『Qiskit Tutorial』勉強会

Nature 編 –
励起状態ソルバー, PES サンプリング, 熱力学観測量の計算

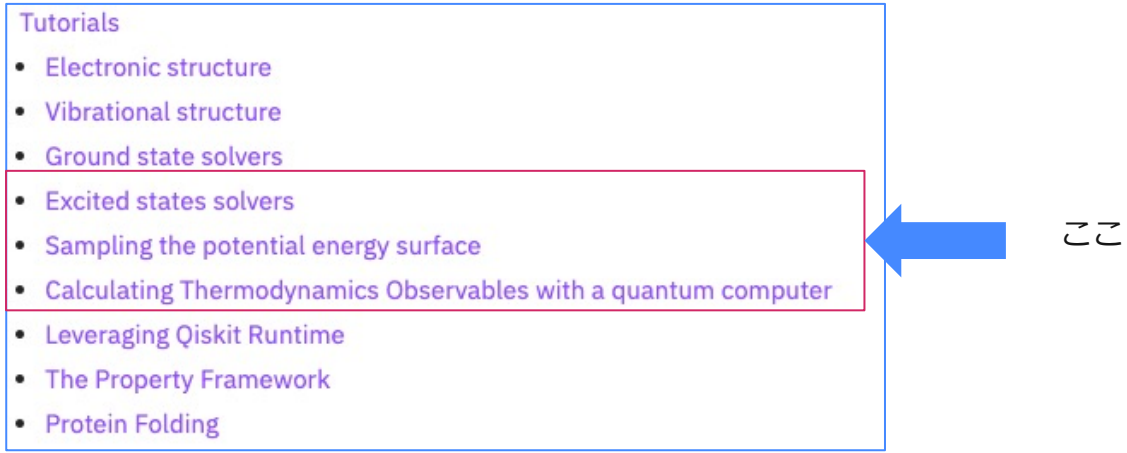
2021年10月1日

東京ラボ / IBM Garage
天野 武彦 (Amano, Takehiko)
Qiskit Advocate (2020)
Twitter: @ibmamnt

Qt

本日の説明の対象

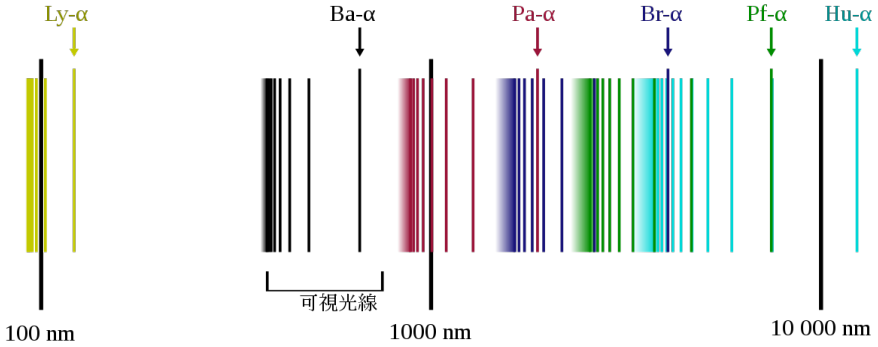
https://qiskit.org/documentation/nature/

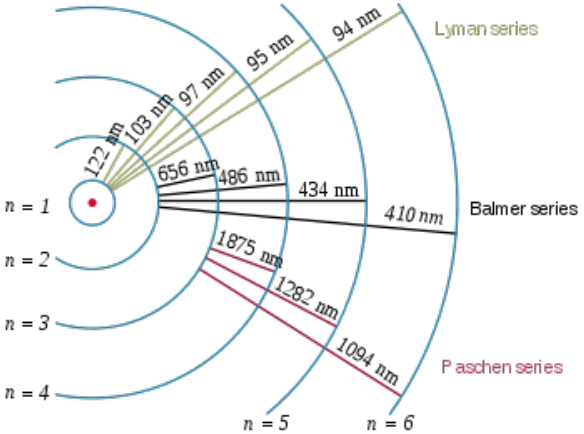


励起状態ソルバー (Excited states solvers)

はじめに ~ 水素のスペクトル

水素のスペクトルは水素原子の電子のエネルギー順位間で光子の吸収・放出に伴って発生します。





前期量子論において、ニールス・ボーアーが電子モデルを考案し有名なリュードベリー公式を再現させました。

$$\frac{1}{\lambda} = R_\infty \left| \frac{1}{(n')^2} - \frac{1}{n^2} \right| \qquad \left(R_\infty = 1.097\,373\,156\,8508 \times 10^7\ \mathrm{m}^{-1}\right)$$

https://ja.wikipedia.org/wiki/水素スペクトル系列 より抜粋

励起状態ソルバーはなぜ重要なのか？

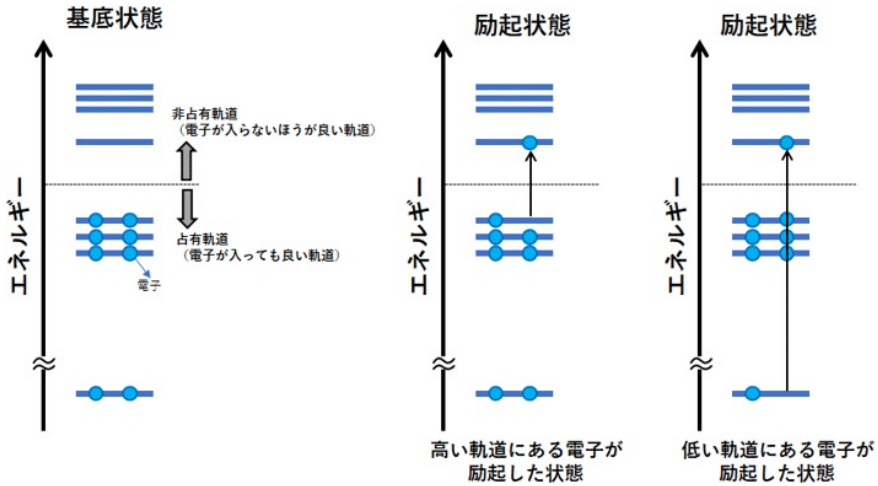
- 例：天文学において遠い星や惑星にどんな成分の分子が存在しているのかわかる

  → 生命が存在する惑星の探知

- 物質の状態をしらべるためX線や電子線を物質にあて出てくるスペクトルを用いてその構造をしらべる方法があります。

- しかしながら励起状態の計算には時間がかかります。最近は、AIを用いて高速に計算する方法も開発されています。

  https://www.aist.go.jp/aist_j/press_release/pr2020/pr20200603/pr20200603.html



図は左記のプレスリリースから引用

Qiskit Nature での励起状態のエネルギー計算

- 励起状態の計算には CI (configuration interaction -配置間相互作用)による計算が必要ですが、NISQ時代の量子コンピューターでは実行が困難です。
- Qiskit Nature では励起状態のエネルギー計算をqEOM (Quantum Equation of Motion) アルゴリズムを利用して計算しています。

分子の基底エネルギーを計算

qEOM アルゴリズムを利用して励起状態エネルギーを計算 (古典)

“Quantum equation of motion for computing molecular excitation energies on a noisy quantum processor”, 2019, https://arxiv.org/abs/1910.12890

アルゴリズムのあらまし(1/2)

1. 基底状態 $|0\rangle$ から 励起状態 $|n\rangle$ に遷移させるユニタリー演算子を作成。逆の演算子も作成する。

$$\hat{O}_n^\dagger = |n\rangle\langle 0| \quad , \quad \hat{O}_n = |0\rangle\langle n|$$

2. ハミルトニアン H の固有値 E_n を用いてエネルギー差分 $E_{0n} = E_n - E_0$が計算できる。

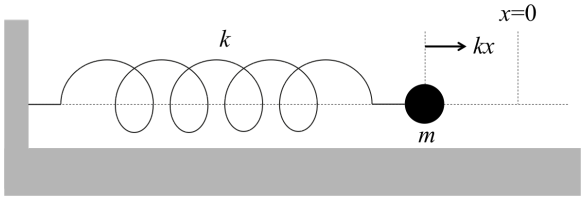
$$\left[\hat{H}, \hat{O}_n^\dagger\right]|0\rangle = \hat{H}\hat{O}_n^\dagger|0\rangle - \hat{O}_n^\dagger\hat{H}|0\rangle = E_{0n}\,\hat{O}_n^\dagger|0\rangle$$

$$E_{0n} = \frac{\langle 0|\left[\hat{O}_n, [\hat{\mathrm{H}}, \hat{O}_n^\dagger]\right]|0\rangle}{\langle 0|\left[\hat{O}_n, \hat{O}_n^\dagger\right]|0\rangle}$$

【ご参考】- 調和振動子のエネルギー計算

- 調和振動子は以下の運動方程式に従います（摩擦なし）

$$F = -kx \quad (k\text{はバネ定数})$$



図は http://physics.thick.jp/Mechanics/Section3/3-5.html から引用

- この方程式を量子力学で解きます。ハミルトニアンと固有エネルギーは以下のようになります。

$$H = \hbar\omega \left(a^{\dagger}a + \frac{1}{2}\right),\ E_n = \hbar\omega \left(n + \frac{1}{2}\right)$$

- $a^{\dagger}$は生成(昇降)演算子、 a は消滅 (下降)演算子と呼ばれ、量子状態$|n\rangle$に対して以下の振る舞いをします。

$$a^{\dagger}|n\rangle = \sqrt{n+1}\,|n+1\rangle,\ a|n\rangle = \sqrt{n}|\,n-1\rangle$$

生成・消滅演算子を使えば量子状態を上げ下げできるということがポイントです。

アルゴリズムのあらまし(2/2)

3. $\hat{O}_n^\dagger$ を 生成演算子 $\hat{a}_n^\dagger$、消滅演算子 $\hat{a}_n$で近似する(運動方程式法 – EOM)
4. 最終的に以下の固有多項式 (secular equation) を解く。

$$\begin{pmatrix} \mathbf{M} & \mathbf{Q} \\ \mathbf{Q}^* & \mathbf{M}^* \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{Y}_n \end{pmatrix} = E_{0n} \begin{pmatrix} \mathbf{V} & \mathbf{W} \\ -\mathbf{W}^* & -\mathbf{V}^* \end{pmatrix} \begin{pmatrix} \mathbf{X}_n \\ \mathbf{Y}_n \end{pmatrix},$$

where

$$M_{\mu_\alpha \nu_\beta} = \langle 0 | \left[(\hat{\mathrm{E}}^{(\alpha)}_{\mu_\alpha})^\dagger, \hat{\mathrm{H}}, \hat{\mathrm{E}}^{(\beta)}_{\nu_\beta} \right] | 0 \rangle ,$$

$$Q_{\mu_\alpha \nu_\beta} = - \langle 0 | \left[(\hat{\mathrm{E}}^{(\alpha)}_{\mu_\alpha})^\dagger, \hat{\mathrm{H}}, (\hat{\mathrm{E}}^{(\beta)}_{\nu_\beta})^\dagger \right] | 0 \rangle ,$$

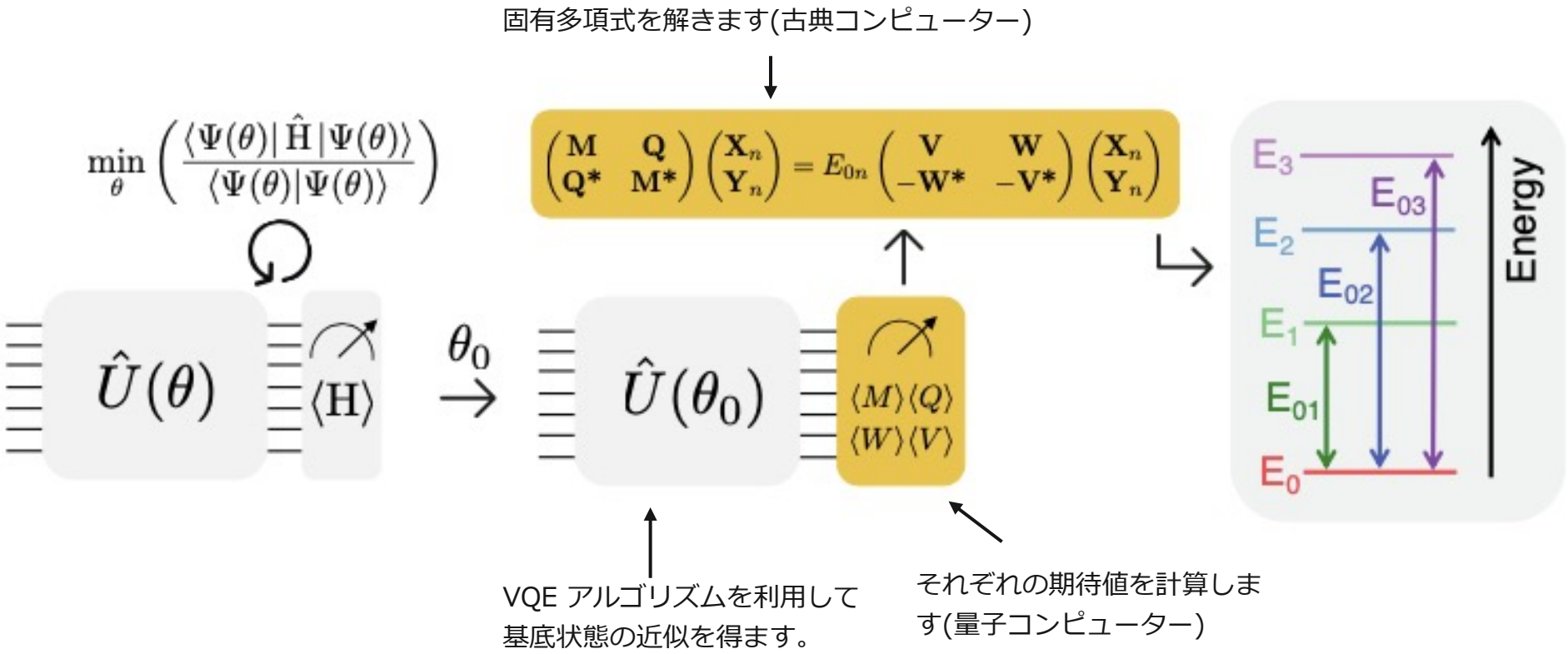
$$V_{\mu_\alpha \nu_\beta} = \langle 0 | \left[(\hat{\mathrm{E}}^{(\alpha)}_{\mu_\alpha})^\dagger, \hat{\mathrm{E}}^{(\beta)}_{\nu_\beta} \right] | 0 \rangle ,$$

$$W_{\mu_\alpha \nu_\beta} = - \langle 0 | \left[(\hat{\mathrm{E}}^{(\alpha)}_{\mu_\alpha})^\dagger, (\hat{\mathrm{E}}^{(\beta)}_{\nu_\beta})^\dagger \right] | 0 \rangle .$$

基底状態 |0> を使った期待値の計算であることに着目してください。

DJ Rowe, “Equations-of-motion method and the ex- tended shell model,” Reviews of Modern Physics 40, 153 (1968).

アルゴリズムのイメージ図



図は https://arxiv.org/abs/1910.12890 から抜粋

Qiskit Nature QEOM API

- アルゴリズムの利用は簡単です。

```
molecule = Molecule(geometry=[['H', [0., 0., 0.]],
                              ['H', [0., 0., 0.735]]],
                    charge=0, multiplicity=1)
driver = ElectronicStructureMoleculeDriver(molecule, basis='sto3g',
                    driver_type=ElectronicStructureDriverType.PYSCF)

es_problem = ElectronicStructureProblem(driver)
qubit_converter = QubitConverter(JordanWignerMapper())

# This first part sets the ground state solver
# see more about this part in the ground state calculation tutorial
quantum_instance = QuantumInstance(Aer.get_backend('aer_simulator_statevector'))
solver = VQEUCCFactory(quantum_instance)
gsc = GroundStateEigensolver(qubit_converter, solver)

# The qEOM algorithm is simply instantiated with the chosen ground state solver
qeom_excited_states_calculation = QEOM(gsc, 'sd')

# Solve the problem
qeom_results = qeom_excited_states_calculation.solve(es_problem)
print(qeom_results)
```

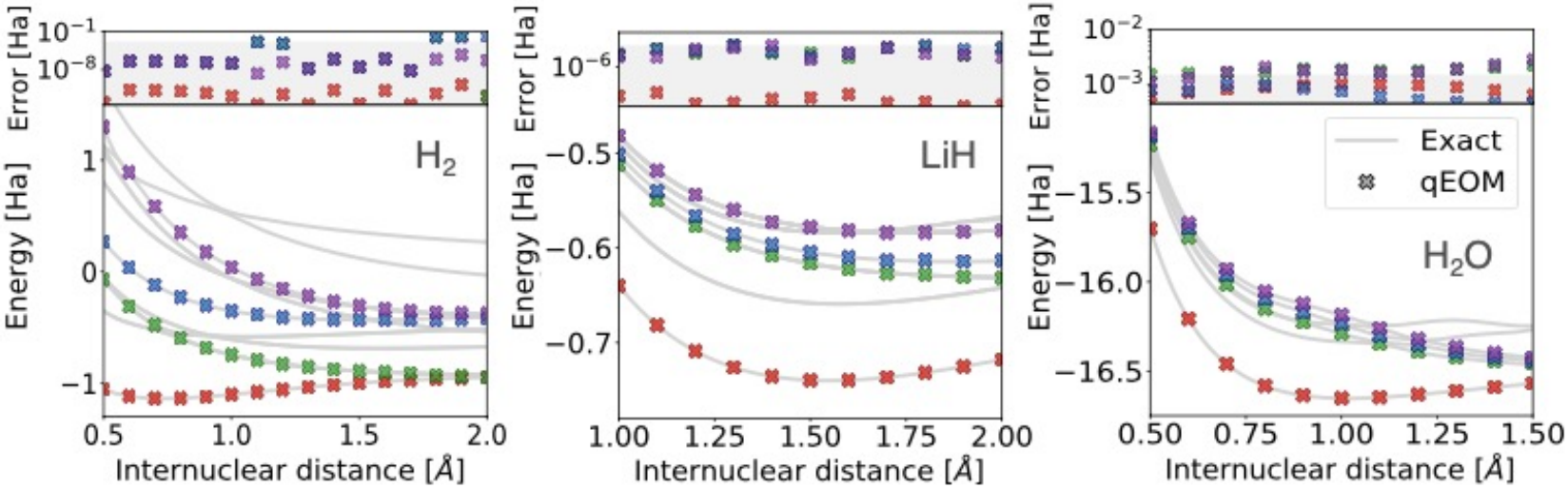
問題を定義（ここでは水素分子）

基底状態のソルバー (GroundStateEigensoler) の結果を QEOM にパラメータとして渡す。

問題を解く

結果(論文より抜粋)

- qEOMアルゴリズムによって励起状態のエネルギー計算ができることがわかりました。



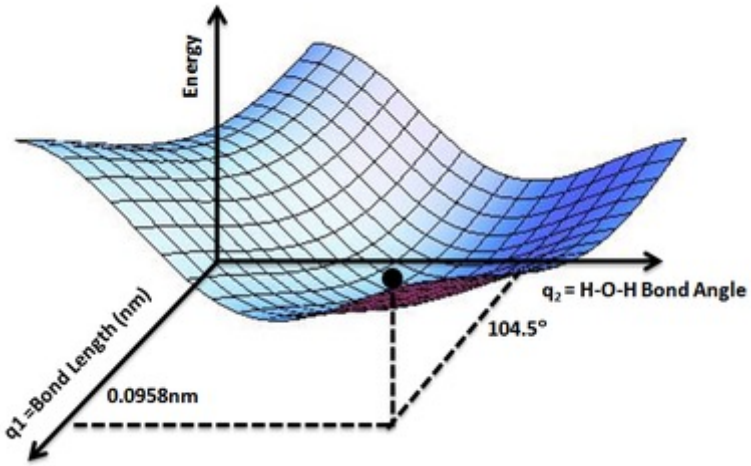
H2, LiH, H2O での結果。赤は基底状態、緑、青、紫はそれぞれ１、２、３励起状態。グレーの線は固有値方程式を解いた厳密解。上段はエラー率。

図は https://arxiv.org/abs/1910.12890 から抜粋

ポテンシャルエネルギー曲面のサンプリング
Sampling Potential Energy Surface

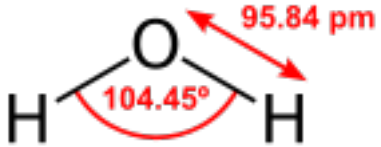
ポテンシャルエネルギー曲面とは？

- ポテンシャルエネルギー曲面（potential energy surface, PES）とは、特定のパラメータ（原子のデカルト座標や結合角、二面角など）に対して系のエネルギーを表したものです。



左図は H_20 (水分子) に対してボンド長 (0-H)、結合角 (Bond Angle) に対してエネルギーを示しています。

よく知られているように水分子は結合角が 104.45° の時に一番安定します。



https://ja.wikipedia.org/wiki/ポテンシャルエネルギー曲面 から抜粋

Qiskit での実装 ~ BOPESSampler

- Qiskit では PES のサンプリング(様々な値を設定してVQE計算) はボルン・オッペンハイマー近似をしています。
- BOPES：Born-Oppenheimer Potential Energy Surface

qiskit.chemistry.algorithms.BOPESSampler ¶

```
CLASS BOPESSampler(gss, tolerance=0.001, bootstrap=True, num_bootstrap=None,
      extrapolator=None)
```

[SOURCE] ¶

基本的な使い方

```
distance1 = partial(Molecule.absolute_distance, atom_pair=(1, 0))
mol = Molecule(geometry=[('H', [0., 0., 0.]),
                         ('H', [0., 0., 0.3])],
                        degrees_of_freedom=[distance1],
                        )

# pass molecule to PSYCF driver
driver = PySCFDriver(molecule=mol)

# Specify degree of freedom (points of interest)
points = np.linspace(0.25, 2, 30)

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

bs = BOPESSampler(
    gss=GroundStateEigensolver(FermionicTransformation(), vqe_solver)
     ,bootstrap=True
     ,num_bootstrap=None
     ,extrapolator=None)
# execute
res = bs.sample(driver,points)
print(res.energies)
```

分子の定義。ここでは水素分子

ドライバーとして PySCF を利用

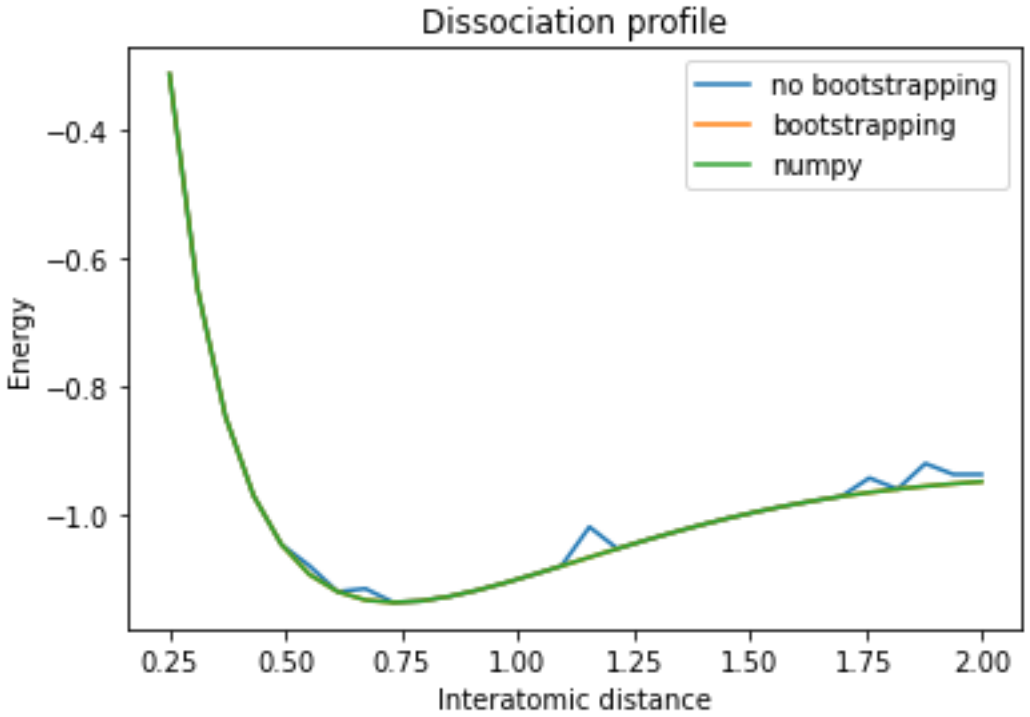
サンプルポイントを設定

固有値のソルバーとして VQE を設定

BOPESSampler インスタンス作成 bootstrap 引数に着目

サンプリング開始

結果例 (ここでは曲面ではなく曲線)



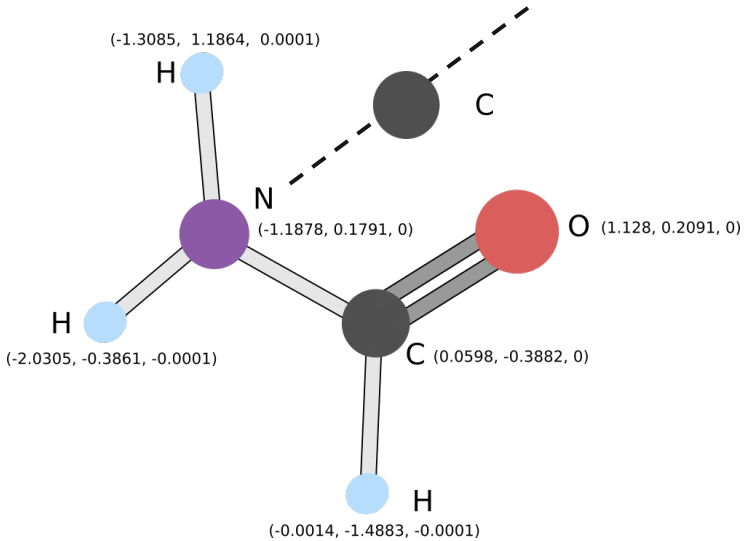
Bootstrap なしとありで結果が異なります。

bootstrap=True

に設定すると前回のサンプリングで使われた最適パラメータを再利用して開始します。

Qiskit Challenge Africa21 でも PESを使いました

- 9/9 ～ 9/21 に開催されたQiskit Challenge Africa21 でも PESを使った計算が課題として出されました。



HIVウイルスに対抗するための抗レトロウイルス薬の模型。

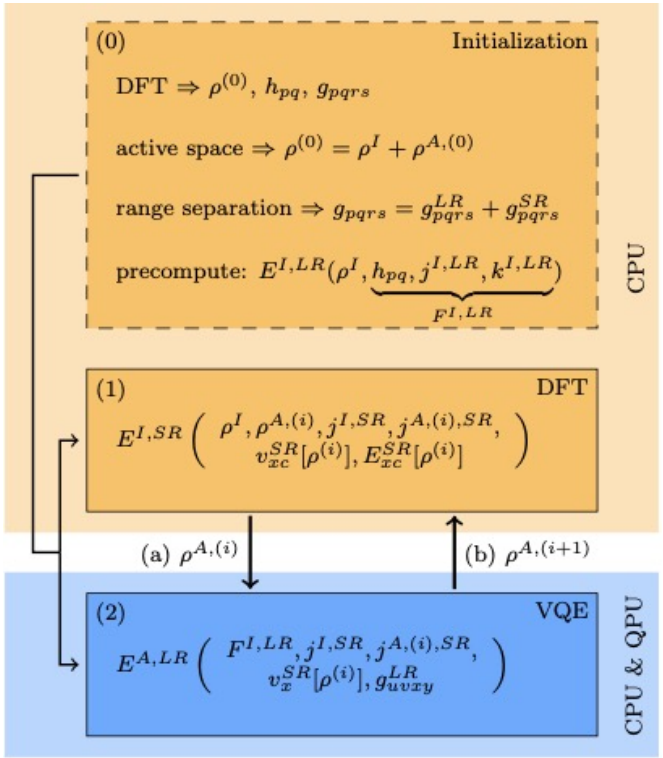
$$\mathrm{HCONH_2} + C$$

電子が30個ありスピン軌道を考えると **60 qubits** 必要。

ActiveSpaceTransformerの活用

- ActiveSpaceTransformerにより HF法など古典計算できる部分は古典計算をし、クリティカルな電子群に焦点をあてた量子計算を行う工夫をすることで大幅に使用する量子ビット数を削減することができます。
- このチャレンジではActiveSpaceTransformer を利用して利用する量子ビット数を 4まで削減しています。

DFT: Density Function Theory



M. Rossmannek, P. Barkoutsos, P. Ollitrault, and I. Tavernelli, arXiv:2009.01872 (2020).

extrapolator 引数について

- extrapolator 引数に関数を設定すると次の試行に利用するパラメータをある程度推測し、評価の数を少なくしてくれるようです。
- サポートされている extrapolator
  - Window Extrapolator
  - PCA Extrapolator
  - Sieve Extrapolator
  - Polynomial Extrapolator
  - Differential Extrapolator

```
no bootstrapping
Total evaluations for no bootstrapping: 20136

bootstrapping
Total evaluations for bootstrapping: 3692

Poly
Total evaluations for poly: 1333

diff_model
Total evaluations for diff_model: 3166
```

Calculating Thermodynamics Observables with a quantum computer
熱力学観測量の量子コンピューターによる計算

熱力学観測量とは？

- Qiskit では比熱 C_v を例に取り上げています。

  - ここで比熱は体積が一定の物質の温度を1度あげるのに必要な熱量のことです。古典的な熱力学では以下の式で与えれます。

$$C_V = \left(\frac{\partial \mathrm{U}}{\partial T}\right)_V$$ Uは内部エネルギー、Tは温度

- 統計力学では内部エネルギーは以下のようになります。

$$U = \left(k_B T^2 \frac{\partial\, lnZ}{\partial T}\right)_V$$ k_Bはボルツマン定数、Tは温度、Zは分配関数

➡ Z を計算で求めれば比熱が計算できます。

分配関数 (partition function) について

- ある系のエネルギー が　E_i　という固有状態をもつ確率は以下の式で与えられます。

$$p_i = \frac{1}{Z} e^{-E_i/k_bT}$$

- Z は規格化因子で次の式で表されます。これを**分配関数**と呼びます。
  ※ 田崎先生の本では、なんで分配という言葉がつかわれているのか？はよくわからないとコメントしてありました。

$$Z = \sum_i e^{-E_i/k_bT}$$

※系とは、気体、液体、個体など数多くの分子から構成されているものです。

系のエネルギー固有値 E_i について

- 系のエネルギー固有値は、系に存在している分子のエネルギーの総和です。
- 分子のエネルギーは次のように分解できます。

分子のエネルギー = 原子核のエネルギー (並進、回転、振動) + 電子のエネルギー

$$E = K_n + U_e$$

- U_e は前節の PES (Potential Energy Surface) から求まります。
- 原子核のエネルギー K_n のうち、振動部分は Watson 方程式を解きます (Vibrational structure節参照)

- Tutorials
  - Electronic structure
  - Vibrational structure
  - Ground state solvers

ここ

熱力学観測量の計算方法

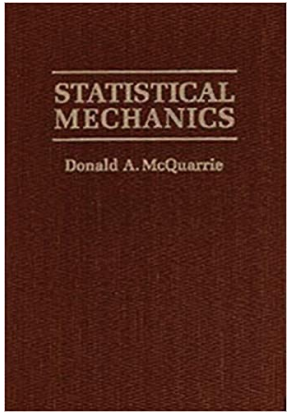
1. 電子エネルギーの計算の実行
2. 原子核エネルギー (vibrational energy level) の計算
3. 分配関数の計算
4. 比熱などの熱力学観測量の計算

分配関数、熱力学的観測量計算など

- 分配関数、熱力学的観測量は Qiskit Nature API に存在しないモジュールで提供されています。(qiskit-nature/docs/tutorials/thermodynamics_utils)

ファイル名	概要
partition_function.py	分配関数の計算。前ページの電子・原子核のエネルギーを用い、さらに並進、回転を考慮して原子核のエネルギーを計算し、分配関数を決定。
thermodynamics.py	比熱の計算。このほか、エンタルピー、ヘルムホルツの自由エネルギーの計算などがある
vibrational_structure_fd.py	振動エネルギーの PESの微分を計算

計算方法については右の書籍のものを実装しているようです。

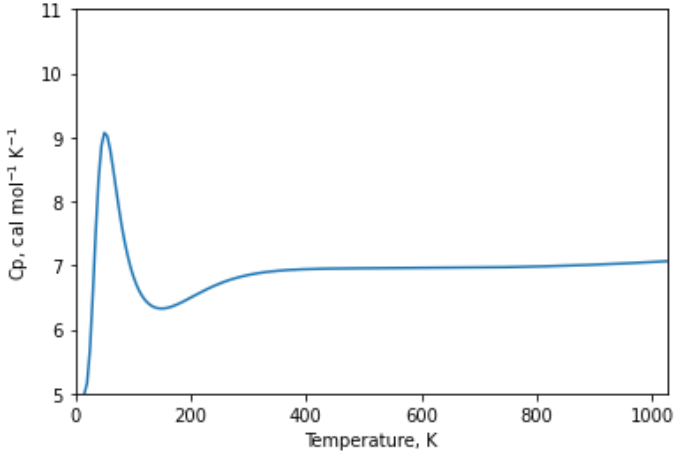


Statistical Mechanics, Donald A. McQuarrie (著), 2020

コードサンプル

様々な温度における水素の定圧比熱を計算しています。

```
Q = DiatomicPartitionFunction(mol, energy_surface, vibrational_structure)

P = 101350  # Pa
temps = np.arange(10, 1050, 5)  # K

mol.spins = [1/2,1/2]

td = Thermodynamics(Q, pressure = 101350)
td.set_pressure(101350)
temps = np.arange(10, 1500, 5)
ymin = 5
ymax = 11

plt.plot(temps,
         td.constant_pressure_heat_capacity(temps) / const.CAL_TO_J)
plt.xlim(0, 1025)
plt.ylim(ymin, ymax)
plt.xlabel('Temperature, K')
plt.ylabel('Cp, cal mol$^{-1}$ K$^{-1}$')

plt.show()
```



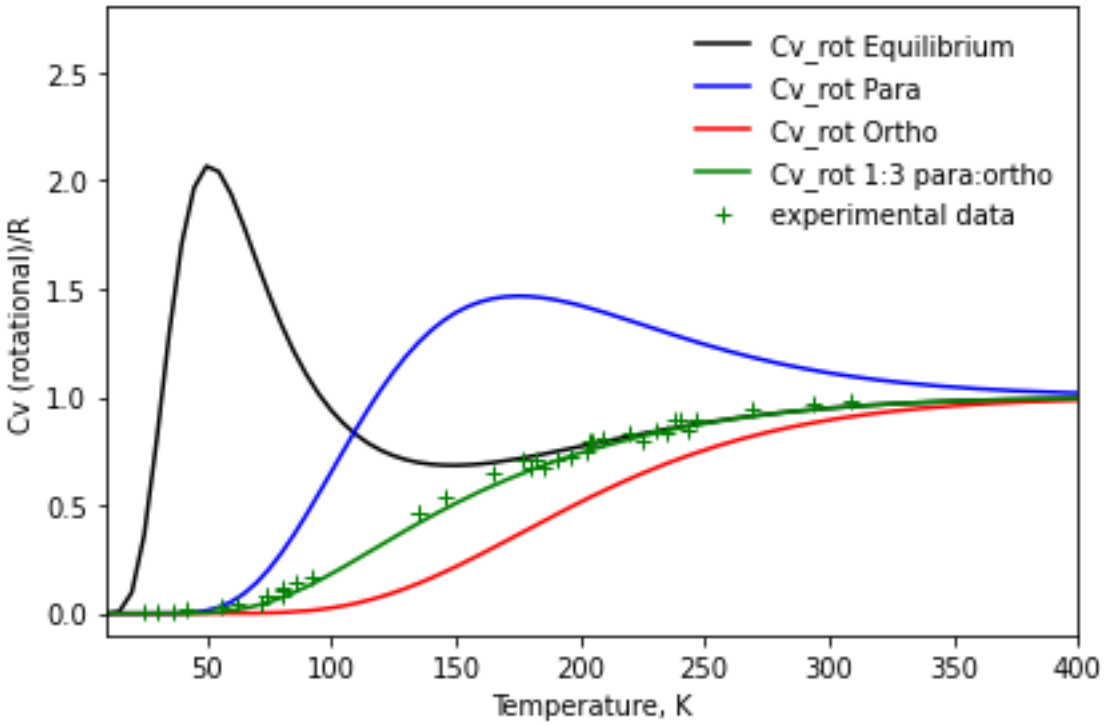
水素の特性

融点：14.01K

沸点：20.28K

平衡水素の場合の計算。

量子コンピューターでの計算結果



- Tutorial では水素分子 H_2 にて実施。
- 二原子分子である水素は２個の陽子を持っています。
- 陽子はスピン自由度をもっているため、その向きが同一のものをオルソ水素、反対のものをパラ水素と呼んでいます。
- 常温で平衡状態にある水素は、パラ水素が25%、オルソ水素が75%の割合(1:3)で存在しています。
- 1:3 のパラ、オルソの組み合わせの計算結果は実験データとほぼ一致しています。

https://ja.wikipedia.org/wiki/核スピン異性体

ご静聴ありがとうございました！