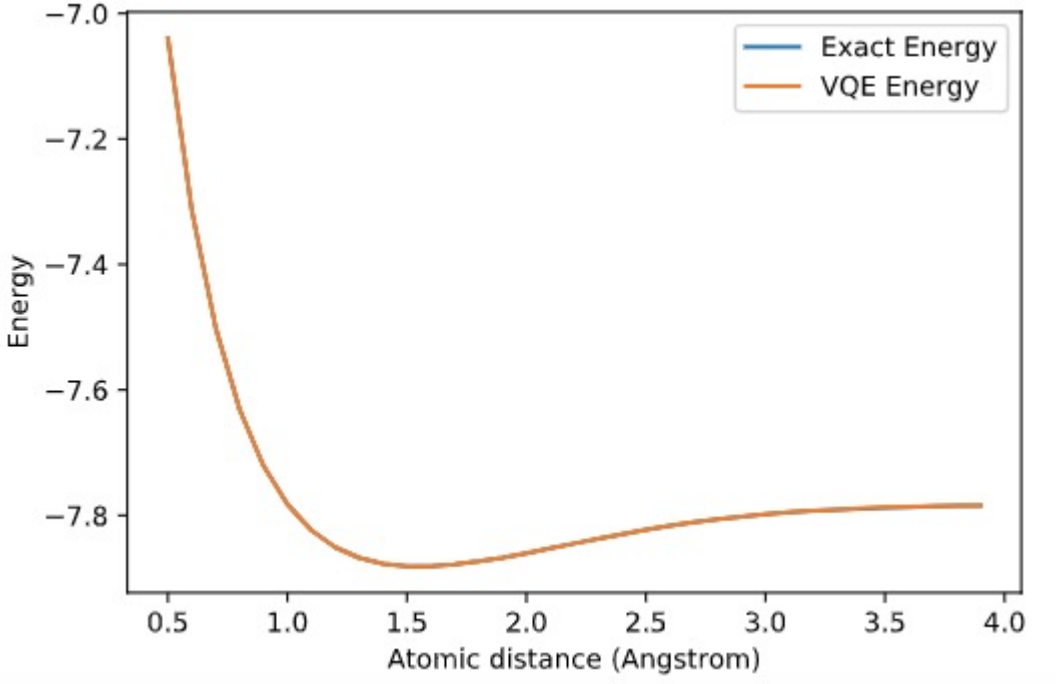
- ここでは 変分フォームは UCCSD, 変換は PARITY

```
var_form = UCCSD(
   qubitOp.num_qubits,
   depth=1,
   num_orbitals=num_spin_orbitals,
   num_particles=num_particles,
   initial_state=initial_state,
  qubit_mapping='parity'
)
vqe = VQE(qubitOp, var_form, optimizer)
results = vqe.run(backend)['energy'] + shift
```

変分フォームを設定します。

Aqua の VQE を呼び出します。Qubit に対する演算子 (qubitOp) 変分フォームと最適化に利用するオプティマイザーを指定します。

結果



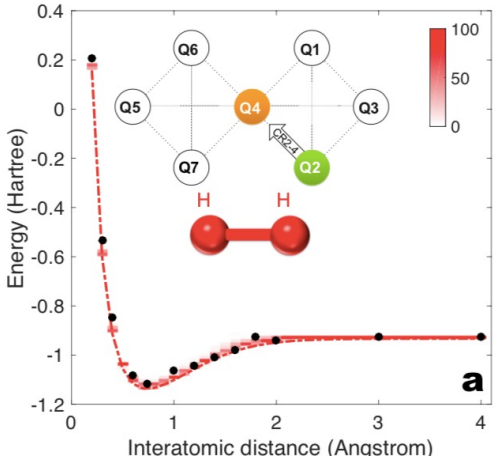
線形演算パッケージを利用した
Exact Energy と VQE 計算結果は
ほぼ同じ結果を出しています。

註： シミュレーターでかつノイズなしで計算しています。

実デバイスでの実行

実デバイスで実行するには工夫が必要です。

- UCCSD では回路が深すぎるので、ハードウェアに最適化された EfficientSU2 変分フォームなどを利用する。




```
exact_solution = NumPyEigensolver(qubitOp).run()
print("Exact Result:", np.real(exact_solution.eigenvalues))
optimizer = SPSA(max_trials=100)
var_form = EfficientSU2(qubitOp.num_qubits,
entanglement="linear")
vqe = VQE(qubitOp, var_form, optimizer=optimizer)
ret = vqe.run(quantum_instance)
vqe_result = np.real(ret['eigenvalue']) print("VQE Result:",
vqe_result)
```

結果

Exact Result: [-1.86712098]
VQE Result: -1.80040360007339

業界で必要とされる精度 (0.0016 Hartree)には達していないがかなり近いと言える

Qiskit Summer School 2020 資料から抜粋

まとめ

- NISC 時代の量子コンピューターでも VQE などのテクニックを利用して分子のシミュレーションができます。
- これを応用するテクニックを身につけることで、将来お客様とともに新しい分子の発見や創薬などの基礎となりますので、皆さんぜひ身につけましょう！

参考文献

- 「量子コンピュータの量子化学計算への応用の現状と展望」 中田真秀
  https://www.slideshare.net/NakataMaho/ss-117321322
- 「化学者のための量子コンピュータを用いた量子化学計算入門」 杉﨑 研司
  （※以前 github.com で公開されていましたが、いまはアクセスできなくなっています）

Takehiko Amano