

Qiskit Document Tutorials 勉強会

Qiskit Nature Tutorials

振動構造・基底状態ソルバー(Filter function編)

Emi ADACHI



# Agenda

1. 振動構造について
2. 振動構造のシュレディンガー方程式
3. ワトソンハミルトニアンについて
4. 基底状態を求めるまでのコードの流れ
5. ハミルトニアンの用意・コードの説明
6. 基底状態ソルバーを求める
7. 参考文献、引用文献

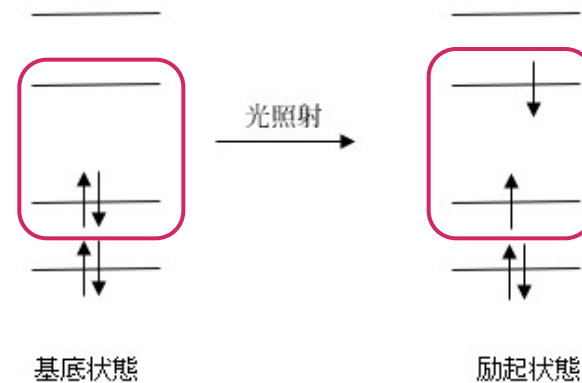
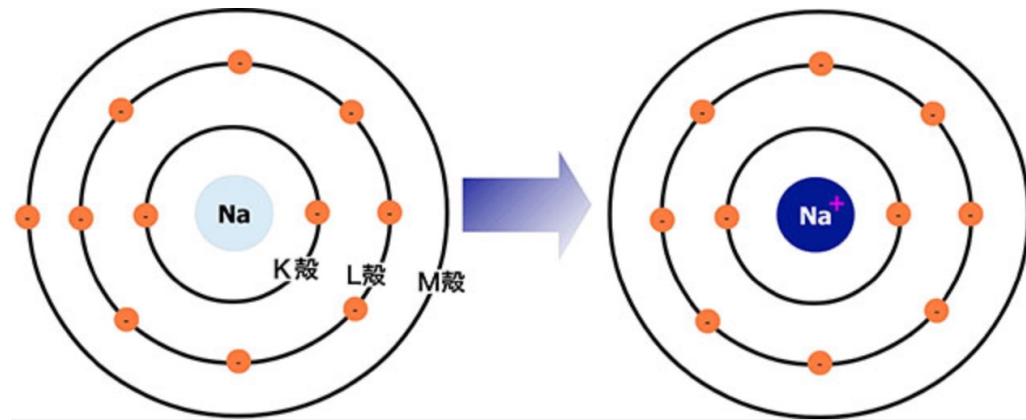


# 1. 振動構造について

1. 振動構造の状態とは？
2. 振動構造が起こるしくみ
3. 振動プログレッションのグラフ
4. 振動構造からわかること

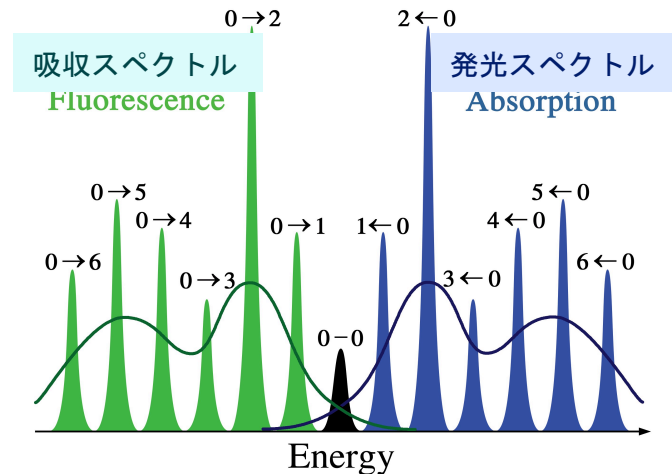
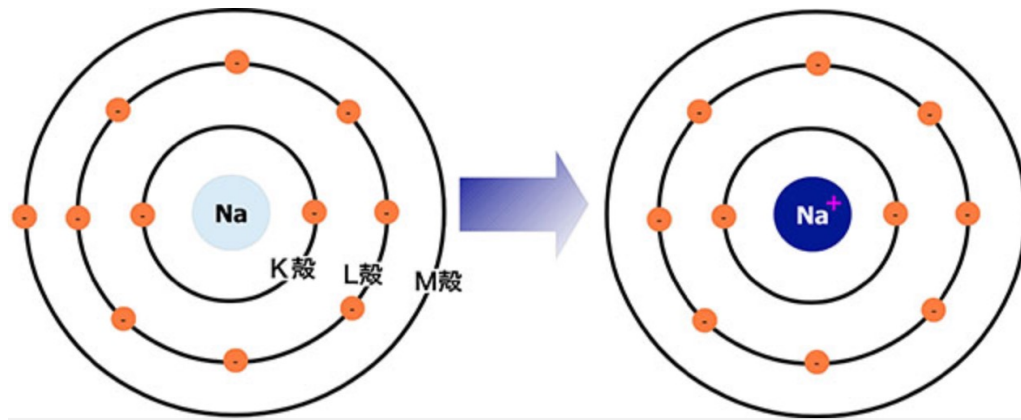


# 1. 振動構造・振動プログレッションの状態とは？



原子価電子と内殻電子のイオン化における挙動の状態

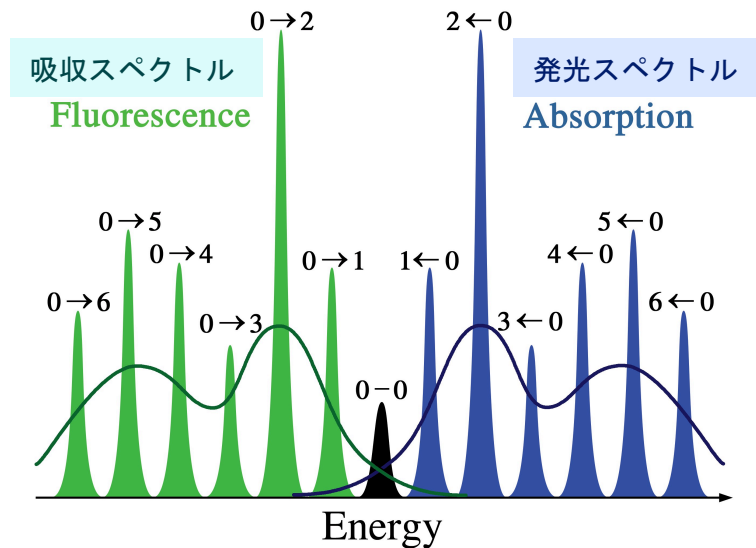
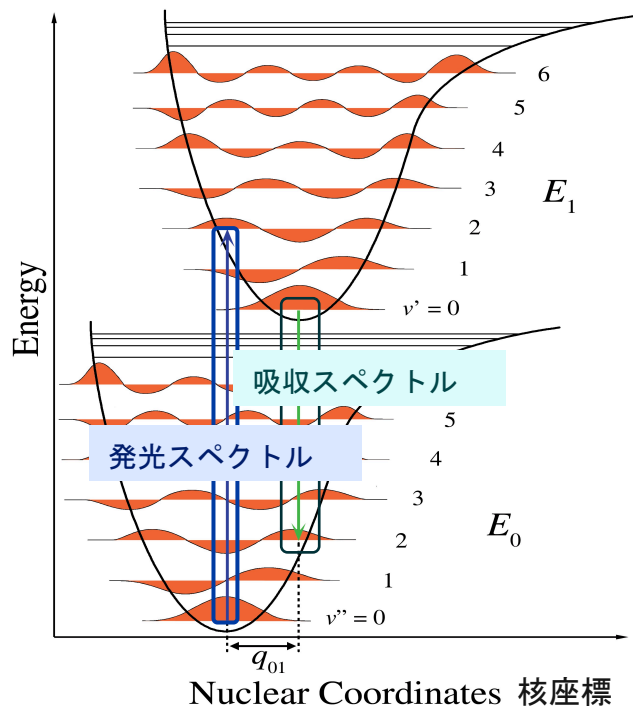
## 1.2 振動構造が起こる仕組み



1. 電子が物質に光や電子を照射すると、内殻から電子がたたき出される
2. 放出された電子は周囲を運動する電子の波動性により、光線の吸収強度、光電子の散乱強度などに振動が現れる。

# 1.3 振動プログレッションのグラフ

発光スペクトルは吸収スペクトルの鏡像のような形をとることが多い



## 1.4 振動構造からわかること

- 内殻励起が起こった原子の周囲の幾何、電子構造についての情報
- 基底状態のハミルトニアン
- 励起状態のハミルトニアン



## 2. 振動構造のシュレディンガー方程式

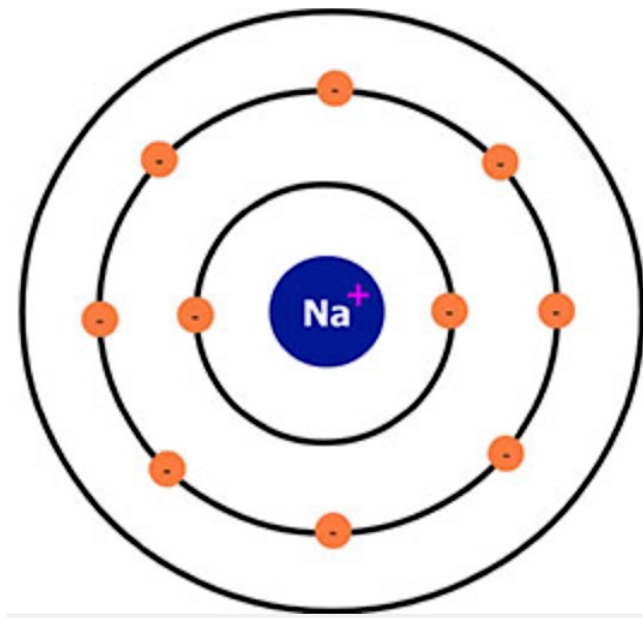
1. 分子の運動とは？
2. 分子のエネルギーとは？
3. 振動構造のシュレディンガー方程式
4. シュレディンガー方程式を解く手順
5. 基準振動解析
6. 調和振動子とは？
7. 調和振動子近似とは？
8. ポテンシャルエネルギー曲面を解く手順





## 2.1 分子の運動とは？

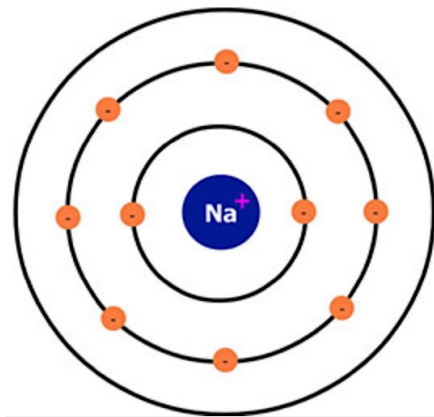
分子の運動 = 並進運動 + 回転運動 + 振動運動



- 分子全体の運動：並進運動 + 回転運動
- 分子内部の運動：振動運動

## 2.2 分子のエネルギーとは？

分子のエネルギー＝ 並進エネルギー ＋ 回転エネルギー ＋ 振動エネルギー ＋ 電子のエネルギー



特に、振動と回転エネルギーは一般には分離できない。

→ 振動と回転の相互作用：コリオリ結合

## 2.3 振動構造のシュレディンガー方程式

$$\mathcal{H}_{\text{vib}}|\Psi_n\rangle = E_n|\Psi_n\rangle$$

ワトソンハミルトニアン:振動回転結合項を無視

$$\mathcal{H}_{\text{vib}}(Q_1, \dots, Q_L) = -\frac{1}{2} \sum_{l=1}^L \frac{\partial^2}{\partial Q_l^2} + V(Q_1, \dots, Q_L)$$



## 2.3.1 ワトソンハミルトニアンとは？

コリオリ結合（振動回転結合項）を無視したハミルトニアン

$$\mathcal{H}_{\text{vib}}(Q_1, \dots, Q_L) = -\frac{1}{2} \sum_{l=1}^L \frac{\partial^2}{\partial Q_l^2} + V(Q_1, \dots, Q_L)$$

原子核の運動エネルギー  
(振動、回転運動)

ポテンシャルエネルギー

## 2.4 シュレディンガー方程式を解く手順

1. シュレディンガー方程式にボルン・オッペンハイマー近似をする
2. 原子核座標 $R$ を固定した状態の電子の波動関数を求める
3. ポテンシャルエネルギー曲面を構築する
4. 基準振動解析：ポテンシャルエネルギー曲面上の原子核の運動を調べる

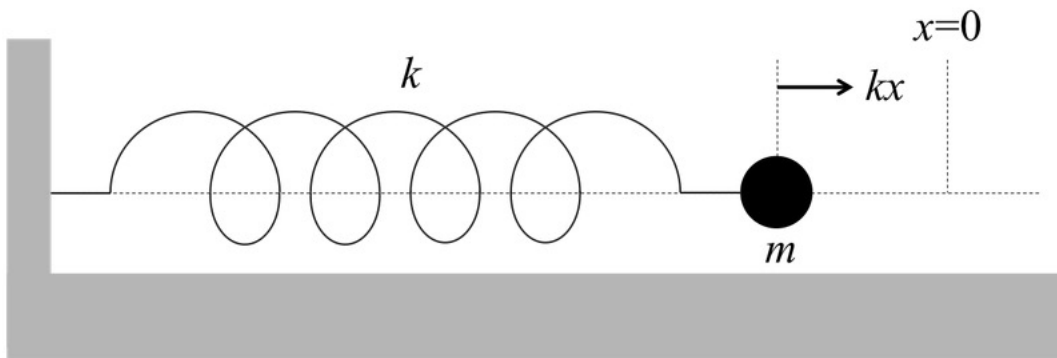


## 2.5 基準振動解析とは？

1. 原子核運動の量子力学的エネルギー準位の計算をする
2. ポテンシャルエネルギー曲面上の原子核のハミルトニアンを定義
3. 原子核の波動関数についてのシュレディンガー方程式を解く
4. 固有関数とエネルギー準位を求める
5. ポテンシャルエネルギー曲面の底のまわりで変位の2次まで取る
6. 調和振動子として近似する：後述

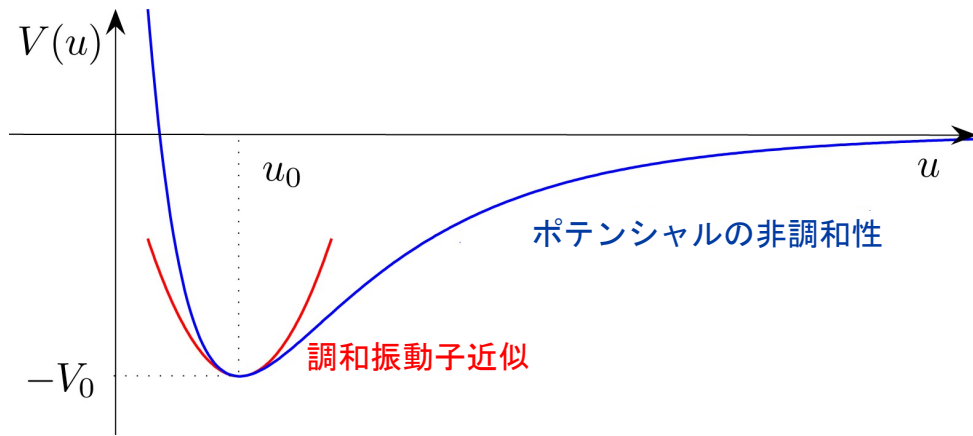


## 2.6 調和振動子とは？



- 調和振動子：質点が定点からの距離に比例する引力を受けて運動する系
- 調和振動子は定点を中心として振動する系、解析的に解くことができる

## 2.7 調和振動子近似とは？



- ポテンシャルエネルギー曲面を放物線として取り扱う近似
- 現実の分子は原子核間距離を伸ばすと解離するため、有限な解離エネルギーを持つ  
→ ポテンシャルの非調和性という



## 2.8 ポテンシャルエネルギー曲面を解く手順

- 基準振動解析を使う
  - 同種多体粒子系として扱う：量子力学で同等なN個の粒子からなる力学的な系
  - vibrational operator :ボース粒子として扱う
1. 一粒子基底（基底関数：modal）：  
ポテンシャルエネルギー曲面に適用される調和近似、またはVSCF計算から得られる
  2. 振動非調和相関は、摂動的または変分的アプローチによって後に追加



## 2.8.1 vibrational operatorと量子ビットの特徴

	vibrational operator	量子ビット
粒子の種類	ボース粒子	区別できる人工原子
特徴	波動関数の符号が不変 粒子の入れ替えに対して対称	同種粒子ではない

## 2.8.2 modalとは？

- vibrational operator:ボース粒子
- ボース粒子をどう基底で表現するか？
- ボース粒子:可算無限個( $0 \sim n$ )で同じエネルギー状態をとるもの
- binaryの0、1で表現しないといけない
- それをどんな風に表現するか？を示したもの



## 2.8.3 modalについての流れ

```
vibrational_problem = VibrationalStructureProblem(driver, num_modals=2, truncation_order=2)
```

1. truncation:ボース粒子の無限個の基底をまず有限に近似
  2. QiskitのVibrational Operatorもまた有限の自由度しか表現できない
  3. そのあとで、modeとmodalをつかって、0と1で表現する
- この後のコードの実行については後述



### 3. ワトソンハミルトニアンについて

1. 基準振動解析の流れ
2. ポテンシャルエネルギーのテイラー展開
3. ワトソンハミルトニアンの第二量子化
4. 波動関数の積基底による展開
5. 生成演算子と消滅演算子のペアを導入
6. 第二量子化形式のポテンシャル表現
7. ポテンシャルの近似値



## 3.1 基準振動解析の流れ

1. 原子核運動の量子力学的エネルギー準位の計算をする
2. ポテンシャルエネルギー曲面上の原子核のハミルトニアンを定義
3. 原子核の波動関数についてのシュレディンガー方程式を解く
4. 固有関数とエネルギー準位を求める
5. ポテンシャルエネルギー曲面の底のまわりで変位の2次まで取る
6. 調和振動子として近似する



## 3.2 ポテンシャルエネルギー曲面のテイラー展開

テイラー展開を用いて、ポテンシャルエネルギーの展開

$$V(Q_1, \dots, Q_L) = V_0 + \sum_{l=1}^L V^{[l]}(Q_l) + \sum_{l < m}^L V^{[l,m]}(Q_l, Q_m) + \sum_{l < m < n}^L V^{[l,m,n]}(Q_l, Q_m, Q_n) + \dots$$

- $Q_1, \dots, Q_L$ : 調和質量加重正規座標
- $V_0$ : 基準の電子エネルギー
- $\sum_{l=1}^L V^{[l]}(Q_l)$ : 平衡位置から  $l$  番目の法線座標を変化させたときの PES の変化
- $\sum_{l < m}^L V^{[l,m]}(Q_l, Q_m)$ :  $l$  番目と  $m$  番目の座標に沿って同時に変位したときの正確な PES の変化
- $\sum_{l < m < n}^L V^{[l,m,n]}(Q_l, Q_m, Q_n)$ :  $l$  番目と  $m$  番目、 $n$  番目の座標に沿って同時に変位したときの正確な PES の変化

### 3.3 ワトソンハミルトニアン第二量子化

$N_l$ 次元基底の $S_l$ によって記述されているとする

$$S_l = \{\phi_1^{(l)}(Q_l), \dots, \phi_{N_l}^{(l)}(Q_l)\}$$

多体基底関数

占有数ベクトルとして符号化

$$\phi_{k_1}(Q_1) \cdots \phi_{k_L}(Q_L) \equiv |0_1 \cdots 1_{k_1} \cdots 0_{N_1}, 0_1 \cdots 1_{k_2} \cdots 0_{N_2}, \dots, 0_1 \cdots 1_{k_L} \cdots 0_{N_L}\rangle$$



### 3.3.1 第二量子化とは？

- 場の量子化
- スピン軌道を用いることで電子が存在しうる場を量子化
- スピン軌道に電子を生成・消滅させることで電子がどこにいるかを記述
- 波動関数の反対称性を第二量子化演算子の代数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

## 3.4 波動関数の積基底による展開

波動関数 $|\psi\rangle$ は積基底 $S$ により配置間相互作用法のように展開

$$|\Psi\rangle = \sum_{k_1=1}^{N_1} \cdots \sum_{k_L=1}^{N_L} c_{k_1, \dots, k_L} \phi_{k_1}^{(1)}(Q_1) \cdots \phi_{k_L}^{(L)}(Q_L)$$

積基底

## 3.4.1 配置間相互作用法とは？

- 変分法により、電子相関を見積もる方法
- 波動関数を電子配置（Slater行列式またはその線形結合）で展開する
- ハミルトニアン行列を構築して、対角化することによって、各配置の係数を決定する

$$\| \chi_1 \cdots \chi_N \| = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_1(x_1) & \cdots & \chi_1(x_N) \\ \vdots & \ddots & \vdots \\ \chi_N(x_1) & \cdots & \chi_N(x_N) \end{vmatrix} \quad \text{Slater行列式}$$



## 3.5 生成演算子と消滅演算子のペアを導入

第二量子化演算子に基づいて、生成演算子・消滅演算子のペアを導入

$$a_{k_l}^\dagger |\cdots, 0_1 \cdots 0_{k_l} \cdots 0_{N_l}, \cdots\rangle = |\cdots, 0_1 \cdots 1_{k_l} \cdots 0_{N_l}, \cdots\rangle$$

$$a_{k_l}^\dagger |\cdots, 0_1 \cdots 1_{k_l} \cdots 0_{N_l}, \cdots\rangle = 0$$

$$a_{k_l} |\cdots, 0_1 \cdots 1_{k_l} \cdots 0_{N_l}, \cdots\rangle = |\cdots, 0_1 \cdots 0_{k_l} \cdots 0_{N_l}, \cdots\rangle$$

$$a_{k_l} |\cdots, 0_1 \cdots 0_{k_l} \cdots 0_{N_l}, \cdots\rangle = 0$$

with

$$[a_{k_l}^\dagger, a_{h_m}^\dagger] = 0$$

$$[a_{k_l}, a_{h_m}] = 0$$

$$[a_{k_l}^\dagger, a_{h_m}] = \delta_{l,m}, \delta_{k_l, h_m}$$

$a_{k_l}^\dagger$ :生成演算子,  $a_{k_l}$ :消滅演算子

### 3.6 第二量子化形式のPES表現

$$\mathcal{H}_{\text{vib}}^{SQ} = \sum_{l=1}^L \sum_{k_l, h_l}^{N_l} \langle \phi_{k_l} | T(Q_l) + V^{[l]}(Q_l) | \phi_{h_l} \rangle a_{k_l}^+ a_{h_l} + \sum_{l < m}^L \sum_{k_l, h_l}^{N_l} \sum_{k_m, h_m}^{N_m} \langle \phi_{k_l} \phi_{k_m} | V^{[l,m]}(Q_l, Q_m) | \phi_{h_l} \phi_{h_m} \rangle a_{k_l}^+ a_{k_m}^+ a_{h_l} a_{h_m} + \dots$$

第二量子化演算子

第二量子化演算子

PESには、3体や高次の多体結合項が含まれている

→ 6つ以上の第二量子化演算子を持った演算子となっている

→ これが精度の高いハミルトニアンを求める理由につながる

## 3.7 ポテンシャルエネルギーの近似値

$$V(Q_1, \dots, Q_L) = \frac{1}{2} \sum_{ij} k_{ij} Q_i Q_j + \frac{1}{6} \sum_{ijk} k_{ijk} Q_i Q_j Q_k + \frac{1}{16} \sum_{ijkl} k_{ijkl} Q_i Q_j Q_k Q_l$$

$$k_{ij} = \frac{H_{ij}(+\delta Q_k) - H_{ij}(-\delta Q_k)}{2\delta Q_k}$$

$$k_{ijkl} = \frac{H_{ij}(+\delta Q_k + \delta Q_l) - H_{ij}(+\delta Q_k - \delta Q_l) - H_{ij}(-\delta Q_k + \delta Q_l) + H_{ij}(-\delta Q_k - \delta Q_l)}{4\delta Q_k \delta Q_l}$$

- 異なる原子核座標の電子シュレディンガー方程式を解くことで、ポテンシャルエネルギー曲面を求めることができる
- Qiskitでは、PESを四次力場で近似することが可能



## 4.基底状態を求めるまでのコードの流れ

1. Gaussianを使って、分子の構造最適化
2. 第二量子化を行う
3. ハミルトニアンをHarmonic mode基底で表現するためのBoson変換を定義
4. modalとtruncationを設定
5. 振動演算子の計算
6. 第二量子化演算子を量子ビット演算子に変換
7. NumPyソルバーを設定
8. Filter functionを使うかどうかの設定
9. GroundStateSolverの設定
10. 振動構造の基底状態を求める



## 5. ハミルトニアンを用意

1. コード実行の流れ
2. 分子の構造最適化
3. Gaussianとは？
4. ハミルトニアンをHarmonic mode基底で表現するためのBoson変換
5. 振動演算子の計算
6. 第二量子化演算子を量子ビット演算子に変換
7. num\_modal=2の時の結果表示
8. num\_modal=3を使う方法
9. 8.と9.の結果比較





## 5.1 コード実行の流れ

1. Gaussianを使って、分子の構造最適化
2. 第二量子化を行う
3. ハミルトニアンをHarmonic mode基底で表現するためのBoson変換
4. modalとtruncationを設定
5. 振動演算子の計算
6. 第二量子化演算子を量子ビット演算子に変換

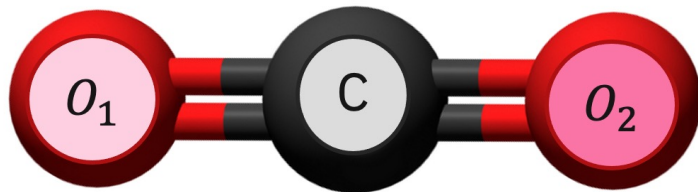


## 5.2 分子の構造最適化

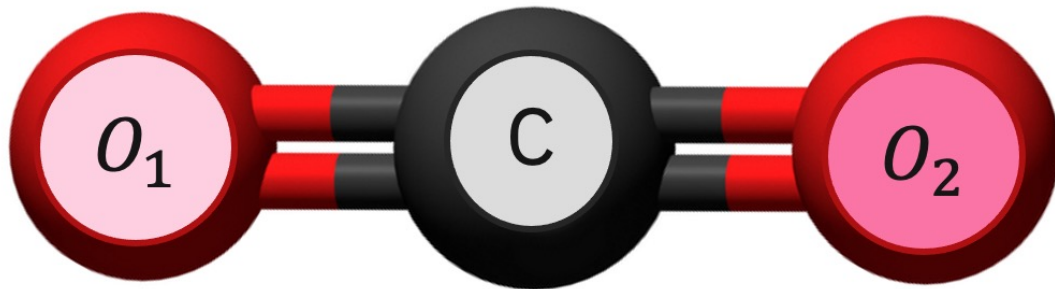
```
# GaussianForceDriverをインポートする
# Gaussianパッケージで二酸化炭素CO2の構造最適化を求める
from qiskit_nature.drivers.second_quantization import GaussianForcesDriver

# Gaussianを使う場合、output fileを使う
#
driver = GaussianForcesDriver(logfile='aux_files/CO2_freq_B3LYP_ccpVDZ.log')

# QiskitからGaussianを動かす場合は以下のコードを使う
# driver = GaussianForcesDriver(
#     ['#p B3LYP/6-31g Freq=(Anharm) Int=Ultrafine SCF=VeryTight',
#     ''],
#     # 二酸化炭素の構造最適化
#     ['CO2 geometry optimization B3LYP/6-31g',
#     ''],
#     # 位置構造設定・ラベル設定
#     ['O 1',
#      'C -0.848629 2.067624 0.160992',
#      'O 0.098816 2.655801 -0.159738',
#      'O -1.796073 1.479446 0.481721',
#      ''],
#     #
```



## 5.2.1 二酸化炭素の座標設定



$$O_1 = \begin{bmatrix} -1.796073 \\ 1.479446 \\ 0.481721 \end{bmatrix}$$

$$C = \begin{bmatrix} -0.848629 \\ 2.067624 \\ 0.160992 \end{bmatrix}$$

$$O_2 = \begin{bmatrix} 0.098816 \\ 2.655801 \\ -0.159728 \end{bmatrix}$$

## 5.2.2 GaussianForcesDriverについて

### Gaussian™ 16 Installation 🇯🇵

**Gaussian™ 16** is a commercial program for computational chemistry. This chemistry driver accesses electronic structure information from Gaussian™ 16 via the Gaussian-supplied open-source [interfacing code](#).

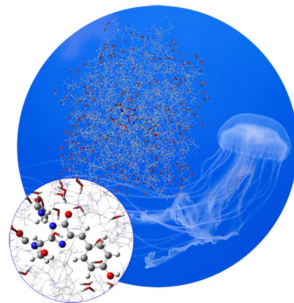
You should follow the installation instructions that come with your Gaussian™ 16 product. Installation instructions can also be found online in [Gaussian product installation support](#).

Following the installation make sure the Gaussian™ 16 executable, *g16*, can be run from the command line environment where you will be running Python and Qiskit. For example verifying that the *g16* executable is reachable via the system environment path, and appropriate exports, such as *GAUSS\_EXEDIR*, have been configured as per [Gaussian product installation support](#).

### Gaussian16をサポートしているパッケージ



## 5.3 Gaussianとは？



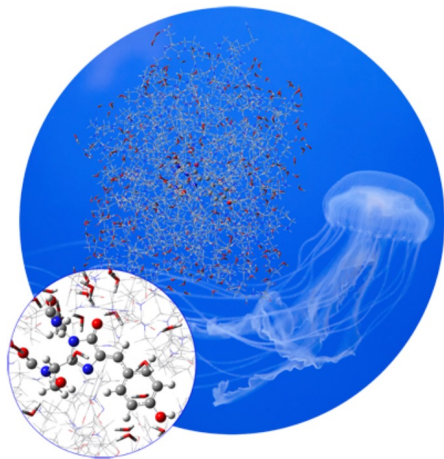
- ab initioモデルや半経験的モデルをサポートした汎用量子化学計算パッケージプログラム
- できること：Gaussianにより分子の構造最適化ができる
- ログファイル
  - ログファイルでは、結果の解析、軌道や電子密度のチェックが可能

## 5.3.1 Gaussianの特徴

- 分子の電子状態を *Hartree Fock* 法、密度汎関数法(DFT)、MP法、CC法などで計算する
- 構造最適化や振動解析、PCM法などによる溶媒効果、時間依存密度汎関数法(Time-Dependent DFT)やSAC-CI法などを用いることで巨大な分子を少ない計算コストで精度良く計算することもできる
- CC(Coupled Cluster)法：結合クラスター法、多体系を記述するために使われる計算手法
- 密度汎関数法・DFT法:電子密度を変数としてエネルギーを計算する方法
- MP法:電子相関を摂動として扱い、系のエネルギーや電子波動関数を摂動展開により求める方法



## 5.3.2 Gaussianの計算方法の利用



- 振動構造のシュレディンガー方程式を解くときに使われている
- ワトソンハミルトニアンを解くときの数値的手法

## 5.4 振動演算子を求める

```
# Bosonから量子ビットへのマッピングが直接行われるプログラム
# second_quantizationからVibrationalStructureProblemをインポート
from qiskit_nature.problems.second_quantization import VibrationalStructureProblem

# second_quantizationからQubitConverterをインポート
from qiskit_nature.converters.second_quantization import QubitConverter

# second_quantizationからDirectMapperをインポート
# DirectMapperはVibrationalOpをPauliSumOpにマッピングする
from qiskit_nature.mappers.second_quantization import DirectMapper

# 振動構造の第二量子化
# ハミルトニアンをHarmonic mode基底で表現するためのBoson変換を定義
# modals per mode: num_modals=2
# ポテンシャルは次数2で切り捨て: truncation_order=2
vibrational_problem = VibrationalStructureProblem(driver, num_modals=2, truncation_order=2)

# 第二量子化演算子・SecondQuantizedOpのリストを返す
second_q_ops = vibrational_problem.second_q_ops()

# 振動演算子の結果を表示する
print(second_q_ops[0])
```





## 5.4.1 *VibrationalStructureProblem*について

```
CLASS VibrationalStructureProblem(bosonic_driver, num_modals, truncation_order,  
                                   transformers=None)
```

- 振動構造の問題を解決するためのクラス
- 初期設定 : *num\_modals*, *truncation\_order*を設定
- パラメーター:
  - *bosonic\_driver*: 分子情報をコード化したBosonicドライバー
  - *num\_modals* (*Union[int, List[int]]*) : modeごとのmodalの数
  - *truncation\_order* (*int*): n体展開が切り捨てられる順序
  - *transformers* (*Optional[List[Union[BaseTransformer, BaseTransformer]]]*)  
ドライバーの結果に適用される変換のリスト



## 5.4.2 QubitConverterについて

### QubitConverter 🍷


```
CLASS QubitConverter (mapper, two_qubit_reduction=False, z2symmetry_reduction=None)
```

- 第二量子化演算子から量子ビットへの変換を行うクラス
- パラメーター
  - *mapper*: 第二量子化演算子を量子ビット演算子に変換するためのマッパーインスタンス
  - *two\_qubit\_reduction*: 可能であれば、2量子ビット削減を行うかどうか
  - *z2symmetry\_reduction*:  
計算された結果の量子ビット演算子に、z2対称性の削減を適用するかどうかを示す



## 5.4.3 *DirectMapper*について

```
DirectMapper.map(second_q_op)
```

[SOURCE] 

Maps a *VibrationalOp* to a *PauliSumOp*.

Parameters

**second\_q\_op** (*VibrationalOp*) – the *VibrationalOp* to be mapped.

Return type

*PauliSumOp*

Returns

The *PauliSumOp* corresponding to the problem-Hamiltonian in the qubit space.

- *VibrationalOp*を*PauliSumOp*にマッピング
- *VibrationalOp*の各モードが1つの量子ビットにマッピングされる
- methods : map(second\_q\_op)
- パラメーター : second\_q\_op
- 量子ビット空間のProblem,ハミルトニアンに対応する*PauliSumOp*



## 5.5 第二量子化演算子を量子ビット演算子に変換

```
# Bosonから量子ビットへのマッピングが直接行われるプログラム・計算する
# VibrationalOpをPauliSumOpにマッピングする
qubit_converter = QubitConverter(mapper=DirectMapper())

# 第二量子化演算子を量子ビット演算子へ変換する
qubit_op = qubit_converter.convert(second_q_ops[0])

# 第二量子化演算子の結果を表示する
print(qubit_op)
```

## 5.6 num\_modal=2の時の結果表示

```
5077.236560625012 * IIIIIIIII
- 472.28961593750034 * ZIIIIIIII
- 157.22939093749991 * IZIIIIIII
- 467.5058515625001 * IIZIIIIII
- 1.2397992187500007 * ZIZIIIIII
- 0.4132664062500002 * IZZIIIIII
```

← パウリ演算子が8個

## 5.7 他のnum\_modalに変えて行う方法

1. num\_modal=3の時
2. 結果表示
3. num\_modal=2,num\_modal=3の時の比較



## 5.7.1 num\_modal=3の時

```
# modeごとに異なる数のmodalを持つプログラム
# modals per mode: num_modals=3
# ポテンシャルは次数2で切り捨て: truncation_order=2
vibrational_problem = VibrationalStructureProblem(driver, num_modals=3, truncation_order=2)

# 第二量子化演算子・SecondQuantizedOpのリストを返す
second_q_ops = vibrational_problem.second_q_ops()

# VibrationalOpをPauliSumOpにマッピングする
qubit_converter = QubitConverter(mapper = DirectMapper())

# 第二量子化演算子を量子ビット演算子へ変換する
qubit_op = qubit_converter.convert(second_q_ops[0])

# 第二量子化演算子の結果を表示
print(qubit_op)
```

## 5.7.2 $num\_modal=3$ の時の結果表示

11327.034966562516	*	IIIIIIIIIIIIII
- 745.8599576562498	*	ZIIIIIIIIIIII
- 446.6739542187503	*	IZIIIIIIIIIIII
- 148.69083703124986	*	IIIZIIIIIIIIII
- 724.2074121874995	*	IIIIZIIIIIIII
- 3.4438867187500013	*	ZIIIZIIIIIIII

← パウリ演算子が12個



## 5.8 $num\_modal=2, num\_modal=3$ の時の結果比較

- $num\_modal=2$ の時、パウリ演算子が8個
- $num\_modal=3$ の時、パウリ演算子が12個

$num\_modal=2$

```
5077.236560625012 * IIIIIIII
- 472.28961593750034 * ZIIIIIII
- 157.22939093749991 * IZIIIIIII
- 467.5058515625001 * IIZIIIIIII
- 1.2397992187500007 * ZIZIIIIIII
- 0.4132664062500002 * IZZIIIIIII
```

$num\_modal=3$

```
11327.034966562516 * IIIIIIIIIIII
- 745.8599576562498 * ZIIIIIIIIIII
- 446.6739542187503 * IZIIIIIIIIIII
- 148.69083703124986 * IIZIIIIIIIIIII
- 724.2074121874995 * IIIZIIIIIIIIIII
- 3.4438867187500013 * ZIIZIIIIIIIIIII
```

## 6. 基底状態ソルバーを求める

1. コードの流れ
2. Filter functionを使う理由
3. Filter functionを使ったコード
4. Filter functionを使わないコード
5. 結果比較



## 6.1 コードの流れ

1. ハミルトニアンを用意:前述
2. ***NumPy***ソルバーを設定
3. ***Filter function***を使うかどうかの設定
4. ***GroundStateSolver***の設定
5. 振動構造の基底状態を求める

詳細はこちら : [https://qiskit.org/documentation/nature/tutorials/03\\_ground\\_state\\_solvers.html#Using-a-filter-function](https://qiskit.org/documentation/nature/tutorials/03_ground_state_solvers.html#Using-a-filter-function)



## 6.2 Filter functionを使う理由

```
CLASS NumPyMinimumEigensolverFactory(filter_criterion=None,  
                                       use_default_filter_criterion=False)
```

[SOURCE] 

NumPy固有値ソルバーは、正しい粒子数を持つ固有値のみを返す

→これが、**Filter function**

→ハミルトニアン<sup>1</sup>の真の基底状態が真空状態である振動構造計算の場合に特に重要

→粒子数をチェックするデフォルトのFilter functionは、さまざまな変換に実装

## 6.2 Filter Functionを使ったコード

```
# 振動構造のハミルトニアンを求める
# Filter Functionを使った場合:use_default_filter_criterion=True
from qiskit_nature.drivers.second_quantization import GaussianForcesDriver
from qiskit_nature.algorithms import NumPyMinimumEigensolverFactory
from qiskit_nature.problems.second_quantization import VibrationalStructureProblem
from qiskit_nature.mappers.second_quantization import DirectMapper
from qiskit_nature.algorithms import GroundStateEigensolver
from qiskit_nature.converters.second_quantization import QubitConverter

driver = GaussianForcesDriver(logfile='aux_files/CO2_freq_B3LYP_ccpVDZ.log')

vib_problem = VibrationalStructureProblem(driver, num_modals=2, truncation_order=2)

qubit_converter = QubitConverter(DirectMapper())

# Filter functionを使った場合
solver_with_filter = NumPyMinimumEigensolverFactory(use_default_filter_criterion=True)

gsc_w = GroundStateEigensolver(qubit_converter, solver_with_filter)
result_w = gsc_w.solve(vib_problem)

# 振動構造の基底状態を求める
print(result_w)
```

use\_default\_filter\_criterion=true

## 6.2.1 NumPyMinimumEigensolverFactoryについて

```
CLASS NumPyMinimumEigensolverFactory(filter_criterion=None,  
                                       use_default_filter_criterion=False)
```

[SOURCE] 📄

- NumPyMinimumEigensolverを構築するためのFactory
- 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

## 6.2.2 *use\_default\_filter\_criterion*の設定

```
CLASS NumPyMinimumEigensolverFactory(filter_criterion=None,  
                                       use_default_filter_criterion=False)
```

[SOURCE] 📄

- パラメータ: *user\_default\_filter\_criterion*(bool)  
*filter\_criterion* = None のとき、変換のdefaultの*filter\_criterion*を使用するかどうか

## 6.3 Filter functionを使わない方法

```
# 振動構造のハミルトニアンを求める
# Filter functionを使わなかった場合: use_default_filter_criterion=False
from qiskit_nature.drivers.second_quantization import GaussianForcesDriver
from qiskit_nature.algorithms import NumPyMinimumEigensolverFactory
from qiskit_nature.problems.second_quantization import VibrationalStructureProblem
from qiskit_nature.mappers.second_quantization import DirectMapper
from qiskit.algorithms import NumPyMinimumEigensolver
from qiskit_nature.algorithms import GroundStateEigensolver
from qiskit_nature.converters.second_quantization import QubitConverter

# Gaussianにinput fileを保存する
driver = GaussianForcesDriver(logfile='aux_files/C02_freq_B3LYP_ccpVDZ.log')

# 振動構造の第二量子化
# ハミルトニアンをHarmonic mode基底で表現するためのBoson変換を定義
# modalsとポテンシャルの次数を設定する
# modals per mode: num_modals=2
# ポテンシャルは次数2で切り捨て: truncation_order=2
vib_problem = VibrationalStructureProblem(driver, num_modals=2, truncation_order=2)

# VibrationalOpをPauliSumOpにマッピングする
qubit_converter = QubitConverter(DirectMapper())

# Filter functionを使わなかった場合
solver_without_filter = NumPyMinimumEigensolverFactory(use_default_filter_criterion=False)

# GroundStateSolverからの設定
gsc_wo = GroundStateEigensolver(qubit_converter, solver_without_filter)
result_wo = gsc_wo.solve(vib_problem)

# 振動構造の基底状態を求める
print(result_wo)
```

use\_default\_filter\_criterion=false





## 6.4 結果比較

正しい粒子数を持つ固有値のみを返す

### Filter functionありの場合

=== GROUND STATE ENERGY ===

\* Vibrational ground state energy ( $\text{cm}^{-1}$ ): (2534.621084885814+0j)  
The number of occupied modals is  
- Mode 0: [1.0, 1.0, 1.0, 1.0]

### Filter functionなしの場合

=== GROUND STATE ENERGY ===

\* Vibrational ground state energy ( $\text{cm}^{-1}$ ): (1.1e-11-0j)  
The number of occupied modals is  
- Mode 0: [0.0, 0.0, 0.0, 0.0]



# 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
- 「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
- アトキンス物理化学
- 現代物理化学
- 物理化学（化学の基本シリーズ）
- Qiskit Nature Tutorials
- Qiskit Textbook
- Qiskit Tutorials
- Quantum Tokyo
- Medium "Introducing Qiskit Nature"
- IBM Quantum Challenge 2021
- 実験医学HP
- すぐできる量子化学計算・ビギナーズマニュアル
- Qiskit Global Summer School Chemistry
- ザボ、オストランド『新しい量子化学』
- IBM Quantum Open Lab
- ボナハルト・ショアー有機化学
- Head First デザインパターン



Thank you