Blueqat ❤ 量子化学

MDR株式会社　加藤 拓己
GitHub/Qiita: gyu-don

About me

加藤 拓己　/　GitHub/Qiita: gyu-don

現在: MDR株式会社でBlueqatの開発など
以前: プラント会社で鉄鋼のアニーリング (炉の計装制御)
　　某IT企業でエンジニア。深層学習とビットコインをちょっとかじる
学部/修士時代: 東北大学の量子光学の研究室で実験屋さん
好きなプログラミング言語:
　Rust, C, C++, Python, OpenQASM

化学は?
　小学生の頃にハマったが、高校生くらいから物理の方が楽しくなってきた
　「ハートリーフォック」という言葉を去年の夏に初めて知った

About Blueqat

オープンソースの量子コンピュータのシミュレータ。
https://github.com/mdrft/Blueqat

- 天才じゃない人でも簡単に使える
  - ちょっと賢い人はちょっと賢く使える
- 他ライブラリとの連携も頑張る
  - BlueqatからIBMの実機を動かすこともできる(要qiskit)
- チュートリアルやMDRブログでの実装例が充実
  - 地味に充実してます。動くサンプルの数では、下手したら世界一かも
- 使ってくれる人、開発手伝ってくれる人、絶賛募集中

量子コンピュータのシミュレータ?

量子コンピュータの動きを模倣する、ただの行列計算ソフト。

- ➔ なんで量子コンピュータを普通のコンピュータで模倣できるの?
  - ◆ 量子コンピュータがやっているのは、ただの行列計算
  - ◆ 量子コンピュータで計算できる問題は普通のコンピュータでも計算できる
- ➔ じゃあ量子コンピュータいらなくない?
  - ◆ 規模が大きくなればシミュレータでは厳しくなる
    - 1量子ビット増えたら、計算時間もメモリ消費も倍になる
    - 普通のパソコンでは30量子ビットは厳しい (時間かかるし、メモリ16GBじゃ足りない)
    - スパコンでは、ランダム量子回路とよばれるものをAlibabaが81量子ビットで計算
- ➔ その他、実機との違いは?
  - ◆ 実機では量子状態を直接見ることはできないが、シミュレータならできる
  - ◆ 実機では計算にエラーがあるが、シミュレータなら理想的な計算ができる

乱立するゲートシミュレータ

- Qiskit (IBM)
  - 一番メジャーなゲートシミュレータ
- Cirq (Google)
- PyQuil (Rigetti)
- Blueqat (MDR)
  - とてもおすすめだよ！
- Q# (Microsoft)
  - Q#言語とC#言語で書く
- SymPy
  - ゲートシミュレータというよりは、単に量子物理で使う計算ができるという感じ

他にも、ProjectQ, Quipper, Qulacs, QuTiP, StrawberryFieldsなど多数
どのライブラリ使っても、ゲートをぽちぽち並べないといけないのは一緒なので好みの問題

初めての人にBlueqatを勧める5つの理由

1. **覚えることが少ない**
   Circuitクラスの使い方だけ覚えたら回路が作れる
2. **回路の書き方が直感的で簡単**
   メソッドチェーンやスライス表記で短い・分かりやすい
3. **安心の日本語コミュニティ**
   MDRのSlackや勉強会で質問をいただければ開発者が直接答えます
4. **簡単QAOA/VQE**
   NISQアルゴリズムの代表格ともいえるQAOAやVQEも簡単に動かせる
   アニーリング用のQUBOを動かすこともできる
5. **環境構築らくらく**
   NumPy/SciPyを使っていて、C++を直接書いたりしていないので、
   依存関係やライブラリなどが原因でインストールに躓くことが少ない

早く簡単に書けて、生産性が高い

他社ライブラリ

```
from qiskit import QuantumCircuit,  \
ClassicalRegister, QuantumRegister, \
execute, BasicAer

q = QuantumRegister(3, ‘q’)
c  = ClassicalRegister(3, ‘c’)
circ = QuantumCircuit([q, c])
circ.h(q[0])
circ.cx(q[0], q[1])
circ.cx(q[0], q[2])
circ.measure(q, c)
backend = BasicAer.get_backend(‘qasm_simulator’)
print(execute(circ, backend,
shots=1024).result().get_count(circ))
```

Blueqat

```
from blueqat import Circuit
c = Circuit().h[0].cx[0, 1].cx[0, 2].m[:]
print(c.run(shots=1024))
```

- 学習
- 研究
- 開発

誰にでも、どんな用途にも使いやすいライブラリ
(を目指しています)

Blueqatの
インストール方法

```
pip install blueqat
```

「計算に必要な資源がどれくらい増加?」←え?

・問題を解くのにかかる時間は「何の道具を使って解くか」で変わる
・問題が大きくなったときの増加のしかたは、道具の影響を受けにくい

例:「1から100まで、足し算してください」

・暗算するのか、電卓で計算するのか、Excelで計算するのか?
・かかる時間は全然違う

「1から200までの足し算をするには、1から100までのときの何倍、時間が必要ですか?」

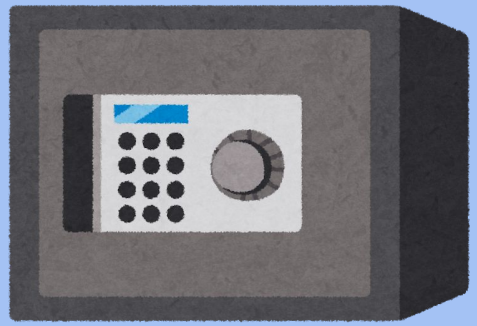
・暗算の人も、電卓の人も、だいたい2倍
・Excelの人も(Excelを立ち上げる手間は省けるかもしれないが、だいたい)2倍

けれど、問題の解き方(=アルゴリズム)によっては、増加の度合いは変わる

・1から100までの足し算は、 1 + 100 + 2 + 99 + … = 101 + 101 + … = 101 x 50
・この方法だと、1から200までになっても、1から1000までになっても、ほぼ同じ時間で解ける

※このスライドでは、桁数のことを考えていない。計算量の観点からは不適切だが、話を簡単にするため省略した

組み合わせ爆発



大切な金庫の暗証番号を忘れた！
仕方ないので、片っ端から暗証番号入れてみる！

番号が3桁→最悪でも1,000回で開く
　　平均500回で開く(頑張れそう)
番号が4桁→平均5,000回(まだ頑張れる)
番号が5桁→平均50,000回(さすがにヤバい)
番号が6桁→平均500,000回(もう諦めたい)
番号が7桁→平均5,000,000回(溢れ出す絶望感)
番号が8桁→平均50,000,000回(金庫と一生向き合う覚悟)
番号が9桁→平均500,000,000回(先祖代々受け継いでいく)

解きたい問題が大きくなると時間がぐんぐんかかるようになる問題は
世の中に意外といっぱいある
→こうした問題に対抗できる(かもしれない)のが量子コンピュータ

(金庫が開くことはないと思いますが……
真面目な話をすると、ノーヒントで総当たりする問題は量子コンピュータにも難しく、Groverのアルゴリズムより効率よくは無理)

量子コンピュータは巨大な並列計算機か?

→そうだけど、そうじゃない。

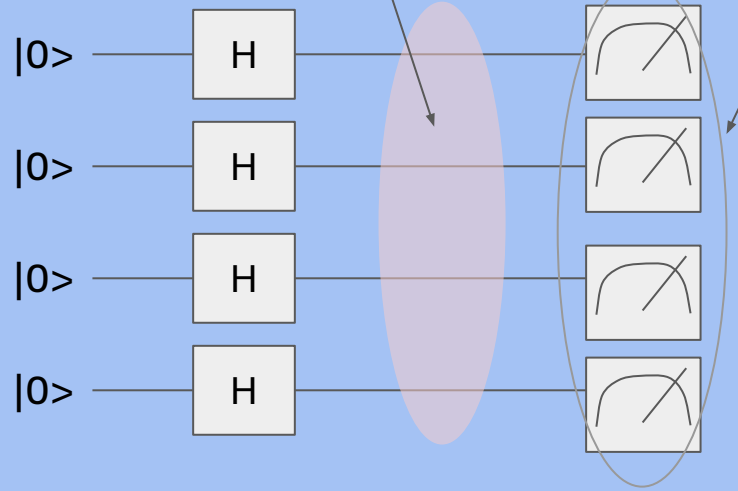
通常のコンピュータでは

各々のビットは0か1のどちらかを取る。

0000
0001
0010
0011
0100
0101
0110
0111
1000
:
:

量子コンピュータでは

|0000> + |0001> + …
全部のパターンの重ね合わせ



測定して得られる
値は1つだけ。
ただ「重ね合わせる」だけでは
意味がない

量子の性質を活かした
専用の計算方法
「量子アルゴリズム」
が必要

そろそろ
量子化学の話を。

量子化学の目的と課題

- ➔ 量子化学計算では、電子の状態を求めたい
  - ◆ 原子核のまわりを電子がぐるぐる
  - ◆ 電子はエネルギーの低い「軌道」をぐるぐるしていたい
  - ◆ けれど、他の電子がいると、自分のぐるぐるに専念できない
    - 他の電子との干渉！　電子数が大きいと計算が一気に重くなる！
- ➔ 量子化学計算は重い計算のオンパレード
  - ◆ 真面目にやってたら、スパコンを何年ぶん回しても解けない
  - ◆ たくさんの近似計算を入れてもまだ重い
- ➔ 電子の量子状態を扱うなら、量子コンピュータが向いている？
  - ◆ 電子の重ね合わせ状態⇔量子ビットの重ね合わせ状態
  - ◆ 自然界は、電子状態をいとも簡単に「計算」している

電子状態求めて何が嬉しいの?

電子状態を知ることは、物質の性質を知ること

- ➔ 新素材の分子設計
  - ◆ 太陽電池の材料や有機ELなどの開発
- ➔ 新薬の開発
  - ◆ 創薬候補分子の探索
- ➔ 触媒材料の分子設計
  - ◆ 化学工業でのプロセス改善→低コスト化、省エネルギー化、環境への負担減

参考: HPCシステムズ https://www.hpc.co.jp/chem/service/seminar/

(参考) 京コンピュータによる成果発表件数

	課題の種類（複数の課題が関与する成果発表は各「課題の種類」の件数としてカウントされます） 表示オプション										合計	正味の成果発表件数
	HPCI利用研究分野											
	バイオライフ	物質材料化学	環境防災減災	工学ものづくり	物理素粒子宇宙	エネルギー	情報工学計算機科学	数理科学	社会システム科学	その他		
査読付き論文	176	374	170	168	158	41	15	24	8	19	1153	1069
査読なし論文	35	18	42	30	11	5	0	1	1	1	144	139
国際会議・シンポジウム	263	285	376	235	285	128	2	14	1	14	1603	1552
国内学会・シンポジウム	461	307	258	354	197	98	4	9	4	26	1718	1648
研究会等	203	125	139	68	114	10	2	4	1	15	681	662
一般向け講演会等	168	32	45	33	36	0	0	0	0	1	315	313
新聞/TV/雑誌/WEB配信等	316	54	157	10	55	0	0	1	0	4	597	590
書籍	33	5	7	0	0	0	0	0	0	0	45	45
プログラム・ＤＢ公開	5	4	0	1	1	0	0	0	0	0	11	11
特許出願	5	4	0	0	0	0	0	0	0	0	9	9
特許取得	6	6	0	0	0	0	0	0	0	1	13	12
合計	1671	1214	1194	899	857	282	23	53	15	81	6289	6050
	6289											

データベース最終更新日：2019年01月29日

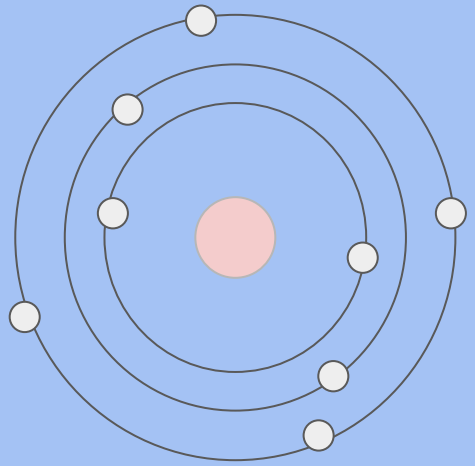
出典: 高度情報科学技術研究機構「成果発表データベース」

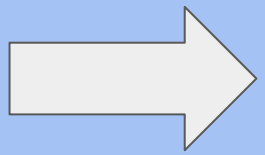
一瞬で分かる量子化学

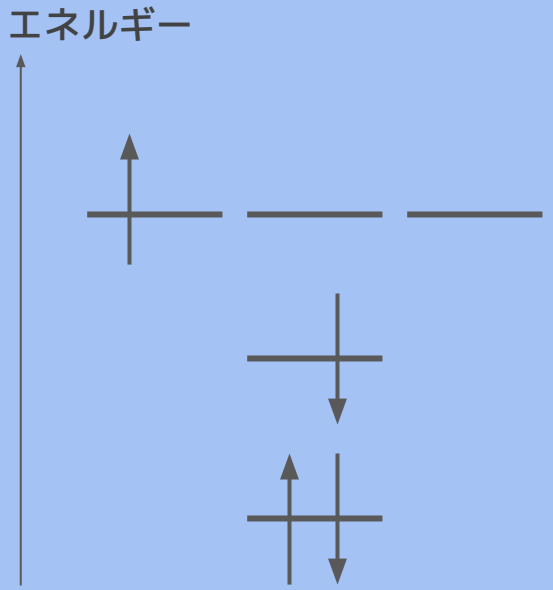
電子は、エネルギーの低い場所に住みたい
⇒ここまでは簡単

けれど、ご近所関係とか考えると……
⇒一気に話が複雑に

電子軌道







同じ軌道に
同じスピンの
電子は入れない

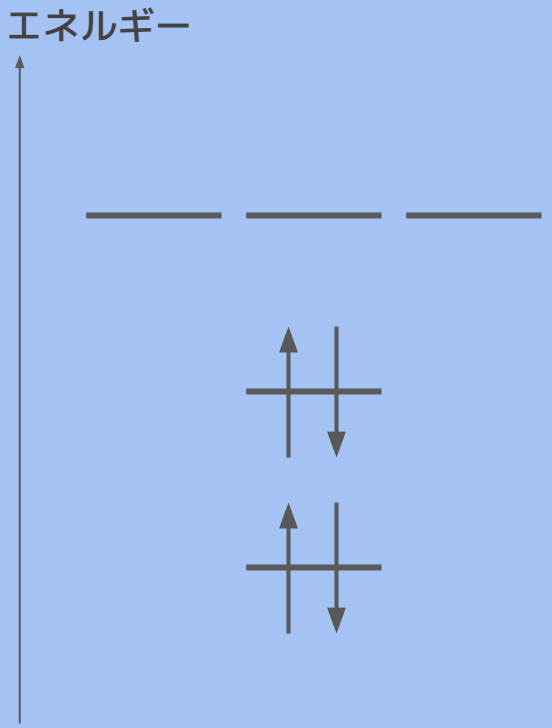
電子は原子核のまわりを、くるくる回っている
(惑星の公転みたいな感じ)
量子力学的にはそうじゃないが、あまり深入りしない。

電子が回れる「軌道」は決まっている
→右の図では横棒で表現

電子は「スピン」という量を持つ
(惑星の自転みたいな感じ)
→上向き、下向きの2種類あるので、↑と↓で表現

各軌道には定員があり、↑と↓が1つずつ入れる

電子相関を考えない場合の電子配置

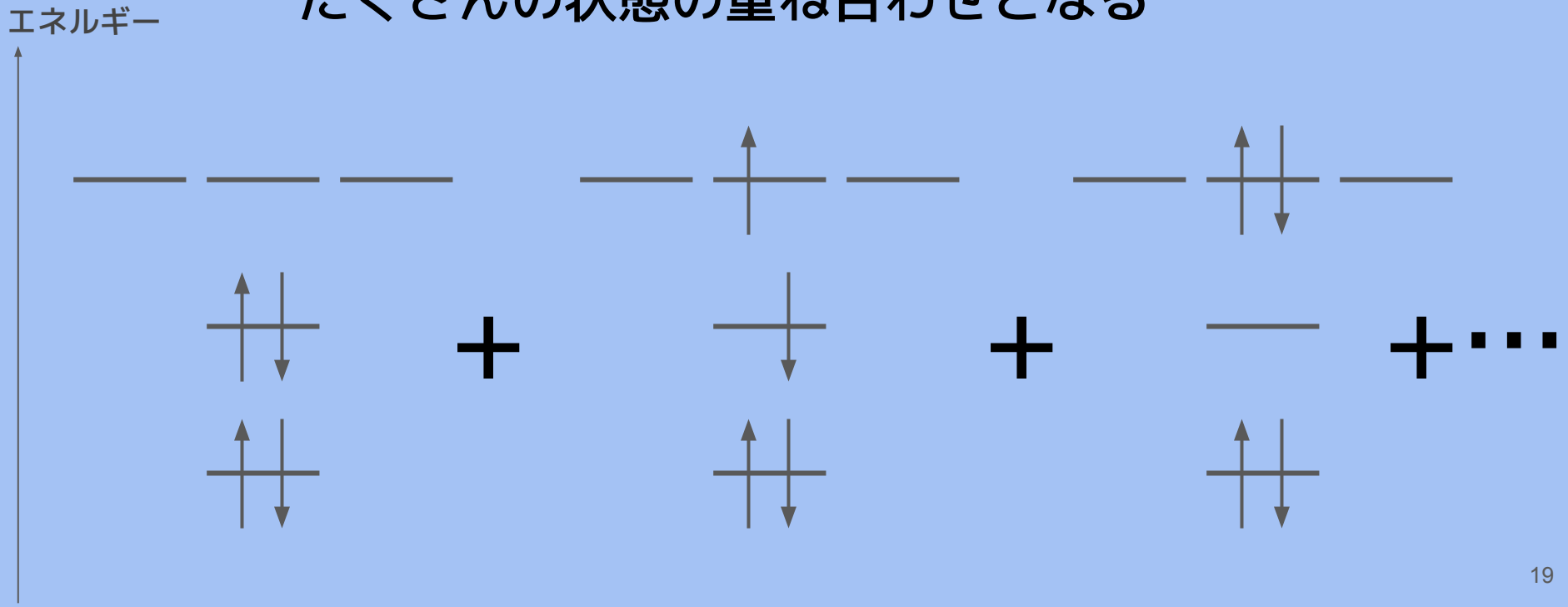


電子は、エネルギーの低い場所に住みたい
⇒低い軌道から詰めていく

けれど、電子相関を考えると……
⇒一気に話が複雑に

電子相関も考えた電子配置

たくさんの状態の重ね合わせとなる



量子コンピュータで
やっていきましょう

VQEによる量子化学計算の大まかな流れ

目標: 基底状態のエネルギーを求めたい
(ポストHartree-Fock法の一種)

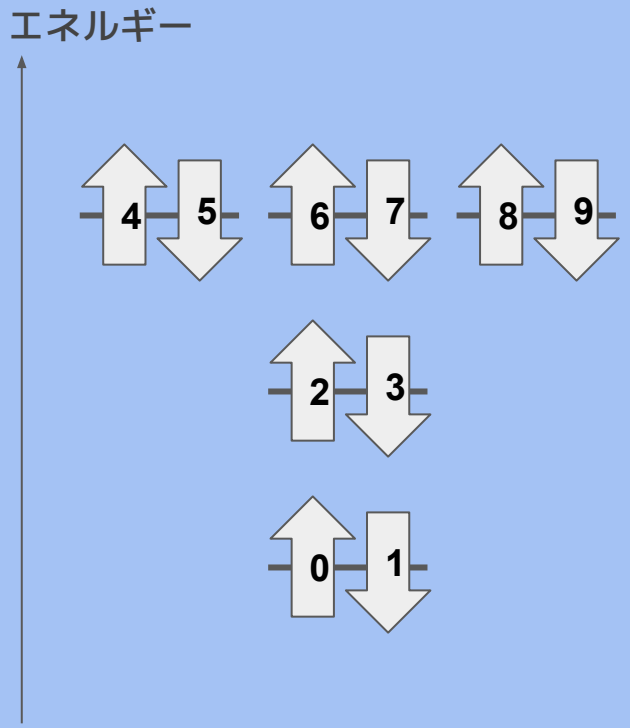
1. 電子状態に対応する量子回路を準備する
   →試行関数、ansatzなどと呼ばれる
2. 量子回路のエネルギーを測定する
3. 試行関数を少し変えて、1に戻る
   →測定するエネルギーが最も低くなるように最適化を行う

VQEによる量子化学計算の大まかな流れ

目標: 基底状態のエネルギーを求めたい
　　(ポストHartree-Fock法の一種)

1. 電子状態に対応する量子回路を準備する
   →**試行関数**、ansatzなどと呼ばれる
2. 量子回路のエネルギーを測定する
3. 試行関数を少し変えて、1に戻る
   →測定するエネルギーが最も低くなるように最適化を行う

軌道に番号を割り振る

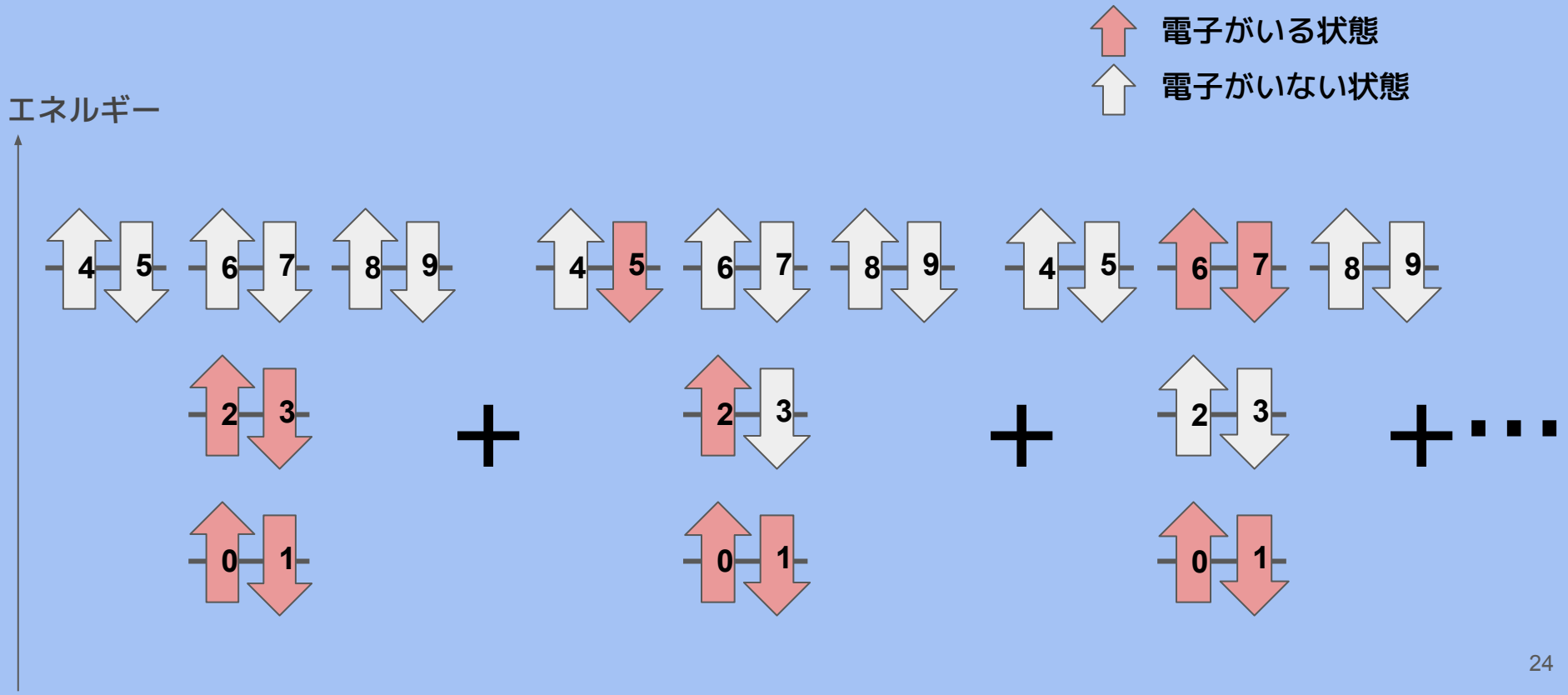


軌道の番号→量子ビットの番号に対応
電子がある→|1>, ない→|0>

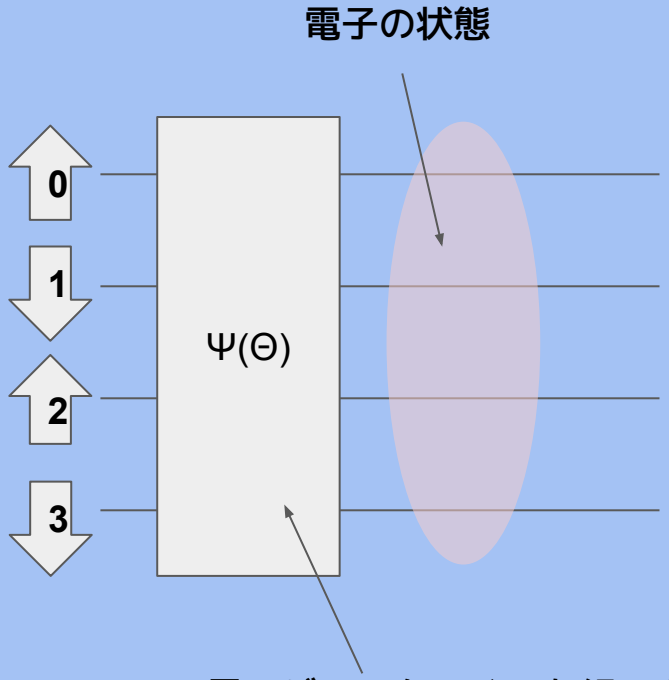
左の例では10量子ビットあれば載せられる

※量子コンピュータでの量子化学計算ではBravyi-Kitaev基底を使うことが多く、
そのように基底を変えた場合、この関係は崩れる。
けれど、概要を理解する上では特に支障はないので、今回はその話はしない。

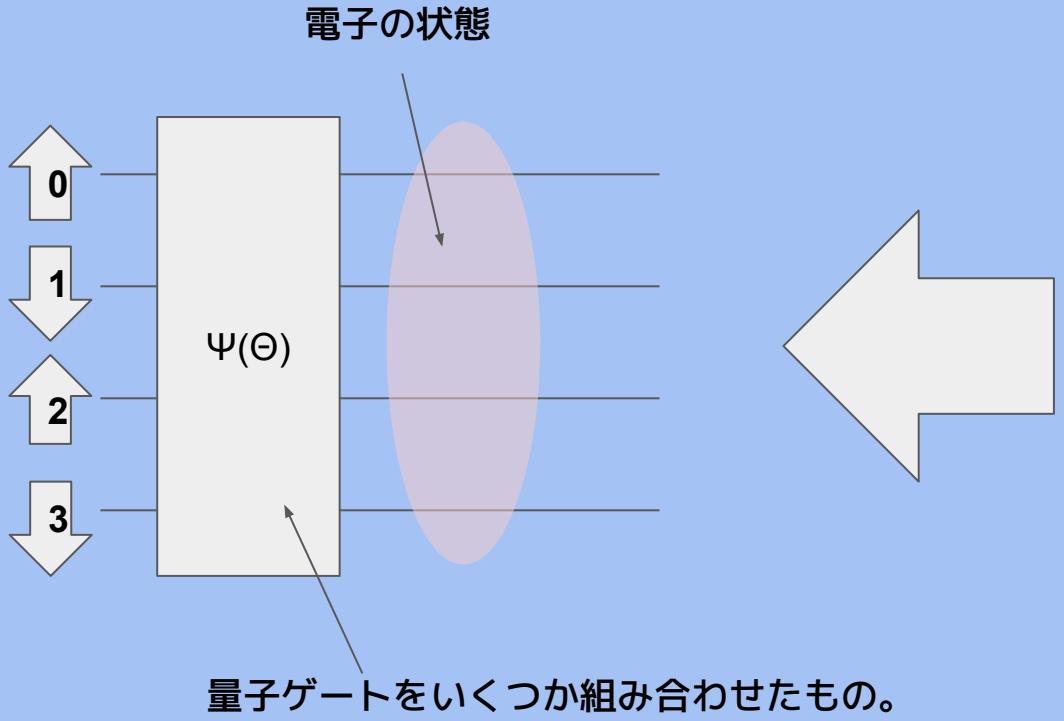
重ね合わせ状態を量子ビットで表現できる



重ね合わせ状態を量子ビットで表現できる



重ね合わせ状態を量子ビットで表現できる



試行関数

VQEによる量子化学計算の大まかな流れ

目標: 基底状態のエネルギーを求めたい
　　(ポストHartree-Fock法の一種)

1. 電子状態に対応する量子回路を準備する
   →試行関数、ansatzなどと呼ばれる
2. 量子回路のエネルギーを測定する
3. 試行関数を少し変えて、1に戻る
   →測定するエネルギーが最も低くなるように最適化を行う

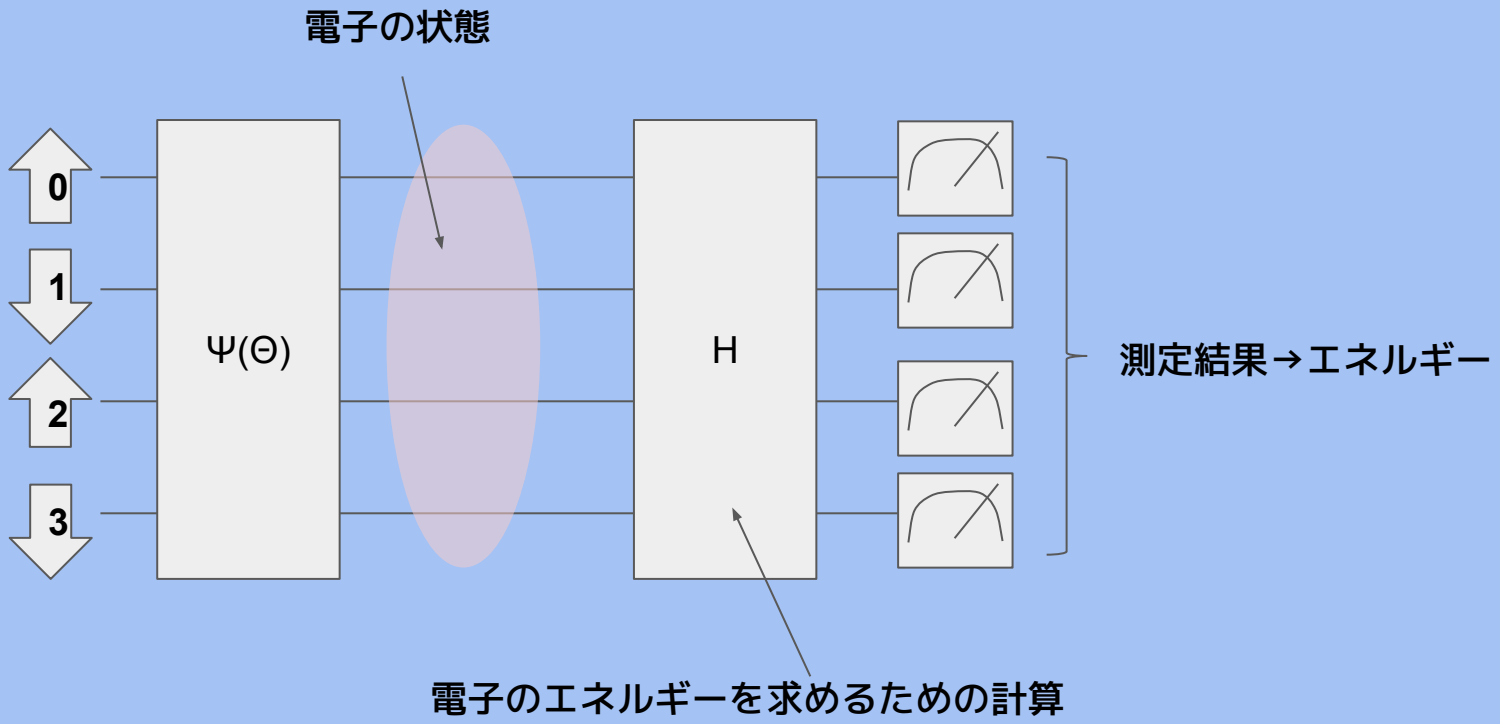
エネルギー?

電子たちにとって快適な場所→低エネルギー

どこに住んだらエネルギーがいくらか?
→知るための計算方法を、VQEをする前に求めておく

どこに住むのが低エネルギーか?
→計算してみないと分からない→VQEで求める

エネルギーを求める

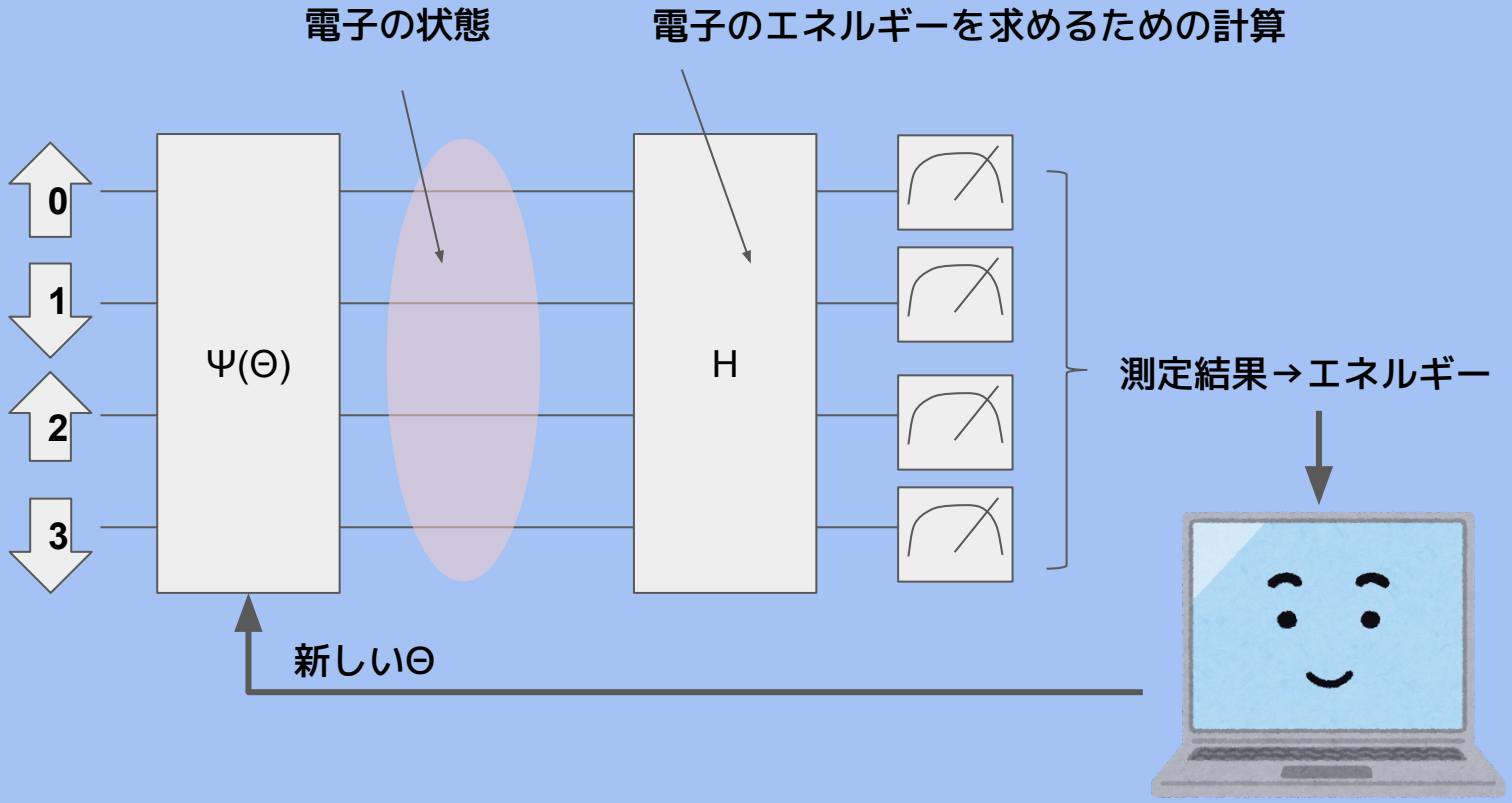


VQEによる量子化学計算の大まかな流れ

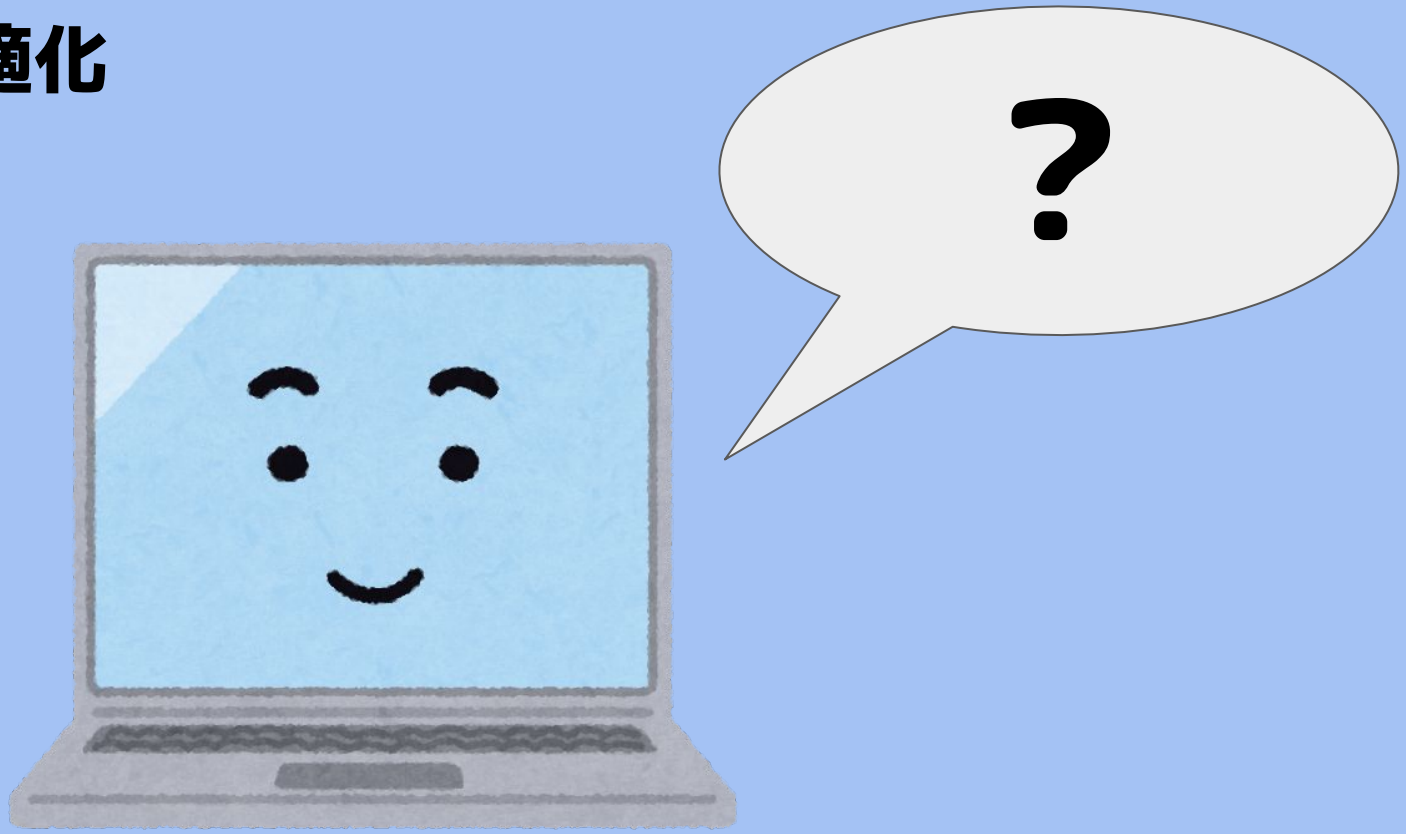
目標: 基底状態のエネルギーを求めたい
　　(ポストHartree-Fock法の一種)

1. 電子状態に対応する量子回路を準備する
   →試行関数、ansatzなどと呼ばれる
2. 量子回路のエネルギーを測定する
3. 試行関数を少し変えて、1に戻る
   →測定するエネルギーが最も低くなるように**最適化**を行う

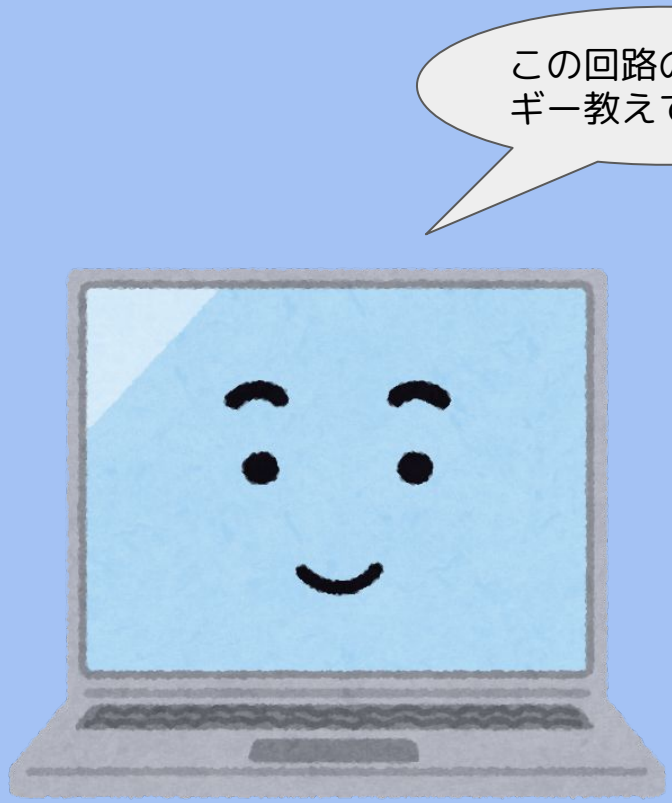
量子古典ハイブリッド

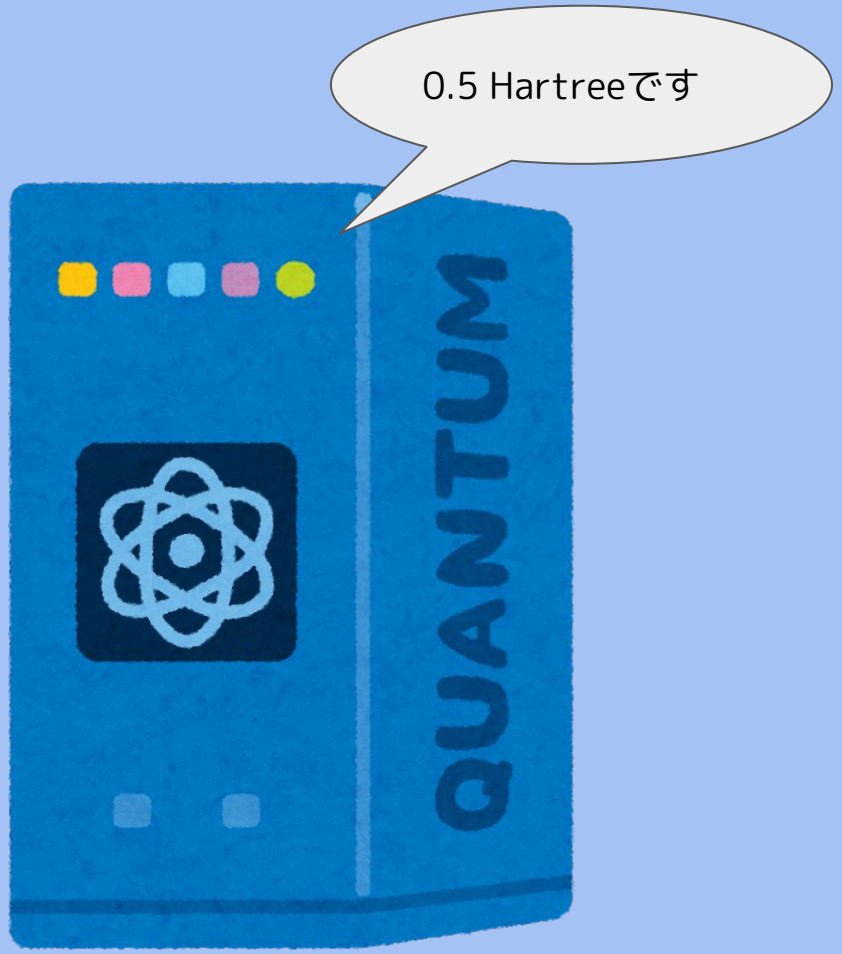


最適化



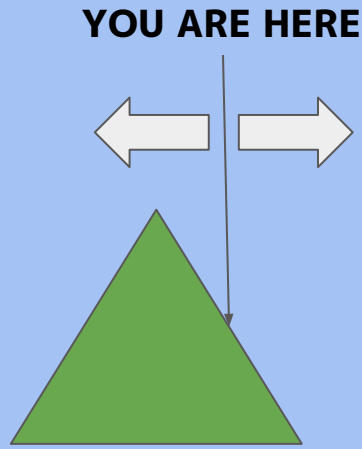
ブラックボックス最適化





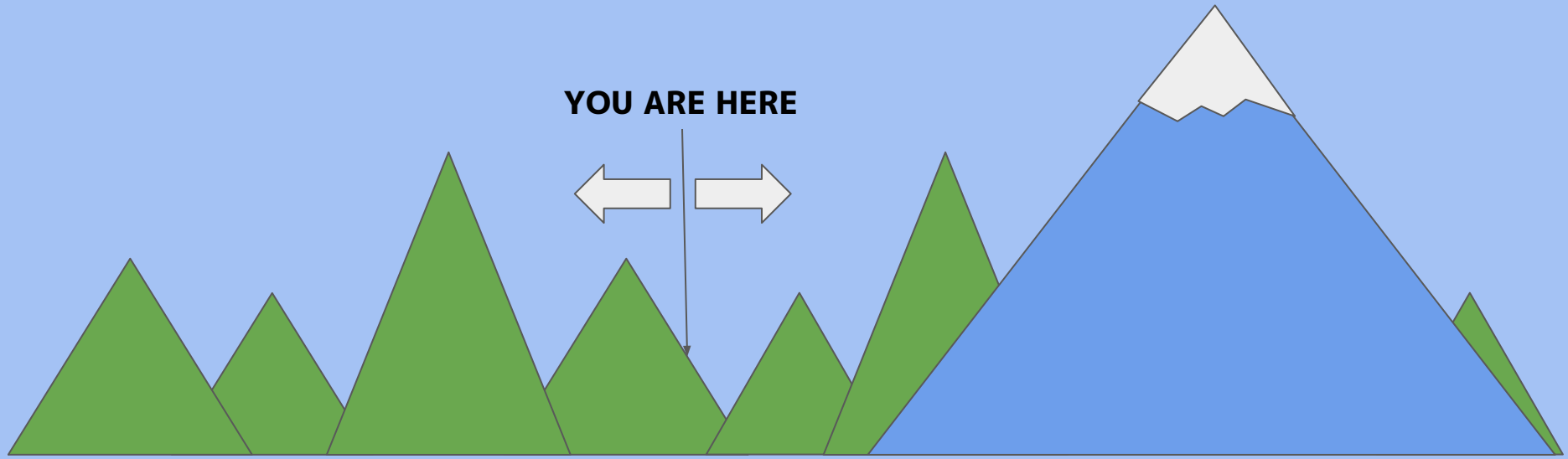
局所最適と全体最適

例: より高いところを目指したい。どちらに行けばいいか?



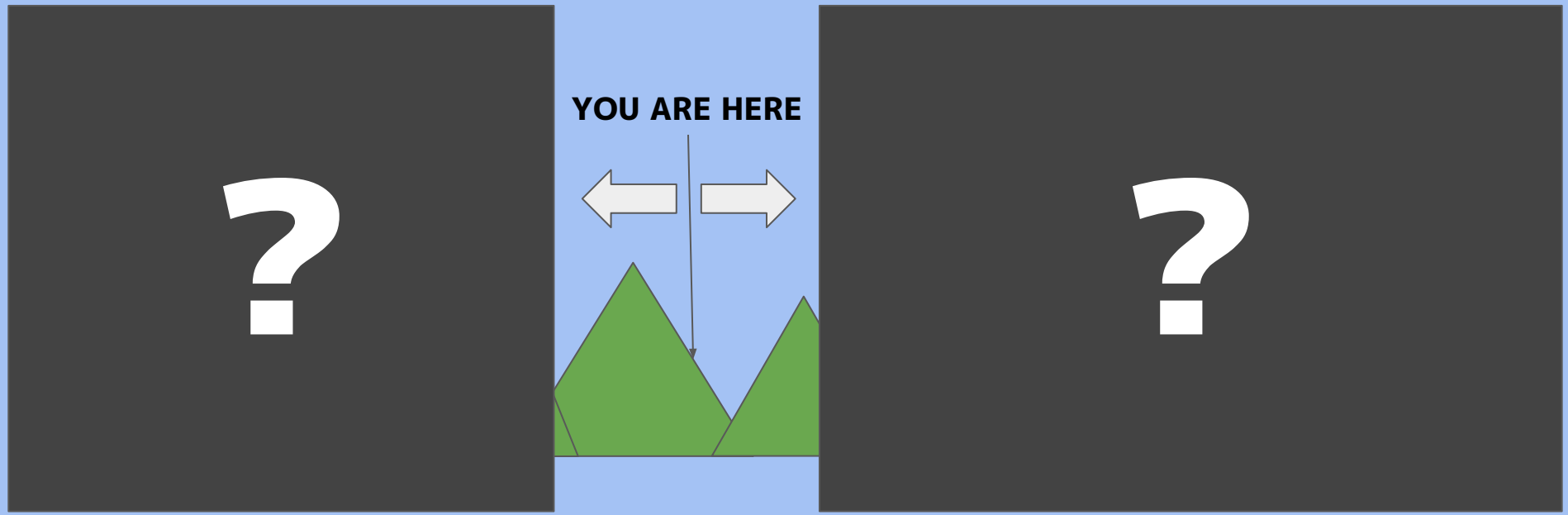
局所最適と全体最適

例: より高いところを目指したい。どちらに行けばいいか?



局所最適と全体最適

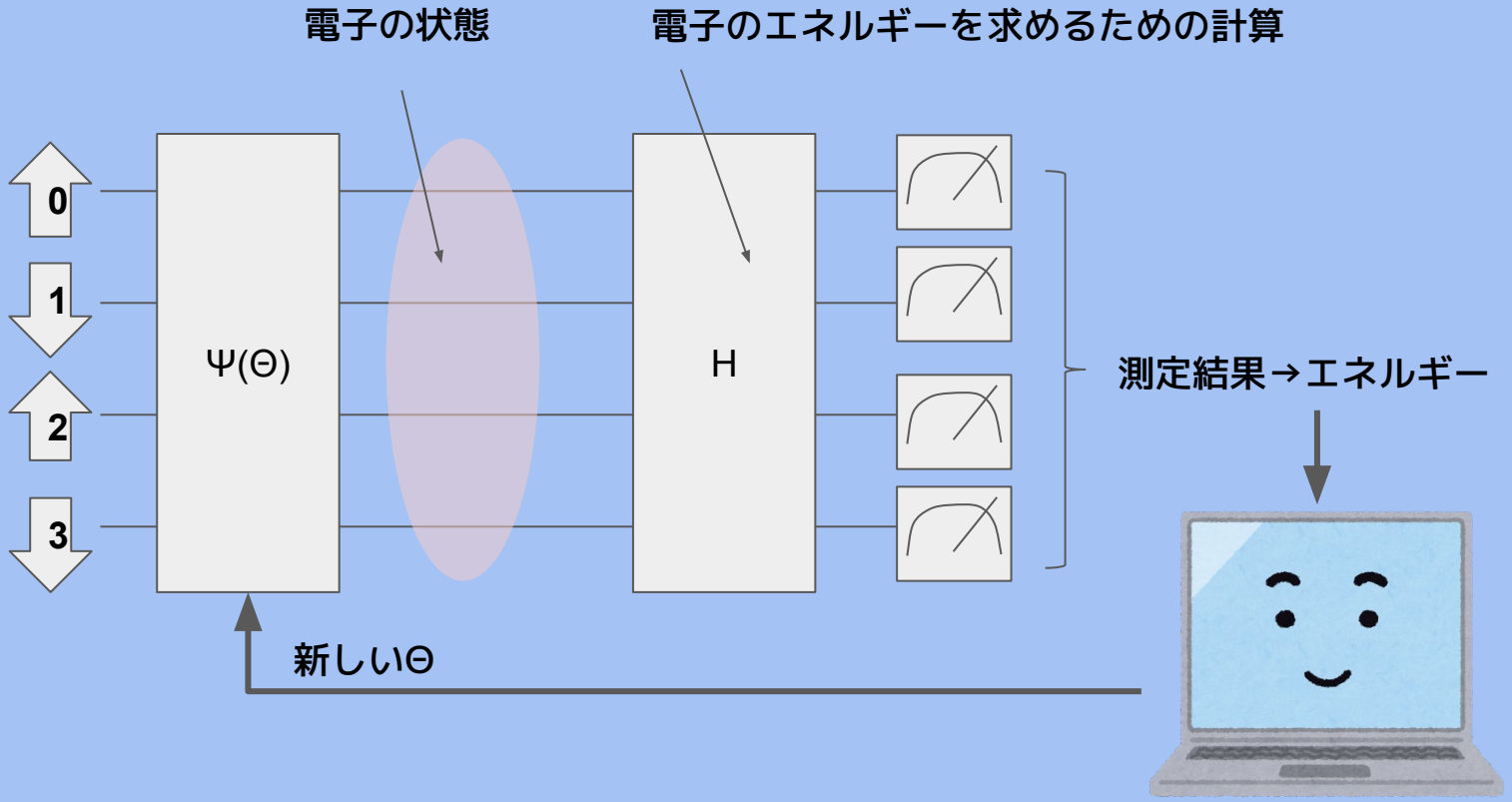
例: より高いところを目指したい。どちらに行けばいいか?



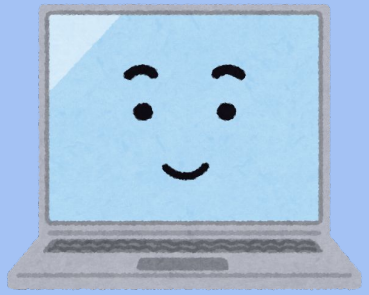
ブラックボックス最適化　いくつかの手法

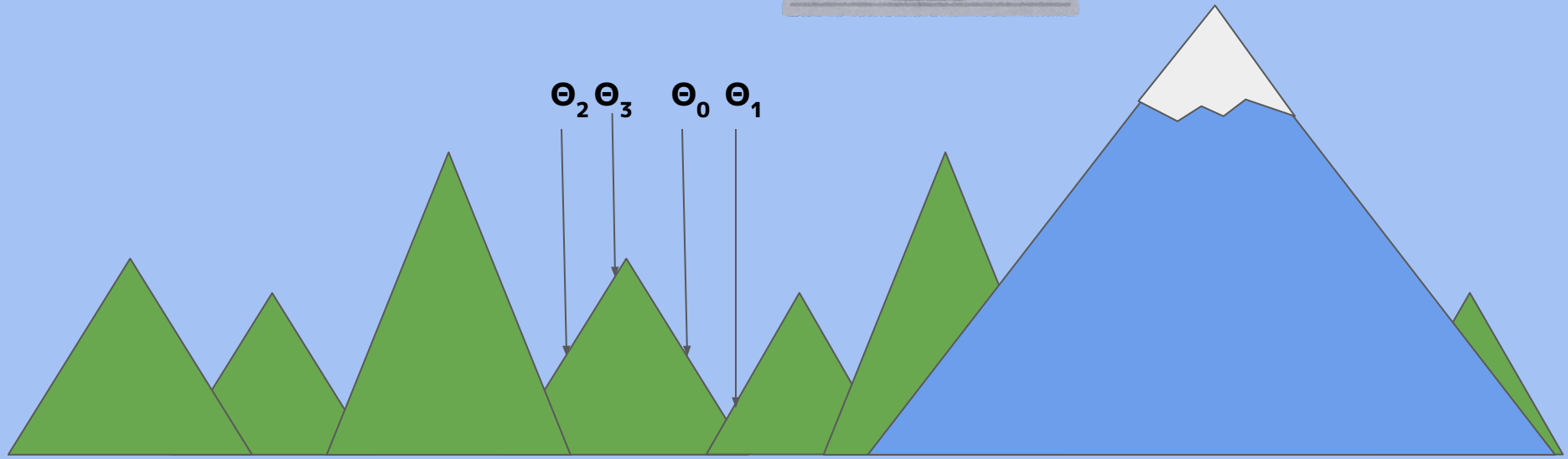
- ➔ Nelder-Mead法
  - ◆ n + 1 個の頂点からなる n 次元の単体（シンプレックス）をアメーバのように動かしながら関数の最小値を探索する。反射、膨張、収縮の3種類を使い分けながら探索する (Wikipedia)
- ➔ Powell法
  - ◆ Nelder-Mead法より収束が速いと言われている。現状、Blueqatのデフォルト手法
  - ◆ 解候補を一点とり，そこから別の点を探索するために，探索次元数分の直線探索方向の調整とその方向への直線探索を繰り返す (https://www.sice.jp/ia-j/papers/03IA001_4.pdf より引用)
- ➔ COBYLA (Constrained optimization by linear approximation)
  - ◆ 直訳すると「線形近似による制約付き最適化」
  - ◆ 大体どのアルゴリズムも、次に試す点の選び方が少し違うくらい
- ➔ ベイズ最適化
  - ◆ 「今見ている点の近くを探す」戦略を取らないので、全体最適に辿り着く可能性がある
  - ◆ RigettiがQAOAで利用

(再掲) 量子古典ハイブリッド

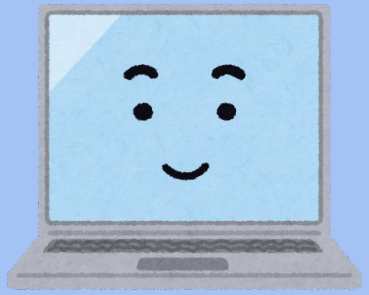


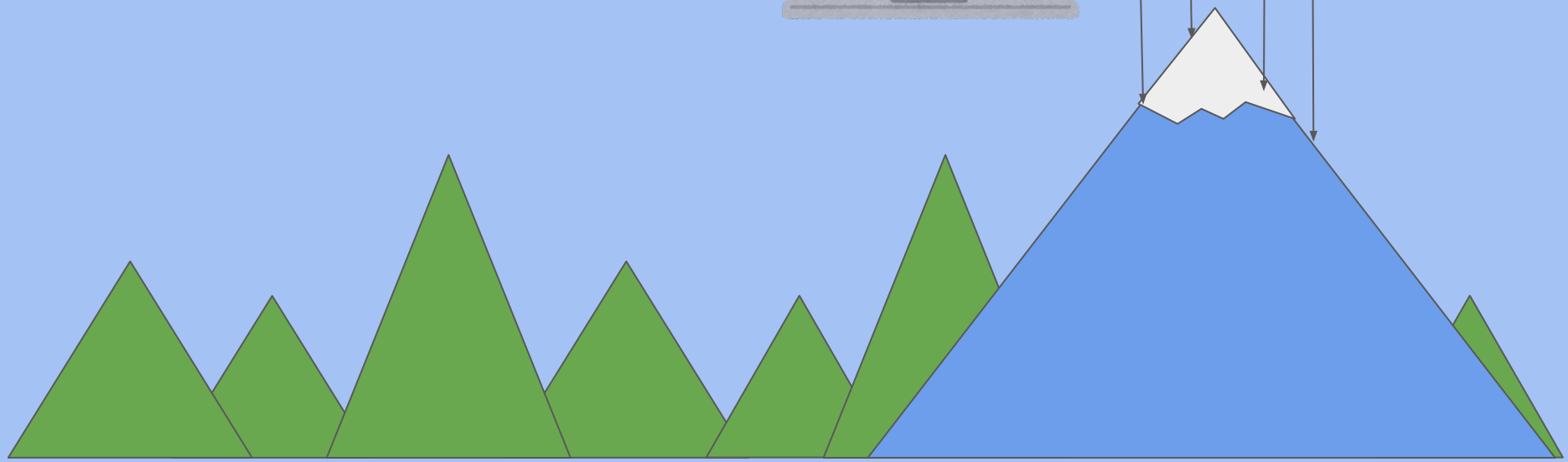
最適化





初期値選びが重要





VQEによる量子化学計算の大まかな流れ

目標: 基底状態のエネルギーを求めたい
　　(ポストHartree-Fock法の一種)

1. 電子状態に対応する量子回路を準備する
   →試行関数、ansatzなどと呼ばれる
2. 量子回路のエネルギーを測定する
3. 試行関数を少し変えて、1に戻る
   →測定するエネルギーが最も低くなるように最適化を行う

実際にやってみる

今回は水素分子(H_2)の基底状態を求めてみる

1. 電子状態に対応する量子回路を準備する
   →UCCと呼ばれるものが最も有名なので、それを使う
2. 量子回路のエネルギーを測定する
   →OpenFermionというライブラリで準備した数式(ハミルトニアン)を使う
3. 試行関数を少し変えて、1に戻る
   →最適化はデフォルトのPowell法で行う

プレイ動画

In [ ]:
```
from blueqat import *
from openfermionblueqat import to_pauli_expr_with_bk
from openfermionblueqat.ucc import *
from openfermion import *
import numpy as np
```

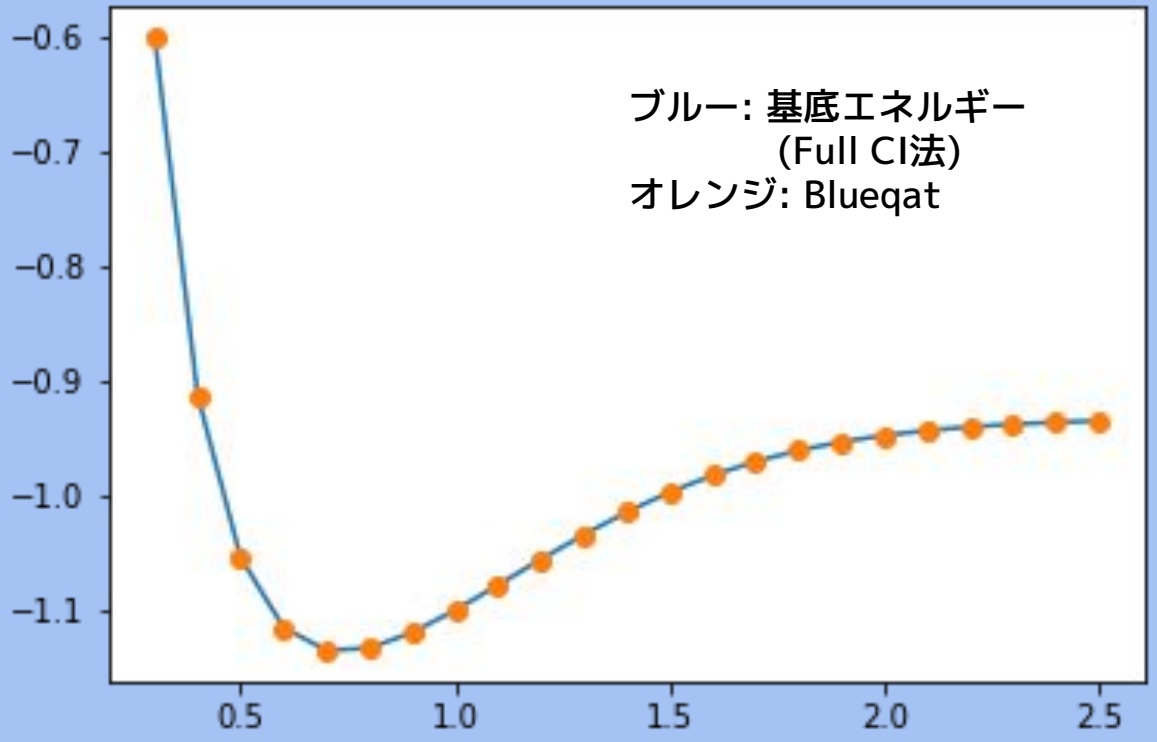
In [ ]:
```
# This example code uses parts of OpenFermion-Cirq's example.
# https://github.com/quantumlib/OpenFermion-Cirq/blob/master/examples/tutorial_4_variational.ipynb
def get_molecule(diatomic_bond_length):
    geometry = [('H', (0., 0., 0.)),
                ('H', (0., 0., diatomic_bond_length))]

    description = '{:0.1f}'.format(diatomic_bond_length)

    molecule = MolecularData(geometry, "sto-3g",
                             1, description=description)
    molecule.load()
    return molecule
```

In [ ]:
```
bond_lengths = np.arange(0.3, 2.6, 0.1)
x = []; y = []; fci = []
print("Length¥tEnergy (Blueqat)¥tEnergy (Full CI)¥tDeviation")
for diatomic_bond_length in bond_lengths:
        m = get_molecule(diatomic_bond_length)
        runner = vqe.Vqe(UCCAnsatz2(m, Circuit().x[0]))
        result = runner.run()
        e = runner.ansatz.get_objective(runner.sampler)(result.params)
        x.append(diatomic_bond_length)
        y.append(e)
        fci.append(m.fci_energy)
```

結果


−0.6
−0.7
−0.8
−0.9
−1.0
−1.1
0.5
1.0
1.5
2.0
2.5
ブルー: 基底エネルギー
(Full CI法)
オレンジ: Blueqat

量子化学VQEの課題

H_2ではかなりうまくいくが、より複雑な分子では、以下のような課題を抱えているように感じる

1. 最適化の難しさ
   パラメータ数の増大→局所最適に落ちやすくなる、計算に時間がかかる
2. 回路が長くなる
   電子が増えると、考慮すべき電子相関が増えて回路が長くなる
   →特に実機で計算したときは、計算エラーが増える

現在のVQEのやり方で、大きな分子で高速化するのかは疑問

・量子化学に関する領域知識
・最適化計算に関する領域知識　を持った人の助けがほしい

まとめ

- ➔ 計算可能性と計算複雑性
  - ◆ 計算複雑性の観点から、量子コンピュータはコンピュータを超えうる
- ➔ 量子化学の大まかな流れと、量子コンピュータでVQEをする方法
  - ◆ 試行関数となる回路の準備
  - ◆ エネルギーの測定
  - ◆ 試行関数のパラメータを変える最適化計算
- ➔ 水素分子のエネルギー計算: 非常に高精度で計算できた
- ➔ VQEの今後
  - ◆ 大きな分子で量子コンピュータのメリットを示せるか?
  - ◆ 挑戦してくれる仲間を募集中
- ➔ 平成最後のイケてるゲートライブラリBlueqatをよろしく