


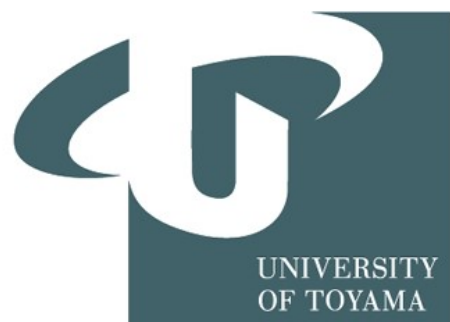
# R5 電子デバイス工学特論1 発表資料 総集編

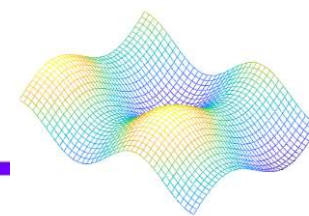


担当教員 富山大学 岡田 裕之

# Classical shadows

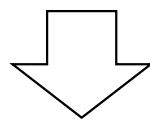
## (古典的な影)



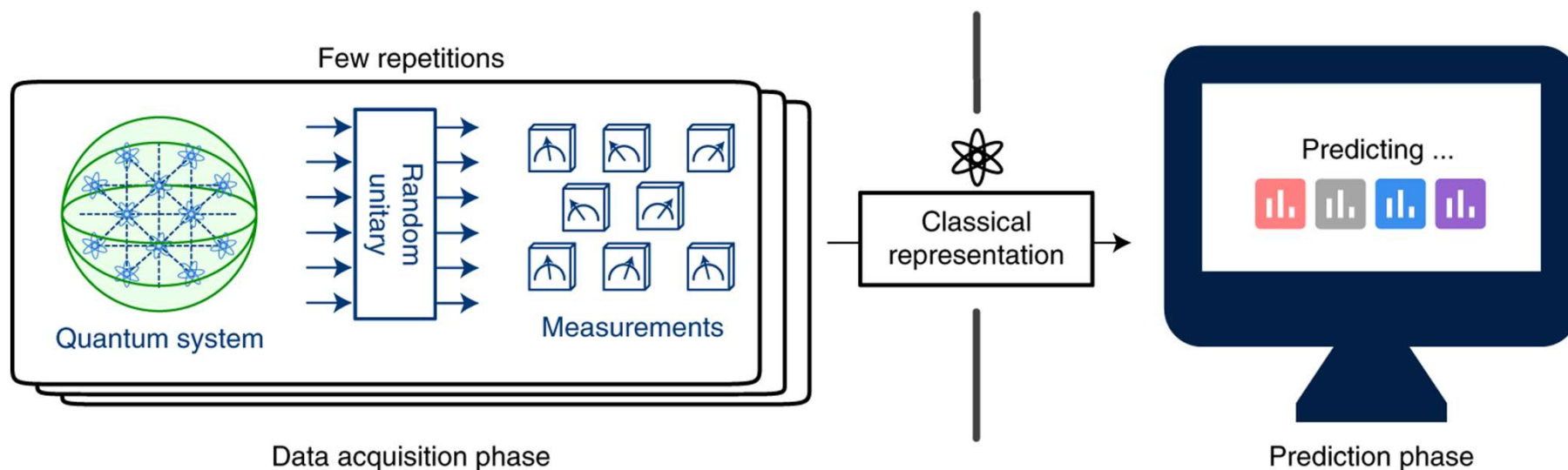


## 未知の量子状態の性質の推定

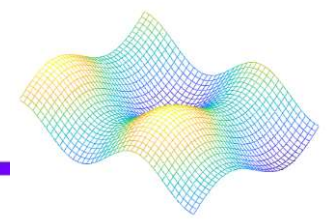
量子ビット数に応じて増大する観測量の正確な期待値が必要



## 古典的な影近似



# 1. 古典的な影の構成



$$\rho \rightarrow U\rho U^\dagger.$$

$$|b\rangle = |0011 \dots 10\rangle$$

$$\mathbb{E}[U^\dagger |b\rangle\langle b|U] = \mathcal{M}(\rho). \quad \dots \textcircled{1}$$

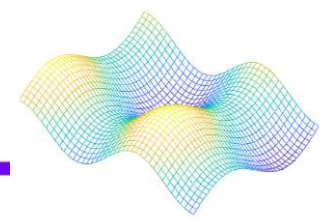
$$\rho = \mathbb{E}[\mathcal{M}^{-1}(U^\dagger |b\rangle\langle b|U)]. \quad \dots \textcircled{2}$$

$$S(\rho, N) = \{\hat{\rho}_1 = \mathcal{M}^{-1}(U^\dagger_1 |b_1\rangle\langle b_1|U_1), \\ \dots, \hat{\rho}_N = \mathcal{M}^{-1}(U^\dagger_N |b_N\rangle\langle b_N|U_N)\}. \quad \dots \textcircled{3}$$

$$\langle O \rangle = \frac{1}{N} \sum_i \text{Tr} \hat{\rho}_i O. \quad \dots \textcircled{4}$$



# 1. 古典的な影の構成



1. 量子状態 $\rho$ は回路で準備される。
2.  $U$ が集合からランダムに選択され、 $\rho$ に適用される。
3. 計算基底測定が実行される。
4. スナップショットは、それぞれ $|0\rangle$ ,  $|1\rangle$ に対して観測された固有値 $1, -1$ と、ランダムに選択された単一の $U$ の指標として記録される。

## 2. 古典的な影からの再構成

---

$$\rho = \mathbb{E}[\mathcal{M}^{-1}(U^\dagger |\hat{b}\rangle\langle \hat{b}| U)].$$

$$\rho = \mathbb{E}|\hat{\rho}| \quad , \quad \hat{\rho} = \otimes \left( \mathbb{3} U_j^\dagger |\hat{b}_j\rangle\langle \hat{b}_j| U_j - \Pi \right).$$

### 3. 古典的な影を用いたパウリ観測量の推定

$$\langle O_{(k)} \rangle = \text{Tr}\{O\hat{\rho}_{(k)}\} \quad \text{and} \quad \hat{\rho}_{(k)} = \frac{1}{\lfloor N/K \rfloor} \sum_{i=(k-1)\lfloor N/K \rfloor+1}^{k\lfloor N/K \rfloor} \hat{\rho}_i.$$

$$\langle O \rangle = \text{median}\{\langle O_{(1)} \rangle, \dots, \langle O_{(K)} \rangle\}.$$

$$O = \bigotimes_j^n P_j, \text{ where } P_j \in \{I, X, Y, Z\}. \dots \textcircled{5}$$

$$\begin{aligned} \text{Tr}\{O\hat{\rho}_i\} &= \text{Tr}\left\{\bigotimes_{j=1}^n P_j (3U_j^\dagger |\hat{b}_j\rangle \langle \hat{b}_j| U_j - \mathbb{I})\right\} \\ &= \prod_j^n \text{Tr}\{3P_j U_j^\dagger |\hat{b}_j\rangle \langle \hat{b}_j| U_j\}. \end{aligned}$$

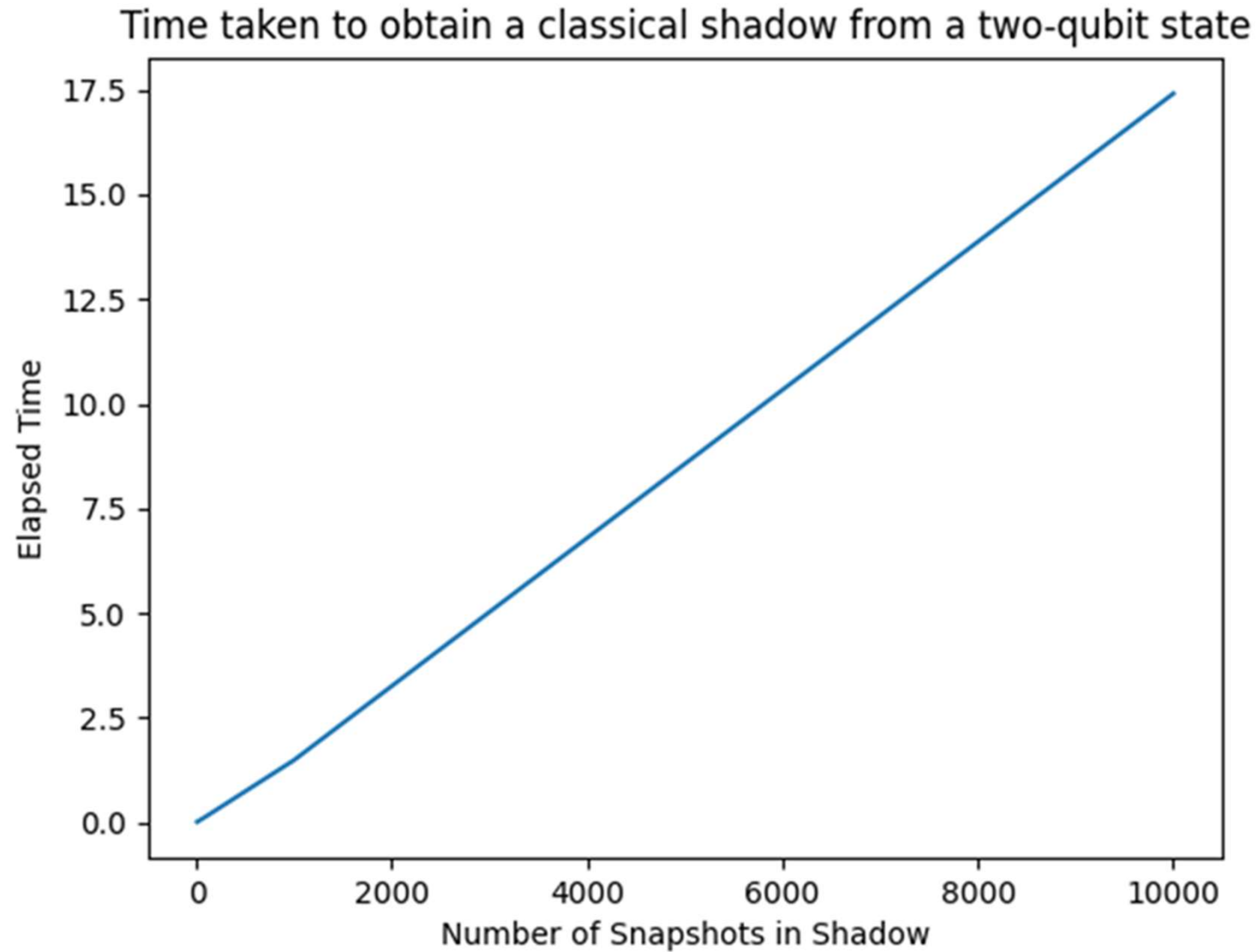
### 3. 古典的な影を用いたパウリ観測量の推定

---

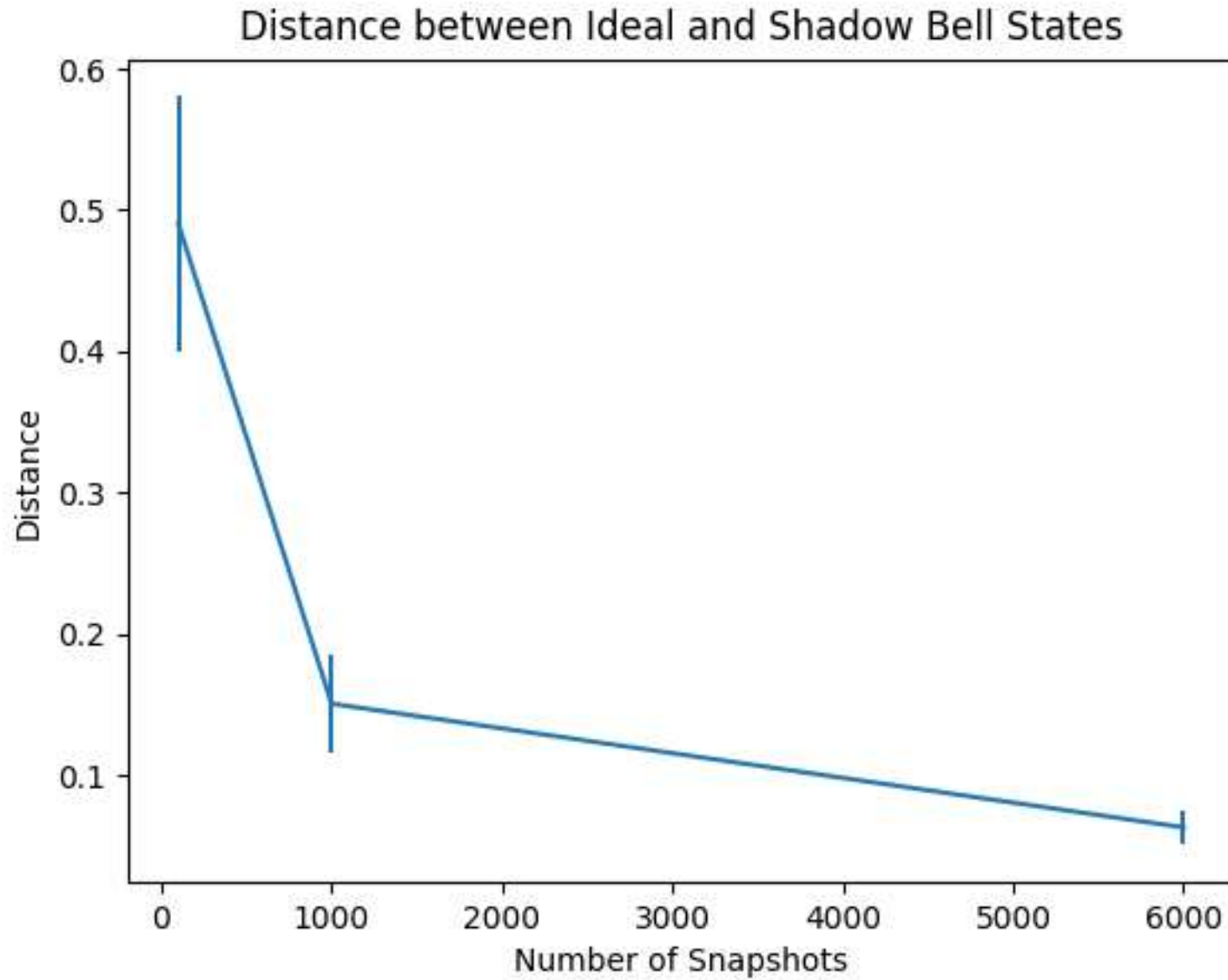
$$O = \sum_{i=0}^{n-1} X_i X_{i+1} + Y_i Y_{i+1} + Z_i Z_{i+1}. \dots \textcircled{6}$$

$$|\langle O_i \rangle_{shadow} - \langle O_i \rangle_{exact}| \leq \epsilon \dots \textcircled{7}$$

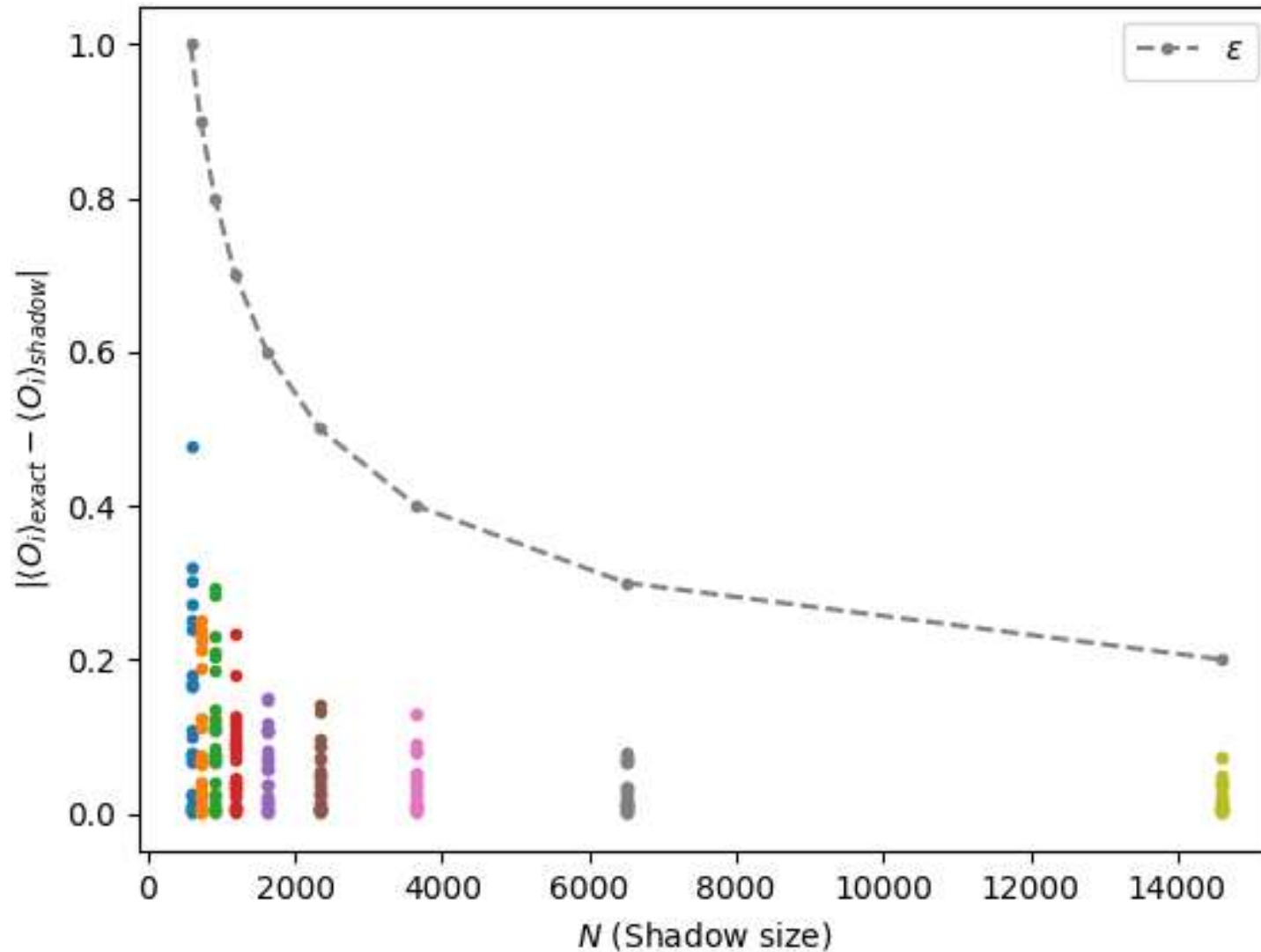
# 1. 古典的な影の構成 実行結果



## 2. 古典的な影からの再構成 実行結果



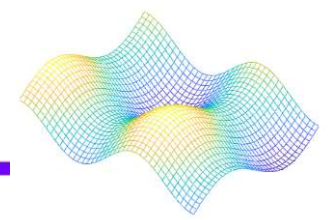
### 3. 古典的な影を用いたパウリ観測量の推定 実行結果



# Beyond classical computing with qsim





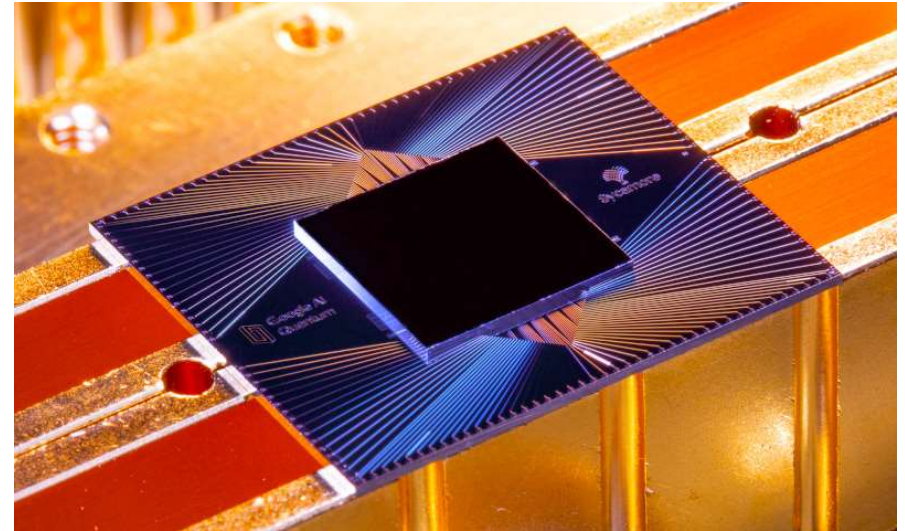
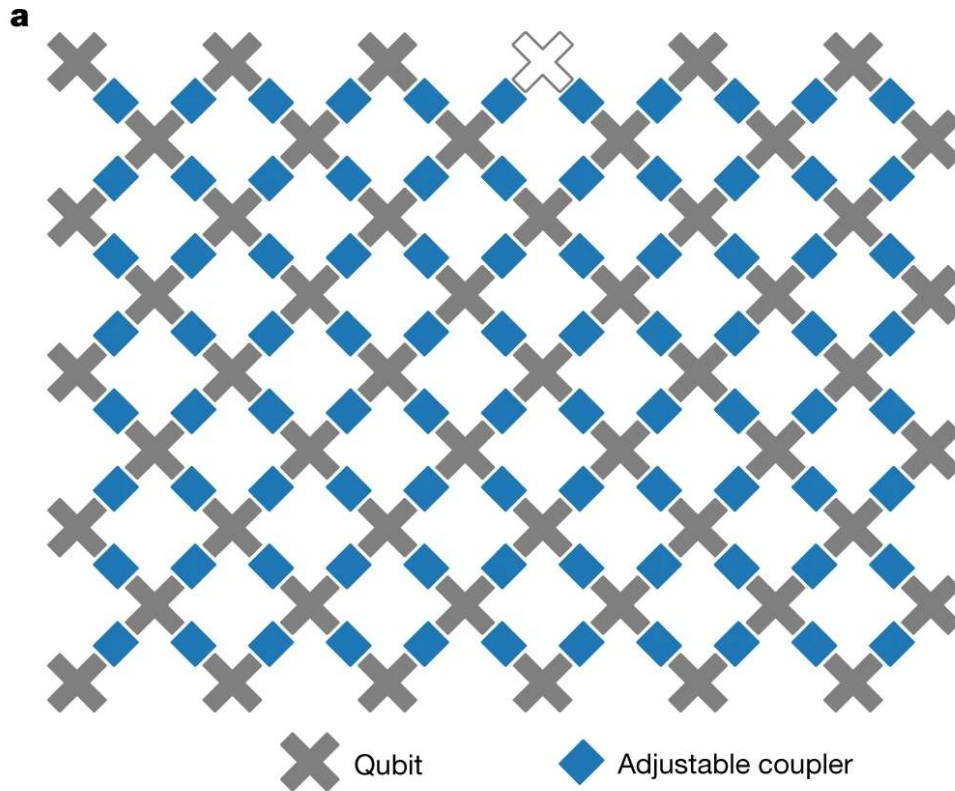
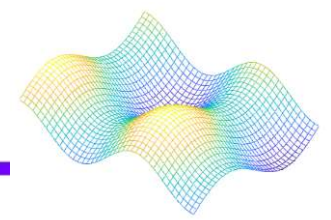


## 量子超越とは

現在のスーパーコンピュータでは実行不可能な計算を量子コンピュータで高速に実行できること

GoogleがSycamore量子プロセッサを用いて量子超越を達成したのでその手法を紹介する

# Sycamore量子プロセッサ

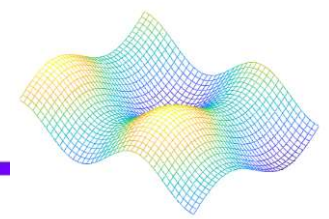


Sycamore google

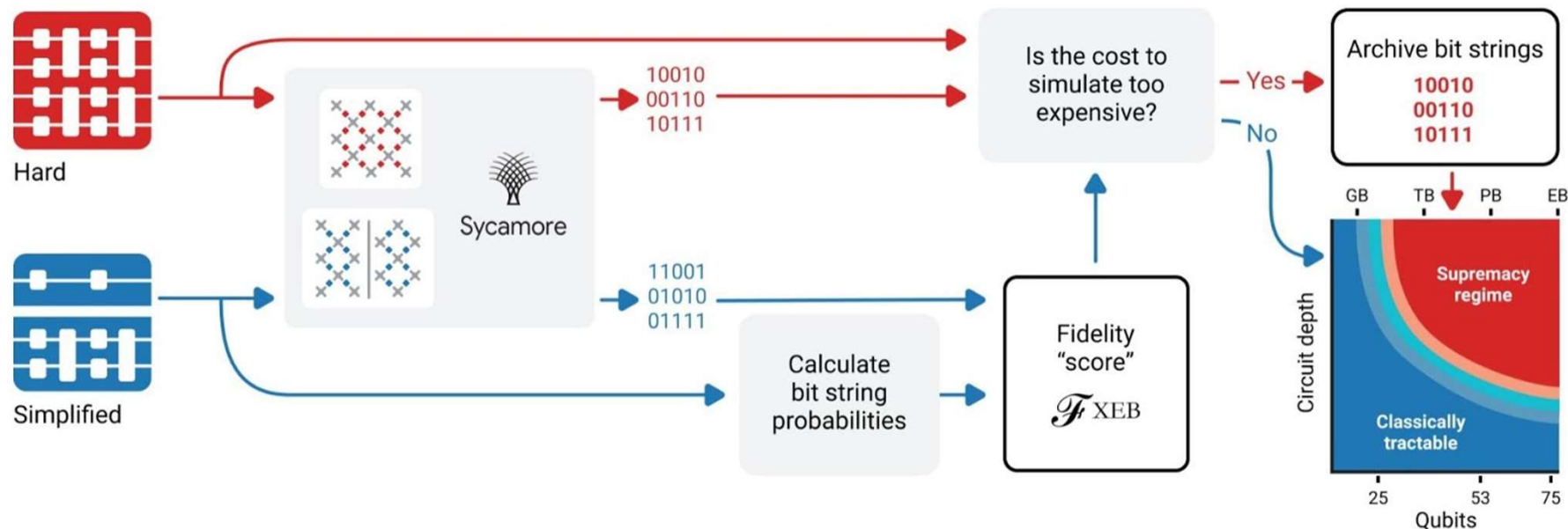
## Sycamoreプロセッサ

- ・54量子ビットを持つプロセッサ
- ・各量子ビットは、直方格子内で4つの最も近い隣接量子ビットに結合されている

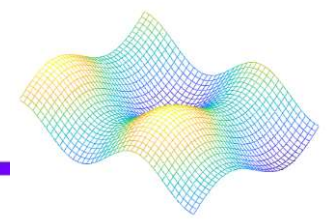
# 量子超越性達成手法



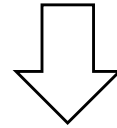
- 1 Choose a specific quantum circuit
- 2 Run circuit on quantum processor
- 3 Estimate quantum processor's fidelity, and determine cost of labor (Classical computation)
- 4 Result: Quantum supremacy achieved



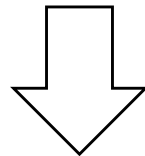
簡単な量子回路を古典コンピュータで評価するのに必要な時間から、計算困難な量子回路の時間を見積もり、Sycamoreプロセッサでかかる時間と比較する



量子ゲートの  
準備

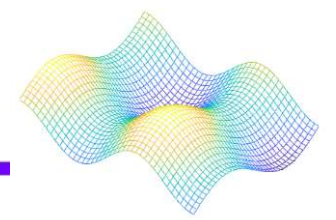


ランダム量子回路  
の構築



評価

# 単一量子ビットゲート



## 単一量子ビットゲート

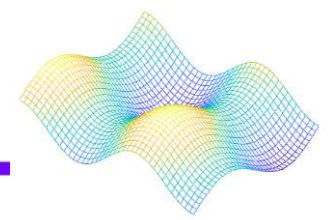
$\sqrt{X}$ ゲート、 $\sqrt{Y}$ ゲート、 $\sqrt{W}$ ゲートの3つが必要

$$\sqrt{X} = Rx\left(\frac{\pi}{2}\right) \quad Rx(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$\sqrt{Y} = Ry\left(\frac{\pi}{2}\right) \quad Ry(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$\sqrt{W} = \sqrt{\frac{X+Y}{2}}$$

# 2量子ビットゲート



iSWAPゲート

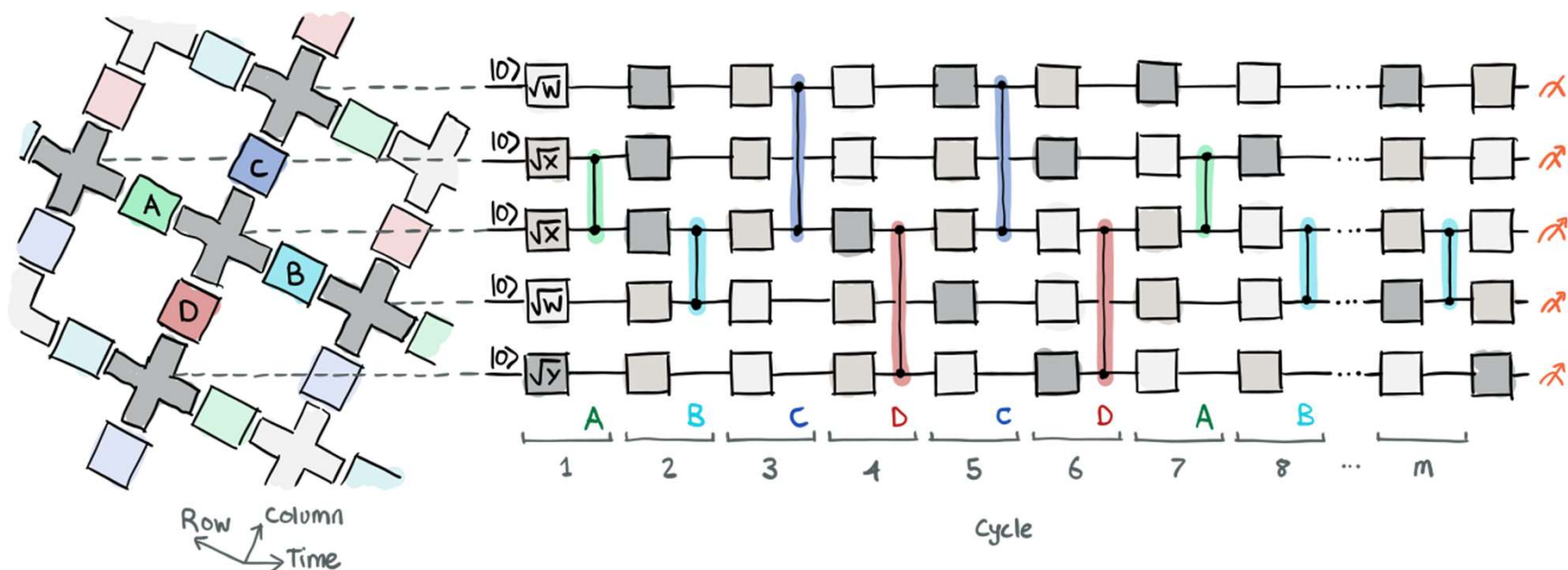
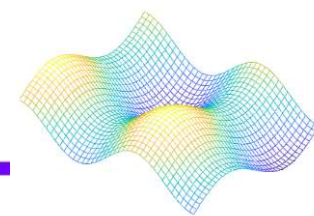
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CPhaseゲート

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\phi} \end{bmatrix}$$



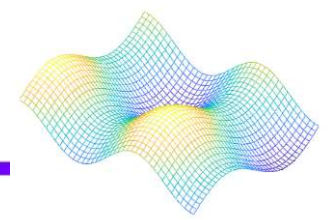
# ランダム量子回路の構築



ランダムな単一量子  
ビットゲートを適用

特定のペアに2量子  
ビットゲートを適用

ポータートンプソン  
確率分布により  
特定のビット列を  
出力



クロスエントロピーのベンチマーク忠実度 ( $F_{XEB}$ )

$$F_{XEB} = 2^n \langle P(x_i) \rangle - 1 \quad n: \text{量子ビット数}$$

$P(x_i)$ : 理想的な量子回路で計算されたビット列  
 $x_i$ の確率

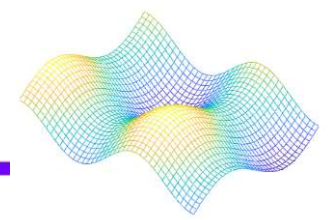
ランダム量子回路によるビット列の分布場合

$$F_{XEB} \cong 1$$

古典的な一様な確率分布の場合

$$F_{XEB} \cong 0$$





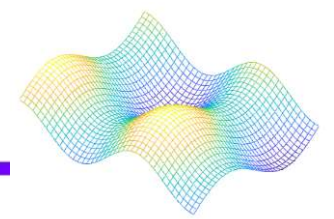
ランダム量子回路からのビット列のサンプリングは次の分布に従う

$$\Pr(p) = (N - 1)(1 - p)^{N-2} \quad N = 2^n$$

この分布は正規分布によって近似できる

$$\Pr(p) = N e^{-Np}$$

$$\langle P(x_i) \rangle = \int_0^1 p^2 N(N - 1)(1 - p)^{N-2} dp = \frac{2}{N + 1}$$

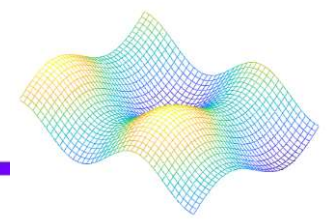


$$F_{XEB} = 2^n \langle P(x_i) \rangle - 1 = \frac{2N}{N+1} - 1$$

$N = 2^n = 2^{12}$ を代入すると

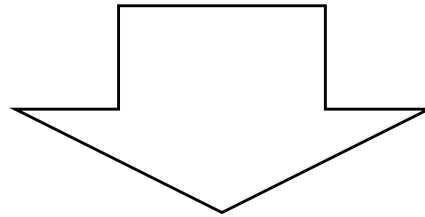
$$\begin{aligned} F_{XEB} &= \frac{2^{13}}{2^{12} + 1} - 1 \\ &\cong \frac{2^{13}}{2^{12}} - 1 = 2 - 1 = 1 \end{aligned}$$

プログラムを動作させて実際にランダム量子回路の忠実度が1になるか確認する



ランダム量子回路の忠実度が1に近づく理由は、ポーター・トンプソン確率分布により特定のビット列を出力しやすい為である

古典的にポーター・トンプソン確率分布をシミュレートすることは困難である

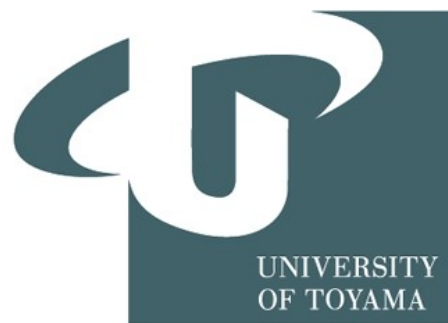


十分大きな回路で高い忠実度を得ることで量子超越性を示すことができる

# Quantum advantage with Gaussian Boson

## Sampling

ガウシアンボソンサンプリングによる量子の利点



# はじめに

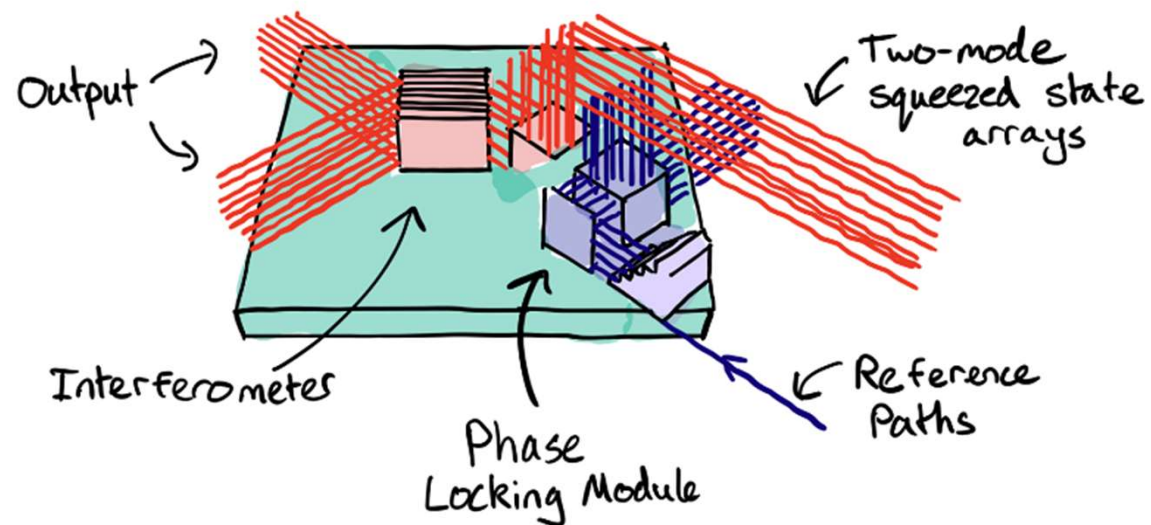
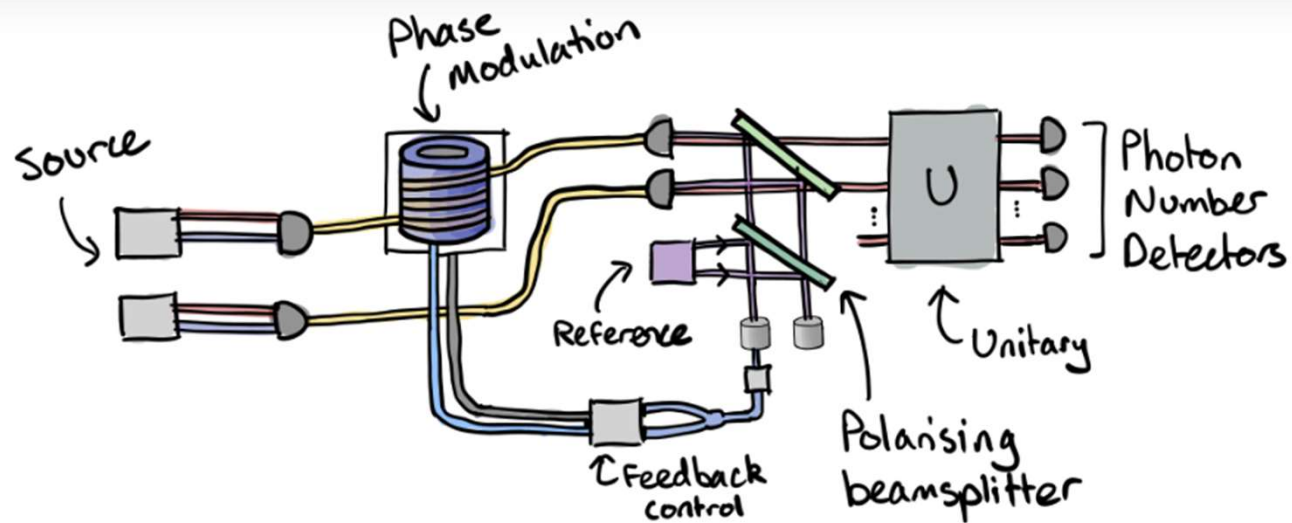
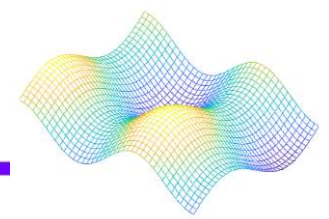
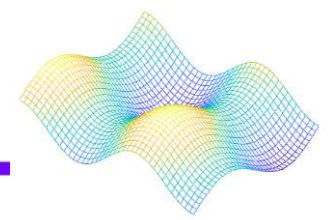
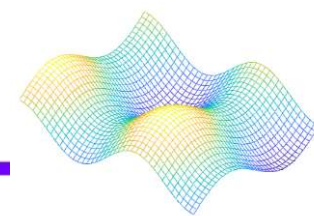


Illustration of the experimental setup used by Zhong et al. in Quantum computational advantage using photons [2].



- ① Gaussian Boson Sampling (GBS) の基本要素を説明
- ② GBS が古典的に難しい理由の説明
- ③ PennyLaneを使用して GBS の探索

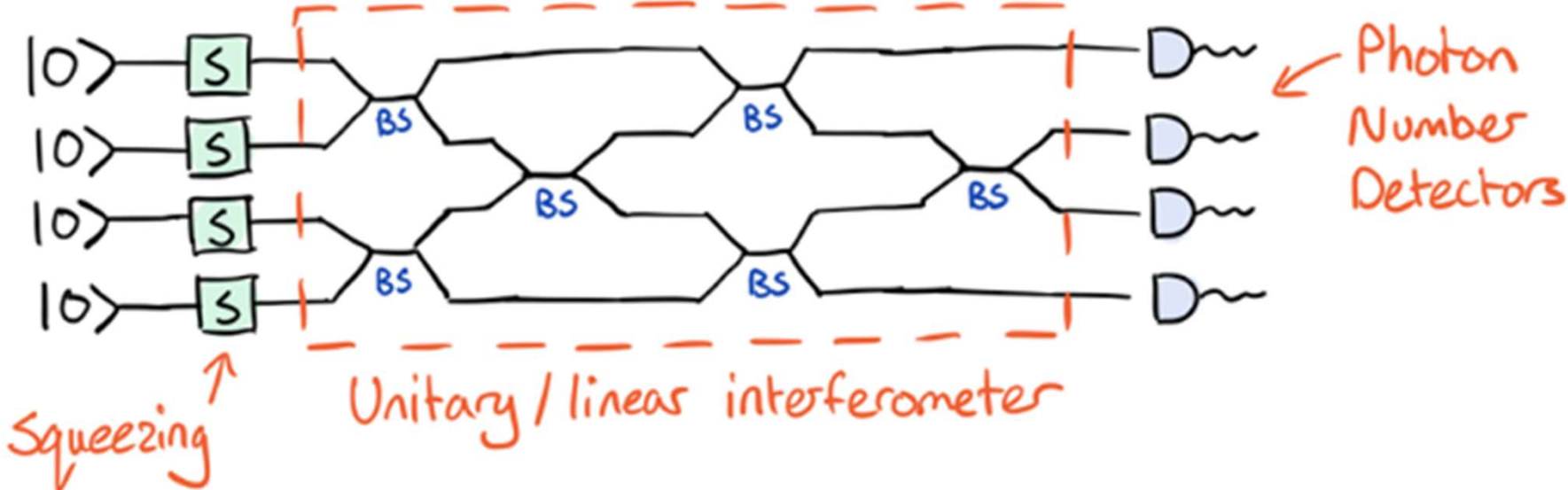
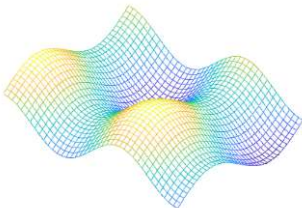


GBSとはGaussian Boson Samplingの略

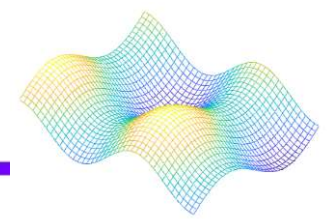
Boson … ボース統計に従う粒子  
例 光子、ヒッグス粒子など

ボソンサンプリングは入力した光子が出力される分布を予測するもの

# GBS アルゴリズムのコーディング







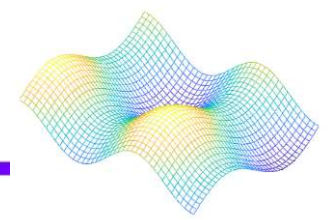
モードごとに 0 または 1 個の光子のみを含む最終状態の測定した確率

$$|\langle n_1, n_2, \dots, n_N | \psi' \rangle|^2 = \frac{|\text{Haf}[(U(\bigoplus_i \tanh(r_i)) U^T)]_{st}|^2}{\prod_{i=1}^N \cosh(r_i)}$$

ハフニアン

$$\text{Haf}(A) = \sum_{\sigma \in \text{PMP}_{2N}} \prod_{i=1}^N A_{\sigma(2i-1)\sigma(2i)},$$

$$\text{Per}(A) = \text{Haf} \left( \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \right).$$



理論式と、シミュレーションした GBS QNode からの  
確率比較

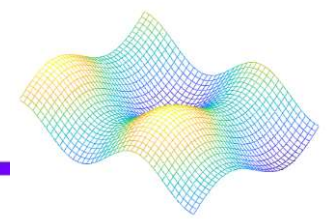
$$|\langle n_1, n_2, \dots, n_N | \psi' \rangle|^2 = \frac{|\text{Haf}[(UU^T \tanh(r_i))]_{st}|^2}{n_1! n_2! \cdots n_N! \cosh^N(r)}$$

PennyLane の結果は、GBSの方程式から予想される結果と一致します

# Photonic quantum computing



# 概要



## 直交位相振幅

$$\left\{ \begin{array}{l} \hat{x} = (\hat{a} + \hat{a}^\dagger)/\sqrt{2} \\ \hat{p} = (\hat{a} - \hat{a}^\dagger)\sqrt{2}i \end{array} \right. \left| \right. \left\{ \begin{array}{l} \hat{a} = (\hat{x} + \hat{p}i)/\sqrt{2} \\ \hat{a}^\dagger = (\hat{x} - \hat{p}i)\sqrt{2} \end{array} \right.$$

$$\begin{array}{l} [\hat{a}, \hat{a}^\dagger] = 1 \\ [\hat{x}, \hat{p}] = i \end{array}$$

## 古典電磁波

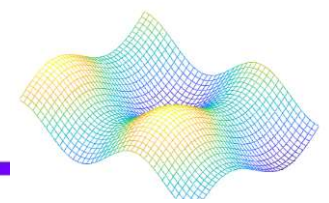
$$E(z, t) = i\varepsilon(\alpha e^{i(kz - \omega t)} - \alpha^* e^{i(kz - \omega t)})$$

量子化 ↓

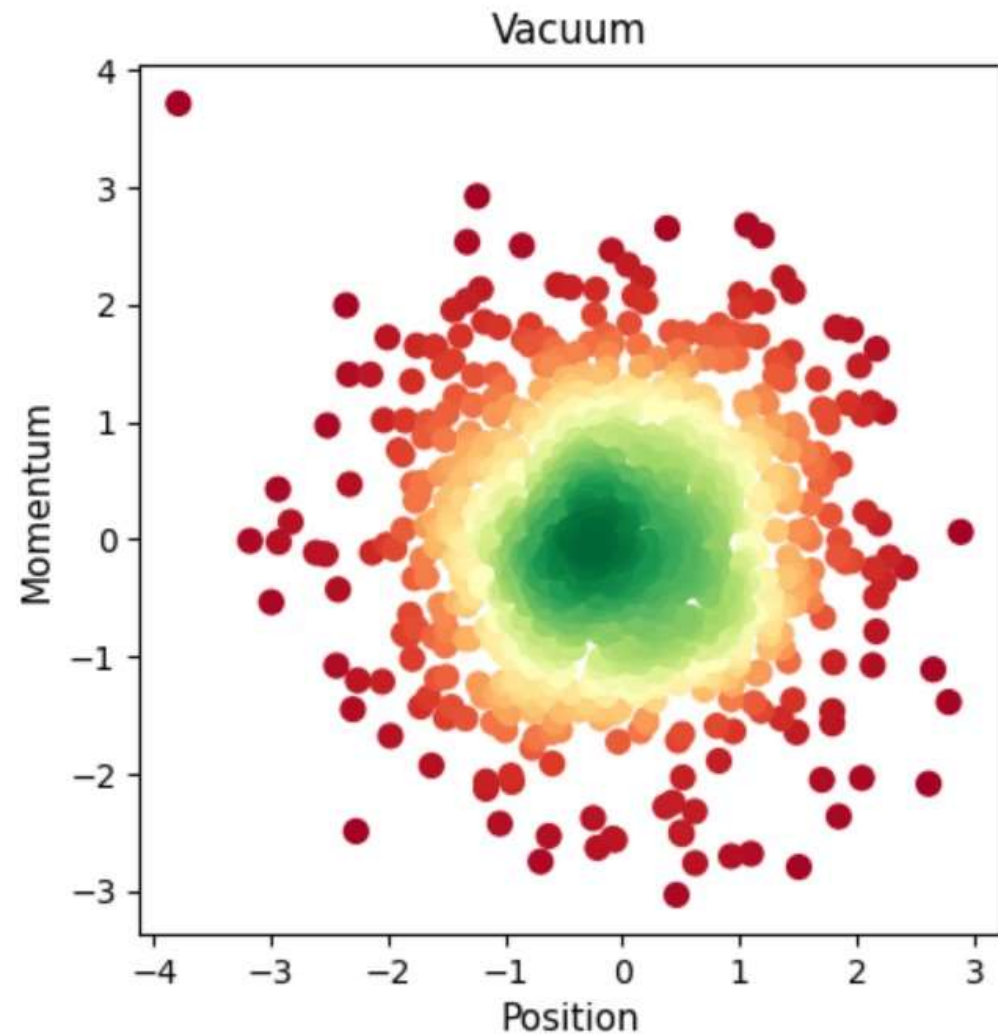
$$\begin{aligned} \hat{E}(z, t) &= i\varepsilon(\hat{a}e^{i(kz - \omega t)} - \hat{a}^\dagger e^{i(kz - \omega t)}) \\ &= -\sqrt{2}\varepsilon(\hat{x} \sin(kz - \omega t) + \hat{p} \cos(kz - \omega t)) \end{aligned}$$

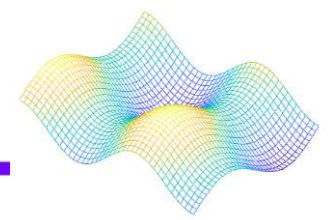
→  $\hat{x}, \hat{p}$ はそれぞれsin成分とcos成分の振幅

# 真空場



```
import pennylane as qml
from pennylane import numpy as np
import matplotlib.pyplot as plt
dev = qml.device("default.gaussian", wires=1, shots=1000)
@qml.qnode(dev)
def vacuum_measure_x():
    return qml.sample(qml.X(0)) # Samples X quadratures
@qml.qnode(dev)
def vacuum_measure_p():
    return qml.sample(qml.P(0)) # Samples P quadrature
# Sample measurements in phase space
x_sample = vacuum_measure_x()
p_sample = vacuum_measure_p()
# Import some libraries for a nicer plot
from scipy.stats import gaussian_kde
from numpy import vstack as vstack
# Point density calculation
xp = vstack([x_sample, p_sample])
z = gaussian_kde(xp)(xp)
# Sort the points by density
sorted = z.argsort()
x, y, z = x_sample[sorted], p_sample[sorted], z[sorted]
# Plot
fig, ax = plt.subplots()
ax.scatter(x, y, c = z, s = 50, cmap="RdYlGn")
plt.title("Vacuum", fontsize=12)
ax.set_ylabel("Momentum", fontsize = 11)
ax.set_xlabel("Position", fontsize = 11)
ax.set_aspect("equal", adjustable = "box")
plt.show()
```



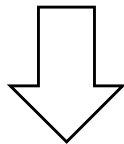


コヒーレント

可干渉性をもつこと。また、最小不確定状態を持つ。

$$[\hat{a}, \hat{a}^\dagger] = 1$$

$$\begin{cases} \hat{a} \equiv \hat{x}_1 + i\hat{x}_2 \\ \hat{a}^\dagger \equiv \hat{x}_1 - i\hat{x}_2 \end{cases}$$



$$[\hat{x}_1, \hat{x}_2] = i/2 \text{ より、}$$

$$\Delta x_1 \Delta x_2 \geq 1/4$$

コヒーレント状態では

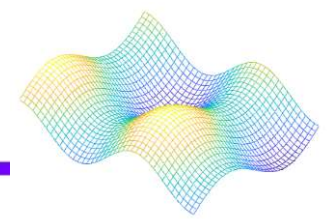
$$\Delta x_1 = \Delta x_2 = 1/2$$

なので、

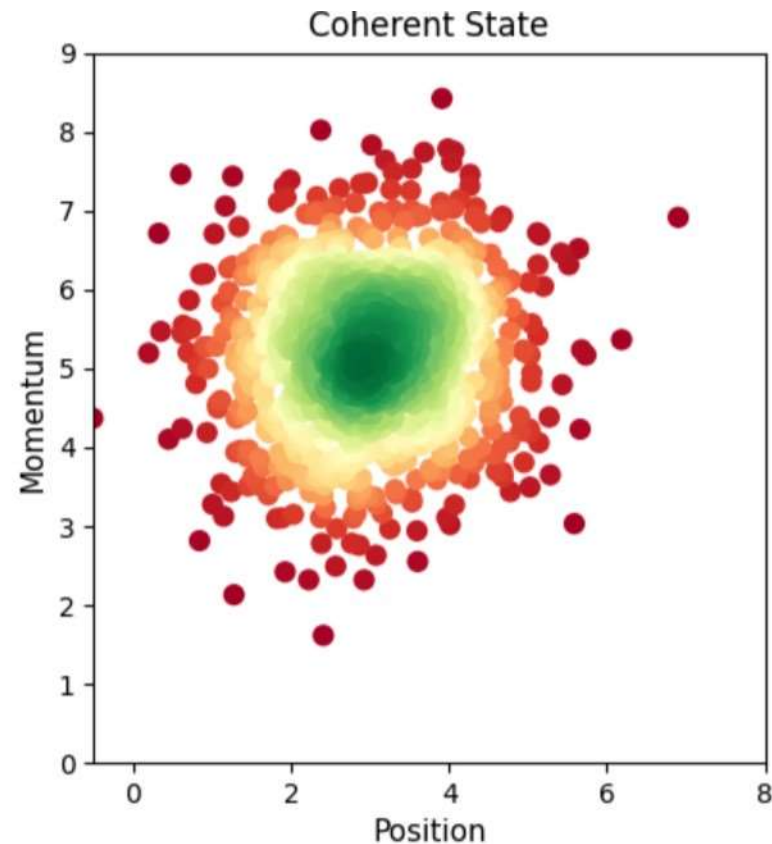
$$\Delta x_1 \Delta x_2 = 1/4$$

コヒーレント状態では、  
2つの直交位相振幅の間の  
最小不確定性となる。

# コヒーレント状態



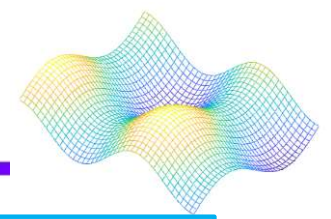
```
@qml.qnode(dev)
def measure_coherent_x(alpha, phi):
    qml.CoherentState(alpha, phi, wires=0) # Prepares coherent state
    return qml.sample(qml.X(0)) # Measures X quadrature
@qml.qnode(dev)
def measure_coherent_p(alpha, phi):
    qml.CoherentState(alpha, phi, wires=0) # Prepares coherent state
    return qml.sample(qml.P(0)) # Measures P quadrature
# Choose alpha and phi and sample 1000 measurements
x_sample_coherent = measure_coherent_x(3, np.pi / 3)
p_sample_coherent = measure_coherent_p(3, np.pi / 3)
# Plot as before
xp = vstack([x_sample_coherent, p_sample_coherent])
z1 = gaussian_kde(xp)(xp)
sorted = z1.argsort()
x, y, z = x_sample_coherent[sorted], p_sample_coherent[sorted], z1[sorted]
fig, ax1 = plt.subplots()
ax1.scatter(x, y, c = z, s = 50, cmap = "RdYlGn")
ax1.set_title("Coherent State", fontsize = 12)
ax1.set_ylabel("Momentum", fontsize = 11)
ax1.set_xlabel("Position", fontsize = 11)
ax1.set_aspect("equal", adjustable = "box")
plt.xlim([-0.5, 8])
plt.ylim([0, 9])
plt.show()
```



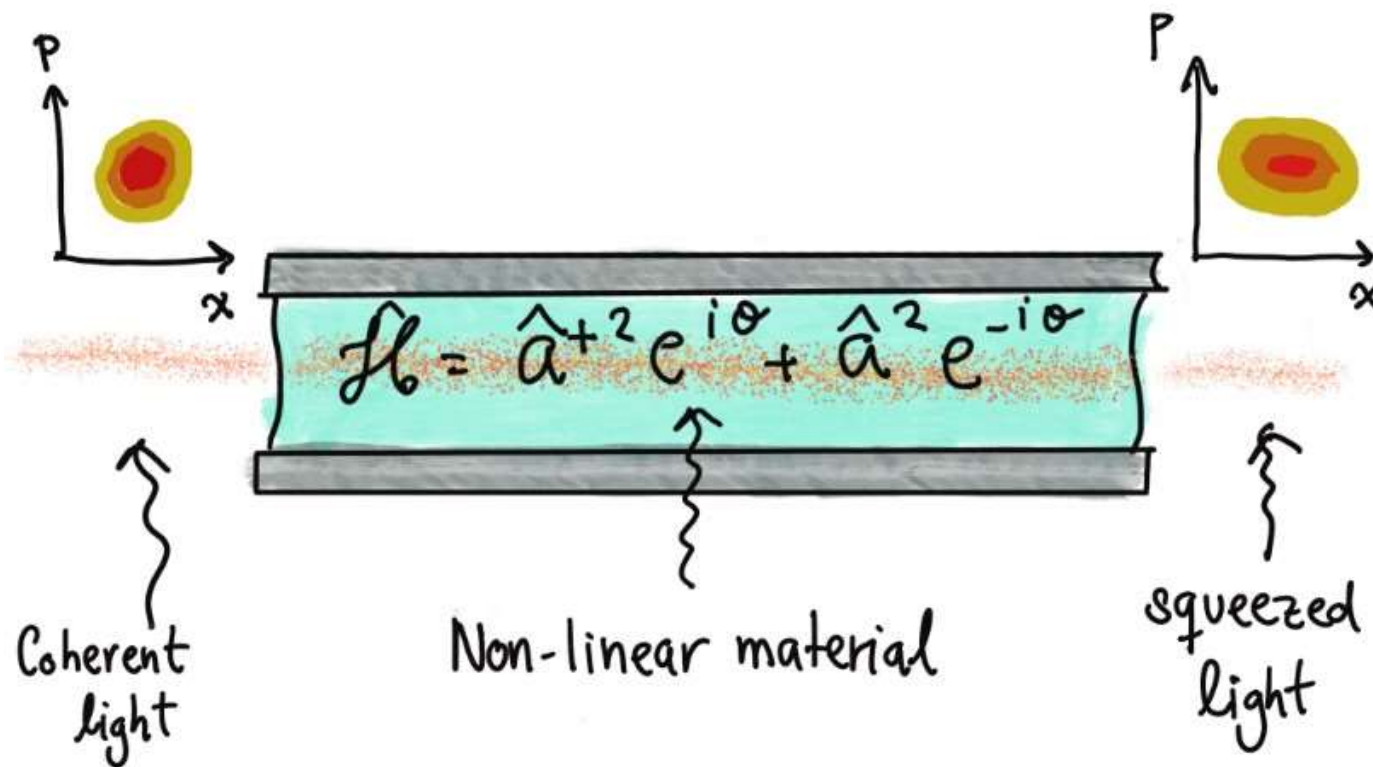
$$\alpha = \sqrt{x^2 + p^2}$$
$$\phi = \tan^{-1}(p/x)$$



# スクイズド状態



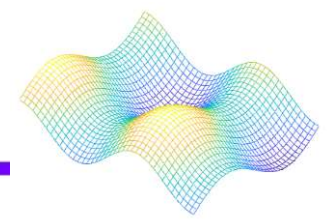
2つの物理量のうち一方の揺らぎを犠牲にして、他方の揺らぎを抑えることをスクイズという。最小不確定状態を維持しながらスクイズされた状態をスクイズド状態という。



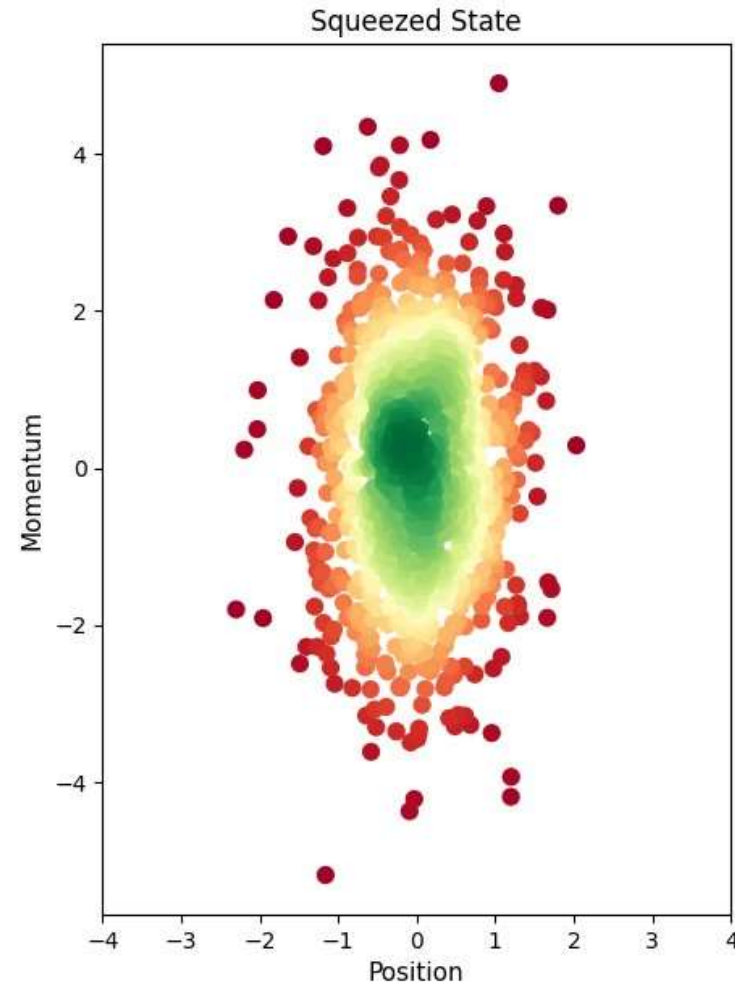
$$\begin{aligned}\Delta x_1 \Delta x_2 &= 1/4 \\ \Delta x_1 &= (1/2)e^{-r} \\ \Delta x_2 &= (1/2)e^r\end{aligned}$$

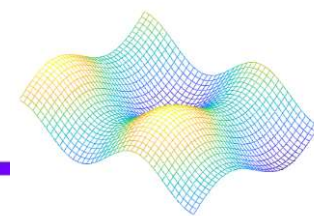


# スクイズド状態

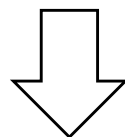


```
import pennylane as qml
from pennylane import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde
from numpy import vstack as vstack
dev = qml.device("default.gaussian", wires=1, shots=1000)
@qml.qnode(dev)
def measure_squeezed_x(r):
    qml.Squeezing(r, 0, wires = 0)
    return qml.sample(qml.X(0))
@qml.qnode(dev)
def measure_squeezed_p(r):
    qml.Squeezing(r, 0, wires = 0)
    return qml.sample(qml.P(0))
# Choose alpha and phi and sample 1000 measurements
x_sample_squeezed = measure_squeezed_x(0.4)
p_sample_squeezed = measure_squeezed_p(0.4)
# Plot as before
xp = vstack([x_sample_squeezed, p_sample_squeezed])
z = gaussian_kde(xp)(xp)
sorted_meas = z.argsort()
x, y, z = x_sample_squeezed[sorted_meas], p_sample_squeezed[sorted_meas],
z[sorted_meas]
fig, ax1 = plt.subplots(figsize=(7, 7))
ax1.scatter(x, y, c = z, s = 50, cmap = "RdYlGn")
ax1.set_title("Squeezed State", fontsize = 12)
ax1.set_ylabel("Momentum", fontsize = 11)
ax1.set_xlabel("Position", fontsize = 11)
ax1.set_xlim([-4, 4])
ax1.set_aspect("equal", adjustable = "box")
plt.show()
```

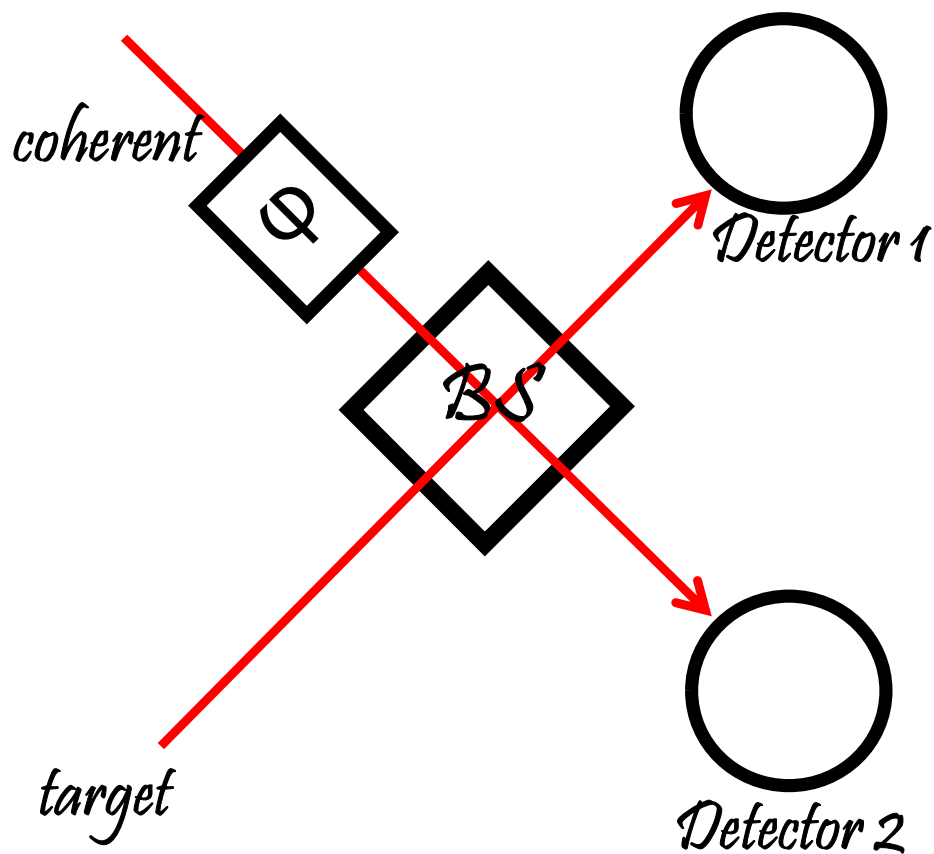




光の位相測定をしたい



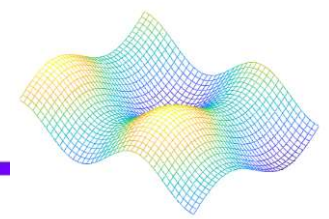
干渉測定



$$\hat{n}_2 - \hat{n}_1 = \sqrt{2}|\alpha|(\hat{x} \cos \phi + \hat{p} \sin \phi)$$

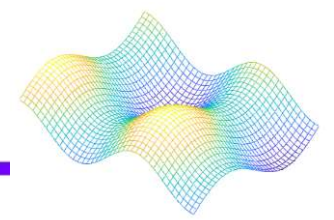
$$\begin{cases} \phi = 0 \Rightarrow \text{Measure } \hat{x} \\ \phi = \pi/2 \Rightarrow \text{Measure } \hat{p} \end{cases}$$

# ホモダイン測定



```
import pennylane as qml
from pennylane import numpy as np
import matplotlib.pyplot as plt
dev_exact2 = qml.device("default.gaussian", wires = 2)
@qml.qnode(dev_exact2)
def measurement(a, phi):
    qml.Displacement(a, phi, wires = 0) # Implement displacement using PennyLane
    return qml.expval(qml.X(0))
@qml.qnode(dev_exact2)
def measurement2_0(a, theta, alpha, phi):
    qml.Displacement(a, theta, wires = 0) # We choose the initial to be a displaced vacuum
    qml.CoherentState(alpha, phi, wires = 1) # Prepare coherent as second qumode
    qml.Beamsplitter(np.pi / 4, 0, wires=[0, 1]) # Interfere both states
    return qml.expval(qml.NumberOperator(0)) # Read out N
@qml.qnode(dev_exact2)
def measurement2_1(a, theta, alpha, phi):
    qml.Displacement(a, theta, wires = 0) # We choose the initial to be a displaced vacuum
    qml.CoherentState(alpha, phi, wires = 1) # Prepare coherent as second qumode
    qml.Beamsplitter(np.pi / 4, 0, wires=[0, 1]) # Interfere both states
    return qml.expval(qml.NumberOperator(1)) # Read out N
print(
    "Expectation value of x-quadrature after displacement: {}".format(measurement(3, 0))
)
print("Expected current in each detector:")
print("Detector 1: {}".format(measurement2_0(3, 0, 1, 0)))
print("Detector 2: {}".format(measurement2_1(3, 0, 1, 0)))
print(
    "Difference between currents: {}".format(
        measurement2_1(3, 0, 1, 0) - measurement2_0(3, 0, 1, 0)
    )
)
```

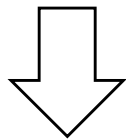
# ホモダイン測定



$[\hat{x}, \hat{p}] = i\hbar$   $\longrightarrow$  位置と運動量が同時確定不可能

$$\phi = 0$$

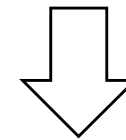
```
Expected current in each detector:  
Detector 1: 2.0000000000000001  
Detector 2: 8.0000000000000002  
Difference between currents: 6.0000000000000001  
C:\Users\takr-\sample1\.venv\Scripts\python.exe:  
on 'sample3' while trying to find 'sample3.py')
```



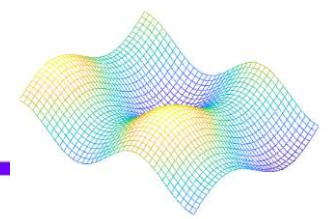
波動性

$$\phi = \pi/2$$

```
Expectation value of x-quadrature after displacement: 6.0  
Expected current in each detector:  
Detector 1: 5.0000000000000001  
Detector 2: 5.0000000000000001  
Difference between currents: 0.0  
C:\Users\takr-\sample1\.venv\Scripts\python.exe: Error wh  
on 'sample4' while trying to find 'sample4.py'). Try usin
```



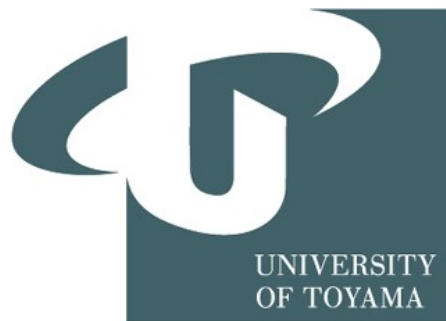
粒子性

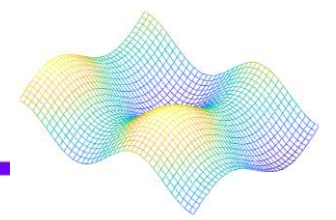


光量子コンピューティング技術応用のための光量子状態等を記述した。

1. 光量子の真空状態、コヒーレント状態、スクイーズド状態について記述した。
2. 直角位相測定について記述した。

# トラップイオン量子コンピュータ



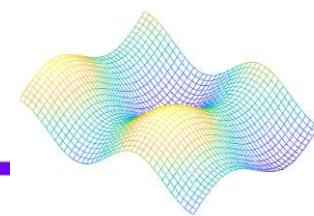


有用な量子コンピュータの一般的なアプローチ方法として、

- ・トラップされたイオン
  - ・超伝導量子ビット
  - ・フォトニクス
- 等

が挙げられる。

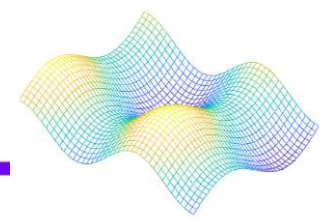
ここでは、トラップされたイオン量子について紹介する。



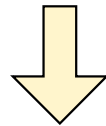
1. イオンをトラップする方法
2. 強固な量子ビットとしてトラップされたイオン
3. 非解体測定
4. 単一量子ビットを操作するためのラビ振動
5. 高調波発信器としてのイオン鎖
6. 多量子ビットゲートによるイオンの絡み合い
7. イオンが多すぎる場合の問題
8. 最先端の技術
9. 結び



# 1. イオンをトラップする方法



イオンを量子ビットとして使用する理由



電界でただ一つの場所に閉じ込めることが可能。

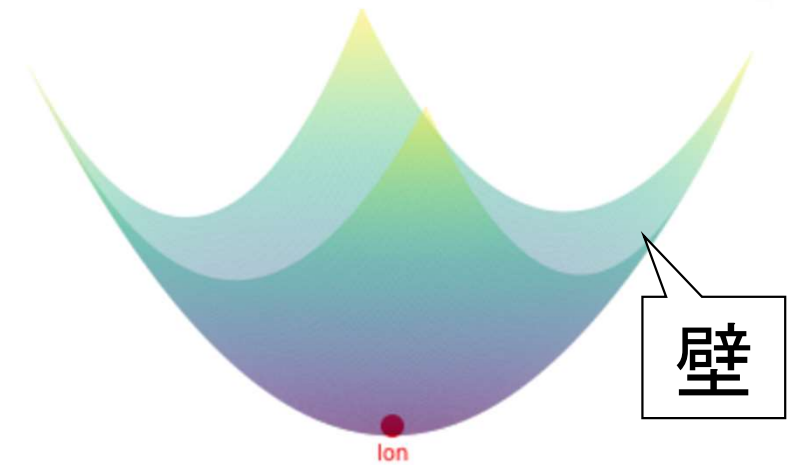


図1 Confining potential

図1は理想的な電場(ポテンシャル)の構成を表す。

しかし、静電場だけで閉じ込めることは不可能。

実際には、図2の鞍型のポテンシャルが生じる。

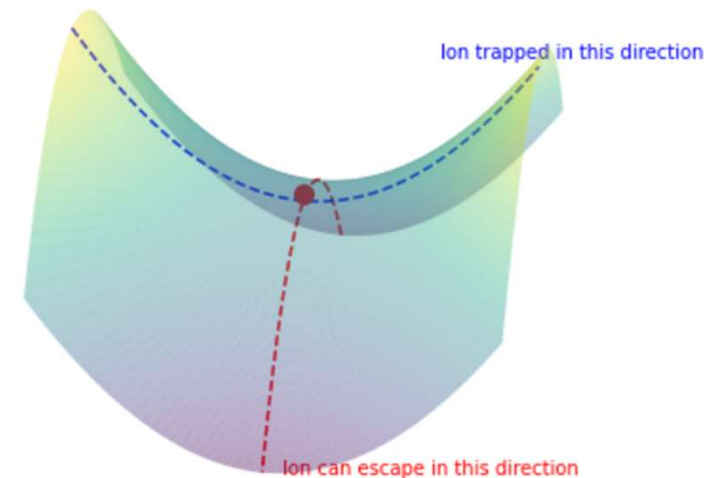
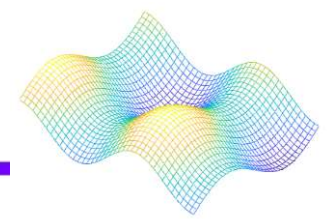


図2 Saddle-shaped potential allowed by electrostatics

# 1. イオンをトラップする方法



イオンは一方向に閉じ込められるが、  
垂直方向に逃げる可能性がある。

〈解決策〉

時間依存の電界を用いて  
ポテンシャルの壁を移動させる。

例えば、プロットしたポテンシャルを  
回転させたらどうなるか。

→図3

このとき発生する電位は、(1.1)で与えられる。

$$\Phi = \frac{1}{2}(u_x x^2 + u_y y^2 + u_z z^2) + \frac{1}{2}(v_x x^2 + v_y y^2 + v_z z^2) \cos(\omega t + \phi).$$

(1.1)

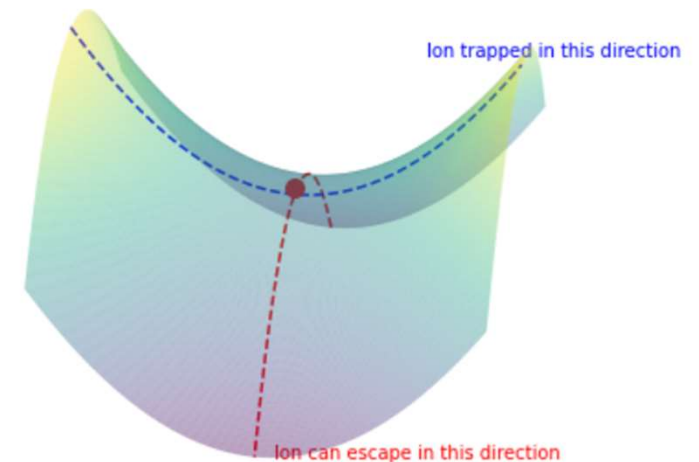


図2 Saddle-shaped potential allowed by electrostatics

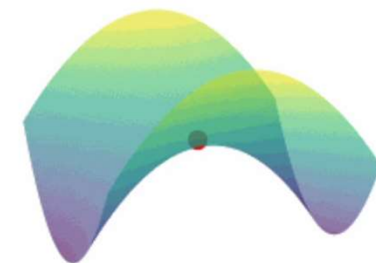
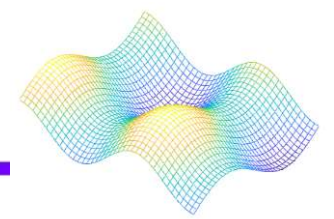


図3 A rotating potential with the correct frequency and magnitude can contain an ion

# 1. イオンをトラップする方法



$$\Phi = \frac{1}{2} (u_x x^2 + u_y y^2 + u_z z^2) + \frac{1}{2} (v_x x^2 + v_y y^2 + v_z z^2) \cos(\omega t + \phi). \quad (1.1)$$

パラメータの調整が必要。

光子を吸収するイオンのシステムが必要。

イオン間で相対運動を引き起こす。  
→イオンを一次元配列に置き、そして冷却。

イオンを励起状態に  
させる相対運動は起きない。  
代わりに反発が発生。  
(メスバウアー効果)

この制御は二量子ビットで  
量子操作を正確に表現するのに必要なこと。

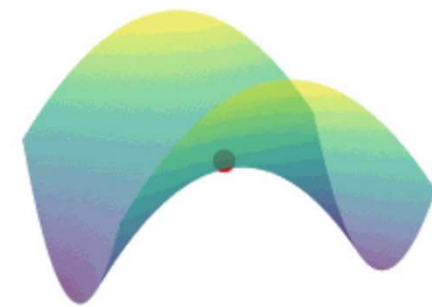


図3 A rotating potential with the correct frequency and magnitude can contain an ion

## 2. 強固な量子ビットとしてトラップされたイオン

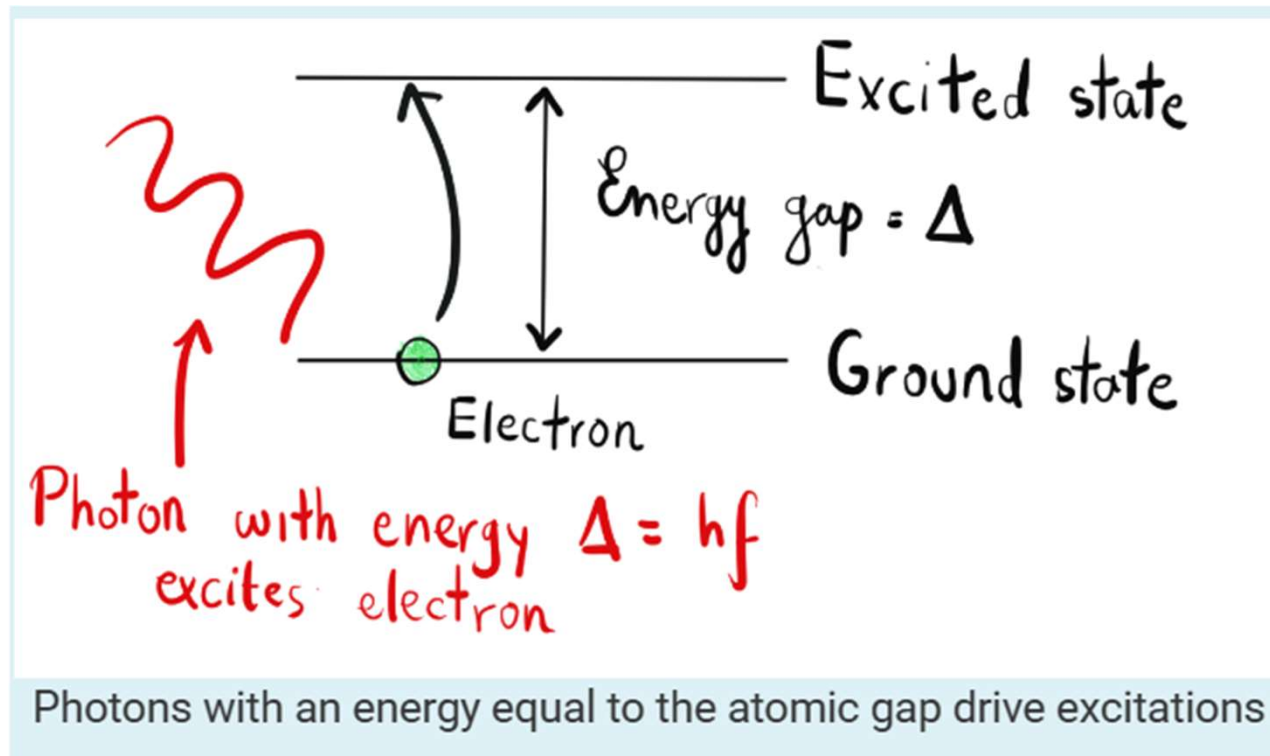
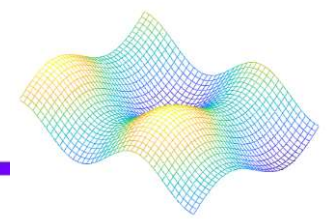


図4

### <原子物理入門>

- ・電子はエネルギー準位に存在。
- ・(電子が失ったエネルギー)=(放出される光子のエネルギー)
- ・このエネルギーは光子の周波数に比例。
- ・光子が電子に当たると、その電子はより高いエネルギー状態に移行する。

## 2. 強固な量子ビットとしてトラップされたイオン

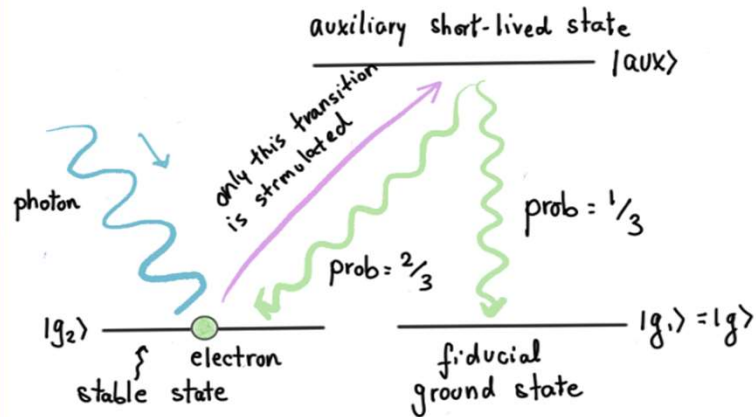
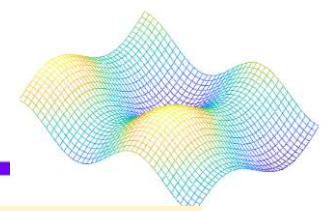


図5 Optical pumping to prepare the ground state

すべての(あるいは大半の)イオンの電子を徐々に基底状態に「ポンピング」していく必要がある。

光ポンピングに似た手順で2つの状態を遷移させることができる。

→コヒーレンス時間は長くなるが、ゲートの実装はより複雑で多くの精度を必要とする。

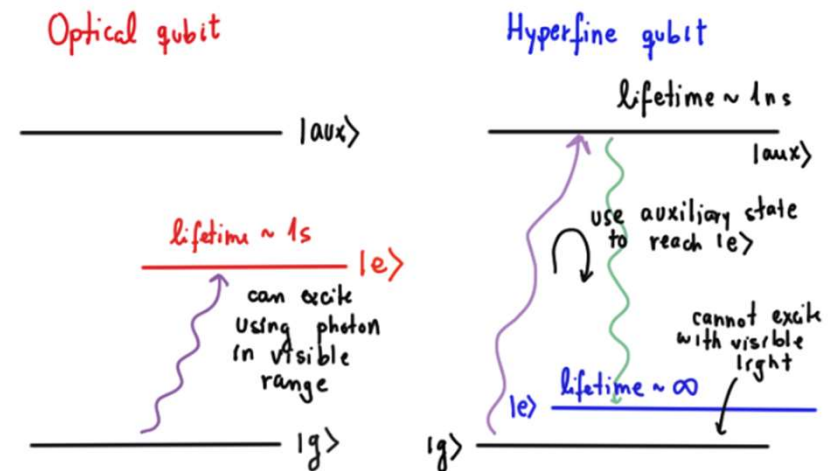
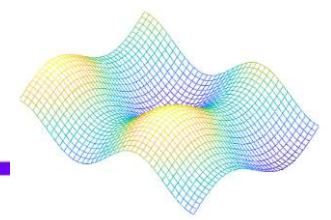


図6 Optical vs. hyperfine qubits

# 3. 非解体測定



<量子ビットの測定について>

測定方法は、  
光ポンピングと同様の原理を用いる。

イオンは照射したレーザー光と  
同波長で連続的に発光する。

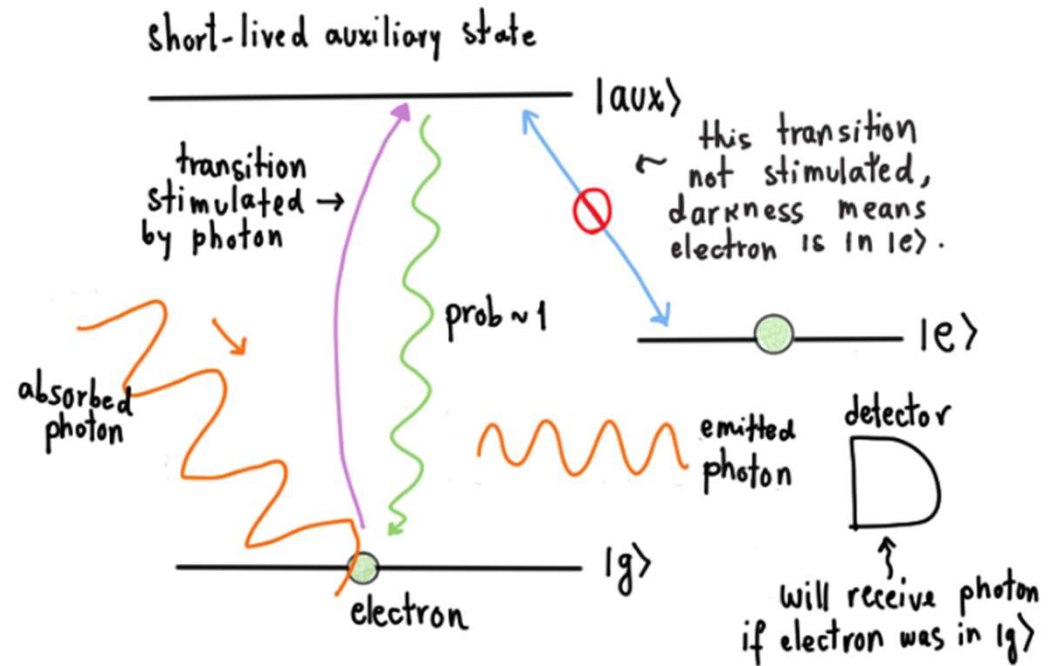
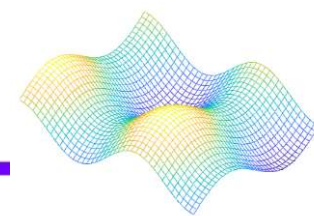


図7 Non-demolition measurement of ion states

## 4. 単一量子ビットを操作するためのラビ振動



レーザー光に共鳴するイオンの電子を記述するハミルトニアンは、次のような演算子で与えられる。

$$\hat{H} = \frac{\hbar\Omega}{2} (S_+ e^{i\varphi} + S_- e^{-i\varphi}) \quad (4.1)$$

ここで $\Omega$ はラビ周波数。行列 $S$ は

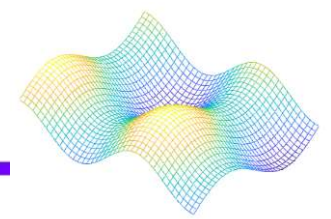
$$S_+ = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad S_- = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad (4.2)$$

ハミルトニアンが時間に依存しない場合、状態 $|g\rangle$ で始まる量子ビットは次のような時間依存の状態に進化。

$$|\psi(t)\rangle = \exp(-i\hat{H}t/\hbar) |g\rangle \quad (4.3)$$



# 4. 単一量子ビットを操作するためのラビ振動



(4.3)

$$|\psi(t)\rangle = \exp(-i\hat{H}t/\hbar) |g\rangle$$

```
import pennylane as qml
from pennylane import numpy as np
from scipy.linalg import expm

Omega = 100
S_plus = np.array([[0, 0], [1, 0]])
S_minus = np.array([[0, 1], [0, 0]])
```

```
def evolution(phi, t):
    Ham = Omega / 2 * (S_plus * np.exp(1j * phi) + S_minus * np.exp(-1j * phi))
    return expm(-1j * Ham * t)
```

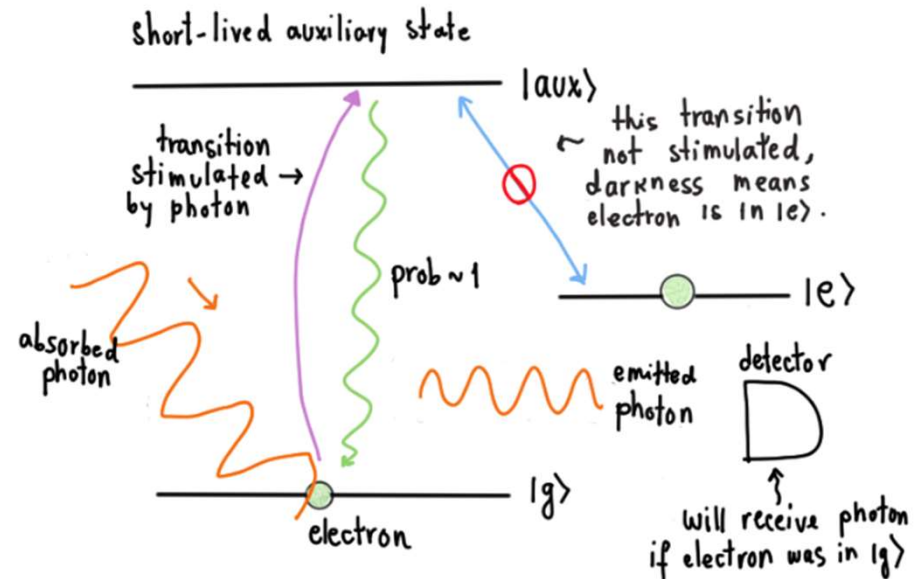


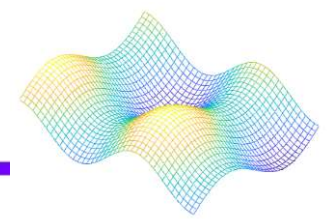
図7 Non-demolition measurement of ion states

図8 code1

基底状態 $|g\rangle$ と $|e\rangle$ はこのハミルトニアンを受ける。  
この演算子を実装すれば、  
一般的なゲートを生成するパルスのシーケンスを決定できる。



# 4. 単一量子ビットを操作するためのラビ振動



```
dev = qml.device("default.qubit", wires=1)
```

```
@qml.qnode(dev, interface="autograd")
def ion_hadamard(state):
```

```
    if state == 1:
        qml.PauliX(wires=0)
```

```
    """We use a series of seemingly arbitrary pulses that will give the Hadamard gate.
    Why this is the case will become clear later"""
```

```
    qml.QubitUnitary(evolution(0, -np.pi / 2 / Omega), wires=0)
    qml.QubitUnitary(evolution(np.pi / 2, np.pi / 2 / Omega), wires=0)
    qml.QubitUnitary(evolution(0, np.pi / 2 / Omega), wires=0)
    qml.QubitUnitary(evolution(np.pi / 2, np.pi / 2 / Omega), wires=0)
    qml.QubitUnitary(evolution(0, np.pi / 2 / Omega), wires=0)
```

```
    return qml.state()
```

```
#For comparison, we use the Hadamard built into PennyLane
```

```
@qml.qnode(dev, interface="autograd")
def hadamard(state):
```

```
    if state == 1:
        qml.PauliX(wires=0)
```

```
    qml.Hadamard(wires=0)
```

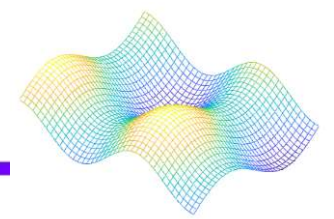
```
    return qml.state()
```

```
#We confirm that the values given by both functions are the same up to numerical error
print(np.isclose(1j * ion_hadamard(0), hadamard(0)))
print(np.isclose(1j * ion_hadamard(1), hadamard(1)))
```

(ex) アダマール・ゲートを生成する位相と  
持続時間の異なるパルスの組み合わせ

図9 code2

# 4. 単一量子ビットを操作するためのラビ振動



```
@qml.qnode(dev, interface="autograd")
def ion_Tgate(state):

    if state == 1:
        qml.PauliX(wires=0)

    qml.QubitUnitary(evolution(0, -np.pi / 2 / Omega), wires=0)
    qml.QubitUnitary(evolution(np.pi / 2, np.pi / 4 / Omega), wires=0)
    qml.QubitUnitary(evolution(0, np.pi / 2 / Omega), wires=0)

    return qml.state()

@qml.qnode(dev, interface="autograd")
def tgate(state):

    if state == 1:
        qml.PauliX(wires=0)

    qml.T(wires=0)

    return qml.state()

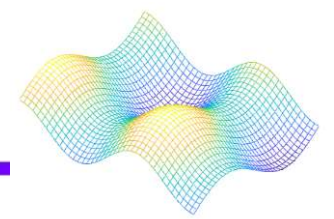
print(np.isclose(np.exp(1j * np.pi / 8) * ion_Tgate(0), tgate(0)))
print(np.isclose(np.exp(1j * np.pi / 8) * ion_Tgate(1), tgate(1)))
```

同様の演習をTゲートでもできる。

このPennyLaneコードは、  
アダマールゲートとTゲートが  
得られることを示している。

図10 code3

# 4. 単一量子ビットを操作するためのラビ振動



```
import matplotlib.pyplot as plt

@qml.qnode(dev, interface="autograd")
def evolution_prob(t):

    qml.QubitUnitary(evolution(0, t / Omega), wires=0)

    return qml.probs(wires=0)

t = np.linspace(0, 4 * np.pi, 101)
s = [evolution_prob(i)[1].numpy() for i in t]

fig1, ax1 = plt.subplots(figsize=(9, 6))

ax1.plot(t, s, color="#9D2EC5")

ax1.set(
    xlabel="time (in units of 1/Ω)",
    ylabel="Probability",
    title="Probability of measuring the excited state"
)

ax1.grid()

plt.show()
```

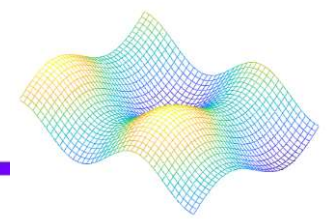
アダマールとTゲートを併用することで、任意の近似度で単一量子ビットのあらゆる演算を実行することができる。

$\phi=0$ の条件で、状態 $|e\rangle$ を得る確率をパルスの持続時間に対してプロットする。

このパターンは繰り返され、ラビ振動として知られている。

図11 code4

## 4. 単一量子ビットを操作するためのラビ振動



実際、シュレーディンガー方程式を陽解法で解くことができる。

基底状態 $|g\rangle$ は

$$|\psi_0(t)\rangle = \cos\left(\frac{\Omega t}{2}\right) |g\rangle - i \sin\left(\frac{\Omega t}{2}\right) e^{i\varphi} |e\rangle \quad (4.4)$$

進化は次式で与えられる。

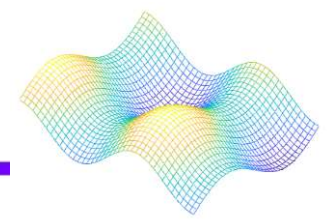
$$|\psi_1(t)\rangle = -i \sin\left(\frac{\Omega t}{2}\right) e^{-i\varphi} |g\rangle + \cos\left(\frac{\Omega t}{2}\right) |e\rangle \quad (4.5)$$

以下の式は、一般的な回転の形をしている。

$$U(\Omega, \varphi, t) = \begin{pmatrix} \cos\left(\frac{\Omega t}{2}\right) & -i \sin\left(\frac{\Omega t}{2}\right) e^{-i\varphi} \\ -i \sin\left(\frac{\Omega t}{2}\right) e^{i\varphi} & \cos\left(\frac{\Omega t}{2}\right) \end{pmatrix} \quad (4.6)$$

ラビ振動は、単一量子ビットゲートの普遍的なセットを構築することを可能にする。

## 5. 高調波発信器としてのイオン鎖



冷却すると、イオン鎖全体が量子調和振動子として働き、  
以下のエネルギーで振動することができる。

$$E = n\hbar\omega \quad (5.1)$$

基底状態であれば、エネルギーギャップを吸収し、イオン鎖は残りを吸収。  
したがって、この光の周波数は次のような青いサイドバンド遷移を引き起こす。

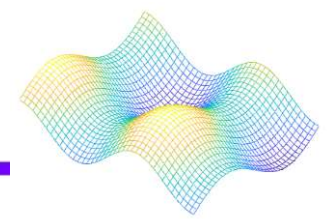
$$|g\rangle |n\rangle \rightarrow |e\rangle |n+1\rangle \quad (5.2)$$

イオンが励起され、イオン鎖が非励起されるとき、  
これはレッド・サイドバンド遷移としても知られている。

$$|g\rangle |n\rangle \rightarrow |e\rangle |n-1\rangle \quad (5.3)$$

イオン鎖がゼロエネルギーのとき、周波数は何もしない。  
もし光の周波数が正確にエネルギーギャップのとき、  
イオン鎖はフォノンを吸収しないが、イオンは励起される。  
これをキャリア遷移と呼ぶ。

# 5. 高調波発信器としてのイオン鎖



イオン鎖の振動は量子状態。

運動状態と電子基底状態にある2つのイオンの系は、次のように進化する。

$$|g\rangle |g\rangle |n\rangle \rightarrow \frac{1}{\sqrt{2}} (|g\rangle |g\rangle |n\rangle + |e\rangle |g\rangle |n+1\rangle) \quad (5.4)$$

CNOTゲートと

1量子ビットゲートを組み合わせると  
普遍的な計算が可能になる。

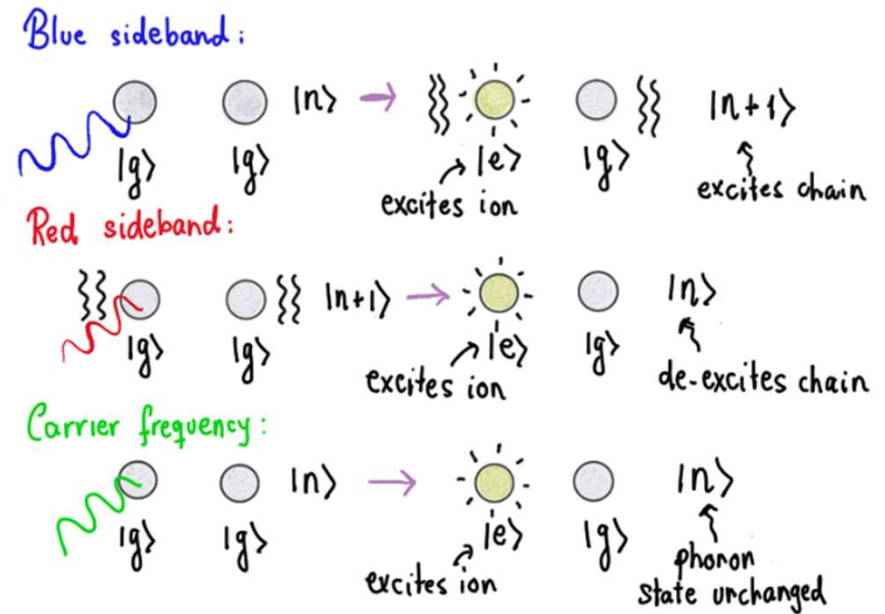
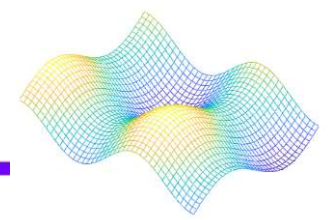


図12 Effects of the sideband and carrier frequencies on an ion chain



## 6. 多量子ビットゲートによるイオンの絡み合い



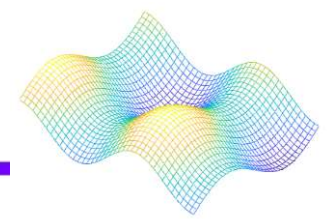
もつれた状態においてすべての結果が同じ確率を持つ場合、

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|e\rangle |g\rangle + |g\rangle |e\rangle) \quad (6.1)$$

運動エネルギーがゼロの鎖に対して、  
持続時間と位相を最初のイオンに適用すると、

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|g\rangle |g\rangle |0\rangle + |e\rangle |g\rangle |1\rangle) \quad (6.2)$$

# 6. 多量子ビットゲートによるイオンの絡み合い



The **Cirac-Zoller gate** の実装手順を図13に示す。

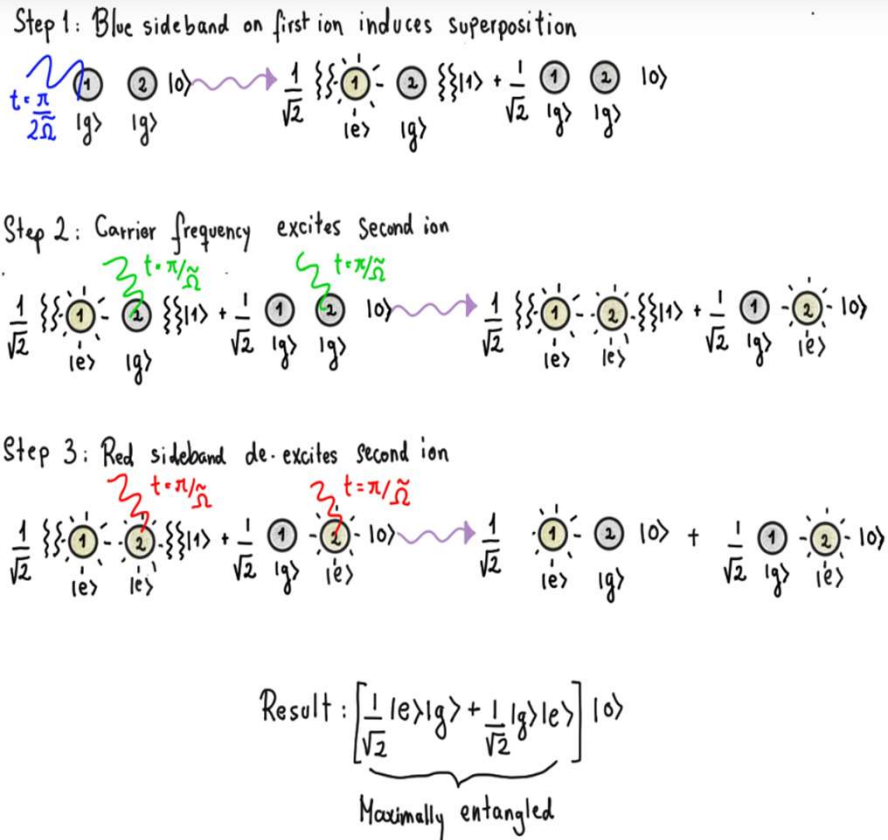


図13 Implementation of the Cirac-Zoller gate using phonon states

実際の応用には、  
Mølmer-Sørensen (MS) gate  
として知られる。

$$\omega_{\pm} = \Delta \pm \delta \quad (6.3)$$

レーザー光との相互作用の正味の効果は、  
 $|g\rangle |g\rangle |n\rangle \rightarrow |e\rangle |e\rangle |n\rangle$ へと励起することになり、  
右図の4つの経路のいずれかを通ることが可能。

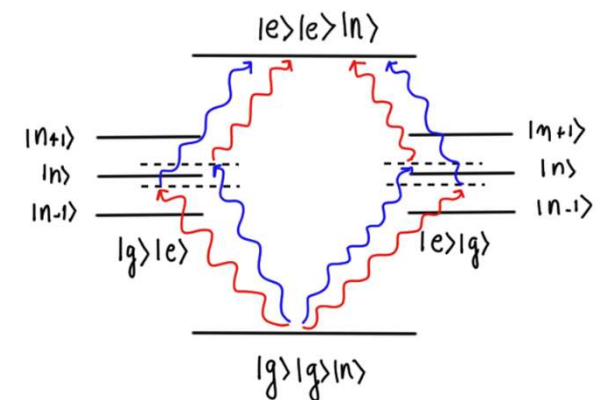
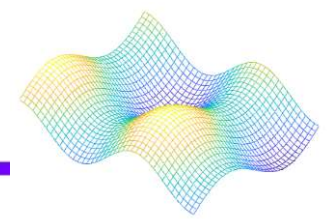


図14

Mølmer-Sørensen gate implemented with two simultaneous laser pulses



## 6. 多量子ビットゲートによるイオンの絡み合い



摂動論として知られる量子力学的手法を用いると、  
次のようなラビ周波数も存在する推測ができる。

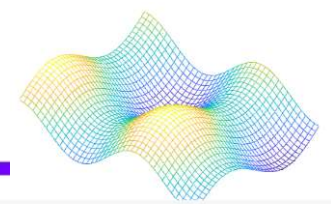
$$U_{MS}(t) = \begin{pmatrix} \cos\left(\frac{\Omega_{Mst}}{2}\right) & 0 & 0 & -i \sin\left(\frac{\Omega_{Mst}}{2}\right) \\ 0 & \cos\left(\frac{\Omega_{Mst}}{2}\right) & -i \sin\left(\frac{\Omega_{Mst}}{2}\right) & 0 \\ 0 & -i \sin\left(\frac{\Omega_{Mst}}{2}\right) & \cos\left(\frac{\Omega_{Mst}}{2}\right) & 0 \\ -i \sin\left(\frac{\Omega_{Mst}}{2}\right) & 0 & 0 & \cos\left(\frac{\Omega_{Mst}}{2}\right) \end{pmatrix} \quad (6.4)$$

レーザーの時間と位相を調整することで、重ね合わせが可能。  
シュレーディンガー方程式を用いて、Mølmer-Sørensenプロトコルをある時間に  
適用したときに、量子ビットがどのように変化するか、Python関数で実装する。

```
Omega = 100

def Molmer_Sorensen(t):
    ms = np.array(
        [
            [np.cos(Omega * t / 2), 0, 0, -1j * np.sin(Omega * t / 2)],
            [0, np.cos(Omega * t / 2), -1j * np.sin(Omega * t / 2), 0],
            [0, -1j * np.sin(Omega * t / 2), np.cos(Omega * t / 2), 0],
            [-1j * np.sin(Omega * t / 2), 0, 0, np.cos(Omega * t / 2)],
        ]
    )
    return ms
```

# 6. 多量子ビットゲートによるイオンの絡み合い



CNOTゲートは  
量子アルゴリズムでよく使われるので、  
Mølmer-Sørensenゲートから  
CNOTゲートを得る方法を考える。

単一量子ビットの回転とMølmer-Sørensen  
ゲートの組み合わせを使用することで、  
CNOTゲートを得ることができる。  
以下の回路を用いて行う。

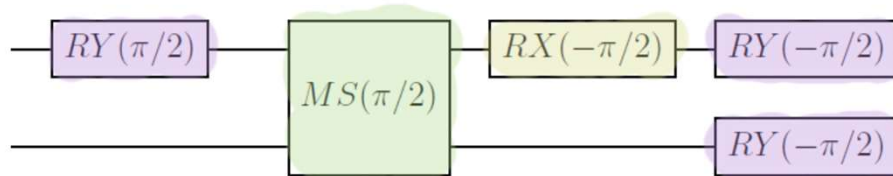


図17 Circuit for the CNOT gate using rotations and an MS gate

PennyLaneで回路を作る。

これはグローバル位相までの  
CNOTゲートを指す。

図16 code6

```
dev2 = qml.device("default.qubit",wires=2)

@qml.qnode(dev2, interface="autograd")
def ion_cnot(basis_state):

    #Prepare the two-qubit basis states from the input
    qml.templates.BasisStatePreparation(basis_state, wires=range(2))

    #Implements the circuit shown above
    qml.RY(np.pi/2, wires=0)
    qml.QubitUnitary(Molmer_Sorensen(np.pi/2/Omega),wires=[0,1])
    qml.RX(-np.pi/2, wires=0)
    qml.RX(-np.pi/2, wires=1)
    qml.RY(-np.pi/2, wires=0)

    return qml.state()

#Compare with built-in CNOT
@qml.qnode(dev2, interface="autograd")
def cnot_gate(basis_state):

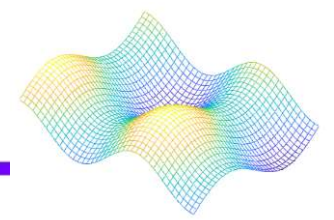
    qml.templates.BasisStatePreparation(basis_state, wires=range(2))

    qml.CNOT(wires=[0,1])

    return qml.state()

#Check that they are the same up to numerical error and global phase
print(np.isclose(np.exp(-1j*np.pi/4)*ion_cnot([0,0]),cnot_gate([0,0])))
print(np.isclose(np.exp(-1j*np.pi/4)*ion_cnot([0,1]),cnot_gate([0,1])))
print(np.isclose(np.exp(-1j*np.pi/4)*ion_cnot([1,0]),cnot_gate([1,0])))
print(np.isclose(np.exp(-1j*np.pi/4)*ion_cnot([1,1]),cnot_gate([1,1])))
```

## 7. イオンが多すぎる場合の問題



主な問題は、スケーラビリティ(適応度合い)。

問題の根源と技術的課題は、特定のフレームワークによって異なる。

### ディヴィンチェンツォの基準

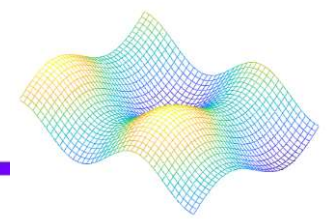
1. よく特性化されたスケーラブルな量子ビット。
2. 量子ビットの初期化。
3. 長いコヒーレンス時間。
4. 汎用ゲート。
5. 個々の量子ビットの測定。

トラップイオン量子コンピュータは、

ディヴィンチェンツォの基準のうち、どれを満たしていないのか？

- ・基準1は部分的にしか満たされていない。
- ・基準2は、光ポンピング技術のおかげで大きな問題はない。
- ・基準3も、イオン鎖が長いと運動状態のコヒーレンス時間が短くなるため、問題。
- ・基準4の2量子ビットの要件は、このデコヒーレンス問題と関連。
- ・基準5には問題が残る。

## 8. 最先端の技術



<QCCDアーキテクチャとしても知られるセグメント化されたトラップについて>

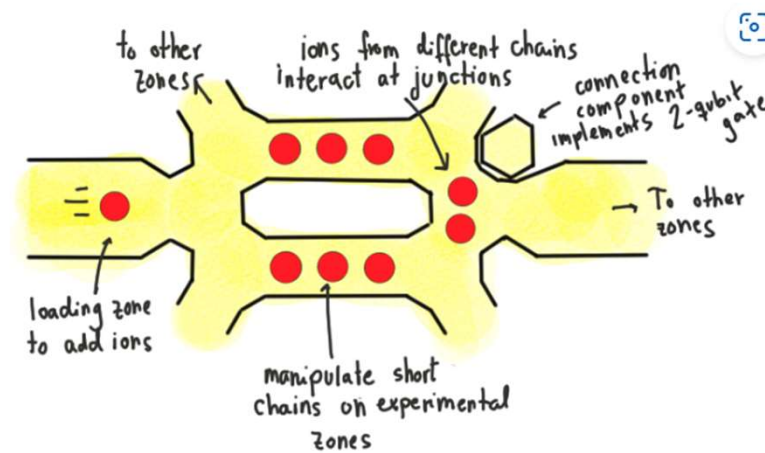


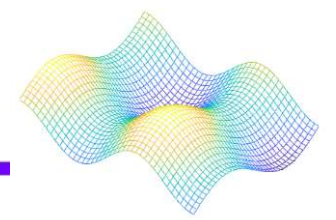
図18 Example of a proposed QCCD architecture, as in [12]

QCCDとゲートの高速化を含むアプローチを組み合わせることで、  
将来的にはスケーラビリティ問題の解決策が得られるかもしれない。

2量子ビットのゲートを効率的に適用するために、  
イオンと光子を接続する別の解決策も提案されている。

優れたQCCDアーキテクチャと組み合わせることで、  
トラップされたイオンを用いて量子コンピューティング・アルゴリズム  
を実行するのに十分なものとなるだろう。

## 9. 結び



イオントラッピングは現在、学术界でも産業界でも、  
量子コンピューターの物理的実装として  
最も普及しているもののひとつである。

量子コンピューターの規模をさらに拡大するには、技術的に難しい問題がある。

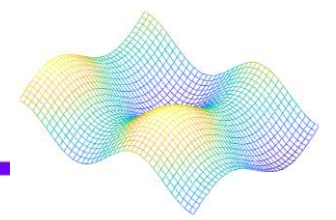
実現に向けてノンストップで取り組まれている。

# Quantum computing

with superconducting qubits

超伝導量子ビットを用いた量子コンピューティング





超伝導・・・特定の金属や化合物などの物質を冷却した時に、電気抵抗が0になる現象

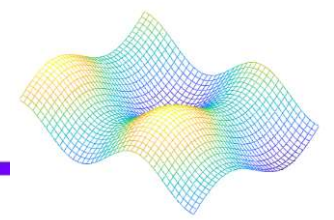
なぜすべての導体が超伝導の性質を持たないのか



パウリの排他原理が存在しているから

逆に超伝導の材料を作り出せば排他原理を無視することが可能

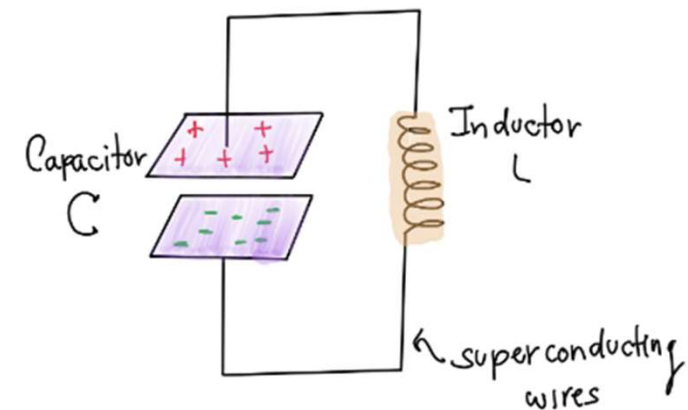
# 人工原子の構築



クラウド上の実際の超伝導量子ビットコンピュータで量子アルゴリズムを実行

原子を量子ビットに適したものにすると同じ特性を持つデバイスを構築するために、人工原子を作ることが可能か試す

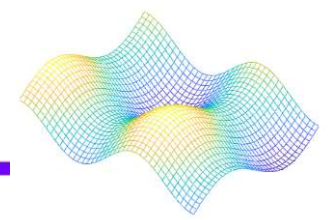
- ①理想的な量子ビットが持つべき機能を分離する
- ②簡単な超伝導回路を構築する
- ③エネルギー準位の間隔問題を解決する



超伝導LC回路



# 量子ビットの測定



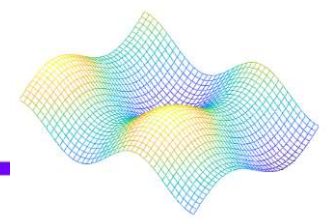
量子ビットの状態を測定するには実際に計算する必要がある  
(ハミルトニアンに依存する)

$$i\hbar \frac{\partial}{\partial t} |\psi(t)\rangle = \hat{H} |\psi(t)\rangle .$$

ApproxTimeEvolution を使用することによって  
PennyLaneで計算可能

光のコヒーレント状態のシミュレート

$|g\rangle$ : 基底状態  $|e\rangle$ : 励起状態



人工原子のハミルトニアンの式

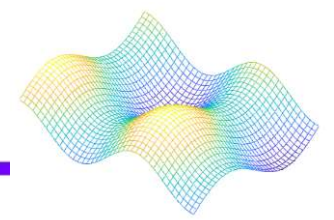
$$\hat{H} = \hbar\Omega_R(\hat{\sigma}_x \cos \phi + \hat{\sigma}_y \sin \phi).$$

$\Phi=0$  , X-回転     $\Phi=\pi/2$  , Y-回転

ハミルトニアンを定義し(qml.Hamiltonian)

Pannylaneを用いて確認(ApproxTimeEvolution)

# 2量子ビットゲートの測定



結合コンデンサを介して接続された2つのトランスモン  
(量子ビットの相互作用を制御)

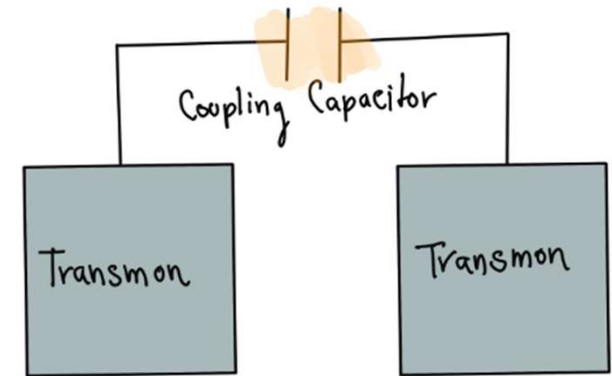
2量子ビットシステムのハミルトニアンの式

$$\hat{H} = \frac{\hbar J}{2} (\sigma_1^x \sigma_2^x + \sigma_1^y \sigma_2^y),$$

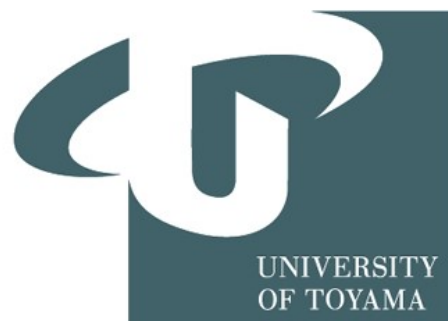
上式により2量子ビットの*i*SWAPゲートを実装

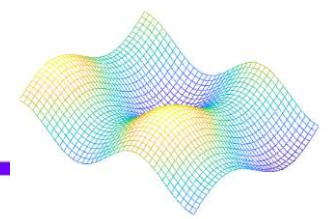
$$iSWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$t=3\pi/2J$ のもとPennyLaneを用いて確認



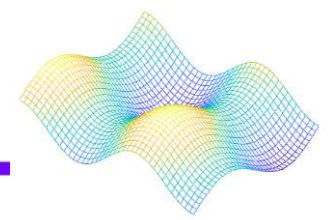
# Modeling the toric code on a quantum computer



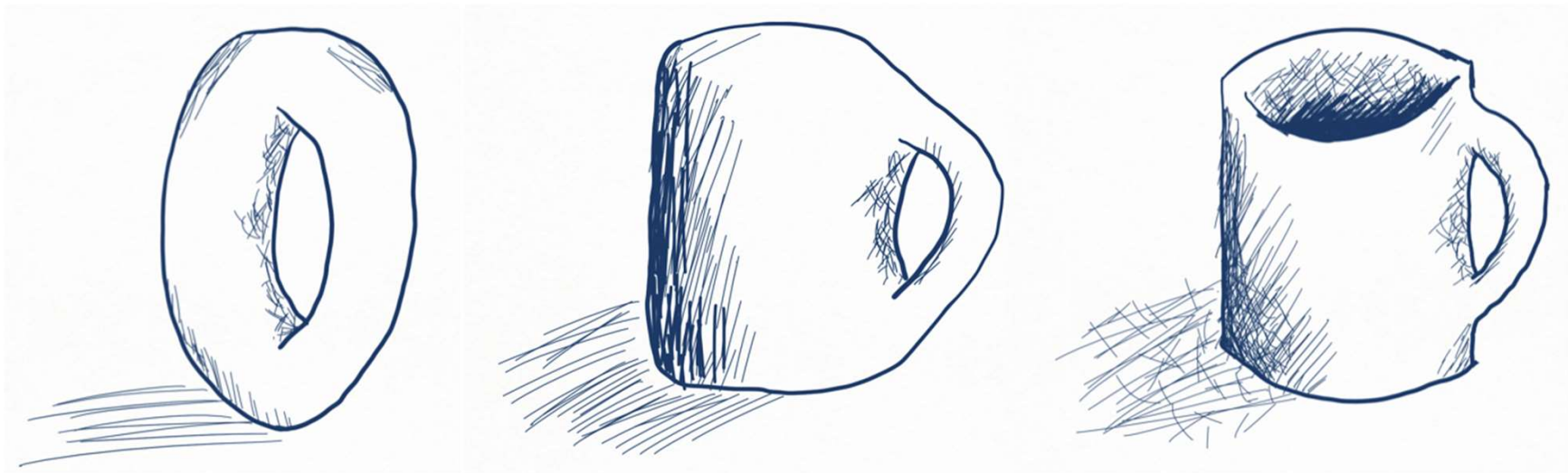


トーリックコードモデルは、誤り訂正モデルの重要なカテゴリである誤り訂正コードの開発のきっかけとなった。

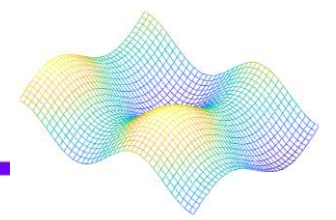
トーリックコードは物質のトポロジー状態の一例



## トポロジー

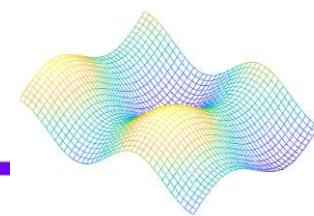


ドーナツをスムーズにマグカップに変形させることができます。



物質の状態(位相)はハミルトニアン $H$ の係数が変化するとき、物理的特性に何らかの不連続性がなければ、異なる位相になることはできない

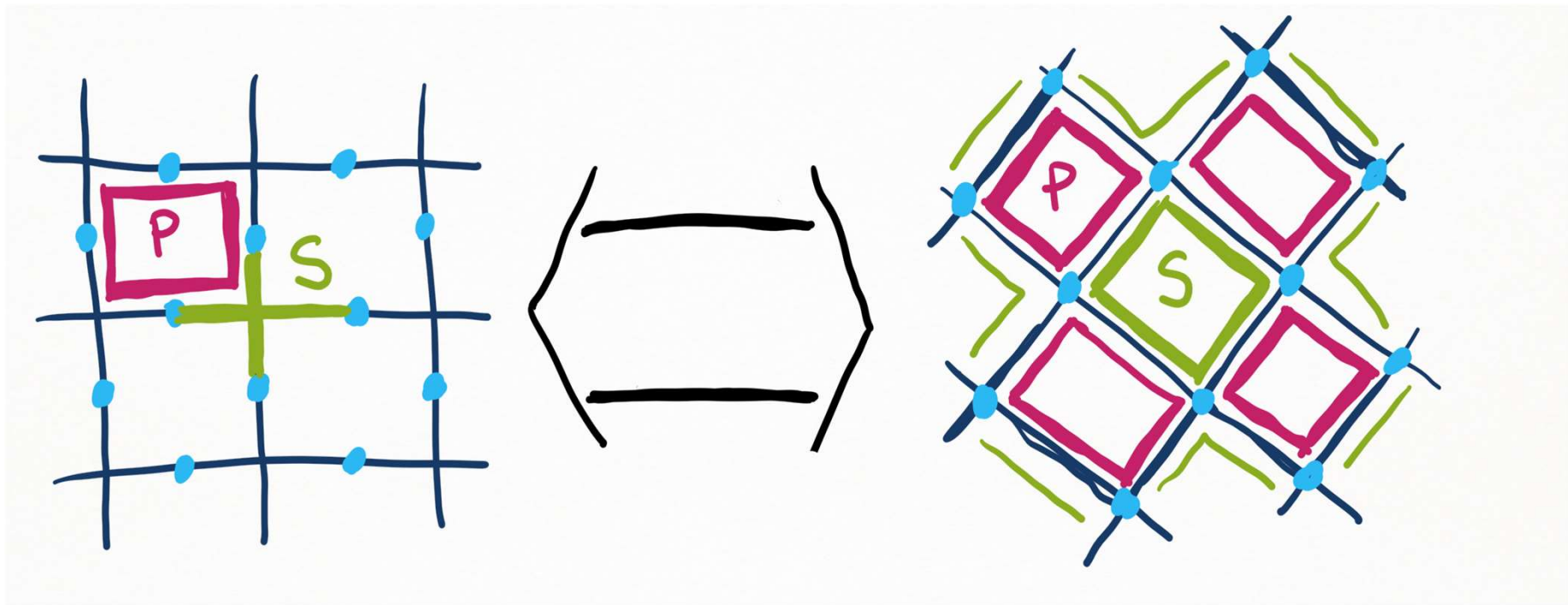
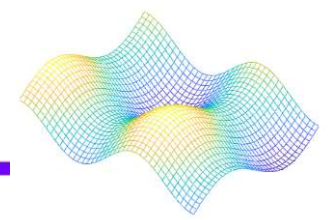
例えば、温度が変化すると、密度が不連続にならなければ氷は水になることはできない



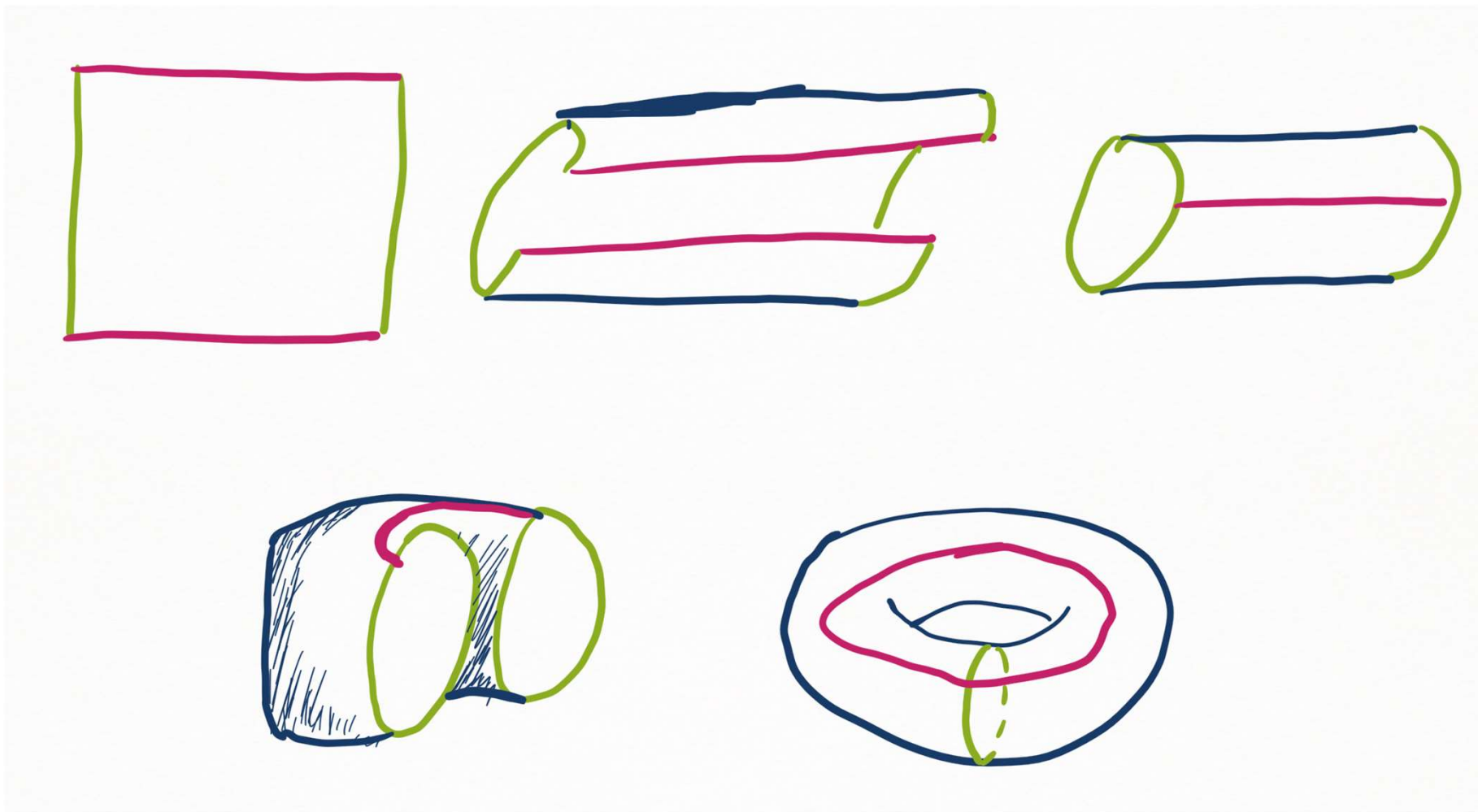
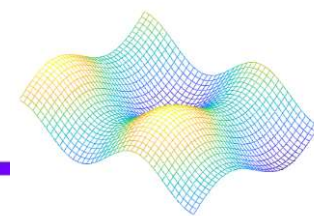
$$\mathcal{H} = - \sum_s S_s - \sum_p P_p,$$

$$S_s = \prod_{i \in s} Z_i \quad P_p = \prod_{j \in p} X_j.$$



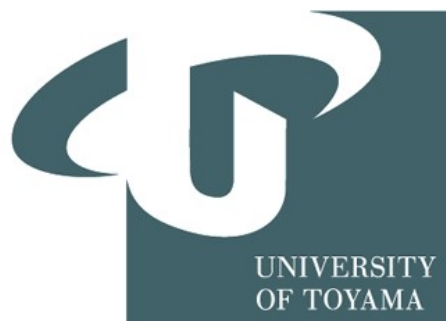


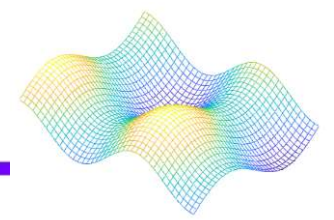
左側では、サイトが頂点の周りの星形と面上のプラケットにグループ化されています。右側では、格子を交互のタイプのグループのチェッカーボードとして表示します。



エッジを周期的な境界条件と一致させることにより、正方形のグリッドをトーラスに変えます。

# Measurement-based quantum computation





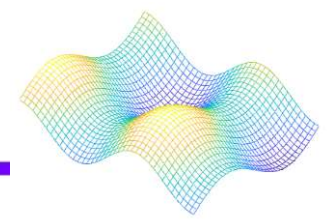
MBQC=測定型量子計算

量子ゲート方式・・・エンタングルしていない初期量子状態からはじめ、量子ゲートを次々に作用させエンタングルさせながら計算を進めていくので、計算の途中ずっと量子状態を保っておく必要がある

MBQC・・・最初に大きくエンタングルした状態を作っておけば、あとは測定を繰り返すだけで計算が進んでいく

※エンタングル・・・2つ以上の量子系が相互作用し、絡み合っ  
て一体化した状態

# クラスタ状態とグラフ状態

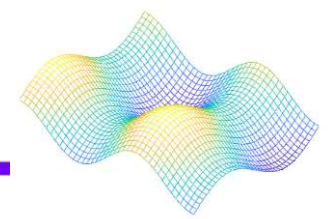


クラスタ状態は、MBQCのためのユニバーサルな基盤

これらはエンタングルした複数の量子ビットの状態を表すために、頂点 $V$  が量子ビットに関連し、辺 $E$ がそれらの間のエンタングルメントに関連する無向グラフ $G=(V,E)$ で表現されるエンタングルグラフの特別なインスタンス

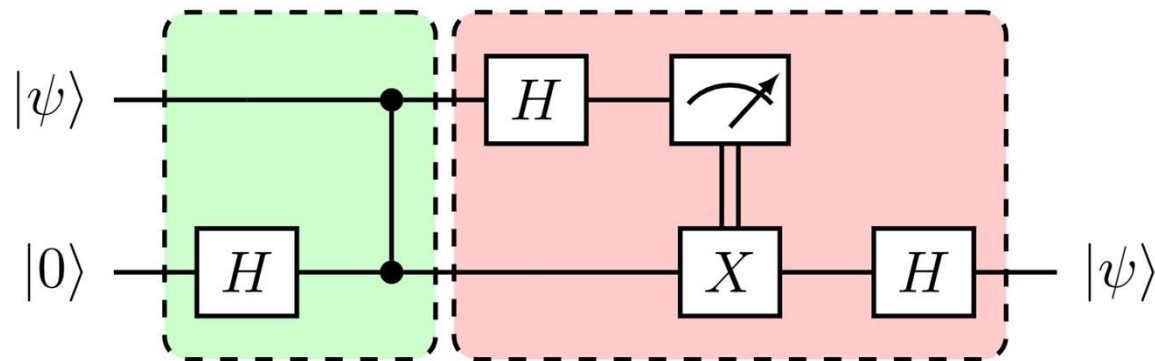
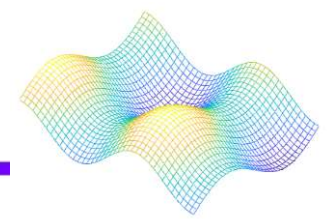
関連する量子状態は以下のように表される

$$|\Phi\rangle = \prod_{(i,j) \in E} CZ_{ij} |+\rangle^{\otimes n}.$$



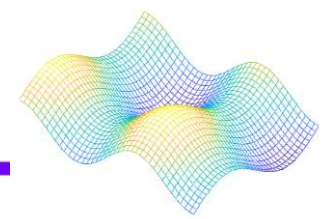
MBQCでは、量子テレポーテーションのプロトコルが使用され、情報の伝播に利用される。

ある量子ビットの状態を測定すると、その結果に基づいて別のエンタングル状態の量子ビットの状態が変化する。この変化した量子ビットは、次の測定ステップで再び測定され、結果が他の量子ビットに伝播する。これにより、情報が量子ビット間で伝播し、計算が進行する。



## 1量子ビットのテレポーテーションの Protokol 例

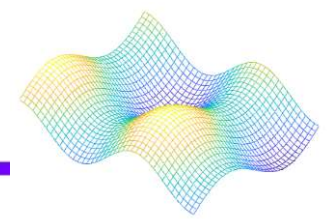
- ・第1の量子ビットの状態を準備 (図の上部)
- ・エンタングルされた状態のクラスタを作成 (図の緑部分)
- ・第1の量子ビットとクラスタの測定を行う (図の下部)
- ・第1の量子ビットとクラスタの測定結果が得られる
- ・測定結果をもとに、第2の量子ビットに補正操作を適用し、第2の量子ビットの状態が第1の量子ビットの状態と同じになる (図の赤部分)



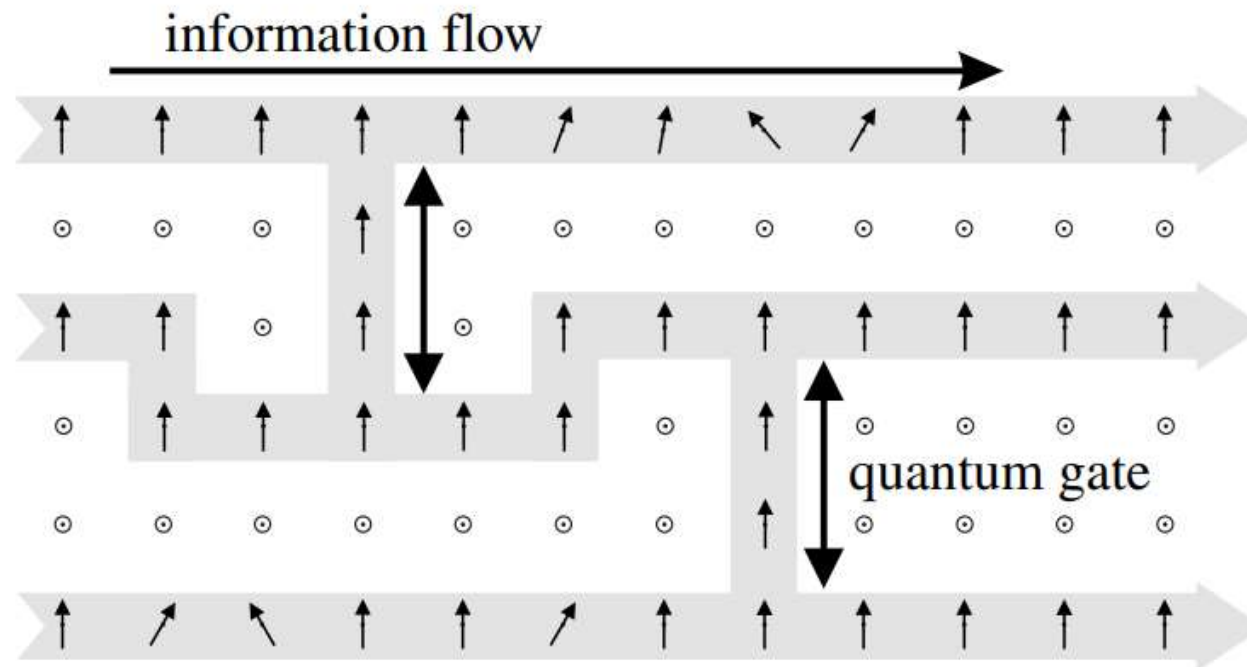
この測定型のアプローチがゲート型のアプローチと同じくらい強力であるかどうかを確認するには、クラスタ状態を通じた伝播方法に加えて、任意の量子回路を実装する方法を示す必要がある。



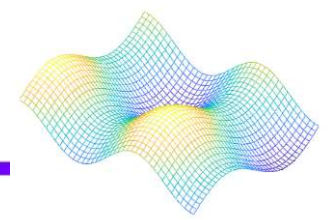
# 任意の量子回路を実装する方法



任意の量子回路を実装するには、ユニバーサルなゲートセットを持っていることを示す必要がある。



完全な測定型量子計算。●記号はパウリZの測定を示し、垂直な矢印↑はパウリXの測定を示し、斜めの矢印↖↗はxy平面での測定を示す。

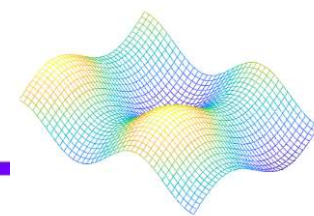


クラスタ状態のエンタングルメントと適応的な任意の単一量子ビット測定が可能なMBQCは、ユニバーサルな量子計算を実現することができる。

MBQCは多くの使い捨ての量子ビットや簡単な物理的エンタングルゲートを可能にするプラットフォームで特に役立つ。

# Testing for symmetry with quantum computers



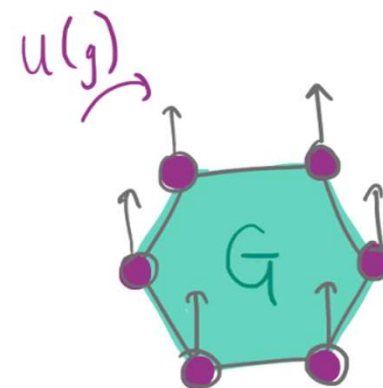


## 対称性の検証

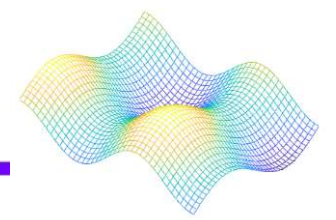
- ・対称性とは>何かを**同じ形**にする変換のことである

すなわち...

- ・量子系であるハミルトニアンが近似的な対称性を持つことは理にかなっている

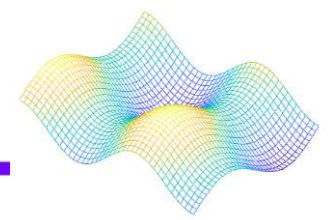


$$[\hat{H}, u(g)] = 0$$



ハミルトニアンの特称性をテストするためのLaBorde and Wilde (2022)のアルゴリズムを実装する

系が対称性の有限群 $G$ を持つかどうか、また持たない場合、どの程度対称性が破られているかを決定することができる



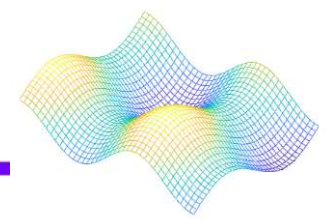
行列と交わることを示す

$$[U, (g)^H] = 0$$

ハミルトニアンは時間発展を発生させる

これは群変換を今適用しても、後で適用しても**効果は同じ**であることを意味する

# 対称性の平均化(1)



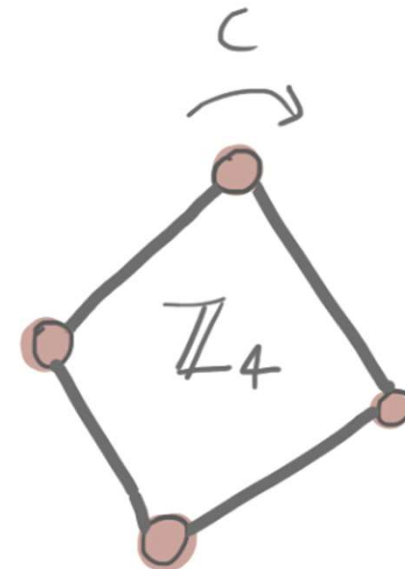
Gに関してハミルトニアンが対称であることを検証する

$$\frac{1}{|G|} \sum_{g \in G} [U(g), \hat{H}] = 0.$$

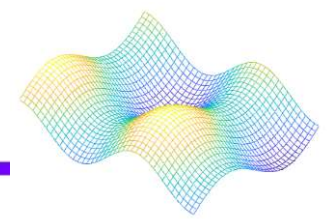
環状群を考えると...  $G = \mathbb{Z}_4$



正方形の回転と考えることができる！



## 対称化の平均(2)



ここでG に関して

- ・厳密に対称なもの ( $\hat{H}_{\text{sym}}$ )
- ・対称に近いもの ( $\hat{H}_{\text{nsym}}$ )
- ・非対称なもの ( $\hat{H}_{\text{asym}}$ ) として

$$\hat{H}_{\text{sym}} = X_0 + X_1 + X_2 + X_3$$

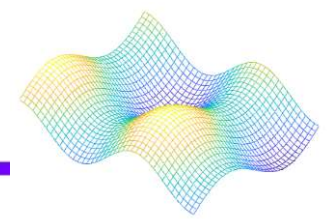
$$\hat{H}_{\text{nsym}} = X_0 + 1.1 \cdot X_1 + 0.9 \cdot X_2 + X_3$$

$$\hat{H}_{\text{asym}} = X_0 + 2 \cdot X_1 + 3 \cdot X_2.$$

これがPennyLaneでどのように見えるか見ていく



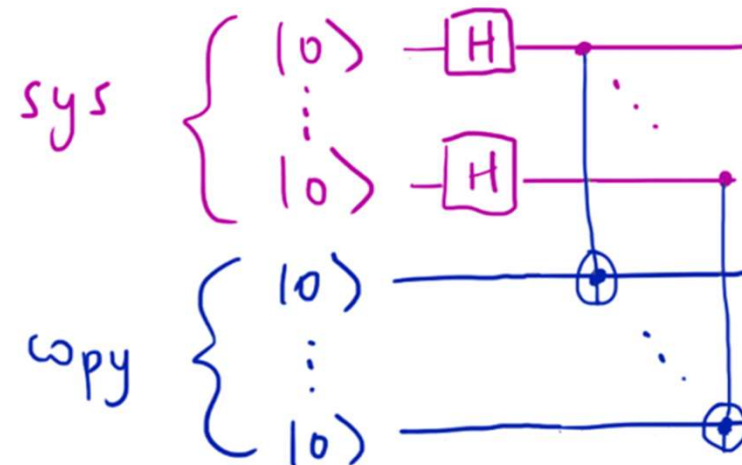
# 対称化の平均(3)



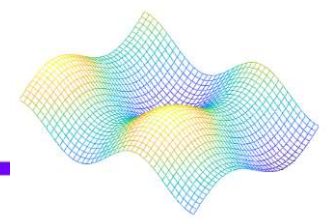
平均的な対称条件は、次の条件と等価であることを数学的に示すことができる。ここで $\Pi_G$ は次式で定義される

$$\Pi_G = \frac{1}{|G|} \sum_{g \in G} U(g) \otimes \overline{U(g)}.$$

$\Pi_{2G} = \Pi_G$ したがって、これはプロジェクターであり、関連する測定値を持つ



# 制御された対称性(1)

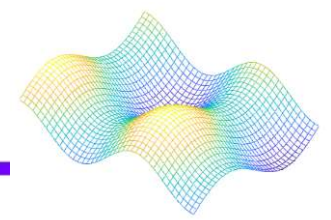


システムHはコピーだけでなく、レジスタHgともつれ合う。これを修正するためにレジスタを観測し、それが重ね合わせ $|+\rangle_G$ にあるかどうかを確認する。この観測を条件とした状態と確率は...

$$\begin{aligned} {}_G\langle + | \frac{1}{\sqrt{|G|}} \sum_{g \in G} (U(g) \otimes \overline{U(g)}) |\Phi_t\rangle \otimes |g\rangle_G &= \frac{1}{|G|} \sum_{g, g' \in G} (U(g) \otimes \overline{U(g)}) |\Phi_t\rangle \langle g' | g \rangle_G \\ &= \Pi_G |\Phi_t\rangle, \end{aligned}$$

$$P_+ = |\Pi_G |\Phi_t\rangle|^2 = \langle \Phi_t | \Pi_G^\dagger \Pi_G | \Phi_t \rangle = \langle \Phi_t | \Pi_G | \Phi_t \rangle$$

# 制御された対称性(2)



```
# Circuit for average symmetry
@qml.qnode(dev, interface="autograd")
def avg_symm(hamiltonian, time):

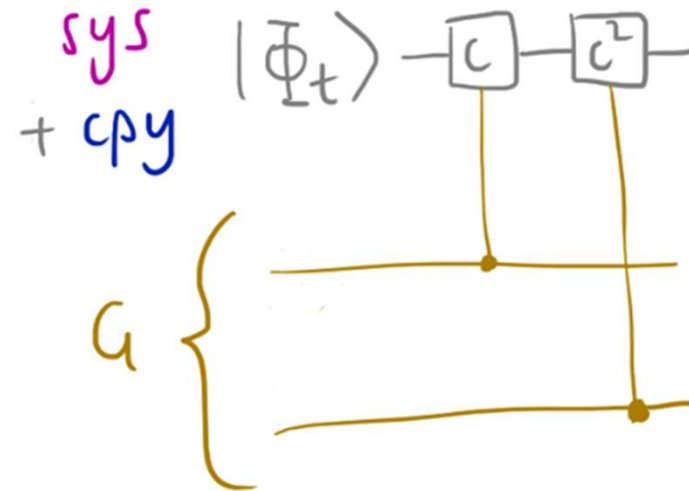
    # Use Choi-Jamiołkowski isomorphism
    choi_state(hamiltonian, time)

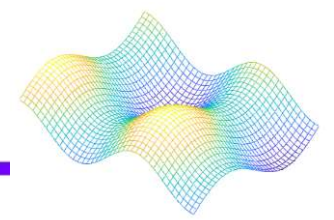
    # Apply controlled symmetry operations
    prep_plus()
    CU_sys()
    CU_cpy()

    # Ready register for measurement
    prep_plus()

    return qml.probs(wires=aux)
```

```
print("For Hamiltonian Hsymm, the  $|+\rangle$  state is observed with probability", avg_symm(Hsymm, 1)[0], ".")
print("For Hamiltonian Hnsym, the  $|+\rangle$  state is observed with probability", avg_symm(Hnsym, 1)[0], ".")
print("For Hamiltonian Hasym, the  $|+\rangle$  state is observed with probability", avg_symm(Hasym, 1)[0], ".")
```



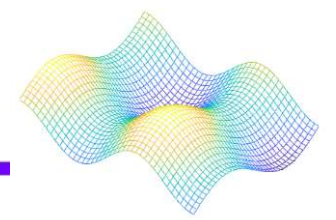


## 先程の回路に残る不安

1. ハミルトニアンが平均的に対称かどうかを測定するだけである
2. 回路から出てくる数値が何を意味するのかわからない

しかし、この2つの疑問は、非常に短い時間  $t \rightarrow 0$  を考えることで解決できる

## 短い制限時間(2)



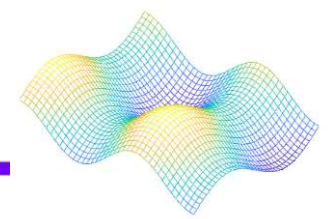
$|+\rangle_G$  の状態を観測する確率  $P_+$  は次のように証明でき、

$$P_+ = \langle \Phi_t | \Pi_G | \Phi_t \rangle = 1 - \frac{t^2}{2d|G|} \sum_{g \in G} \|[U(g), \hat{H}]\|_2^2 + O(t^3),$$

コミュテータノルムの2乗の合計である。この和を非対称性  $\xi$  と呼ぶことにする

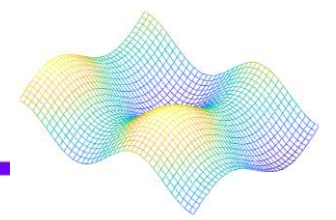
$$\xi = \sum_{g \in G} \|[U(g), \hat{H}]\|_2^2 = \frac{2d|G|(1 - P_+)}{t^2} + O(t).$$

# 短い制限時間(3)



```
# Define asymmetry circuit
def asymm(hamiltonian, time):
    d, G = 16, 4
    P_plus = avg_symm(hamiltonian, time)[0]
    xi = 2 * d * (1 - P_plus) / (time ** 2)
    return xi

print("The asymmetry for Hsymm is", asymm(Hsymm, 1e-4), ".")
print("The asymmetry for Hnsym is", asymm(Hnsym, 1e-4), ".")
print("The asymmetry for Hasym is", asymm(Hasym, 1e-4), ".")
```

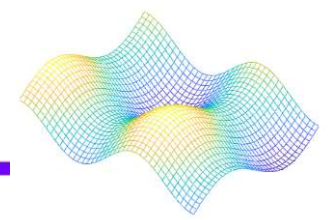


量子力学において対称性は物理的に重要である

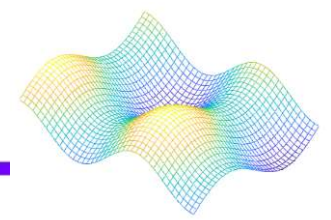
# Barren plateaus in quantum neural networks



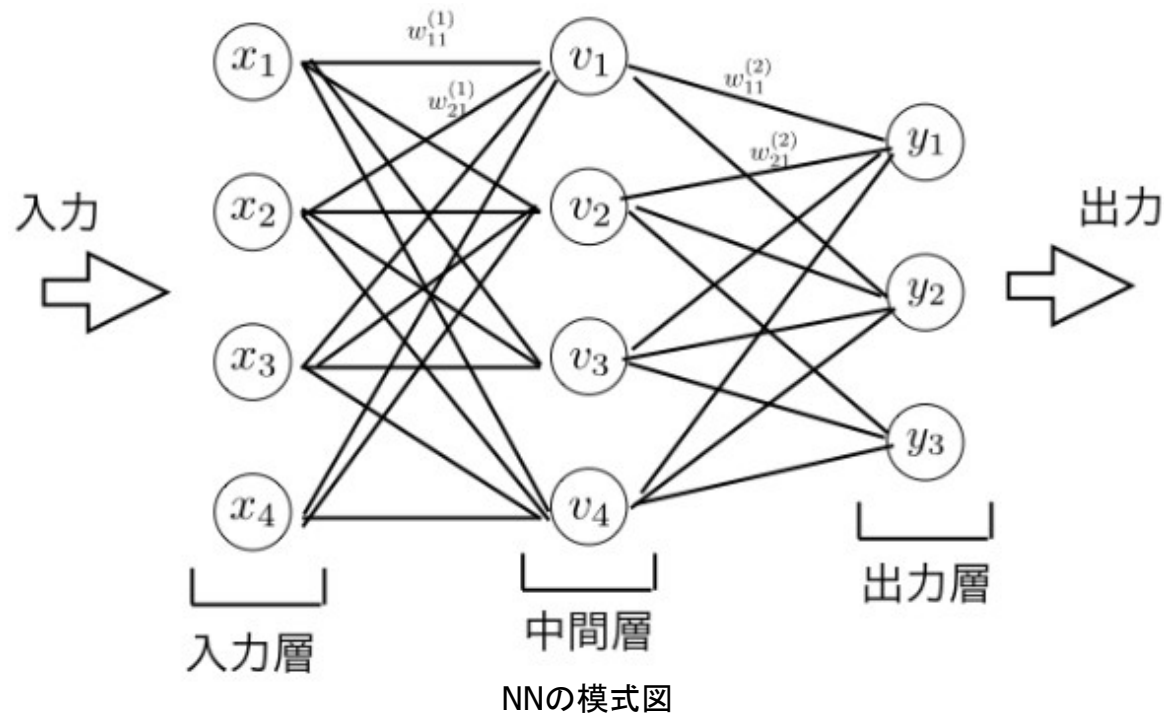




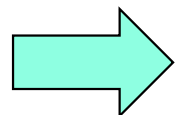
- ・量子ニューラルネットワークについて
- ・量子ニューラルネットワーク学習の問題と対策
- ・シミュレーション, 結果



NN・・・人間の脳の神経伝達構造を数式的に表したモデル

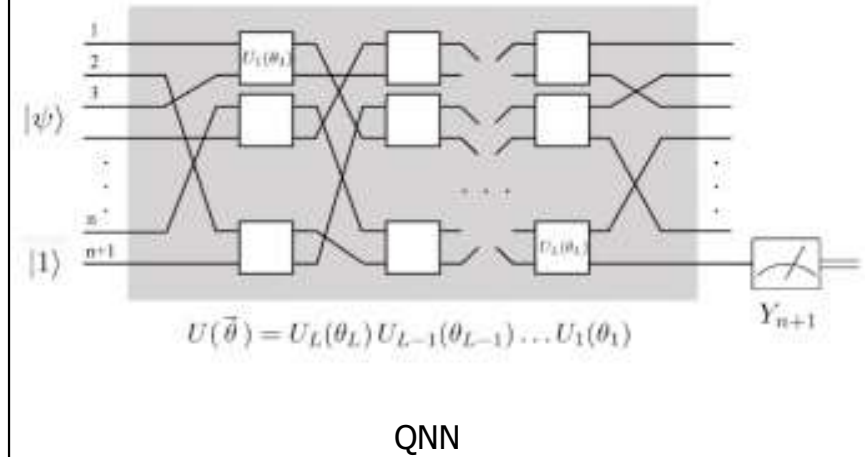
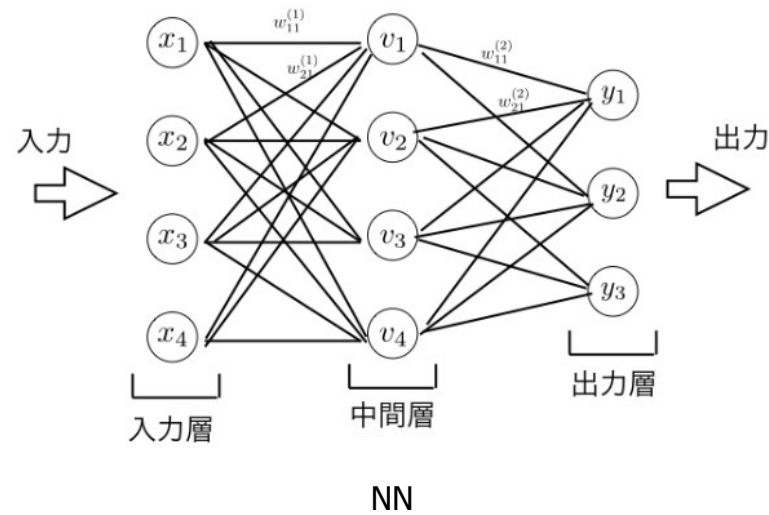
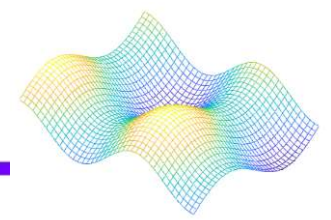


1. 入力層, 中間層, 出力層からなる
2. 各ユニットは複数の入力を受け付ける
3. 重みとバイアスによって線型結合
4. 活性化関数と呼ばれる非線形関数を通して出力される



NNは高い表現性を持ち, 複雑な数理モデルを構築可能

# 量子ニューラルネットワークについて



ユニットの構造

入力値と重みの線型結合を計算し、活性化関数を通じて出力

量子ビットとして表現される(0と1の重ね合わせの状態を保つ)

レイヤーの構造

ユニットの集合で入力層, 中間層, 出力層から構成

量子ゲートの連鎖

パラメータの調整

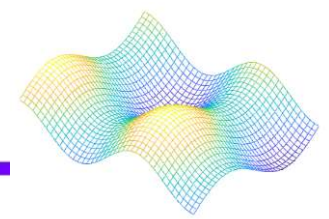
学習中に重みとバイアスというパラメータが調整

量子ビットの状態, 量子ゲートのパラメータが調節

出力の取得

最終的なレイヤーのユニットの出力

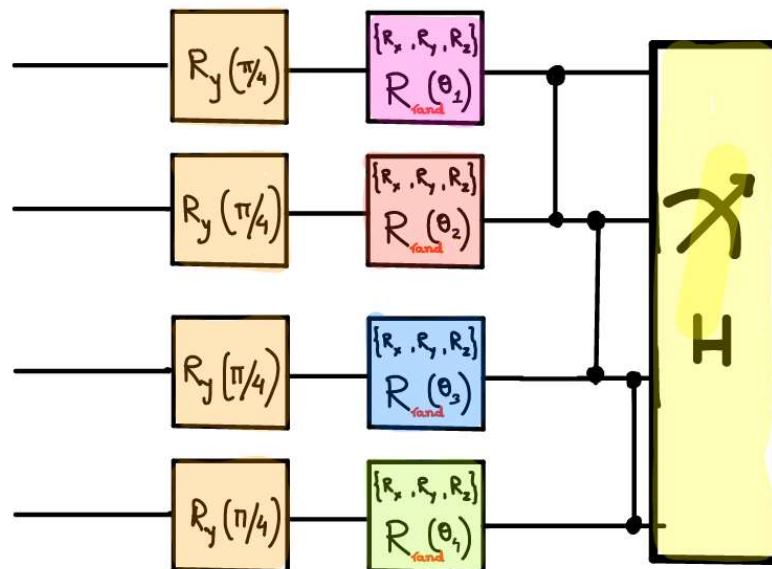
量子ビットの状態測定結果



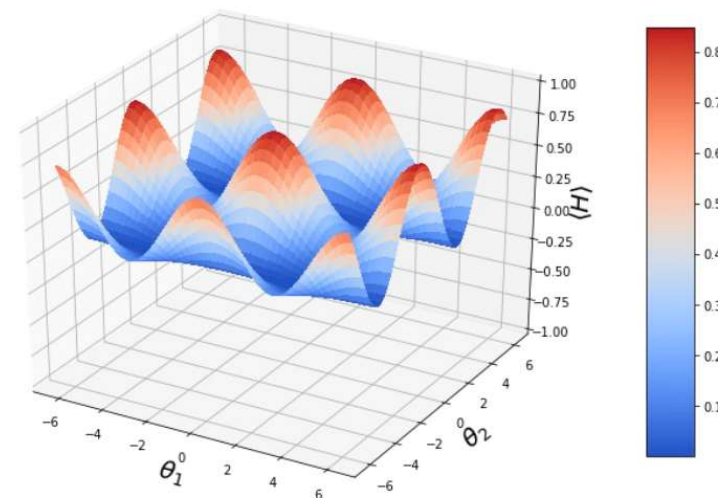
## ・量子ニューラルネットワーク(QNN)の学習

所望する数理モデルとしてうまく出力を表現できるように、ハイパーパラメータの値(量子回路の各ゲートが持つ値であり、回路の振る舞いを決定する)を調節(学習)する必要がある

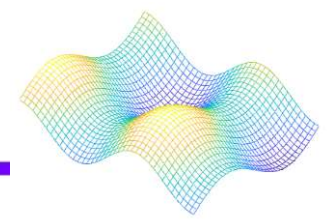
→QNNの出力とターゲット値の間の差異を評価する目的関数の最小化問題を最適化手法などを使用して解く



A random variational circuit with gates chosen from the set  $\{RX, RY, RZ\}$



Expectation value of a Hermitian observable along a slice in the parameter space



## ・問題点

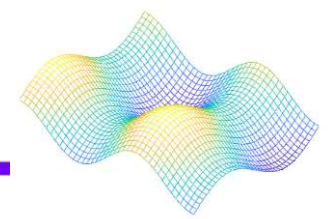
ランダム初期化: モデルのパラメータや変数をランダムな値で初期化

→勾配が一定の精度まで非ゼロである確率は、量子ビット数の関数として指数関数的に小さくなる(パラメータの微小な変化が勾配にほとんど影響を与えない)

QNNのパラメータ空間は一般に非常に大きく、多くの次元を持つ

→勾配の推定や最適化の計算コストが指数関数的に増加  
このようにQNNの学習において、特定の条件下で勾配の推定が困難になり、学習の進展が停滞する状態を**バレン高原**と表現する

→バレン高原の問題への解決策や回避策が大切



## ・対策案(初期化戦略<sup>1</sup>)

Grantら(2019)の研究では、バレン高原問題に対処するための初期化戦略の一つが提案された。

手順は以下のようになる。

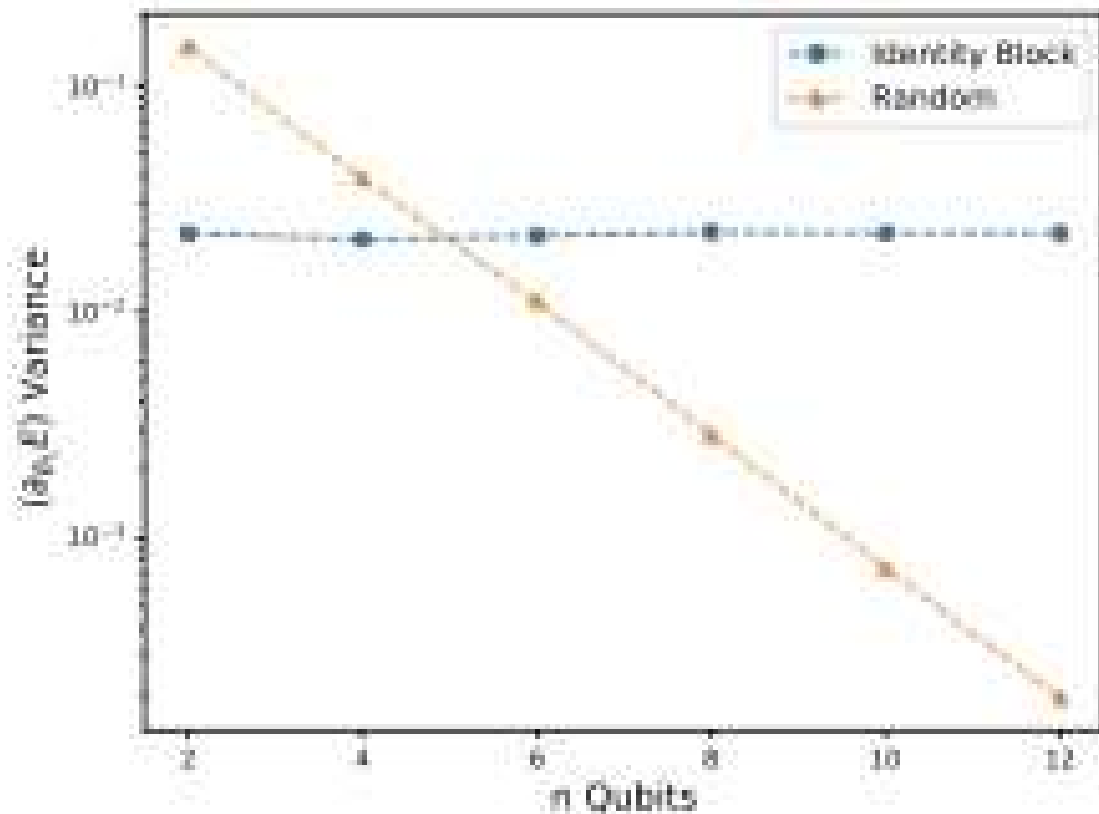
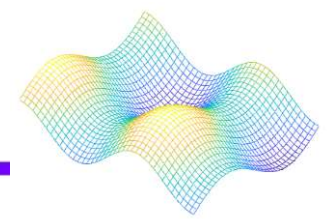
1) パラメータのランダムな選択: 最初に、初期パラメータの一部をランダムに選択する。これにより、回路内の一部のユニタリーブロックがランダムな演算を評価する

2) ユニタリーブロックのシーケンス化: 次に、残りのパラメータ値を選択します。この選択では、浅い(層の数)ユニタリーブロックのシーケンスを作成し、各ブロックを恒等演算子(単位行列)と等しい操作に設定する

→学習の最初のパラメータ更新における回路の有効な深さを制限して学習の初めにバレン高原に陥ることが防止される。

1. Grant, Edward, et al. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. arXiv preprint arXiv:1903.05076 (2019)

# シミュレーション結果

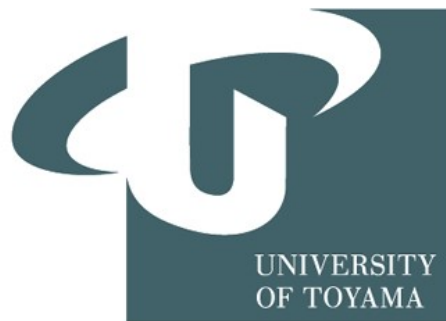


ランダム初期化を使用した場合，勾配の分散はシステムサイズとともに消滅した．一方，同一性ブロック初期化を用いた回路では，分散はシステムサイズとともに指数関数的に消滅せず，初期化時にプラトーが回避された

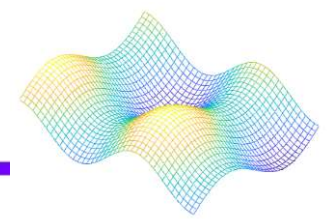


# Quantum circuit structure learning

---







- 量子機械学習および最適化問題では回路内パラメータの選定をする必要がある。
- パラメータ選定におけるコスト関数を最小限に抑えることで、ノイズの影響を軽減することができる。
- PennyLane の最適化アルゴリズム (勾配降下法)  
量子回路から量子ノードの微分を行い、コスト関数を最小化していく
- Rotoselect アルゴリズム  
→ 勾配降下法を用いずに最適値を直接得ることができる

# 固定量子回路とRotosolve法

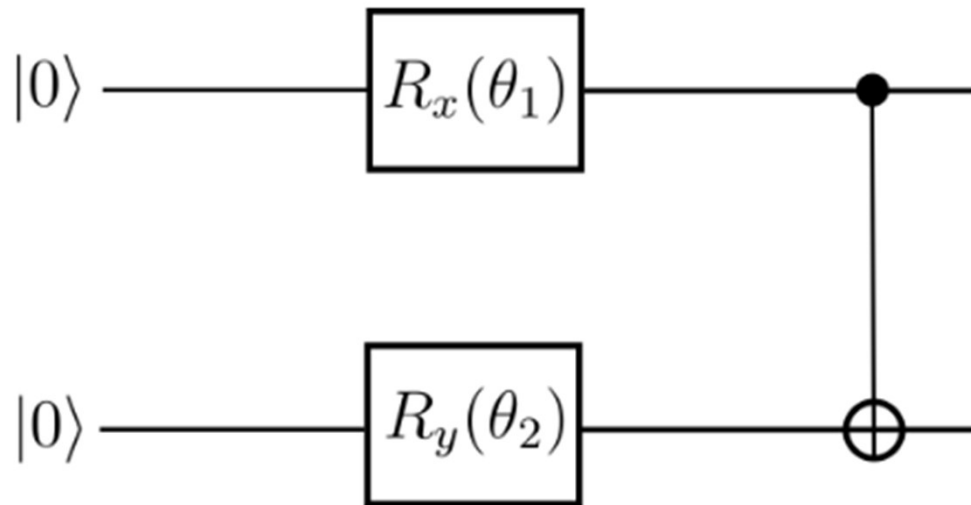
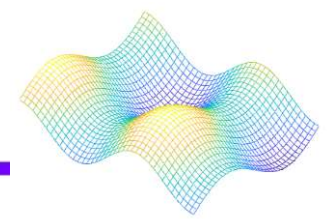


Fig. 固定アンザッツ構造を備えた量子回路

$$H = 0.5Y_2 + 0.8Z_1 - 0.2X_1$$

$$\begin{aligned} \theta_d^* &= \operatorname{argmin}_{\theta_d} \langle H \rangle_{\theta_d} \\ &= -\frac{\pi}{2} - \arctan \left( \frac{2\langle H \rangle_{\theta_d=0} - \langle H \rangle_{\theta_d=\pi/2} - \langle H \rangle_{\theta_d=-\pi/2}}{\langle H \rangle_{\theta_d=\pi/2} - \langle H \rangle_{\theta_d=-\pi/2}} \right) \end{aligned}$$

## 量子回路

X軸回転ゲートとY軸回転ゲートとCNOTを使用した2ビット量子回路を構築した。

## Rotosolve法

回転パラメータ $\theta_d^*$ を  
 $\theta_d = 0$ 、 $\theta_d = \frac{\pi}{2}$ 、 $\theta_d = -\frac{\pi}{2}$ の時  
コスト関数を用いて最適なパラメータ  
を選択する。

# Rotosolveの最適化と勾配降下法での比較

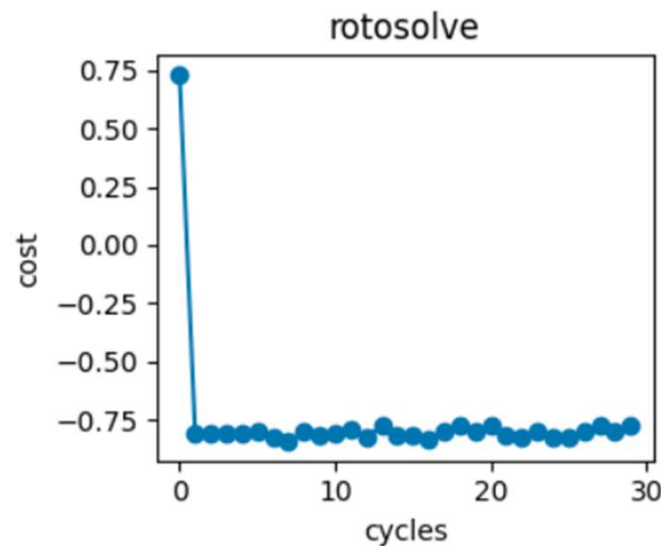
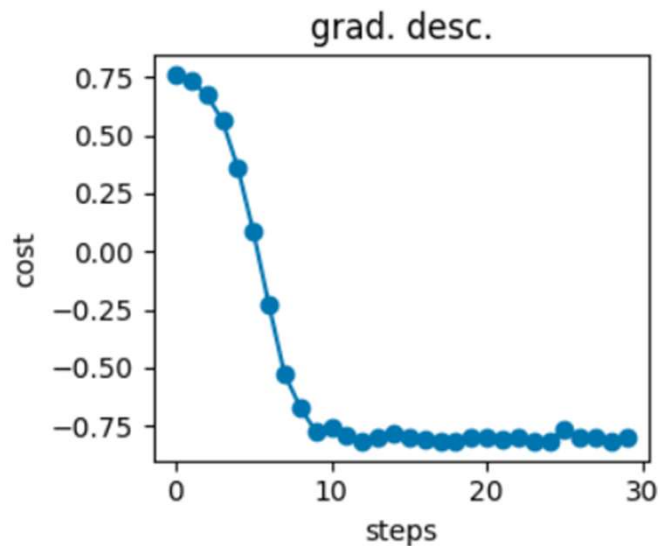
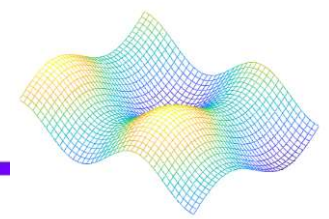


Fig. 勾配降下法(左)とRotosolve(右)によるコスト関数の遷移

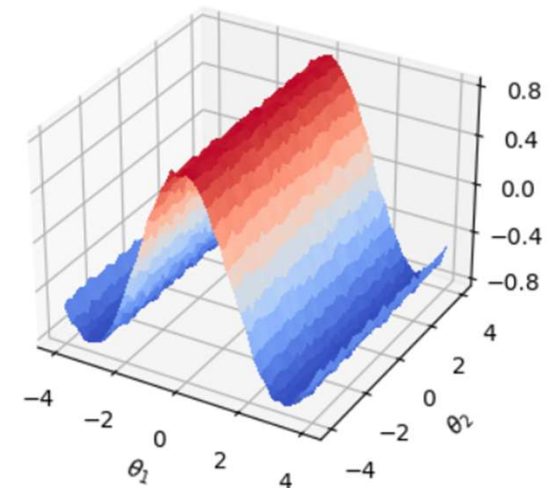
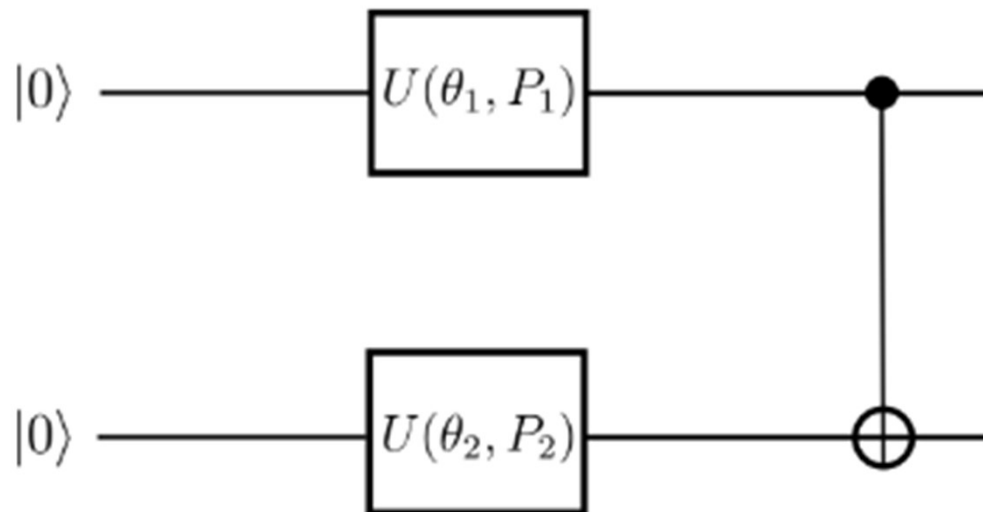
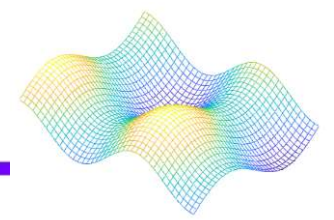


Fig. Rotosolveによるコスト関数曲面

- Rotosolve の 1 サイクルに含まれる回路評価の数と、  
勾配を計算してこの方向に進むのに必要な評価の数は等しい
- **Rotosolve** による最適化のほうは一回で最小値に収束している
- コスト関数が回転ゲートRyの角度パラメータ $\theta_2$ に依存しない



## 量子回路

2つの回転軸が任意の回転ゲートとCNOTを使用した2ビット量子回路を構築した。

## Rotoselect法

回転ゲートをX軸、Y軸、Z軸からそれぞれ選択し、その後Rotosolve法を使用することで、最適な回転ゲートを選択する。

# オリジナル回路とRotoselect法の比較

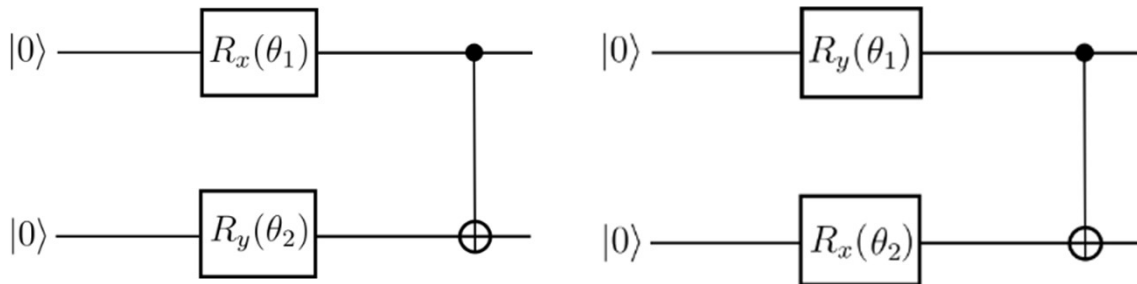
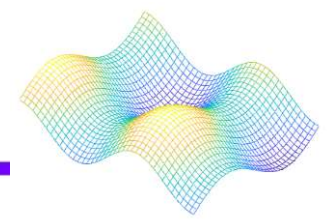


Fig.オリジナル回路(左)とRotoselect により選択された回路(右)

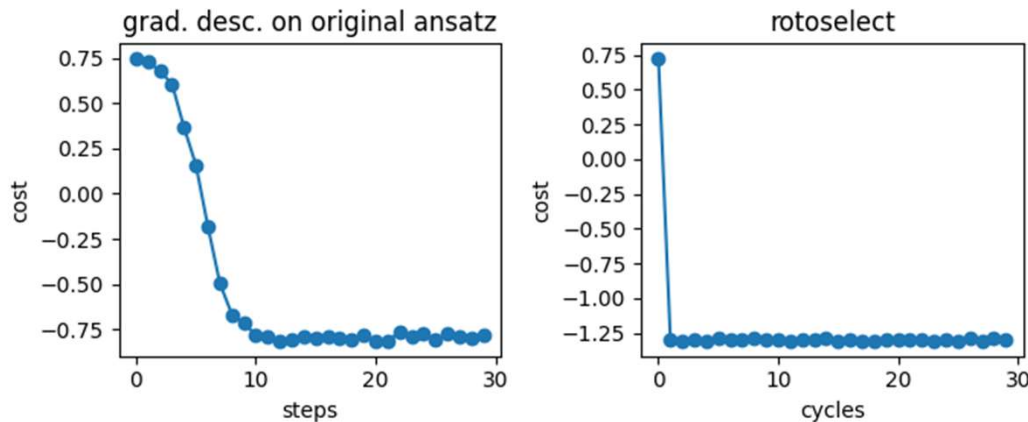


Fig.オリジナル回路(左)とRotoselect(右)によるコスト関数

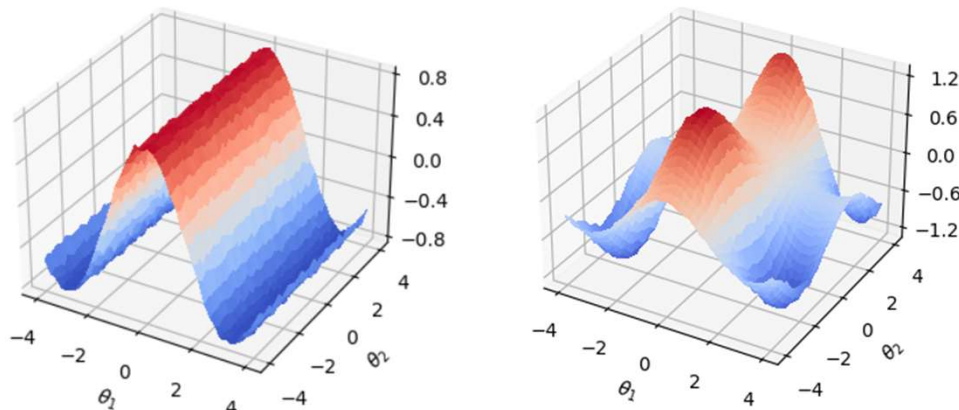


Fig.オリジナル回路(左)とRotoselect(右)によるコスト関数曲面

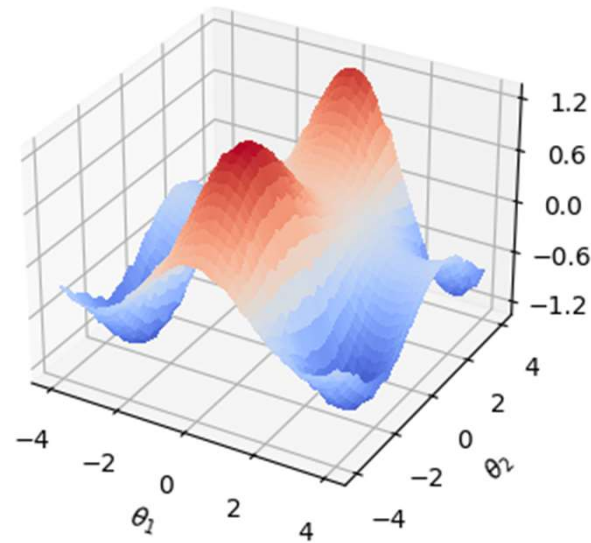
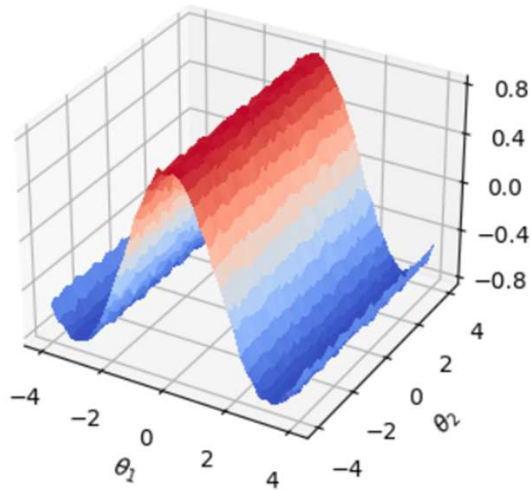
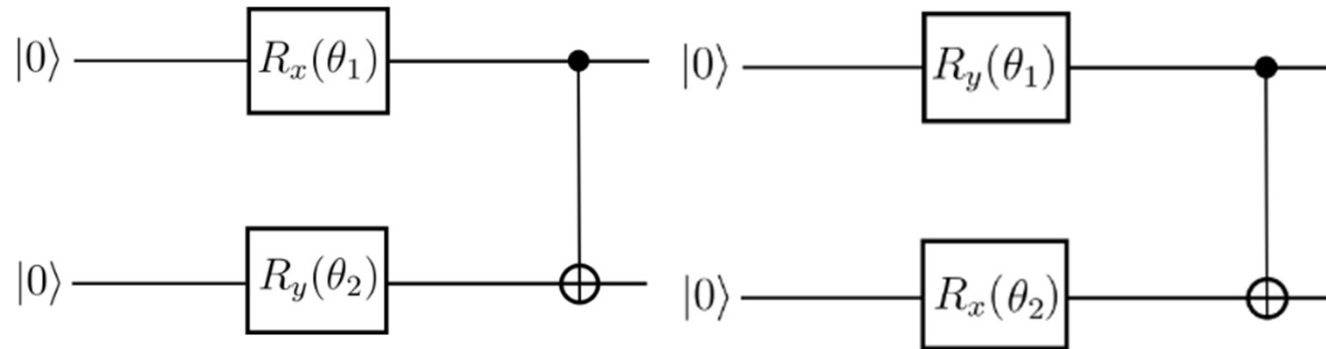
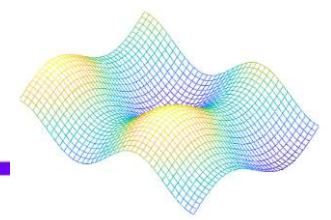
## 結果

最適な回転ゲートを選択することで両方の回転ゲートがコスト関数に作用し、コスト関数の最小値がより小さくなり、回路の最適化が行われた。

## まとめ

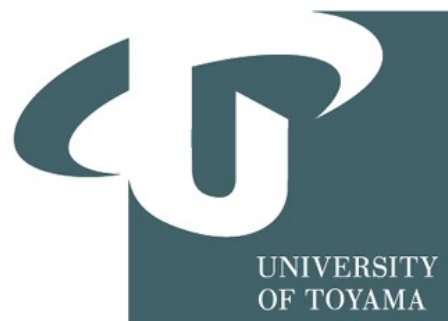
Rotosolve法とRotoselect法を用いることでよりよい回路構造を選択することができることがわかった。

# 結果

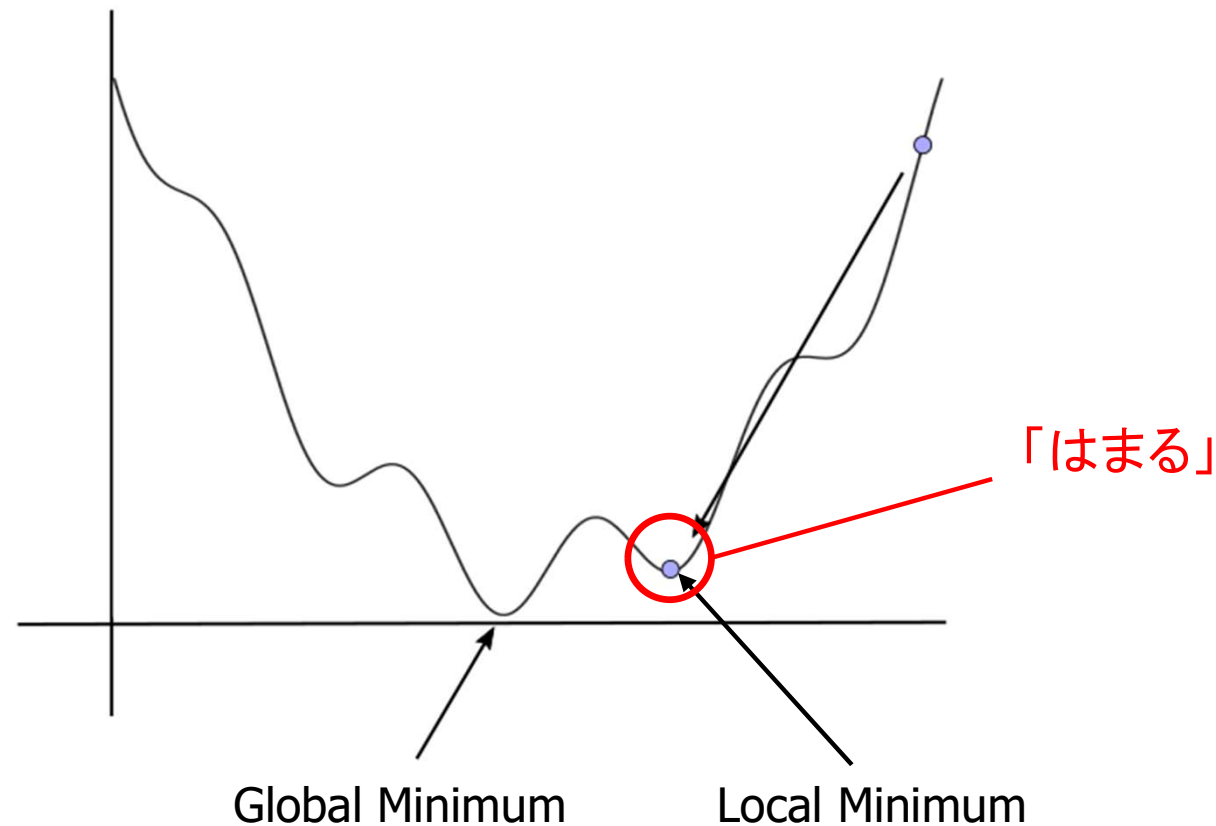
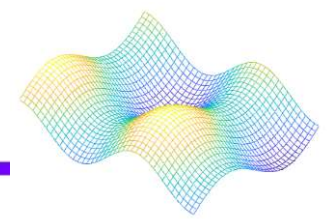


# Feedback-Based Quantum Optimization (FALQON)

フィードバックに基づく量子最適化

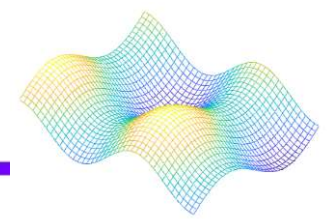


# なぜFALQONを使うのか？

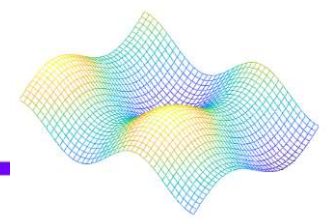


量子近似最適化アルゴリズム (QAOA) は、量子コンピュータで組合せ最適化問題を解くための最もよく知られたプロセスの1つですが、大きな欠点がある。最適化手順が局所最小値に「はまる」ことがあるため、収束が保証されない。

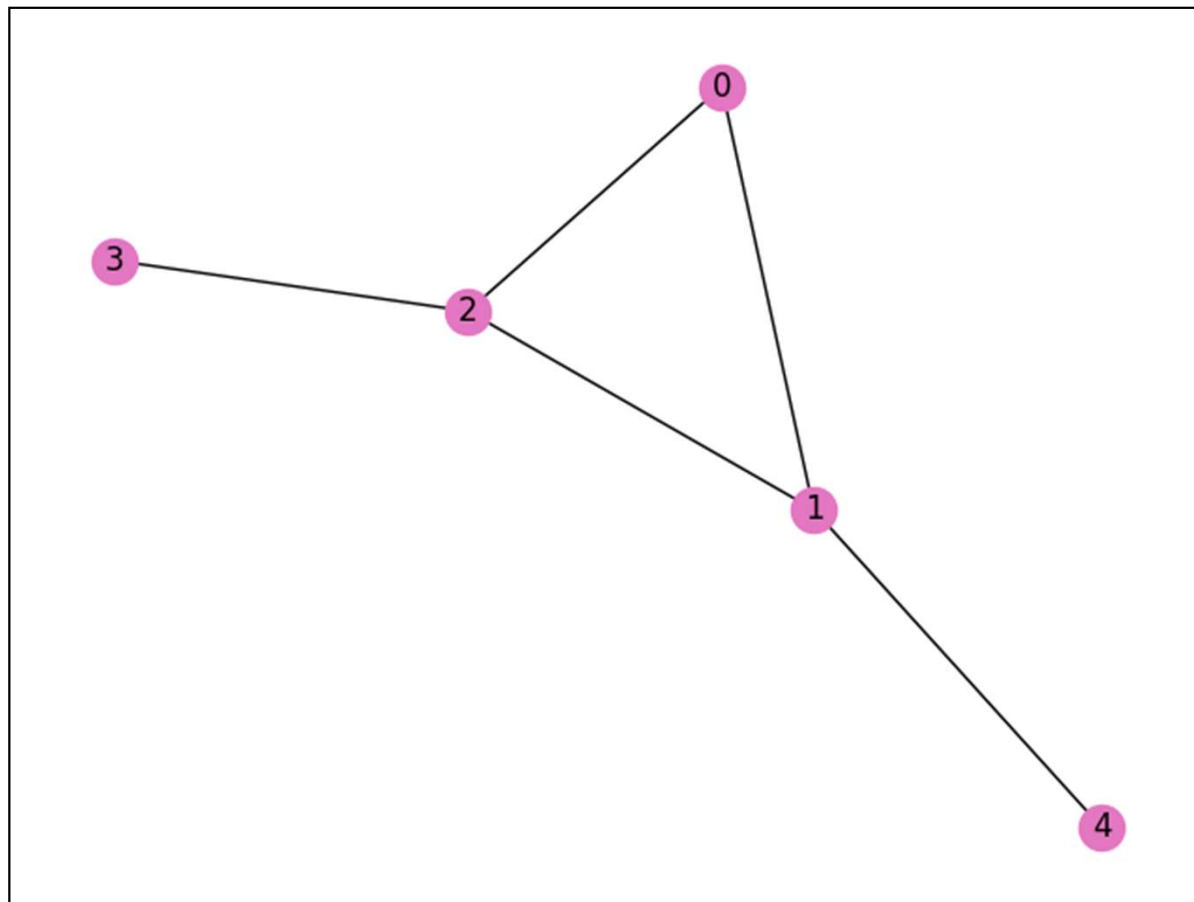




- 今回は、FALQONを実装してグラフ理論のMaxClique問題を解き、ベンチマークを行い、FALQONとQAOAを組み合わせて強力な最適化手順を作成します。
- QAOAとFALQONは似ているがパラメーターに対するグローバルな最適化ではなく、反復的なフィードバックを使用するのが特徴。

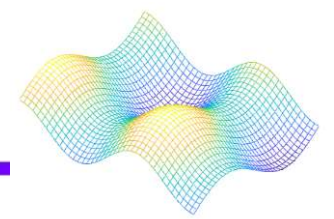


- グラフのMaxCliqueを見つけるのが目的となる。
- Cliqueとは、グラフ内の頂点の集合であり、MaxCliqueとはグラフ内で最も大きな頂点の集合。

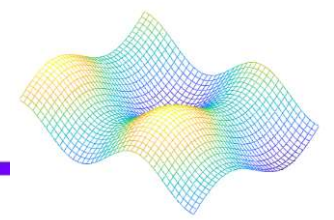


左図に適当な  
グラフを作成する

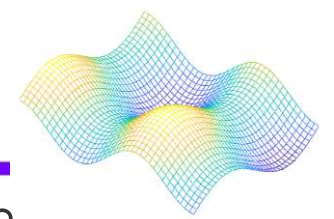
各頂点に色を付け  
番号を割り振る



- グラフをハミルトニアンにエンコードする。  
また、ハミルトニアン演算子の固有値と固有ベクトルは、システムのエネルギー固有値と対応する量子状態を与える。
- これらのハミルトニアンは、QAOA モジュールに入っているためそれを用いる。

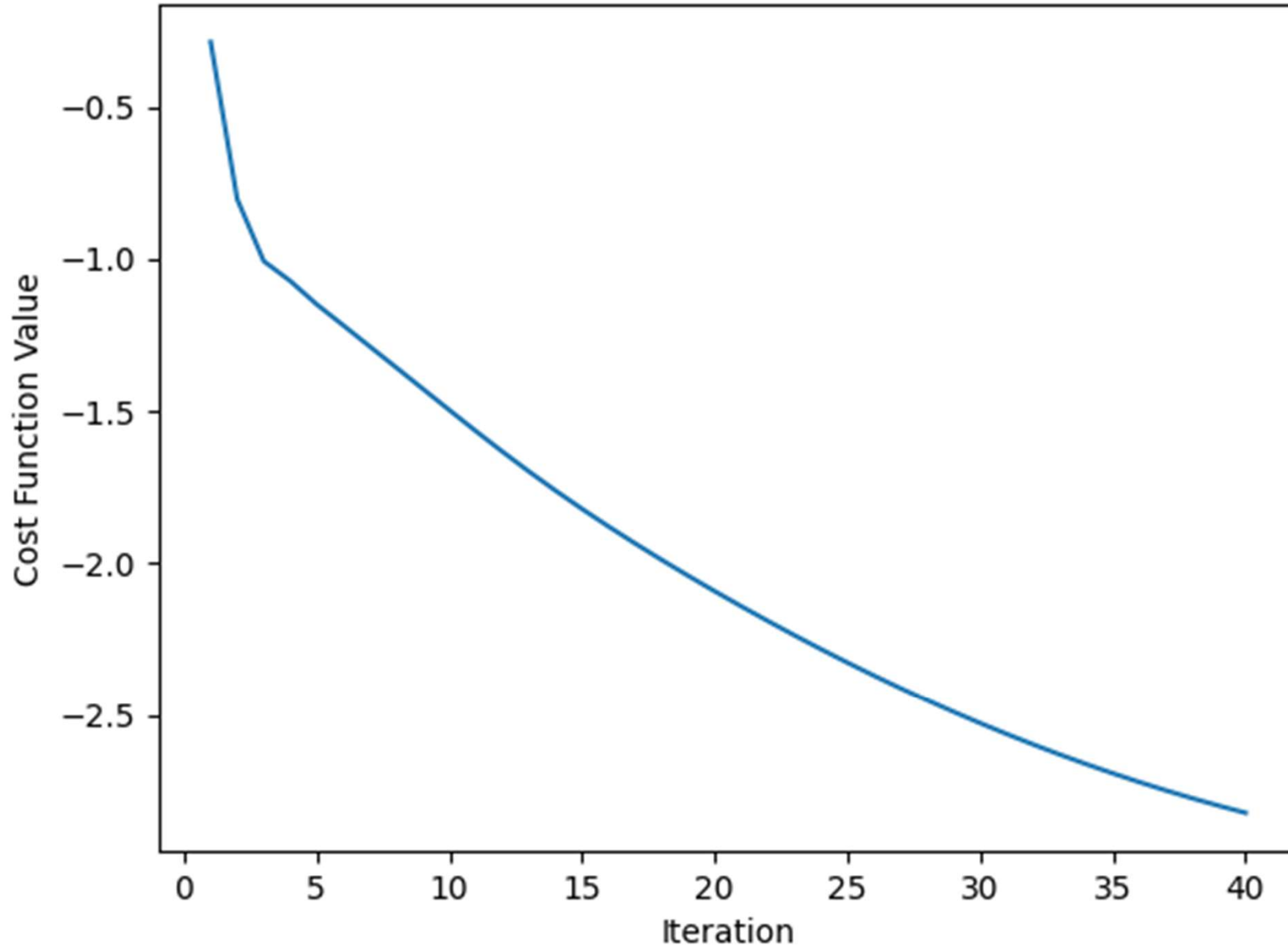
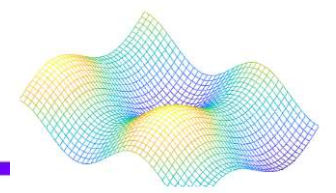


- ハミルトニアンの下でいくつかの初期状態を進化させることです。まず、Trotterized 時間発展の 1 つの層を定義します。
- 特定のコストハミルトニアン、ドライバー ハミルトニアン、およびドライバー ハミルトニアンに対応する FALQON アンザッツを返すメソッドを定義します。これには、上で定義した「FALQON 層」の複数の繰り返しが含まれます。回路の初期状態は均等な重ね合わせです。

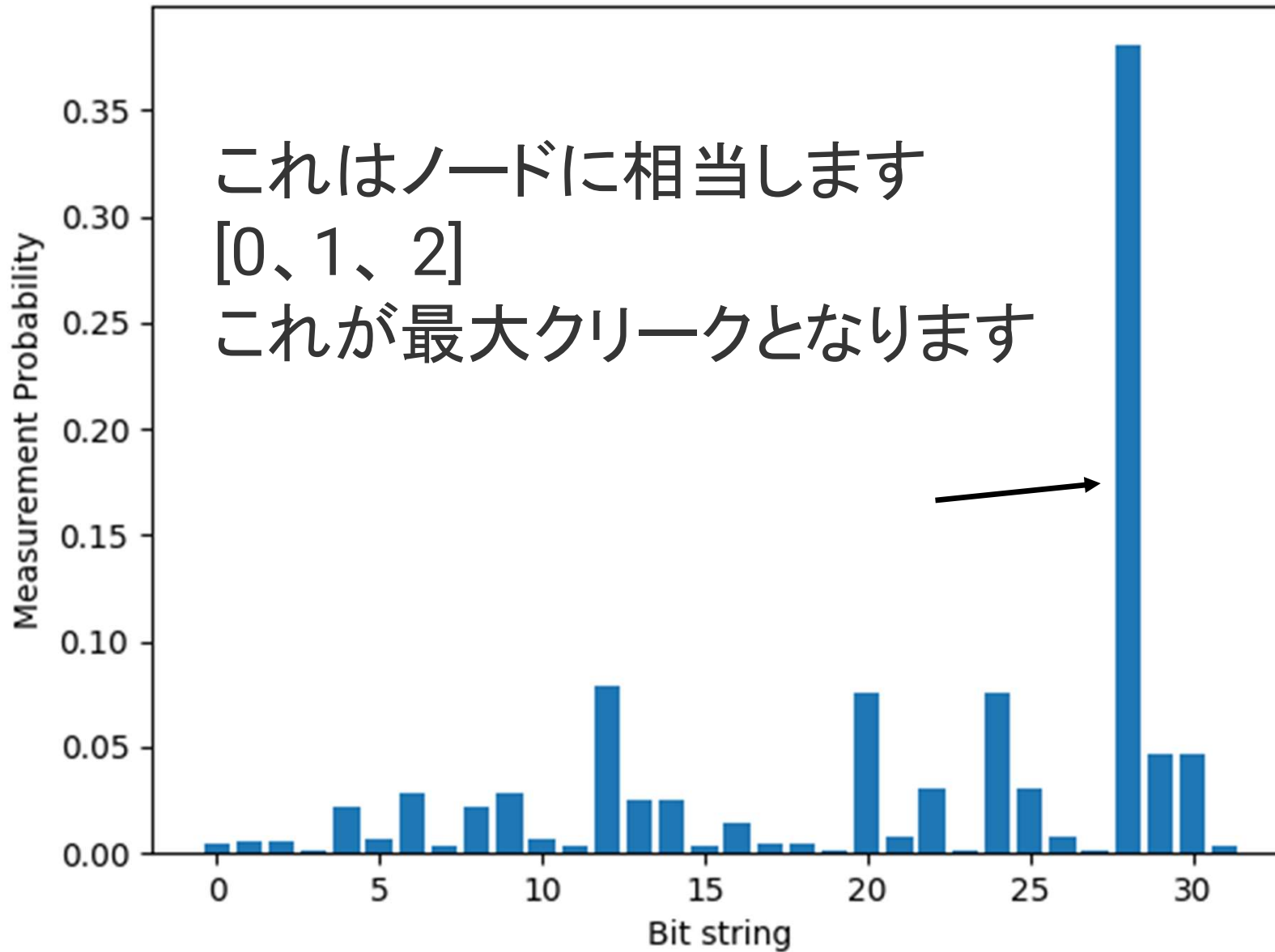
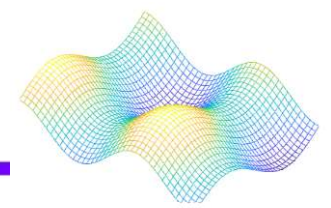


- FALQON が次の値を決定できる再帰プロセスを実装します。層の数が増えるにつれてそれ自体にフィードバックされます。
- MaxClique 問題に対して FALQON を実行できるようになりました。選択することが重要です。おおよその時間発展がリアルタイムMaxClique 問題に対して FALQON を実行できるようになりました。選択することが重要です  $\Delta t \Delta$   $\blacklozenge$  おおよその時間発展がリアルタイム発展に十分に近づくほど十分に小さい場合、そうでない場合は、次の期待値が得られます。

# 各ステップのコスト ハミルトニアンの期待値

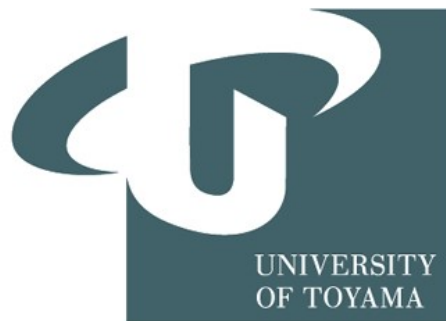


# 各ビット文字列の測定確率



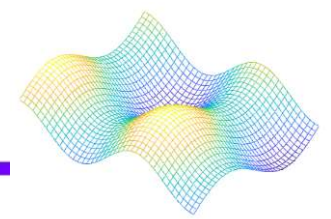
# Coherent Variational Quantum Linear Solver

## コヒーレント変分量子線形解法





# 問題点



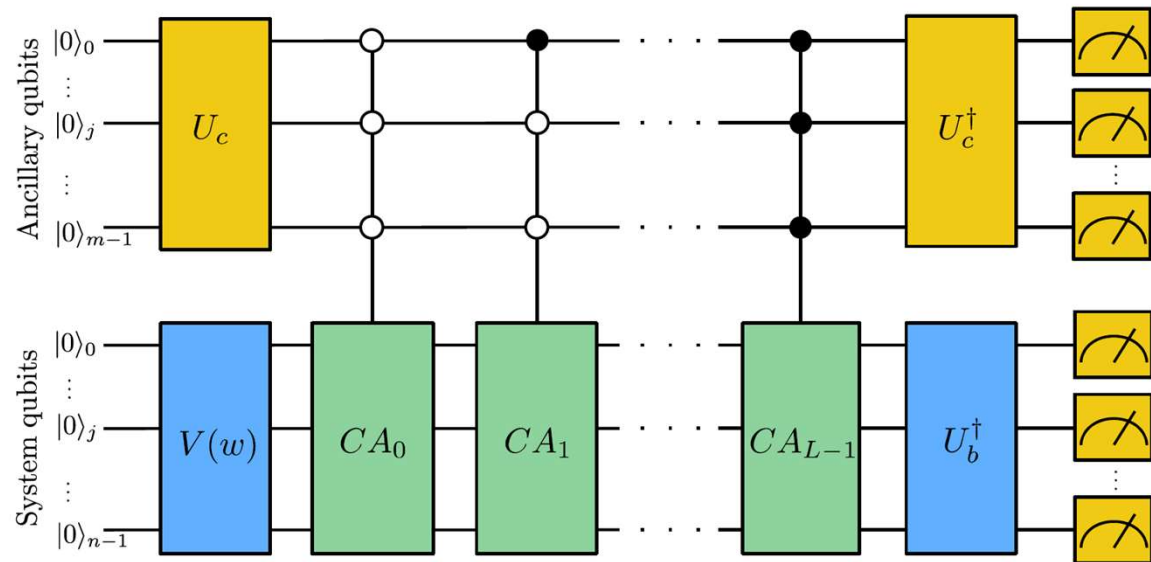
$2^n \times 2^n$ 行列  $A$ 、これは次の線形結合として表現できる  $L$  ユニタリ行列  $A_0, A_1, \dots, A_{L-1}$ ,

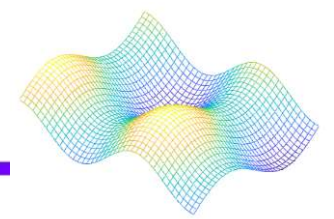
$$A = \sum_{l=0}^{L-1} c_l A_l,$$

ここで、 $c_l$  は任意の複素数。

各ユニタリ行列成分  $A_l$  に作用する量子回路で効率的に実装できると仮定する。

$n$ 量子ビットに作用する量子回路で効率に実装できると仮定する。





また、量子状態の物理的形式で正規化された複素ベクトルも与えられる。

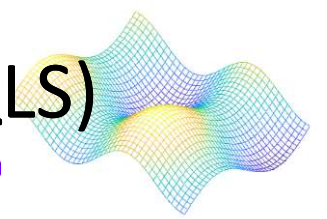
$|b\rangle$ 、単一操作で生成できます。 $U$ の基底状態に適用される  $n$ 量子ビット。、つまり、

$$|b\rangle = U_b|0\rangle,$$

最適化された量子状態を準備する手法がコヒーレント変分量子線形解法である。

$$|\Psi\rangle := \frac{A|x\rangle}{\sqrt{\langle x|A^\dagger A|x\rangle}} \approx |b\rangle.$$

# Coherent Variational Quantum Linear Solver (CVQLS)



変分量子回路で解を近似



ユニタリ回路を使用した場合  $V$  有限数の古典的な実パラメータに依存  
 $w = (w_0, w_1, \dots)$

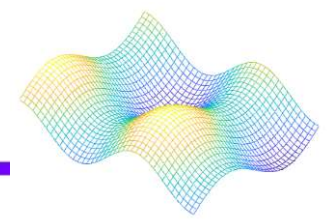
$$|x\rangle = V(w)|0\rangle.$$

量子状態  $|\Psi\rangle$  と  $|b\rangle$  の重複を最大化するにはパラメータを最適化する必要があるため次のコスト関数を定義する

$$C = 1 - |\langle b | \Psi \rangle|^2,$$

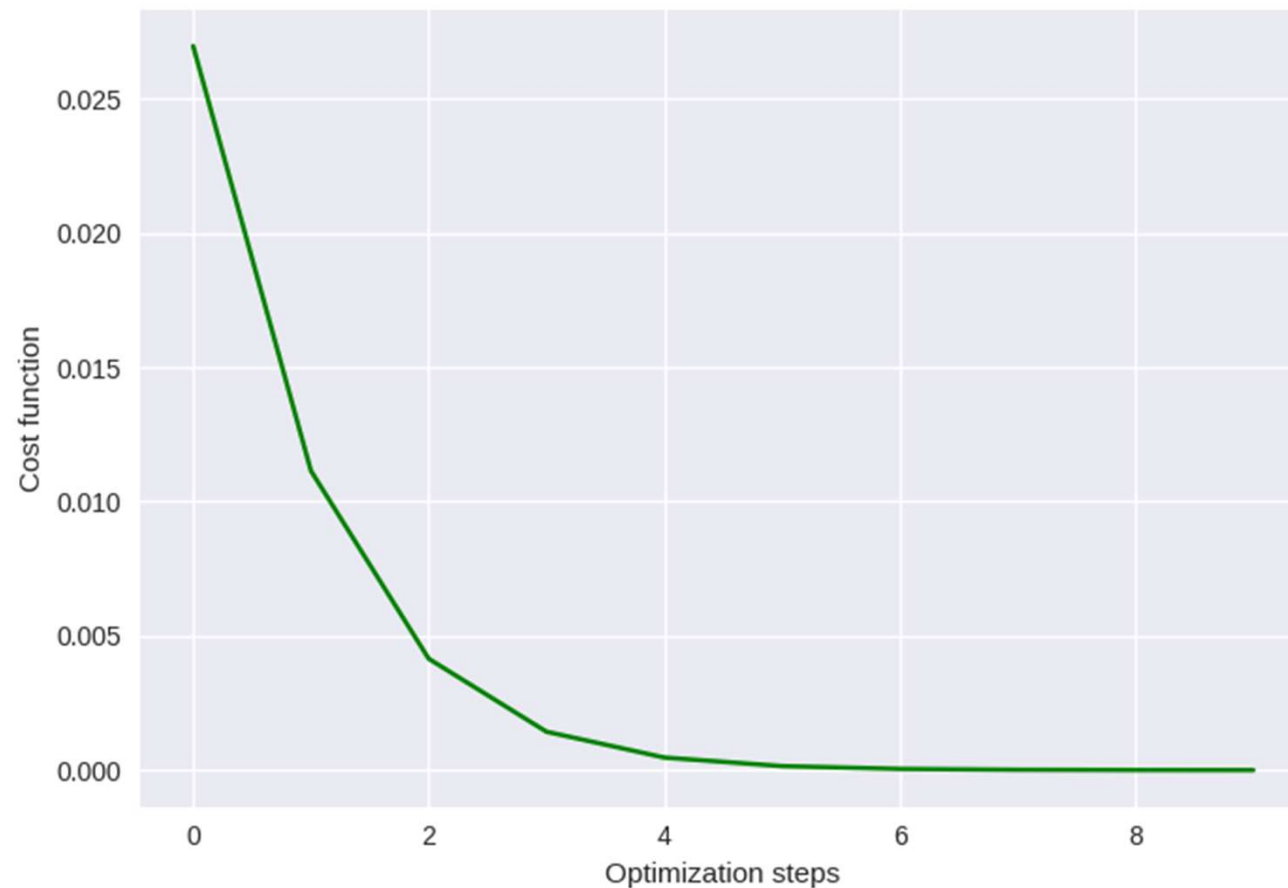
したがって、変分パラメータに関する最小化が  
問題の解決につながる

# 最適化

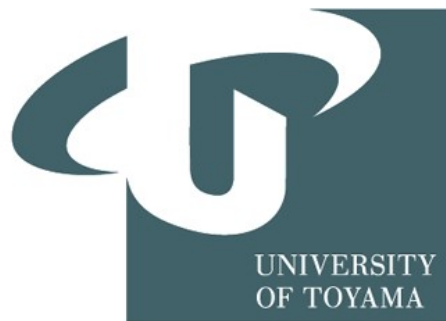


・定義したコスト関数 $C$ を以下の手順で最適化

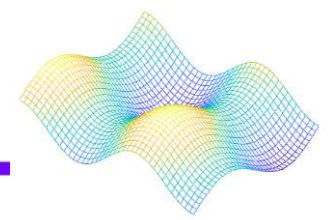
1. ベイズの定理を用いて期待値で表現
2. コスト関数を最小化するために勾配降下法を使用。
3. 最適化ループを行う。(今回はループ10回)



# QAOA for MaxCut



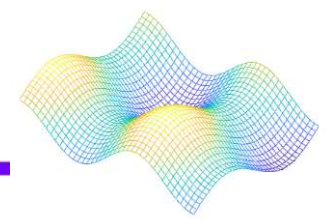
# はじめに



このチュートリアルでは、Farhi, Goldstone, and Gutmann (2014)が提案したMaxCut問題に対する量子近似最適化アルゴリズム(QAOA)を実装する。

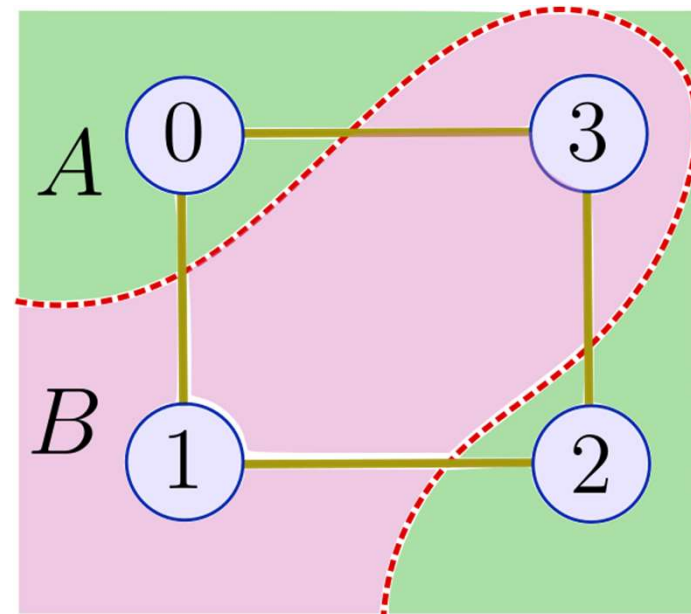
まず、4つの頂点と4つの辺を持つグラフという簡単な例を用いて、MaxCut問題の概要を説明する。

次に、PennyLaneを用いてQAOAアルゴリズムを実行し、最大カットを求める方法を示す。



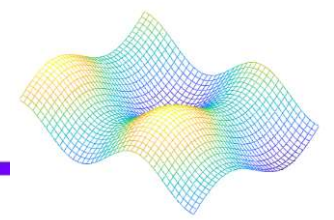
## ■ マックスカット問題

MaxCutの目的は、与えられた頂点(青い丸)の2つの集合への分割によって「カット」されるグラフの辺(黄色の線)の数を最大化することである(右図参照)。

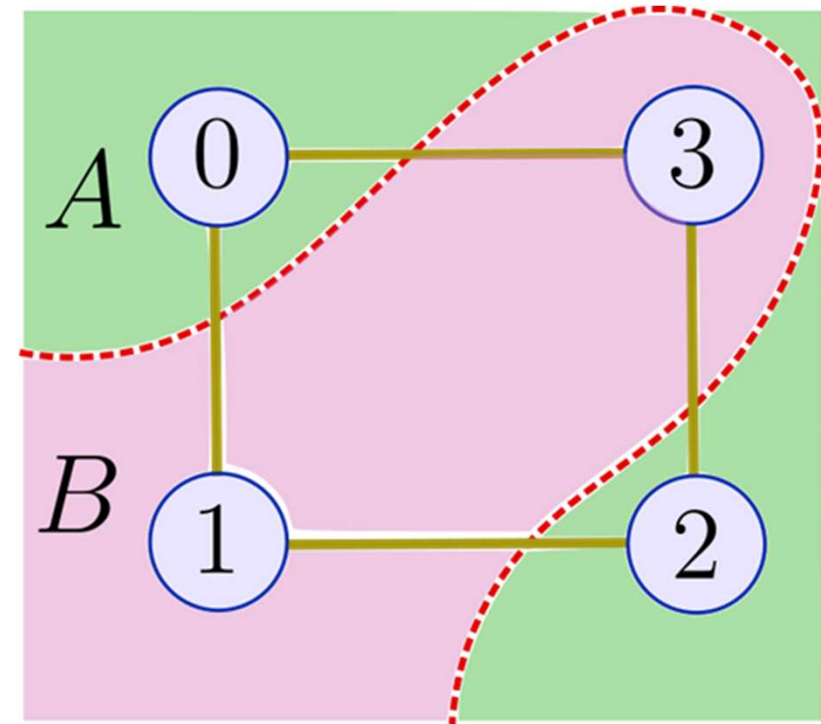


- $m$ 個の辺と $n$ 個の頂点を持つグラフを考える。頂点を2つの集合AとBに分割するとき、次の式を最大にする分割 $z$ を求める。

$$C(z) = \sum_{\alpha=1}^m C_{\alpha}(z),$$

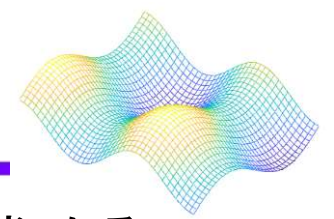


- 頂点の集合AまたはBへの割り当ては、ビット列 $z=z_1\dots z_n$ で表すことができる。ここで、 $i$ 番目の頂点がAにあれば $z_i=0$ 、Bにあれば $z_i=1$ である。
- 例えば、上の図に描かれている状況では、ビット列表現は $z=0101$ であり、0番目と2番目の頂点がAにあり、1番目と3番目の頂点がBにあることを示している。





# QAOA用回路

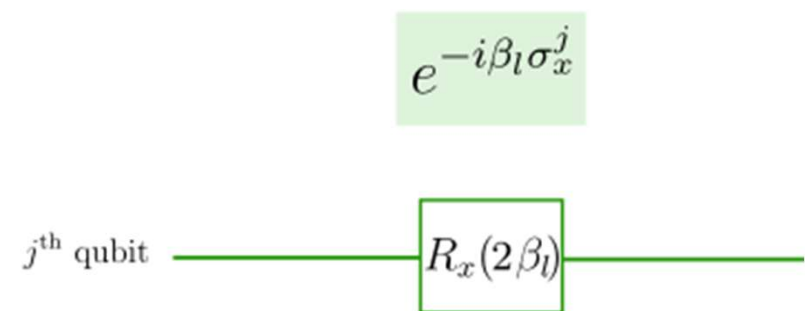
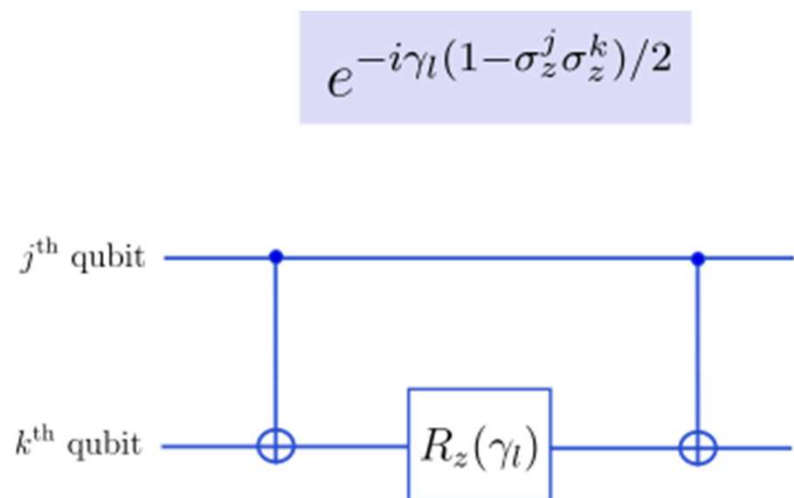


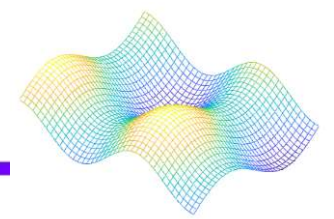
- 基本的なユニタリーゲートを用いて、MaxCut問題の近似解を求めるQAOA回路の実装について述べる。

$$U_{B_l} = e^{-i\beta_l B} = \prod_{j=1}^n e^{-i\beta_l \sigma_x^j},$$

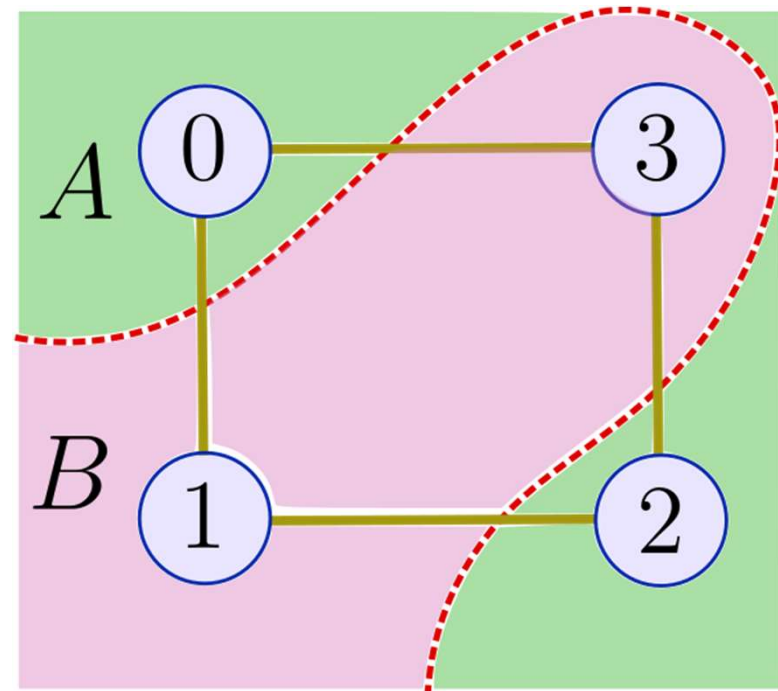
$$U_{C_l} = e^{-i\gamma_l C} = \prod_{\text{edge (j,k)}} e^{-i\gamma_l (1 - \sigma_z^j \sigma_z^k)/2}.$$

- これらは、パラメータに吸収される無関係な定数まで、以下に示すゲートを用いて量子回路に実装することができる。
- C演算子に大きな値が得られる可能性が高い重ね合わせのビット列状態の空間を探索することを目的とする。

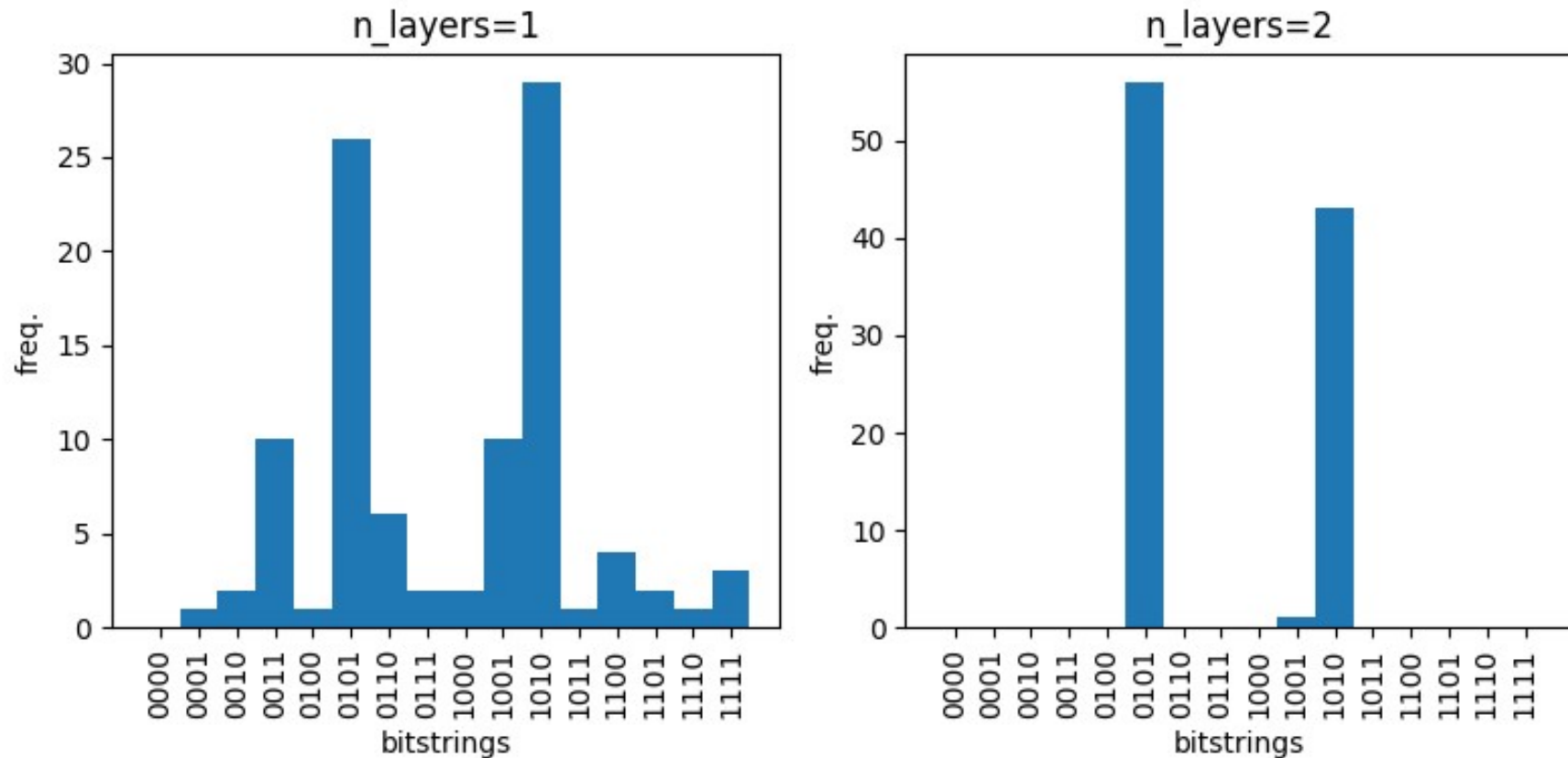
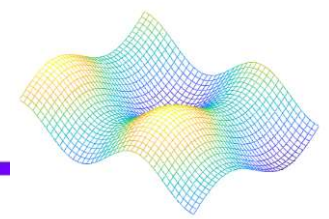




- 目的演算子の期待値を  $\langle \gamma, \beta | C | \gamma, \beta \rangle$  とする。
- PennyLaneを用いて回路パラメータ  $(\gamma, \beta)$  に対する古典的最適化を行う。これによって、最適な分割が得られる可能性が高い状態の期待値が指定されます。
- グラフの場合、0101か1010のどちらかを測定したい。



# PennyLaneにQAOAを実装する

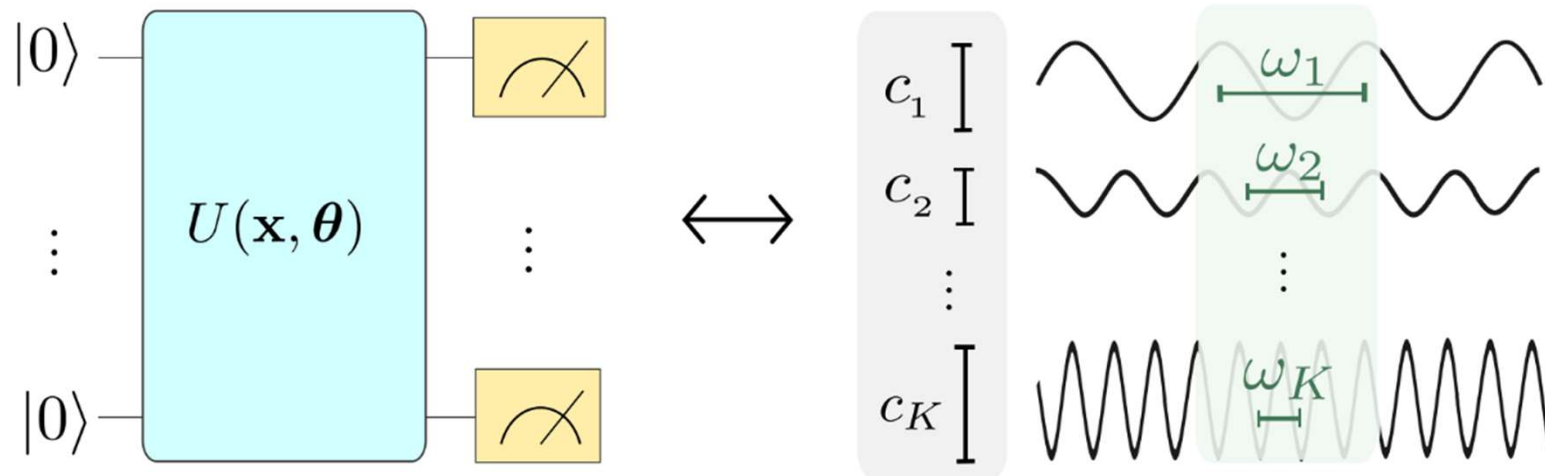
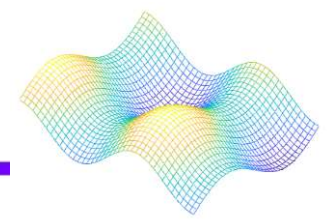


最後に、`n_layers` を用いてレイヤー数 (UBUC の繰り返し適用) を指定します。角度パラメータ  $\gamma$  と  $\beta$  に対して最適化し、最適化された回路を複数回サンプリングしてビット列の分布を得る。最適な分割 ( $z=0101$  または  $z=1010$ ) のいずれかが、最も頻繁にサンプリングされるビット列になった。

Quantum models as Fourier series

量子モデルとしてのフーリエ級数



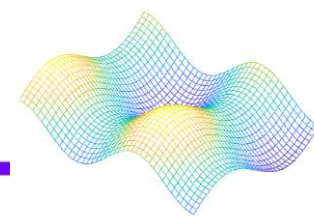


## 量子モデルとフーリエ級数の関係とは？

参考文献：

*The effect of data encoding on the expressive power of variational quantum machine learning models* by [Schuld, Sweke, and Meyer \(2020\)](#)

# 背景



量子機械学習モデル

$$f_{\theta}(x) = \langle 0 | U^{\dagger}(x, \theta) M U(x, \theta) | 0 \rangle$$

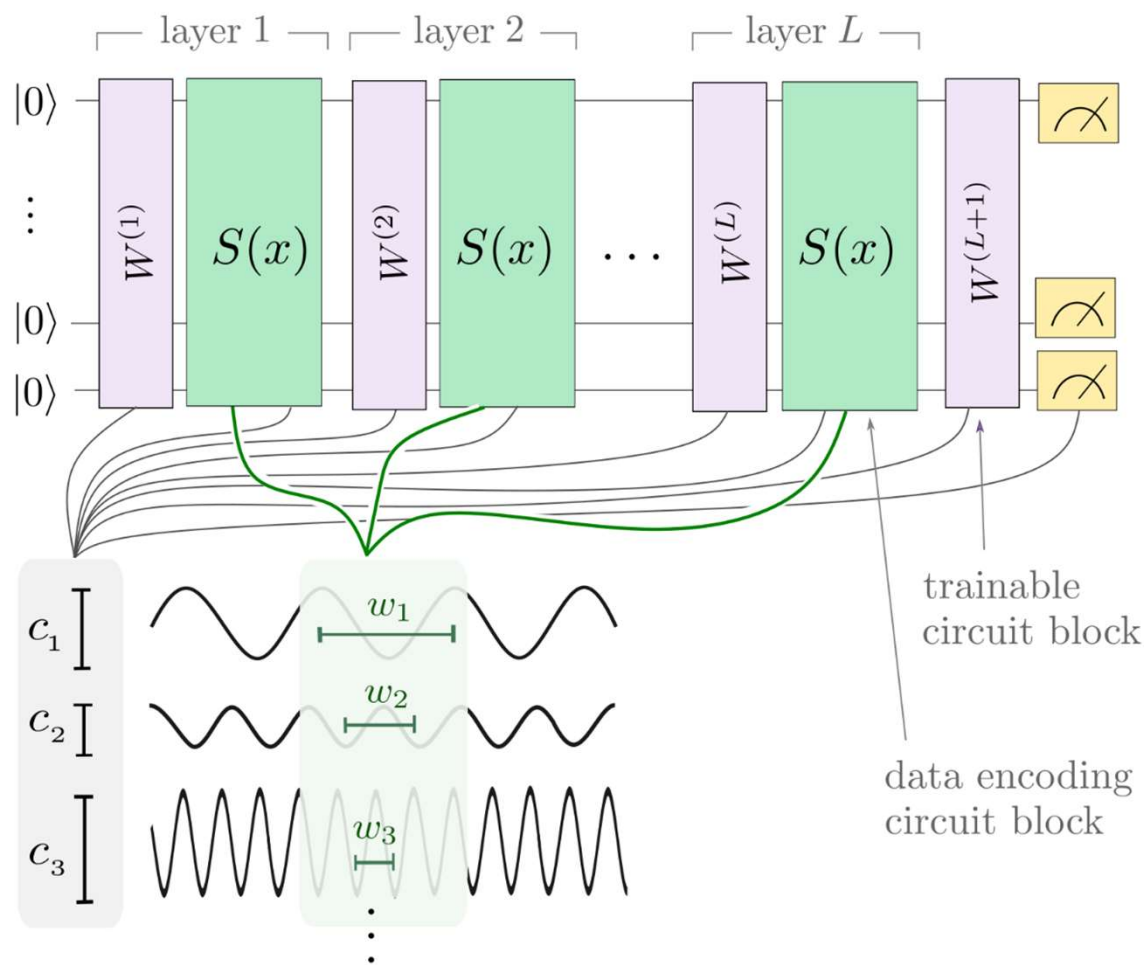
エンコーディング回路

$$S(x) = \mathcal{G}(x) = e^{-ixH}$$

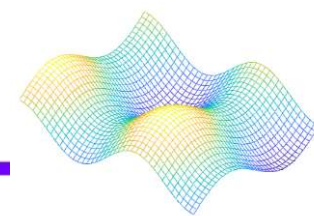
→ポアーリ回転ゲート  
など

フーリエ級数

$$f_{\theta}(x) = \sum_{\omega \in \Omega} c_{\omega}(\theta) e^{i\omega x}$$

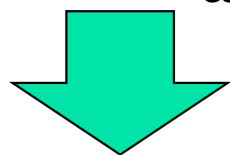


# 背景



フーリエ級数

$$f_{\theta}(x) = \sum_{\omega \in \Omega} c_{\omega}(\theta) e^{i\omega x}$$

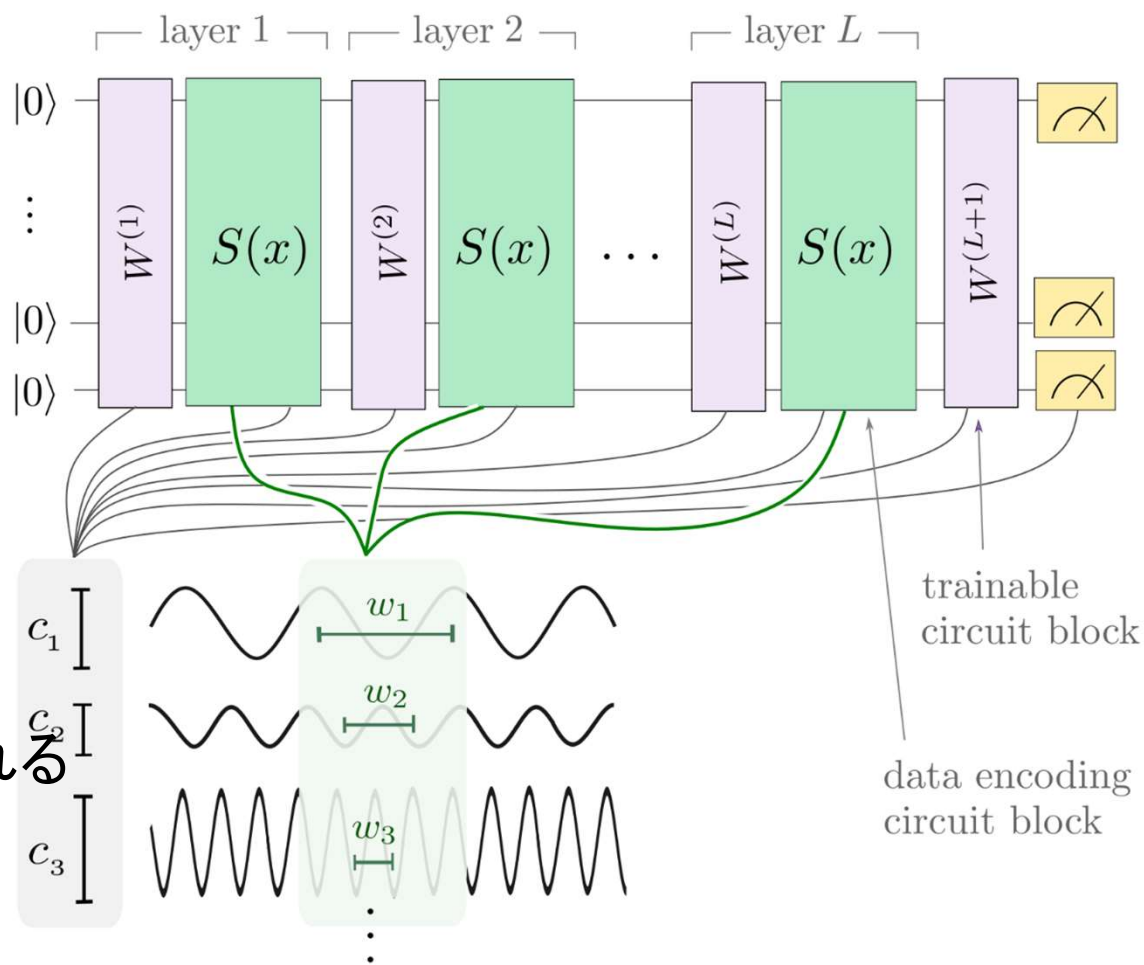


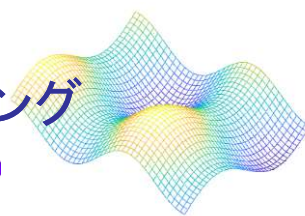
表現できる  
フーリエ級数は

$$f_{\theta}(x) = \sum_{n \in \Omega} c_n(\theta) e^{inx}$$

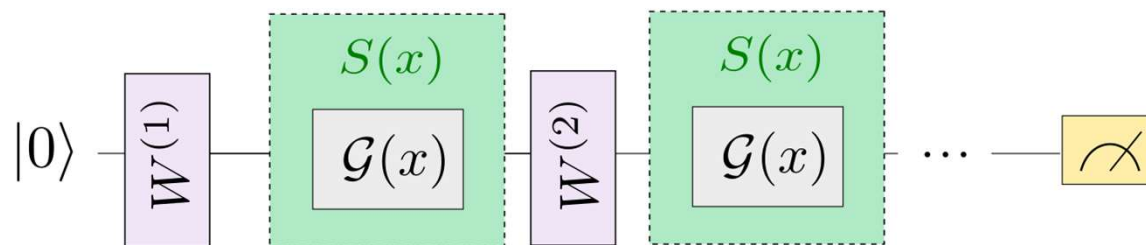
$$\Omega = \{-r, \dots, -1, 0, 1, \dots, r\}$$

$r$ はエンコーディングゲート数(層数)  
または量子ビット数によって決定される

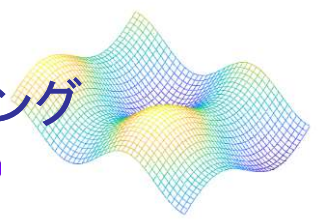




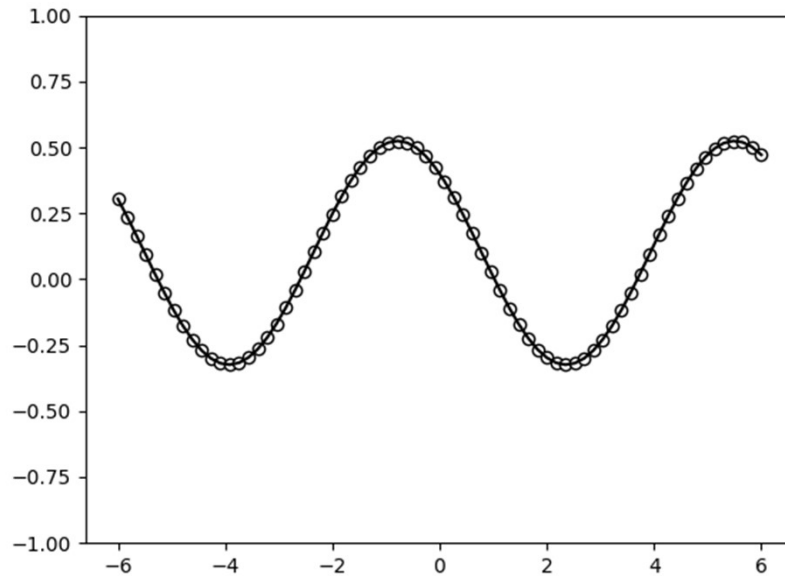
量子ビットを1つで  
エンコーディングゲートをr回「連続」行う





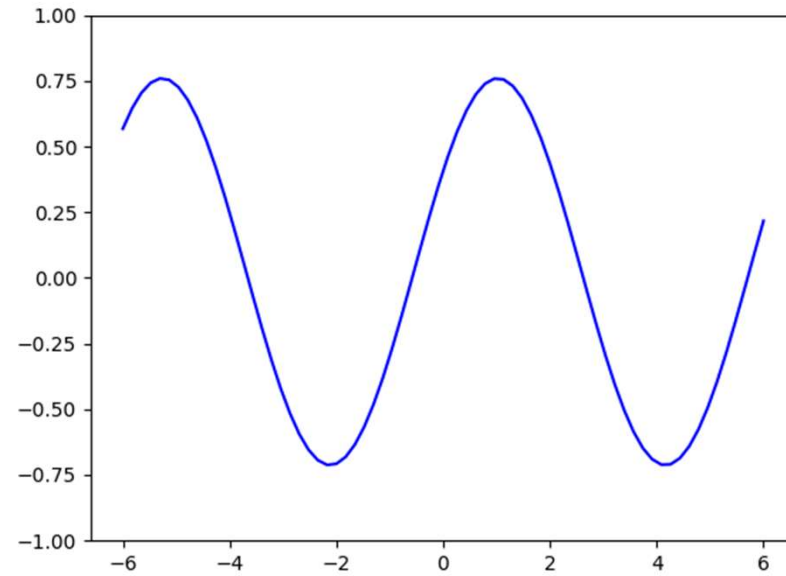


## 1. 目的関数の作成

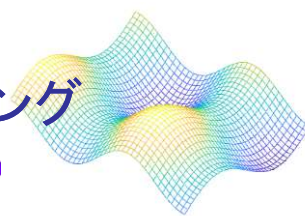


特定の次数のフーリエ級数を作成する

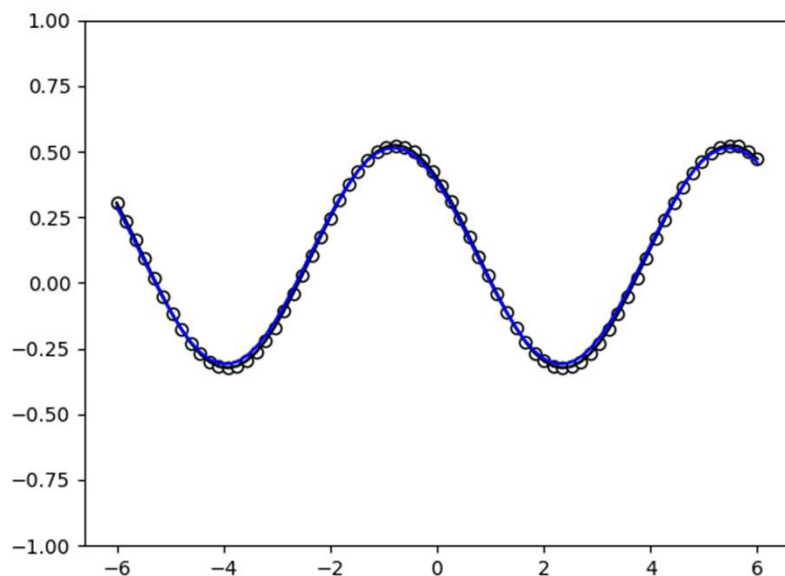
## 2. 量子モデルの作成



各点(量子モデル)における重みはランダムに選定されるが、同じ周波数の関数の値となる

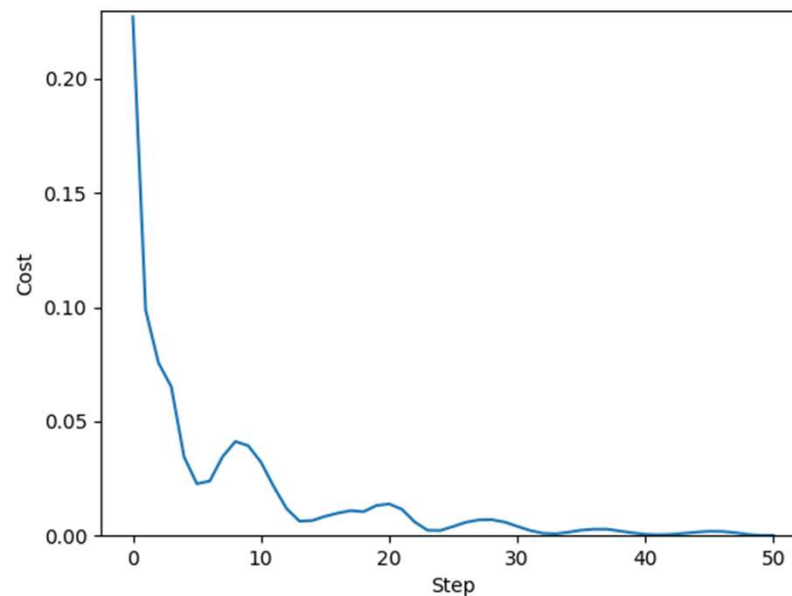


## 3. フィッティング結果



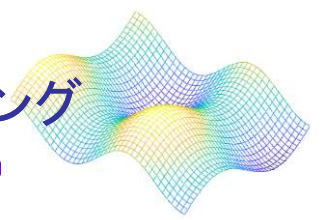
量子モデルを機械学習で真の値に近づくように重みを最適化させる

## 4. 損失

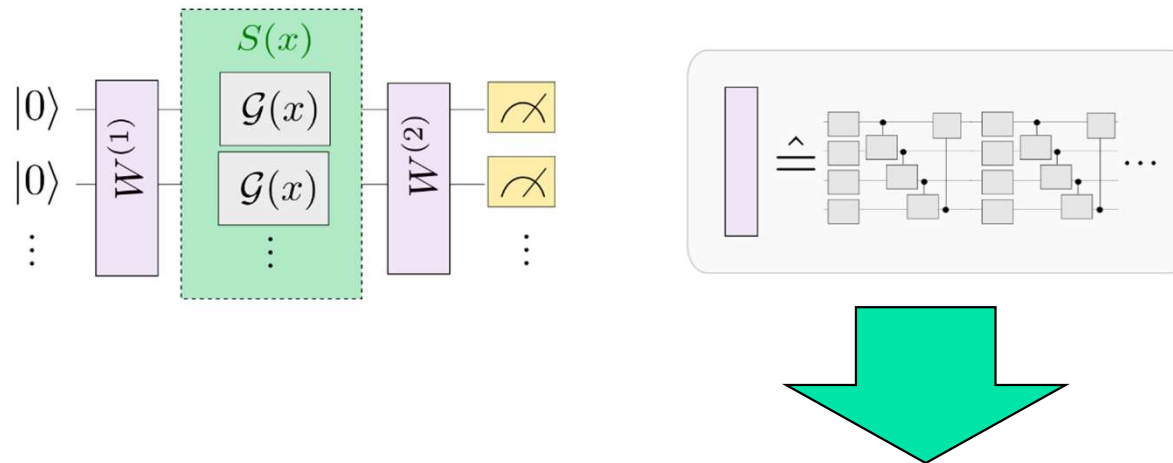


損失 = 真の値と量子モデルの差

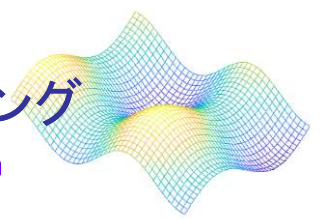
今回は $r=L=1$ で簡単であり  
 $L$ が多くなると多くのステップがかかる



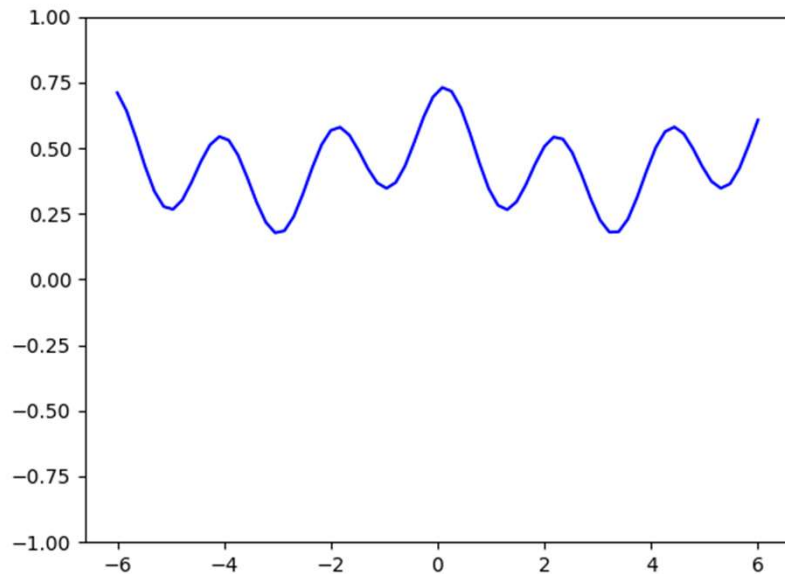
異なるキュビットで「並列」(L=1)に繰り返す



Wにはユーザーが定義することができるansatzを導入、PennyLaneの層構造のひとつである「StronglyEntanglingLayers」

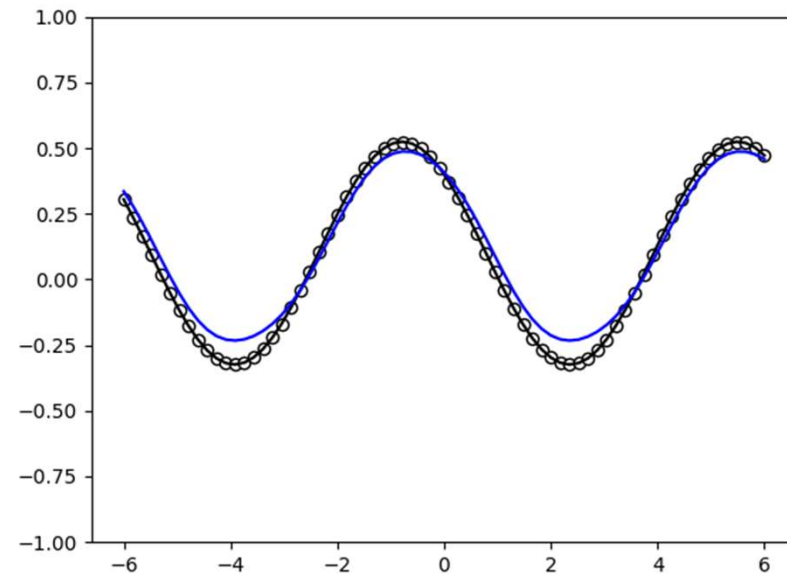


## 1. 量子モデルの作成

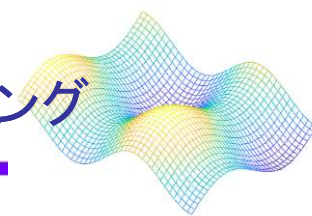


ゲートが2回繰り返される( $r=2$ )ことで量子モデルの表現できるフーリエ級数の次数が増える

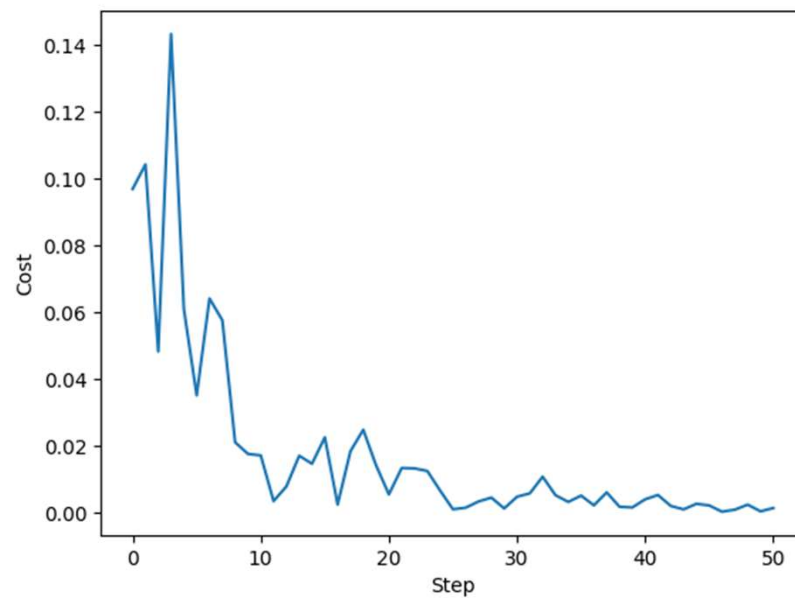
## 2. フィッティング結果



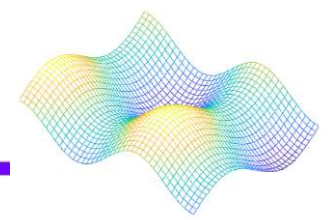
量子モデルを機械学習で真の値に近づくように重みを最適化させる



## 3. 損失



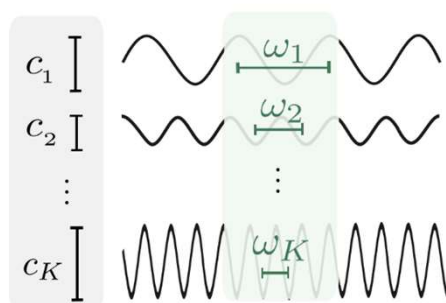
# 実験3: フーリエ係数のサンプリング



どんなに学習しても真の値に近づかない場合がある

作成した量子モデルによって表現できるフーリエ級数が異なる

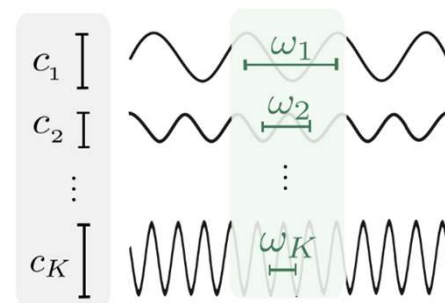
量子モデルA



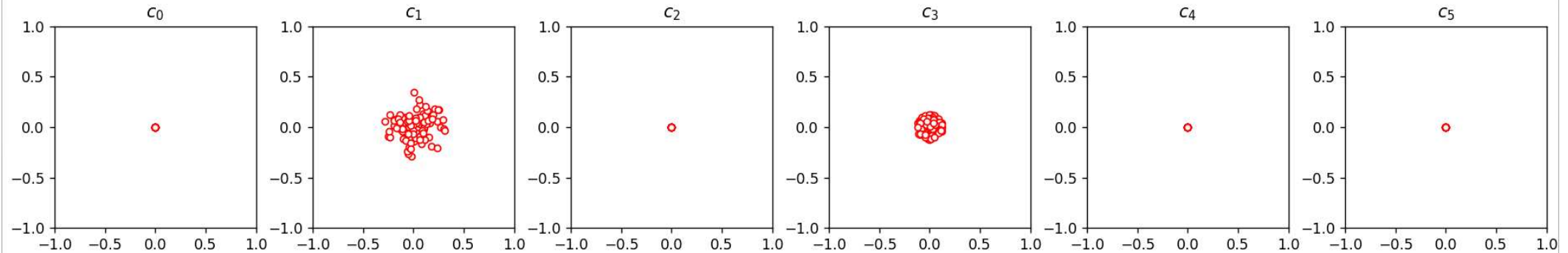
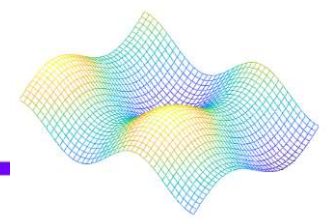
使用した層で  
値が違う!!



量子モデルB



# 実験3: フーリエ係数のサンプリング



$S(x)$ の層  $\rightarrow$  BasicEntanglingLayers

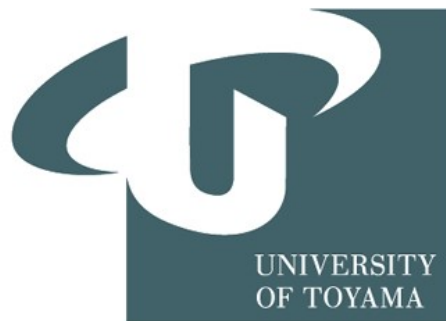
- ・偶数次のフーリエ係数の分散が0
- ・係数の次数が増えると分散が減少

計算する層によって特性が異なる

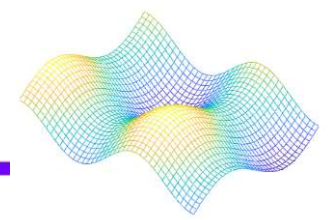
$\rightarrow$  実験2のStronglyEntanglingLayersでは全ての級数に対して値の分散が見られる

Training and evaluating quantum kernels

量子カーネルの訓練と評価

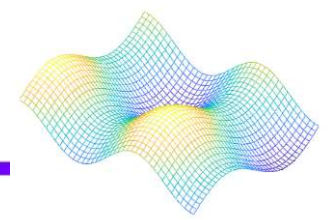




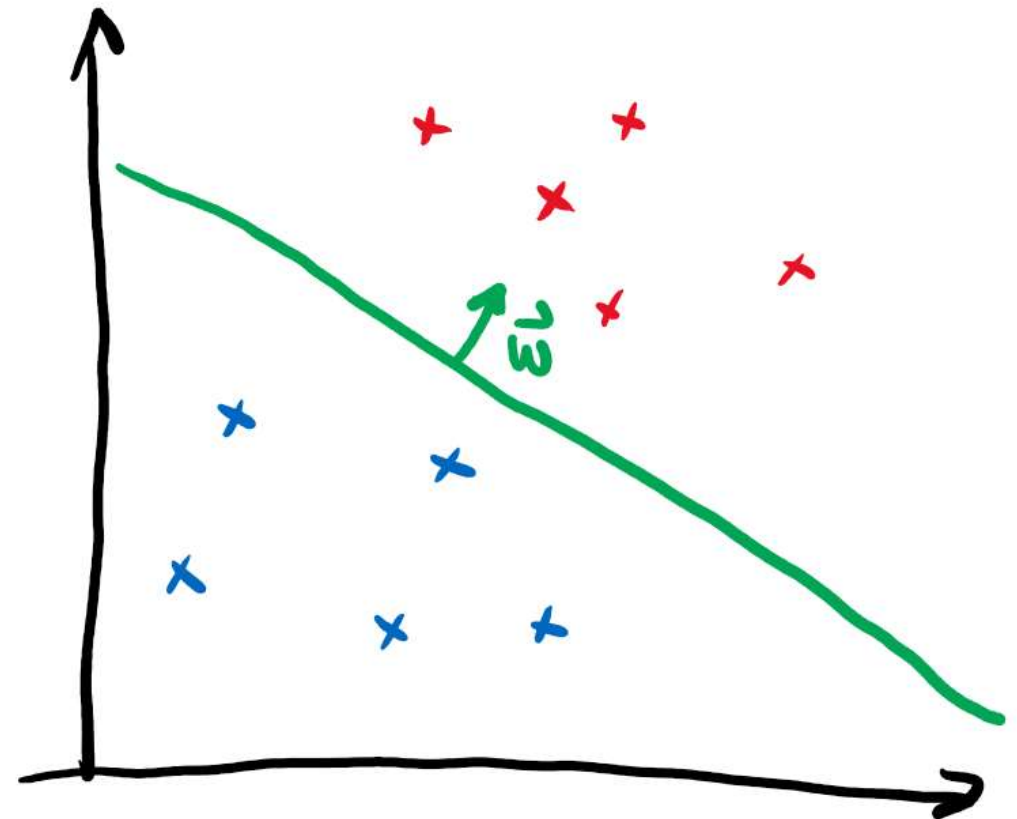


量子カーネルとは、量子コンピューターで評価できるカーネルのこと、略して量子カーネルと呼ぶ。

また、カーネルとはOSの中核となるソフトウェア。動作中のプログラムの実行状態を管理したり、ハードウェア資源を管理してプログラムがハードウェアの機能を利用する手段を提供したりするもののこと。

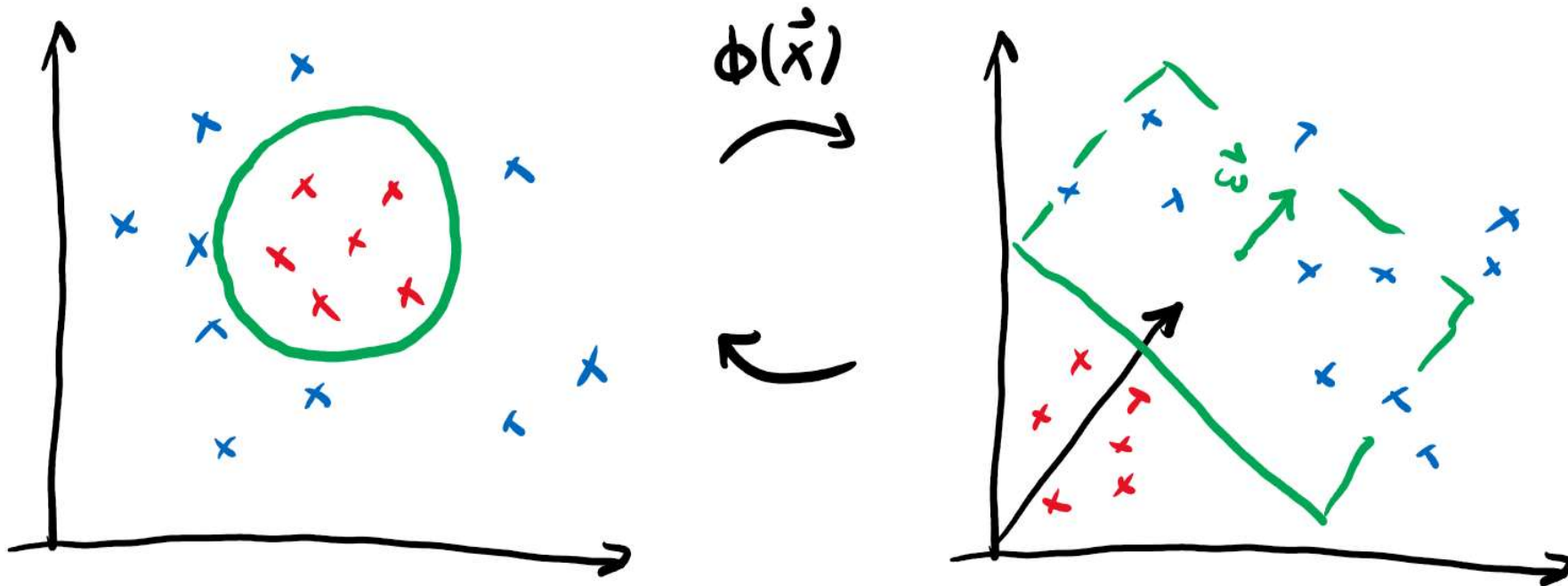
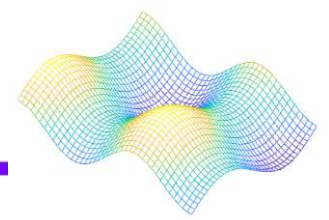


線を引き線の反対側の領域に異なるラベルを割り当てることによりこれを数学的に形式化できる。

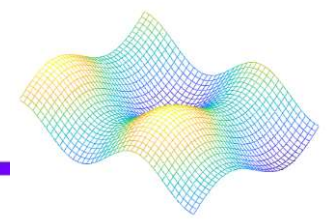


$$y(x) = \text{sgn}(\langle w, x \rangle +$$

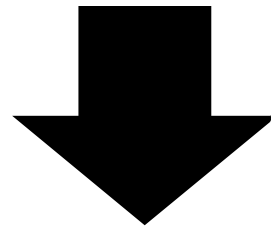
# 線形分類



$$y(x) = \text{sgn}(\langle w, \phi(x) \rangle + b)$$

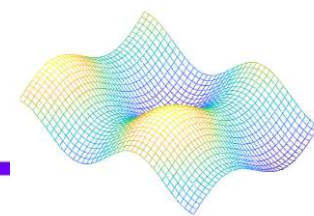


$$y(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b).$$



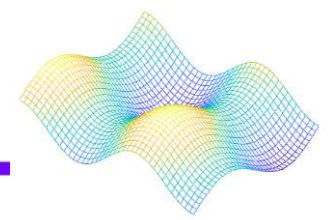
$\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i)$  を代入

$$y(\mathbf{x}) = \text{sgn}\left(\sum_i \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b\right).$$



$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle.$$

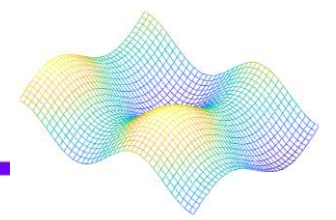
この関数をカーネルと呼び、この関数を用いることで、実際に埋め込みを実行する必要がなくなる。



$$\phi((x_1, x_2)) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$k(\mathbf{x}, \mathbf{y}) = x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 = \langle \mathbf{x}, \mathbf{y} \rangle^2.$$

カーネルの式に置き換えることで、より複雑な決定境界を示すことが可能になる。  
カーネルを用いることで、より簡単に計算を行なうことができる。



## Quantum Embedding Kernels (QEKs)

を導き出す。

これは量子状態の空間にデータを入力することで導き出されるカーネルである。

パラメータ化した量子回路 $U(\mathbf{x})$ を定式化する。

定式化

$$|\psi(\mathbf{x})\rangle = U(\mathbf{x})|0\rangle.$$

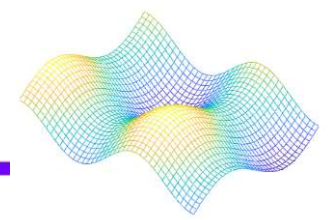
カーネルの式

$$k(\mathbf{x}_i, \mathbf{x}_j) = |\langle \psi(\mathbf{x}_i) | \psi(\mathbf{x}_j) \rangle|^2.$$

# The Quantum Graph Recurrent Neural Network







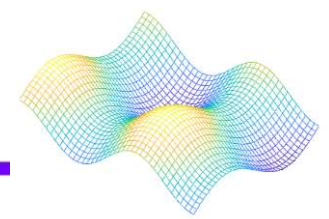
近年、グラフニューラルネットワーク(GNN)の概念が注目を集めている。

GNNは、ノードとエッジに割り当てられた特徴・グラフ全体のトポロジーを解析して、グラフの表現を学習しようとする。

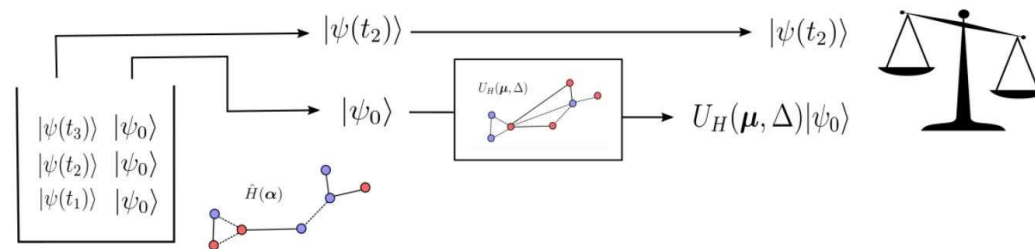
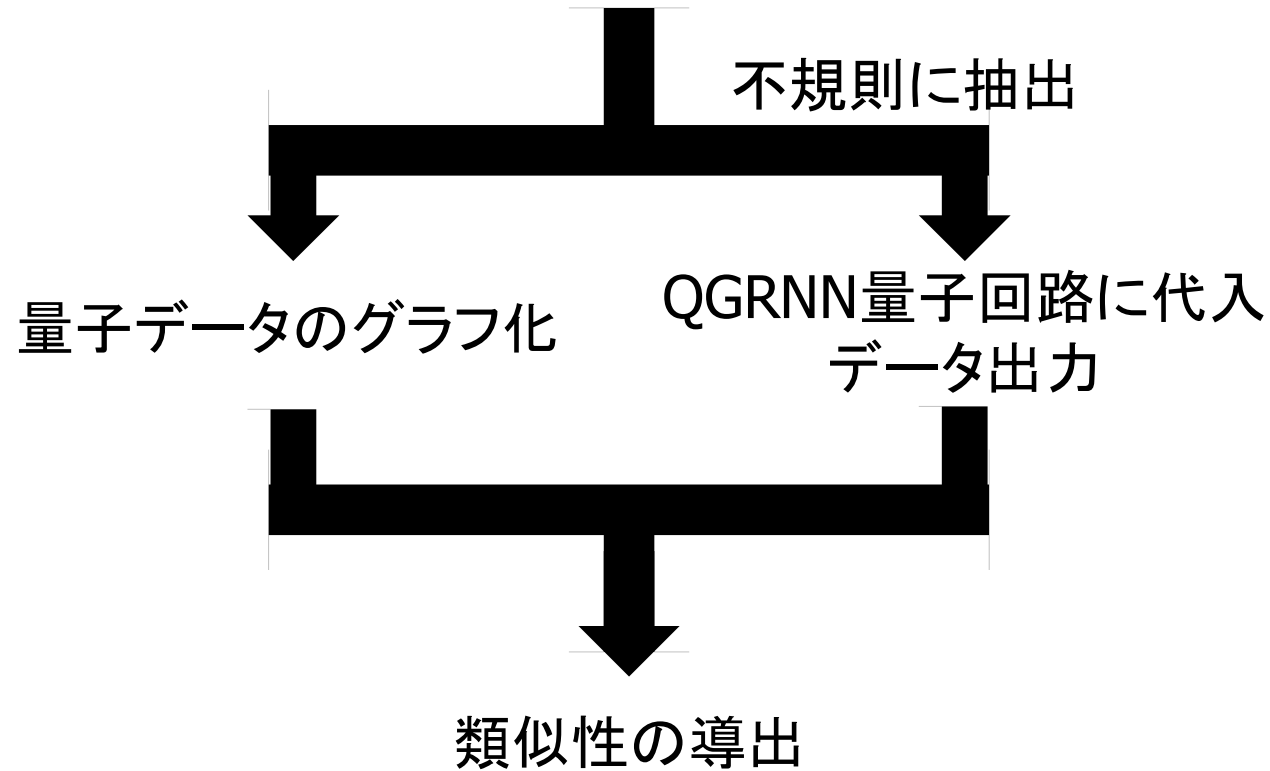


量子学的な機能を持ち合わせたものが量子グラフニューラルネットワーク(QGRNN)

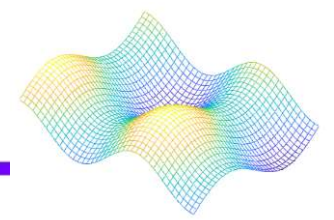
# QGRNNの使用



低エネルギー状態と時間変化した状態のデータ集合

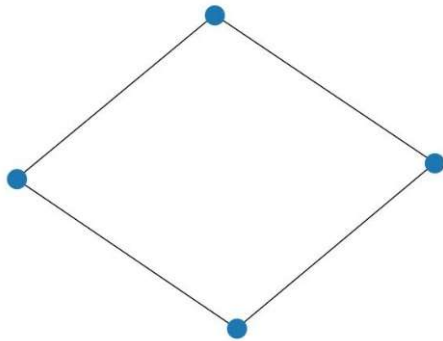


A visual representation of one execution of the QGRNN for one piece of quantum data.

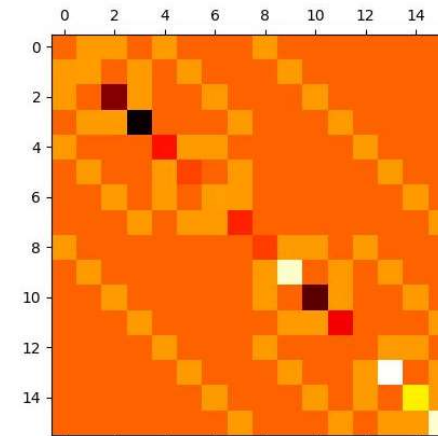


この実験では、QGRNNにすぐに代入できる量子データがないため、自分で生成する必要がある。

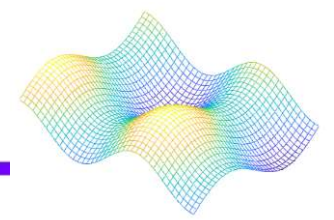
## ノードとエッジの循環グラフ



## ハミルトンのグラフ化



- モジュールのnetworkxを使用してデータを生成する。
- -2から2の小数第2位までの範囲でパラメータを選択し、ハミルトニアンのマトリクス図を生成する。



Variational Quantum Eigensolver algorithm (VQE)を使用してハミルトニアンの低エネルギー状態データを定義する。

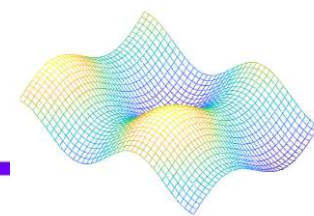


ハミルトニアンの最小固有値を低エネルギー状態値と比較し、設定した低エネルギーの状態値を見つける。

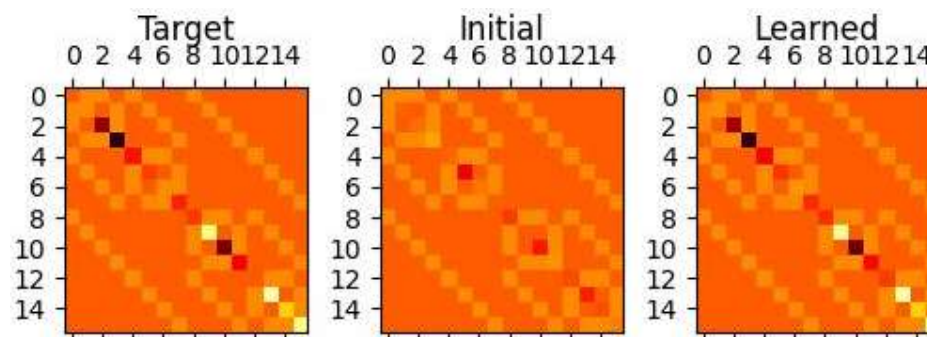
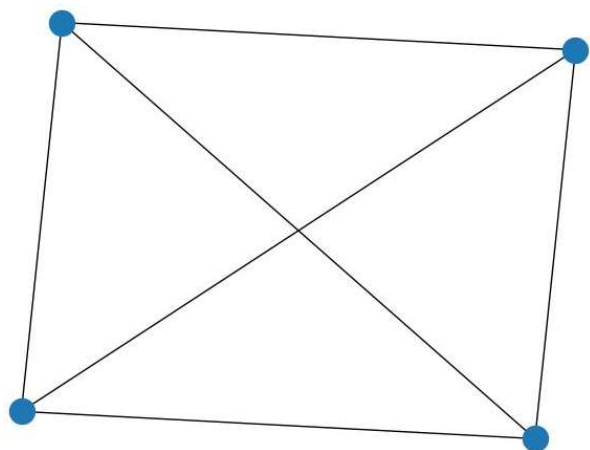


初期状態値に時間変化単位を乗算し、時間変化した状態値を出力する。

# QGRNNでの導出結果



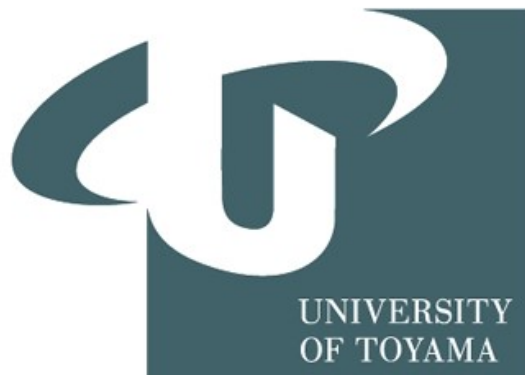
低エネルギー状態値と時間変化状態値の量子データをQGRNN回路に代入し、グラフを出力する。

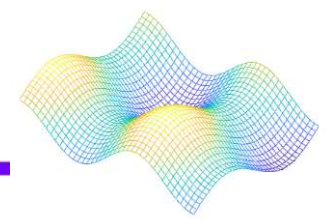


これらのグラフからハミルトニアンデータを学習し、似た画像を出力することができること示された。

# Learning to learn with quantum neural networks

## 量子ニューラルネットワークを使って学習する





## 変分量子アルゴリズム(VQA)

固定形状と調整可能なパラメーターを備えた量子回路を利用する  
⇒変分量子回路の調整可能なパラメータは、量子アルゴリズムの性能を測定するコスト(または損失)関数を最小化することによって、繰り返し最適化される

### VQAの課題

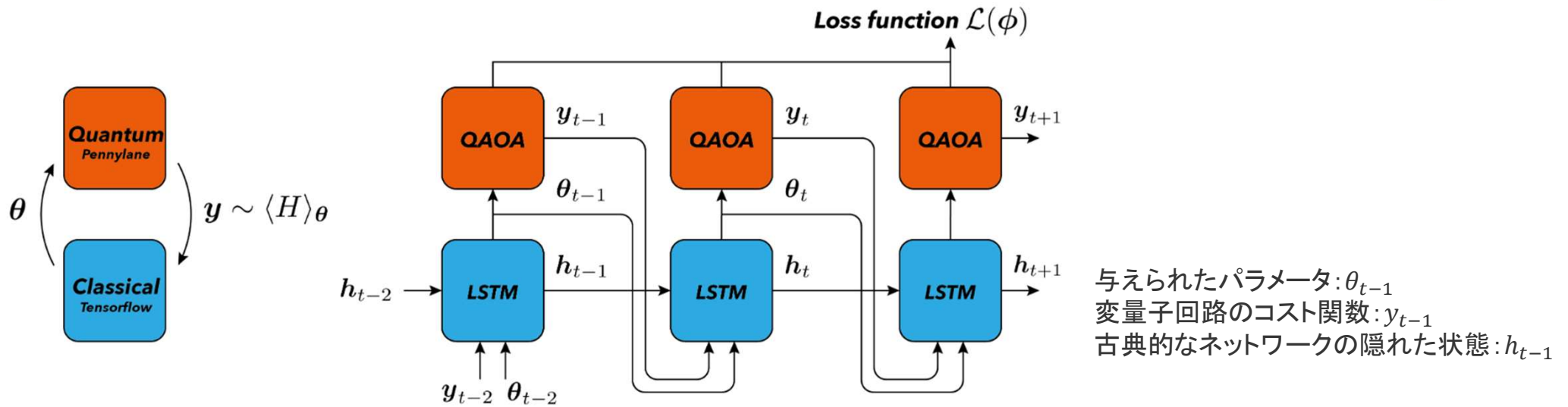
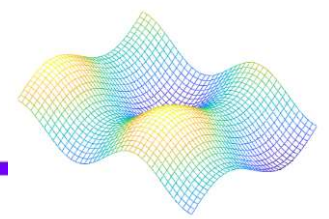
調整可能なパラメータの最適化が非常に困難

### 解決策

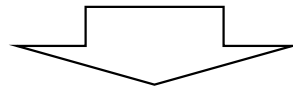
## リカレント・ニューラル・ネットワーク(RNN)

変分量子アルゴリズムのパラメータを最適化するブラックボックス制御装置

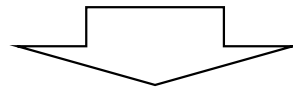
# 古典的なりかレントニューラル ネットワーク(RNN)



$\theta_{t-1} \cdot y_{t-1} \cdot h_{t-1}$  からRNNは新しい推測を提案される  $\Rightarrow \theta_t$



$\theta_t$  からコスト関数を評価するために量子コンピューターに入力される  $\Rightarrow y_t$

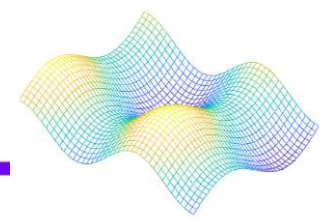


このサイクルを数回繰り返りリカレントニューラルネットワークの重みをトレーニングして損失関数を最小限に抑える

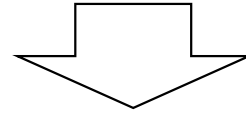
$$h_{t+1}, \theta_{t+1} = \text{RNN}_\phi(h_t, \theta_t, y_t),$$

重みを訓練すると  $\phi$  を学習する  $\Rightarrow$  RNNは量子回路の最適なパラメータを提案 164





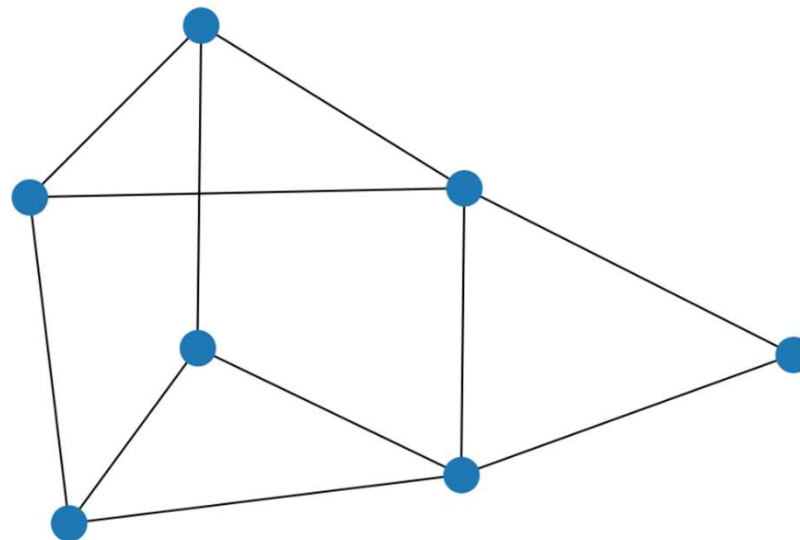
PennyLaneを使用して 量子回路を実行



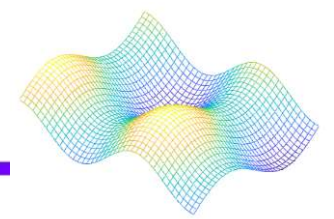
RNN の作成に使用される TensorFlowと統合

モデルのトレーニングとそのパフォーマンスのテストのため  
適切なデータセットをMaxCut 分析によりランダムなグラフを生成

MaxCut分析: 分離によってカットされるエッジの数が最大化される  
グラフ内のノードの適切なバイナリ分割を見つける



# 変分量子回路: QAOA



PennyLane の組み込みサブパッケージを使用して QAOA 量子回路を作成

PennyLane のQAOAモジュールを使用

⇒非常に少ないコード行でMaxCut 問題に対して完全に機能する  
量子回路を作成することが可能

```
def qaoa_from_graph(graph, n_layers=1): """Uses QAOA to create a cost
Hamiltonian for the MaxCut problem."""

# Number of qubits (wires) equal to the number of nodes in the graph wires =
range(len(graph.nodes))

# Define the structure of the cost and mixer subcircuits for the MaxCut
problem cost_h, mixer_h = qaoa.maxcut(graph)

# Defines a layer of the QAOA ansatz from the cost and mixer Hamiltonians def
qaoa_layer(gamma, alpha): qaoa.cost_layer(gamma, cost_h)
qaoa.mixer_layer(alpha, mixer_h)

# Creates the actual quantum circuit for the QAOA algorithm def
circuit(params, **kwargs): for w in wires: qml.Hadamard(wires=w)
qml.layer(qaoa_layer, n_layers, params[0], params[1]) return
qml.expval(cost_h)

# Evaluates the cost Hamiltonian def hamiltonian(params, **kwargs):
"""Evaluate the cost Hamiltonian, given the angles and the graph."""

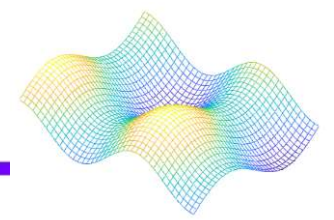
# We set the default.qubit.tf device for seamless integration with TensorFlow
dev = qml.device("default.qubit.tf", wires=len(graph.nodes))

# This qnode evaluates the expectation value of the cost hamiltonian operator
cost = qml.QNode(circuit, dev, interface="tf", diff_method="backprop")

return cost(params)

return hamiltonian
```

# リカレントニューラルネットワーク: LSTM



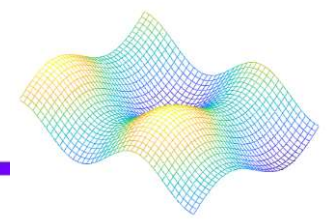
古典手順と量子手順の間で受け渡されるハイブリッドデータを処理できる  
長期短期メモリ (LSTM) ネットワークのカスタムモデルを構築

## 1. モデルの基本的な構成要素である LSTM セルを定義

```
# Set the number of layers in the QAOA ansatz.  
# The higher the better in terms of performance, but it also gets more  
# computationally expensive. For simplicity, we stick to the single layer case.  
n_layers = 1  
  
# Define a single LSTM cell.  
# The cell has two units per layer since each layer in the QAOA ansatz  
# makes use of two parameters.  
cell = tf.keras.layers.LSTMCell(2 * n_layers)
```

## 2. 一連のグラフのコスト関数を含む `qaoa_from_graph` リストを作成

```
# We create the QAOA MaxCut cost functions of some graphs  
graph_cost_list = [qaoa_from_graph(g) for g in graphs]
```



コスト関数: LSTM セルの重みの学習手順を実行するために必要

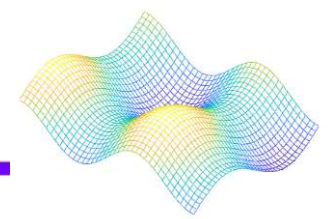
$$\mathcal{L}(\phi) = \mathbf{w} \cdot \mathbf{y}_t(\phi),$$

W: 再帰ループ内のさまざまなステップに重み付けをする係数

RNN は最適化の最初のステップ(係数が低い)  
パラメーター空間のより広い部分をより自由に探索

手順の終わり(係数が高い)  
最適な解を選択することは制約

# トレーニング



単一の勾配降下ステップをカスタム関数内にラップ

LSTM セルの重みのパラメーター空間で確率的勾配降下法を実行  
トレーニングセット内の各グラフについて勾配を評価しそれに応じて重みを更新  
⇒この手順を複数回 (エポック) 繰り返す

```
Epoch 1
> Graph 1/20 - Loss: -1.6641689538955688
> Graph 6/20 - Loss: -1.4186843633651733
> Graph 11/20 - Loss: -1.3757232427597046
> Graph 16/20 - Loss: -1.294339656829834
>> Mean Loss during epoch: -1.7352586269378663

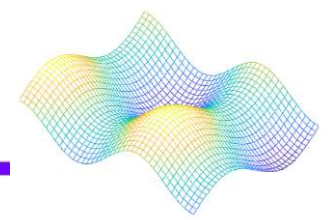
Epoch 2
> Graph 1/20 - Loss: -2.119091749191284
> Graph 6/20 - Loss: -1.4789190292358398
> Graph 11/20 - Loss: -1.3779840469360352
> Graph 16/20 - Loss: -1.2963457107543945
>> Mean Loss during epoch: -1.8252217948436738

Epoch 3
> Graph 1/20 - Loss: -2.1322619915008545
> Graph 6/20 - Loss: -1.459418535232544
> Graph 11/20 - Loss: -1.390620470046997
> Graph 16/20 - Loss: -1.3165746927261353
>> Mean Loss during epoch: -1.8328069806098939

Epoch 4
> Graph 1/20 - Loss: -2.1432175636291504
> Graph 6/20 - Loss: -1.476362943649292
> Graph 11/20 - Loss: -1.3938289880752563
> Graph 16/20 - Loss: -1.3140206336975098
>> Mean Loss during epoch: -1.8369774043560028
```

各グラフの損失はエポック全体で減少し続けている  
⇒トレーニングルーチンが正しく機能している

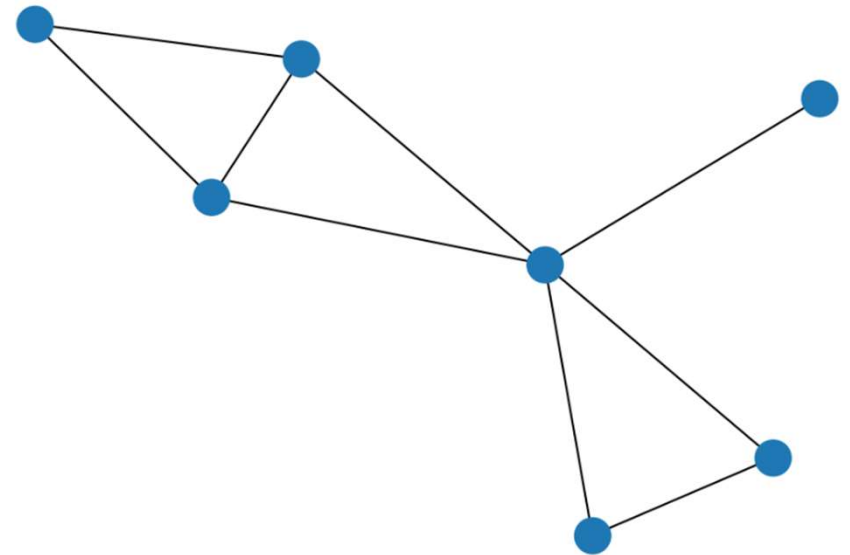
# 結果



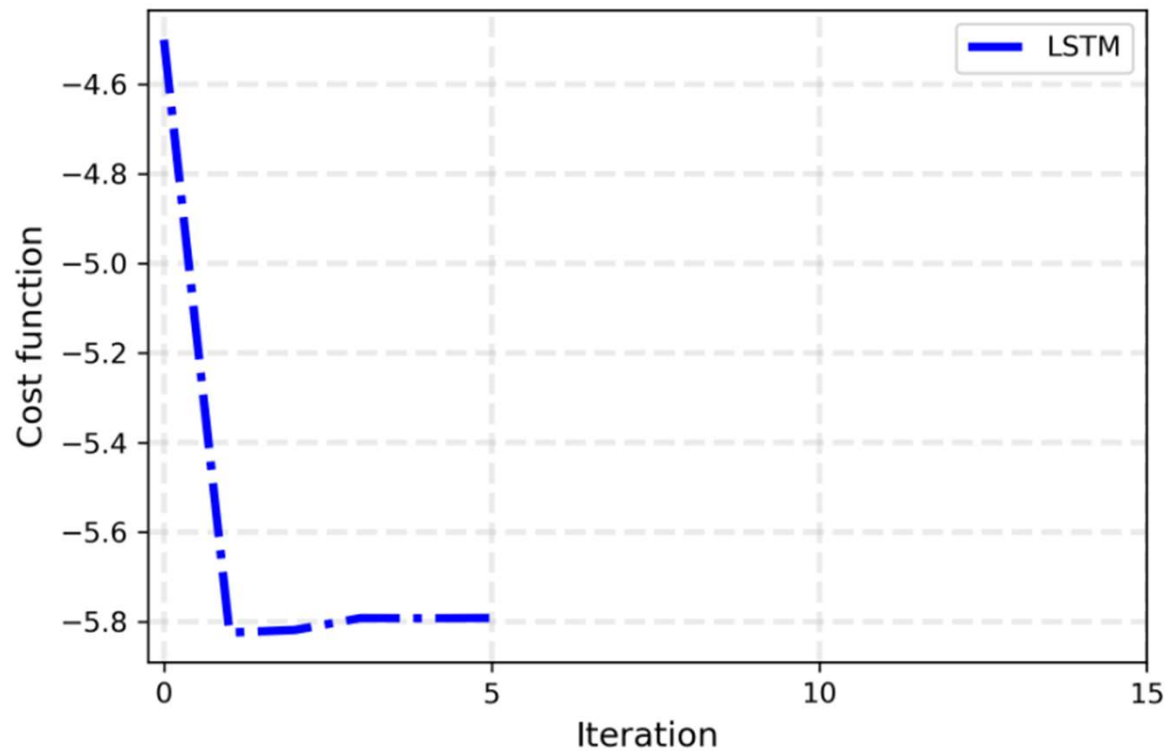
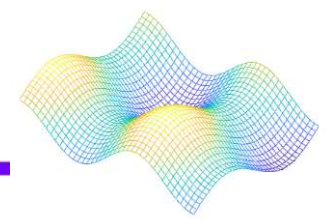
最適化された RNN を QAQA アルゴリズムの角度の初期化子として使用

トレーニングデータセットには存在しない新しいグラフを選択

```
new_graph = nx.gnp_random_graph(7, p=3 / 7)  
new_cost = qaqa_from_graph(new_graph)  
  
nx.draw(new_graph)
```

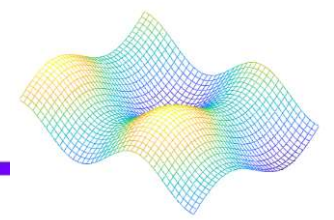


トレーニングされた RNN を新しいグラフに適用する

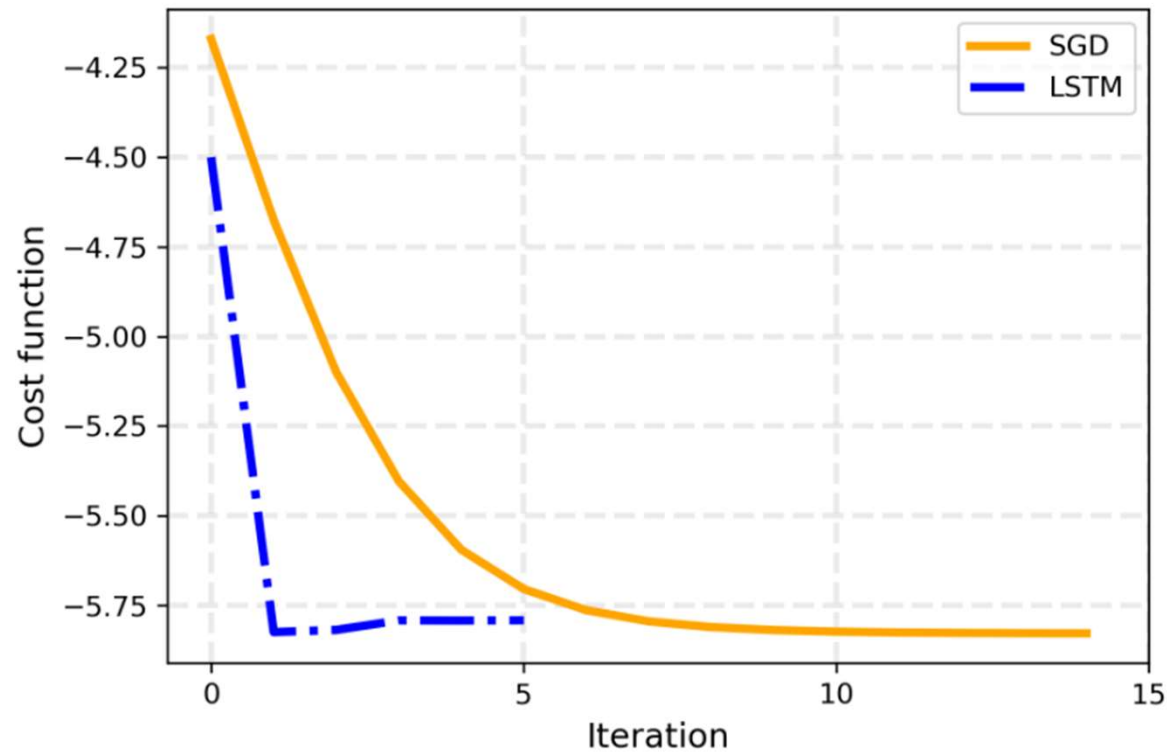


わずか数回の反復で損失が最小に達する  
⇒MaxCut コストが非常に迅速に最小化されるような  
新しいパラメーターを提案することを学習

# 標準的な確率的勾配降下法 (SGD) との比較

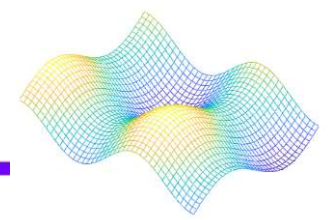


確率的勾配降下法 (SGD) : 標準的な最適化手法



RNN は標準の SGD よりも少ない反復で良好な最小値に達します





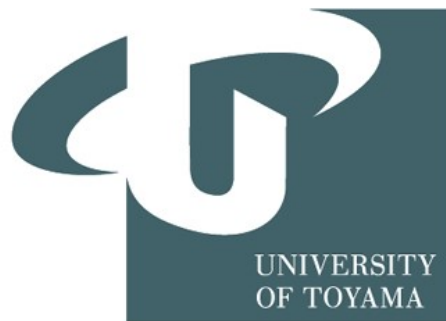
リカレントニューラルネットワークをブラックボックスオプティマイザーとして使用し、最適解に近い変分量子回路のパラメーターを初期化する方法を確認した

- PennyLane の MaxCut QAOA 量子回路を TensorFlow で構築された LSTM に接続し、カスタム ハイブリッドトレーニングルーチンを使用してネットワーク全体を最適化
- 非常に少ない反復で優れた最適解に到達する

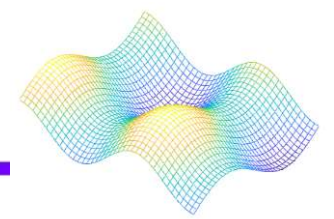
# Introduction to Geometric Quantum Machine

## Learning

### 幾何学量子機械学習の概要

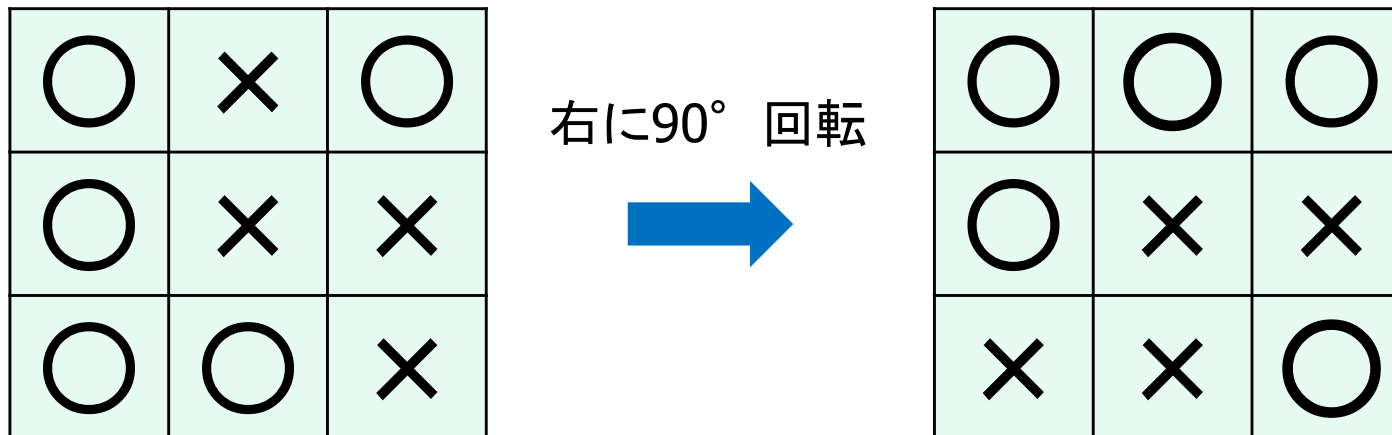


# はじめに



対称性は物理学の核心である。物性物理学や素粒子物理学では、ある物事を対称性だけで定義することがよくある。機械学習の分野で対称性を扱う目的は一般化能力を向上させることである。

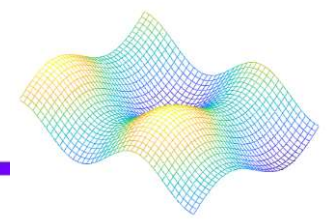
例 三目並べ  
ゲームに勝った場合、



対称性に注目すると同一のものとして識別できる。

利点

アルゴリズムにかけるデータ量が少なくなる。



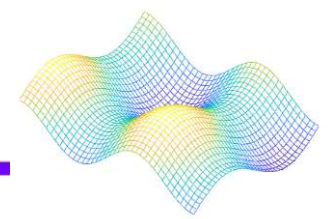
ニューラルネットワークの訓練には膨大なデータの入力が必要。

既知の対象構造をあらかじめネットワークにエンコードしておくことで、より正確に、より早く学習できるという原理がある。

画像解析のための畳み込みニューラルネットワーク(CNN)が成功した理由でもある。

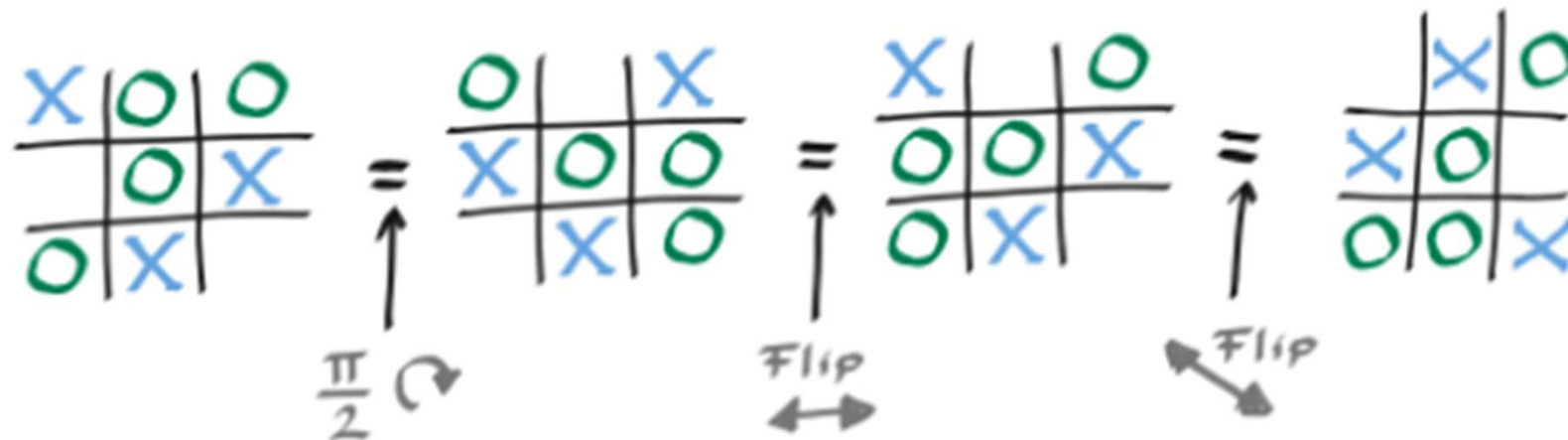
量子回路に焦点を当て、このアイデアを量子機械学習に拡張しようとしている。

# ○×ゲーム



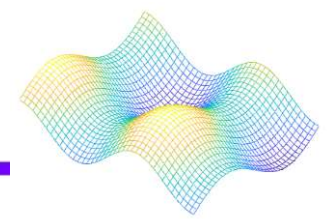
2人のプレイヤーが交互に、3x3のマスのに○か×を置く。目的は、自分のシンボルを3つ並べることである。

学習課題はゲームの勝ち負けを正しく識別するアルゴリズムを学習すること。



9つの要素からなるボードの対称性

# エンコード



1, 0, -1 はそれぞれ○, 無, × を表す。

-1	0	0
0	1	0
0	0	0



×		
	○	

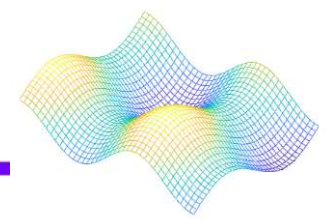
$y = (y_0, y_-, y_X)$   $(y_0, y_-, y_X) = (-1, -1, 1)$  のとき × を表す。

$(-1, -1, 1)$	$(-1, 1, -1)$	$(-1, 1, -1)$
$(-1, 1, -1)$	$(1, -1, -1)$	$(-1, 1, -1)$
$(-1, 1, -1)$	$(-1, 1, -1)$	$(-1, 1, -1)$



×		
	○	

# 量子モデル化



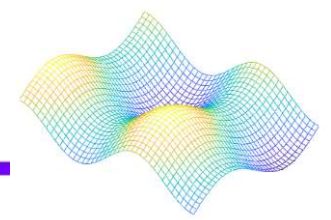
9個の量子ビットをボードのマスを表すようにする。  
全てのマスを0に初期化。

0	0	0
0	0	0
0	0	0

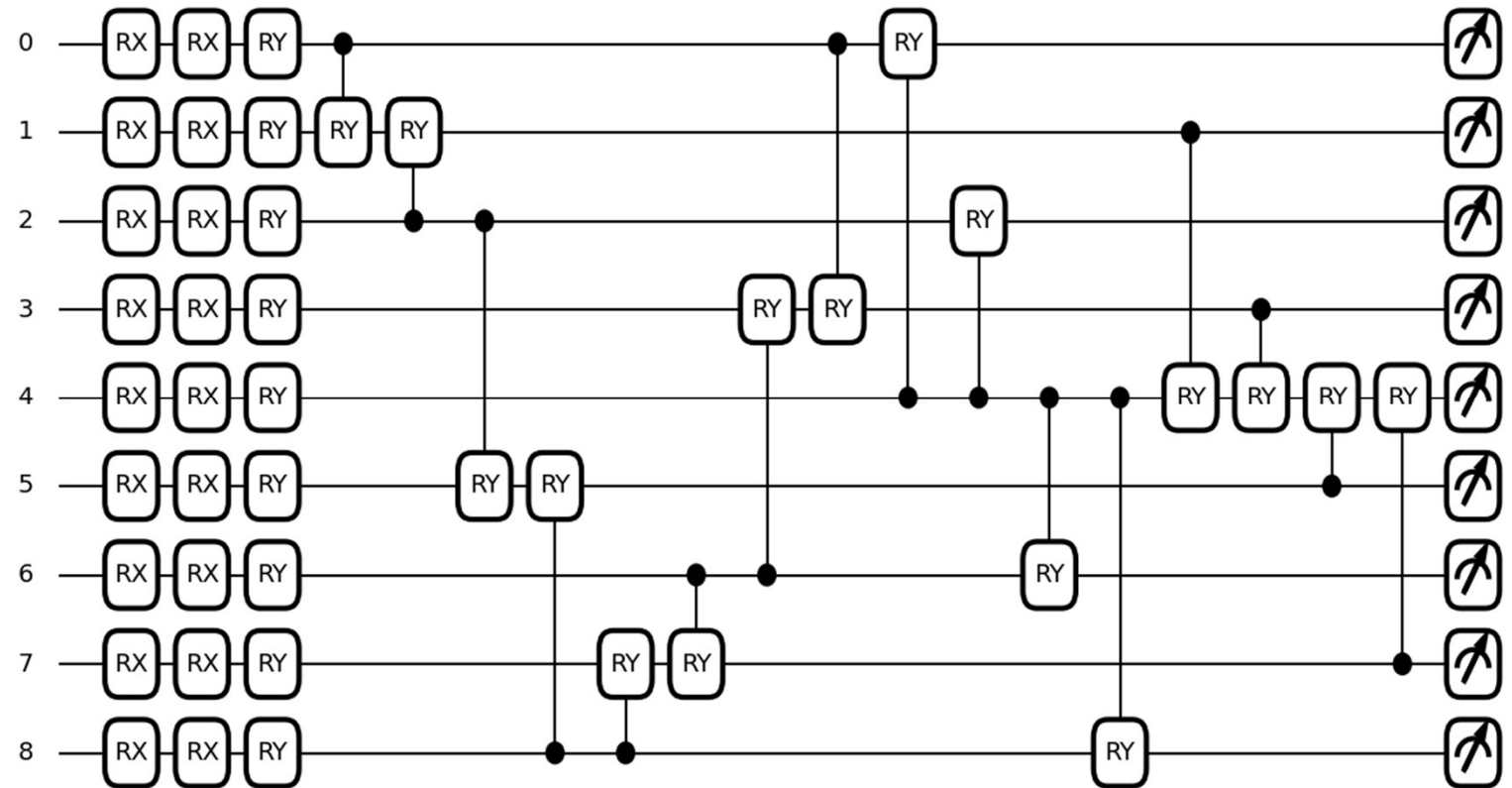
数値の入力

各量子ビットに1量子ビット $R_x(\theta)$ ,  $R_x(\theta_1)$ ,  $R_y(\theta_2)$ 、  
2量子ビット $CR_y(\theta_3)$ の回転を行う。

# 量子回路



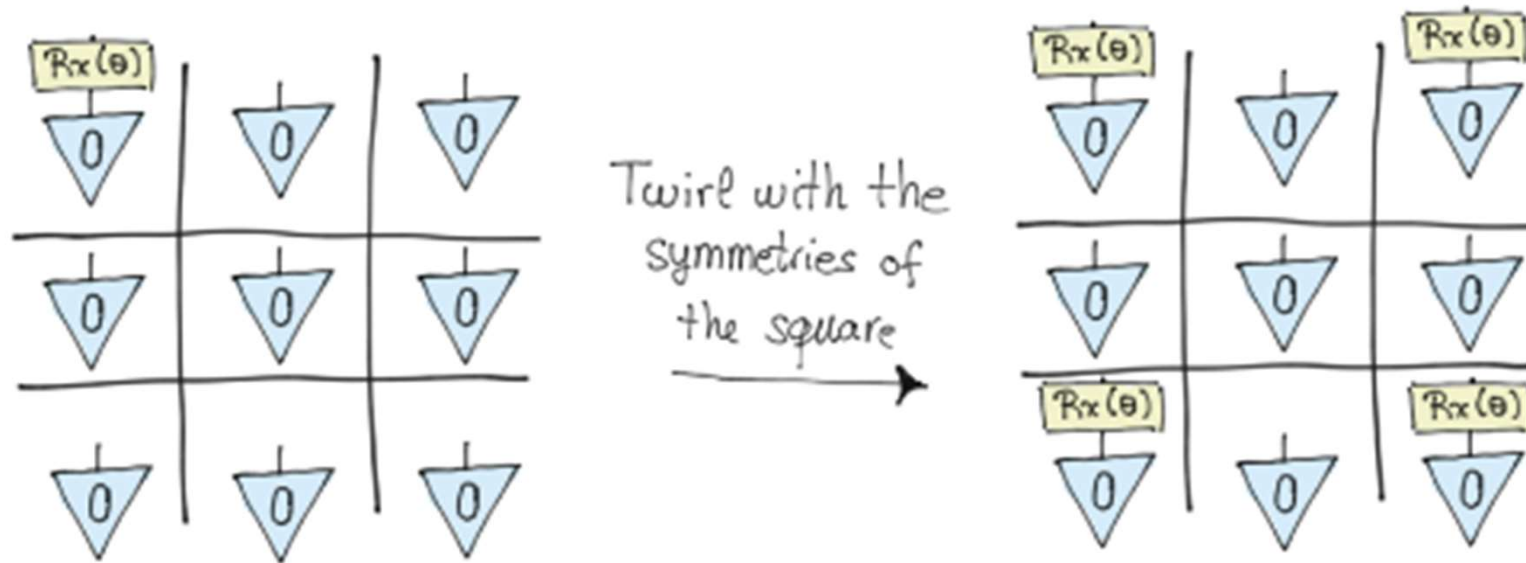
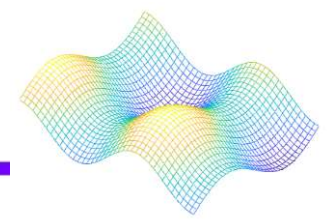
0	1	2
3	4	5
6	7	8



44種類のゲートが必要

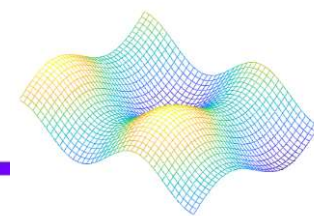


# ボードの対称性

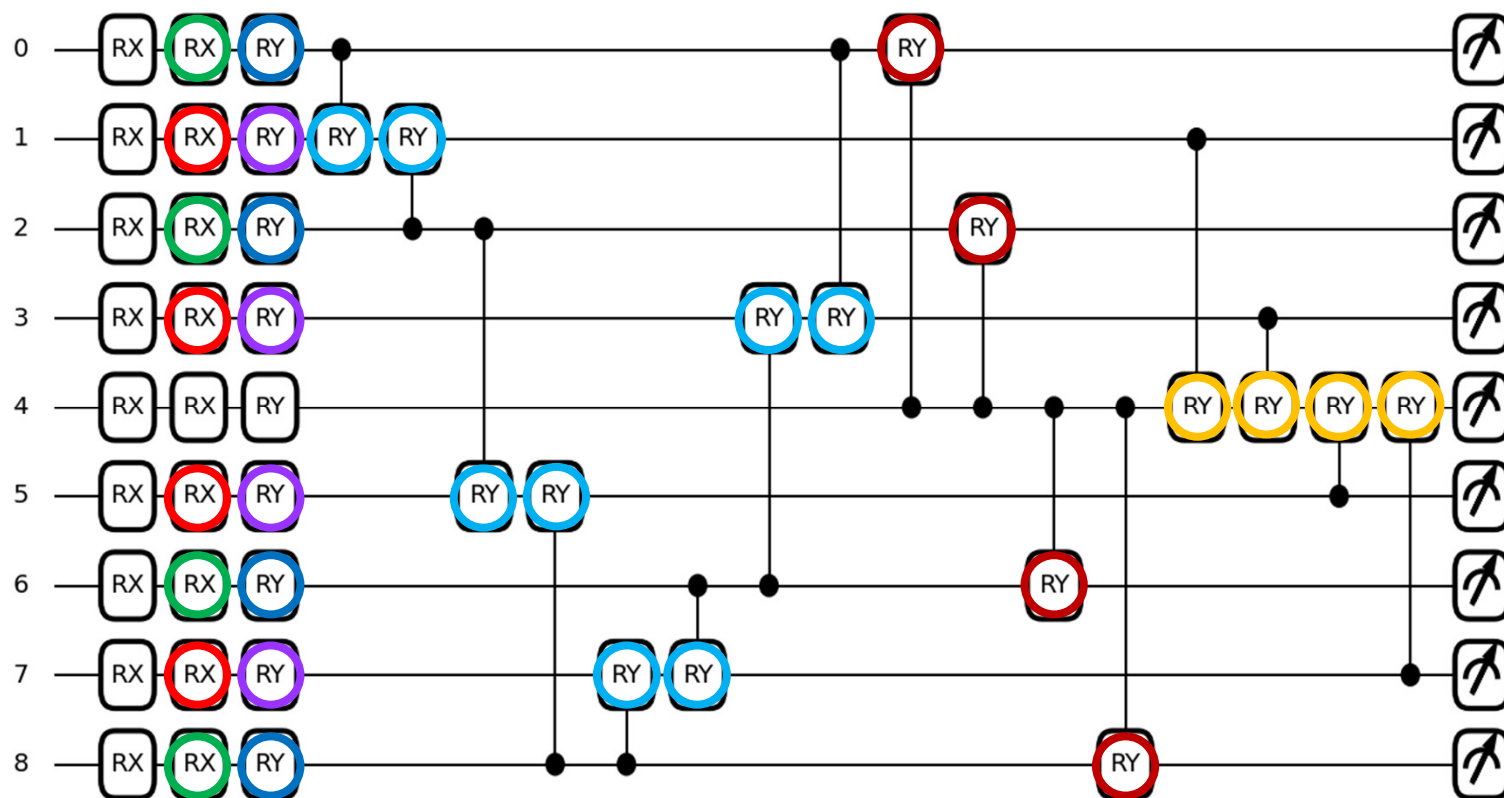


ボードの対称性を考慮すると4つのゲートは同じものとみなせる。

# 量子回路(対称性考慮)

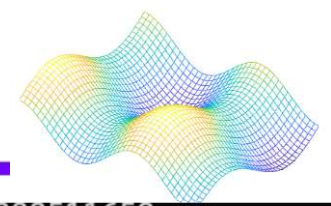


0	1	2
3	4	5
6	7	8



18種類にまで減らすことが可能

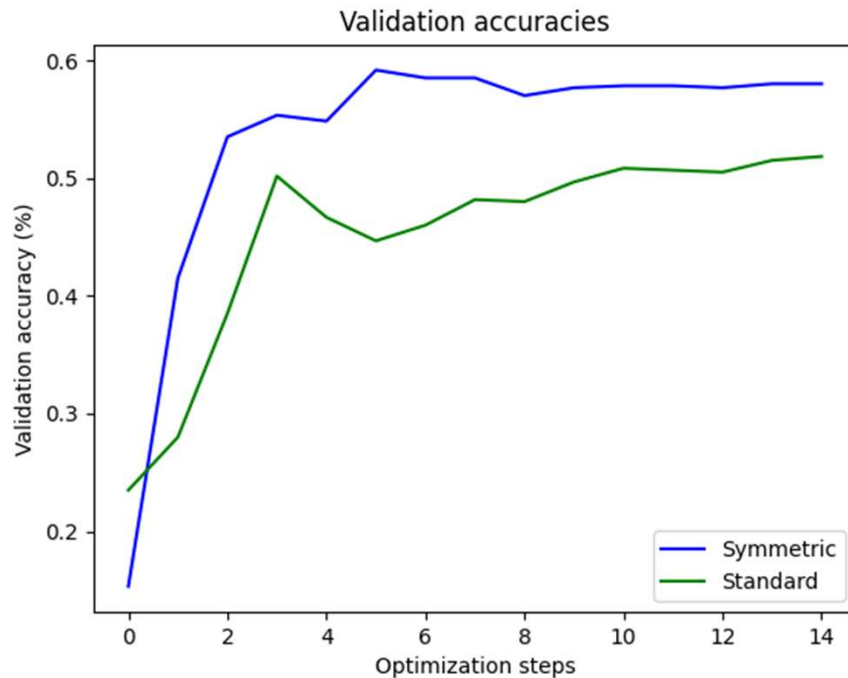
# 結果



accuracy without training = 0.2383333295583725			accuracy without training = 0.22166666388511658		
Epoch: 1	Loss: 2.996221	Validation accuracy: 0.153333	Epoch: 1	Loss: 3.025290	Validation accuracy: 0.235000
Epoch: 2	Loss: 2.838961	Validation accuracy: 0.415000	Epoch: 2	Loss: 2.918151	Validation accuracy: 0.280000
Epoch: 3	Loss: 2.721652	Validation accuracy: 0.535000	Epoch: 3	Loss: 2.824333	Validation accuracy: 0.385000
Epoch: 4	Loss: 2.686487	Validation accuracy: 0.553333	Epoch: 4	Loss: 2.747958	Validation accuracy: 0.501667
Epoch: 5	Loss: 2.608699	Validation accuracy: 0.548333	Epoch: 5	Loss: 2.693046	Validation accuracy: 0.466667
Epoch: 6	Loss: 2.648471	Validation accuracy: 0.591667	Epoch: 6	Loss: 2.659418	Validation accuracy: 0.446667
Epoch: 7	Loss: 2.630698	Validation accuracy: 0.585000	Epoch: 7	Loss: 2.641402	Validation accuracy: 0.460000
Epoch: 8	Loss: 2.544674	Validation accuracy: 0.585000	Epoch: 8	Loss: 2.626516	Validation accuracy: 0.481667
Epoch: 9	Loss: 2.630653	Validation accuracy: 0.570000	Epoch: 9	Loss: 2.616884	Validation accuracy: 0.480000
Epoch: 10	Loss: 2.595081	Validation accuracy: 0.576667	Epoch: 10	Loss: 2.610851	Validation accuracy: 0.496667
Epoch: 11	Loss: 2.586225	Validation accuracy: 0.578333	Epoch: 11	Loss: 2.606585	Validation accuracy: 0.508333
Epoch: 12	Loss: 2.600443	Validation accuracy: 0.578333	Epoch: 12	Loss: 2.599107	Validation accuracy: 0.506667
Epoch: 13	Loss: 2.652541	Validation accuracy: 0.576667	Epoch: 13	Loss: 2.592962	Validation accuracy: 0.505000
Epoch: 14	Loss: 2.585265	Validation accuracy: 0.580000	Epoch: 14	Loss: 2.589474	Validation accuracy: 0.515000
Epoch: 15	Loss: 2.598611	Validation accuracy: 0.580000	Epoch: 15	Loss: 2.584630	Validation accuracy: 0.518333

対称性考慮

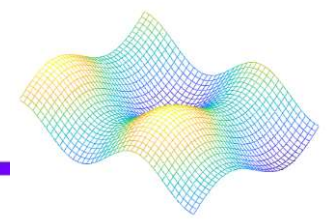
対称性考慮せず



損失関数

$$L(D) = \frac{1}{|D|} \sum_{(g,y) \in D} \|\hat{y}(g) - y\|_2^2$$

対称性を考慮することによって学習精度が向上していることが分かる。

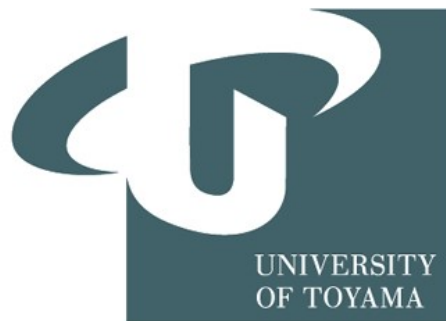


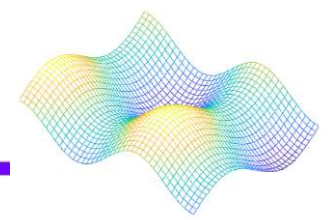
- ・学習問題に内在する対称性に注目して、これを等変数ゲートセットに反映させることで、学習精度を向上させることができた。
- ・どちらの場合も学習精度が理想的であるとは言えないが、対称性を考慮した回路は明らかに普通の回路より優れている。

# Variational Quantum Thermalizer

## 変分量子熱化器

Jack Ceroni





- ・変分量子サーマライザー(VQT)

近年、提案された量子アルゴリズムに関連する理論と実験について説明

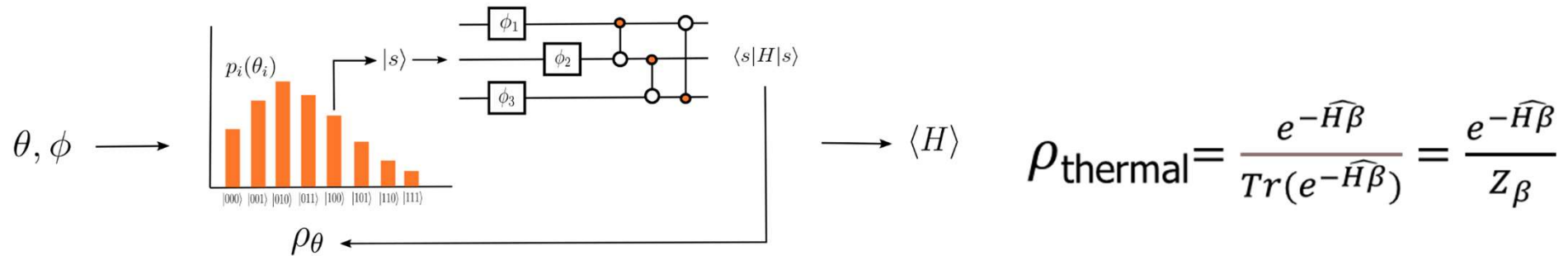
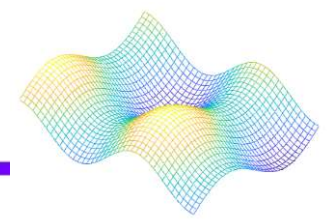
- ・変分量子固有ソルバー(VQE)をゼロ以外の温度のシステムに一般化したもの



短期量子コンピュータを使用した量子化学の主力アルゴリズム  
リッツ変分原理の応用、コンピュータが特定の分子の基底状態を準備するように利用されている

→ VQTの目標・・・特定のハミルトニアンの熱状態を準備すること

# VQT : (初期密度行列の確率分布)

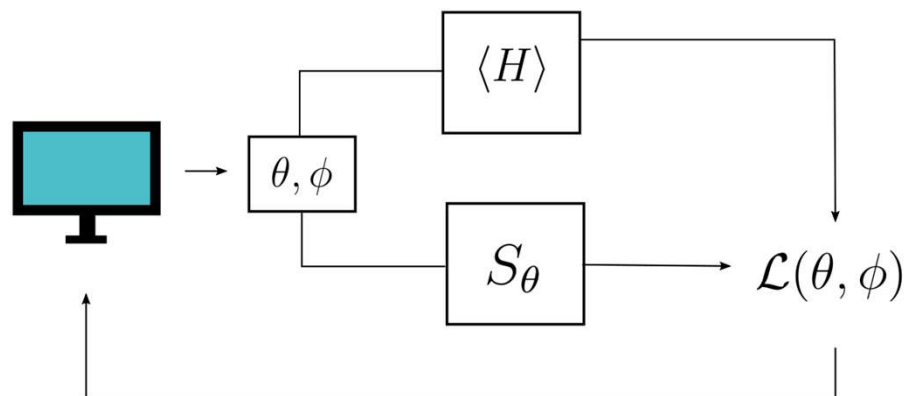
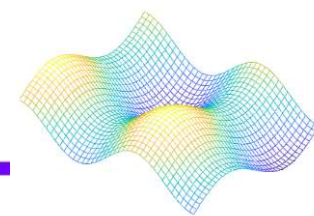


熱状態は混合状態であり、純粋状態の集合体によって説明できる  
→純粋状態をアンザッツ回路に通し、エネルギー期待値を最小限に抑える  
という変分法から逸脱する必要

入力されたパラメータは、初期密度行列とパラメータ化された分析値を作成  
これらは、新しい混合状態に関するハミルトニアン期待値を計算するため使用



# VQTのしくみの概要



VQTがどのように機能するかを高レベルで表現したもの

変分回路で最も重要な部分→コスト関数

古典的なオプティマイザーを用いて $\langle \psi(\theta) | \hat{H} | \psi(\theta) \rangle$ を最小限に抑えようとする

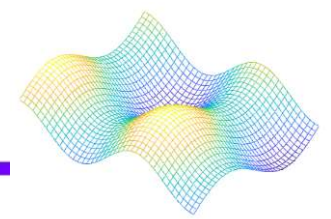
VQTの目標である、熱状態の適切な近似を生成するパラメータ化された確率分布と分析に到達のため以下の式を用いた

$$L(\theta, \phi) = \beta \text{Tr}(\hat{H} \hat{U}(\phi) \rho_\theta \hat{U}(\phi)^\dagger) - S_\theta$$

以上の式を満たす時、コスト関数が最小化される

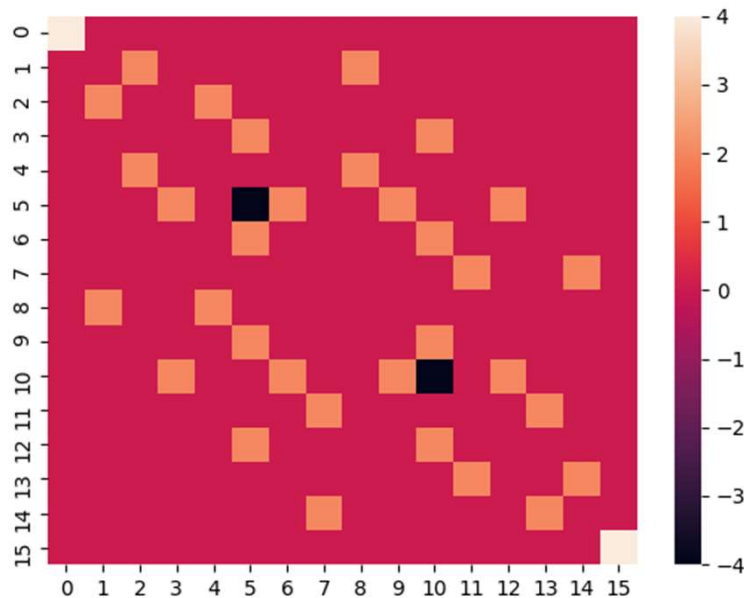
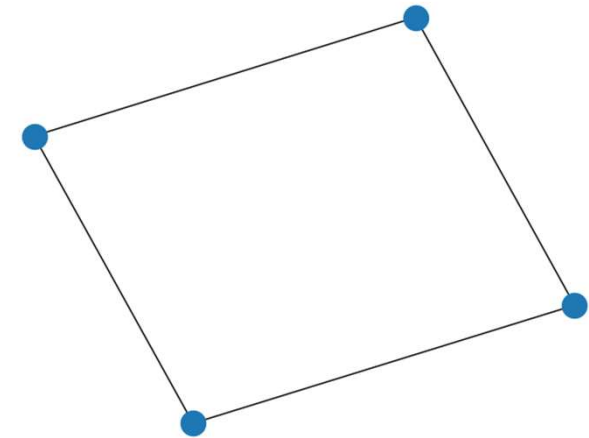


# VQT シミュレーション



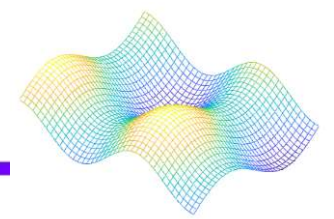
このデモでは、4量子ビットのハイゼンベルクモデルをシミュレートした  
なお、ハイゼンベルクハミルトニアンは次のように定義される

$$\hat{H} = \sum_{(i,j) \in E} X_i X_j + Z_i Z_j + Y_i Y_j$$



この実行結果により、シミュレーション  
に必要な依存関係をインポート  
VQTを構築する

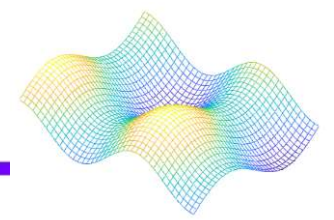
# アンザッツ回路の作成



アンザッツを構築するには定義された手法と、相互作用グラフ内のエッジを共有する量子ビット間に配置されたパラメータ化された結合ゲートのコレクションを組み合わせる。  
その後、アンザッツの深さとシミュレーションが実行されるデバイスを定義

```
def single_rotation(phi_params, qubits):  
  
    rotations = ["Z", "Y", "X"]  
    for i in range(0, len(rotations)):  
        qml.AngleEmbedding(phi_params[i], wires=qubits, rotation=rotations[i])
```

# コスト関数

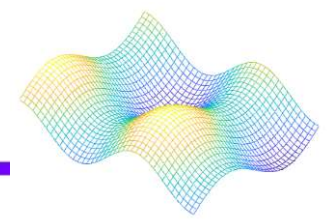


最後にアンザッツとエントロピー関数を組み合わせてコスト関数を取得

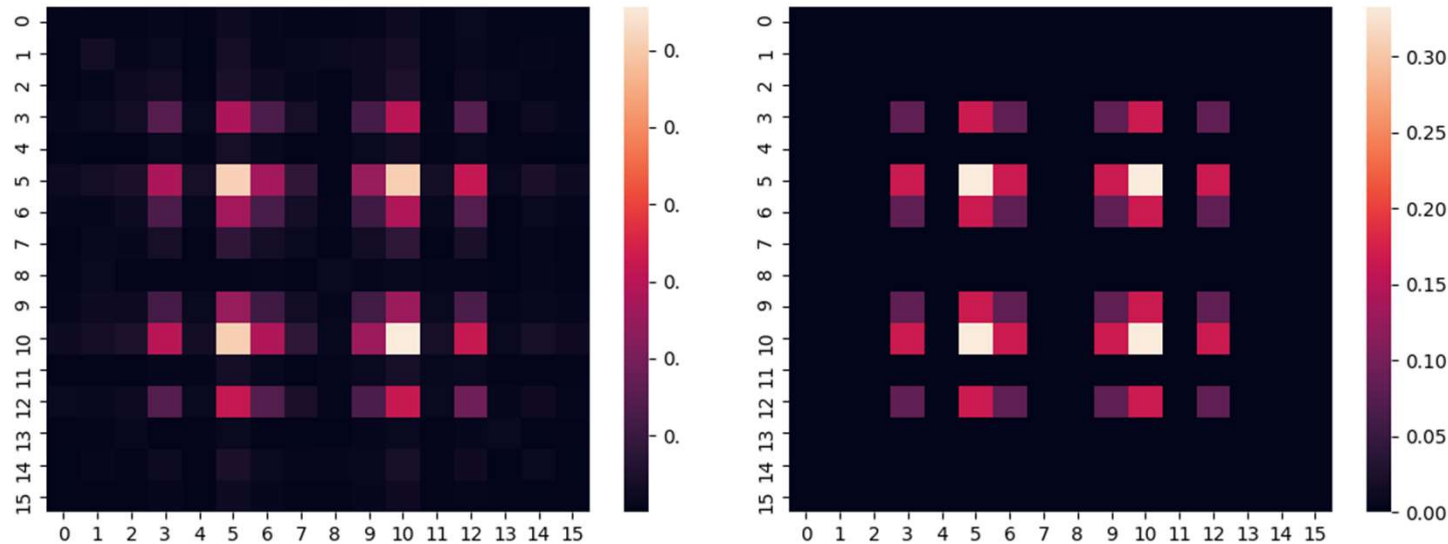
1. 初期の混合状態を記述する確率分布からサンプリングする代わりにアンザッツを使用して計算
2. これらの値の期待値に対応する値を乗算  
そして各項を合計すると変換されたハミルトニアン<sup>1</sup>の期待値が得られる。
3. この関数をアンザッツおよびエントロピー関数とともに最終コスト関数に渡す
4. オプティマイザーに渡される関数を作成
5. オプティマイザーを定義し、最適化メソッドを実行

密度行列を再構成する関数を作成することで、パラメータを得る

# ヒートマップ(密度行列: エントリごとの絶対値)



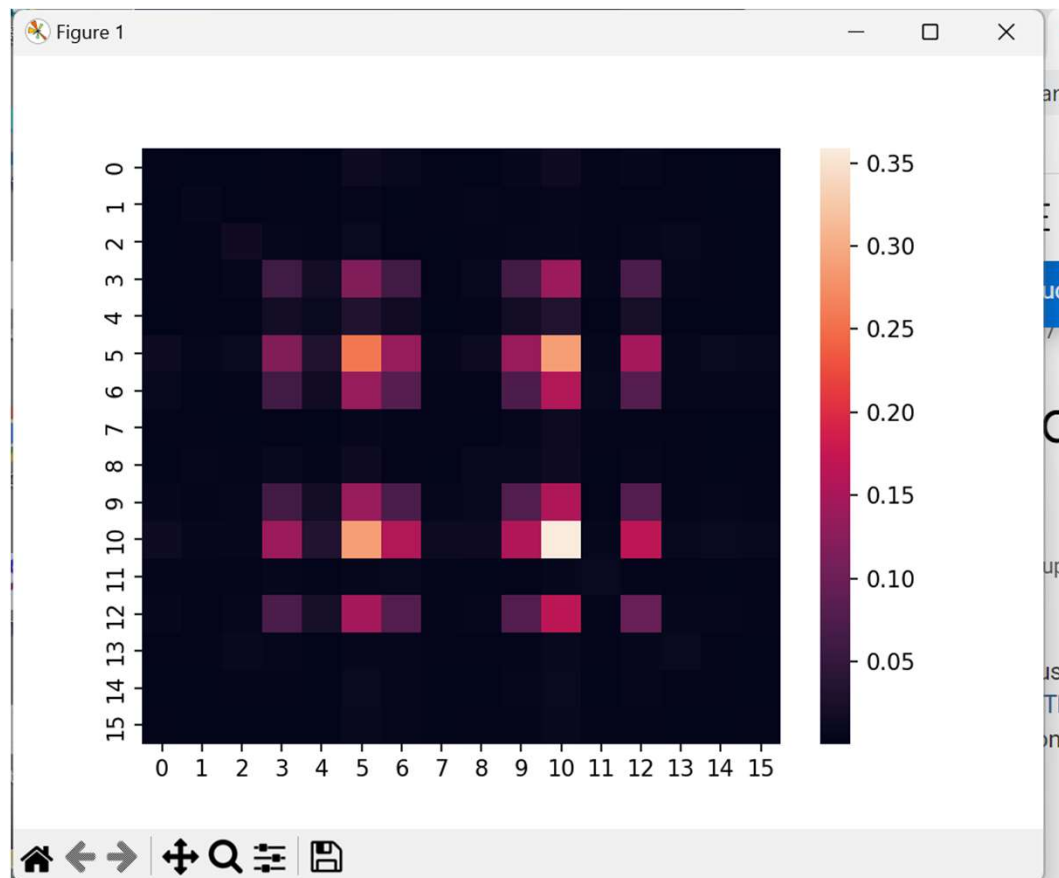
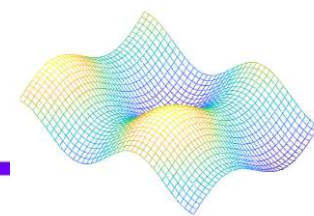
密度行列のエントリごとの絶対値のヒートマップのプロットを表示

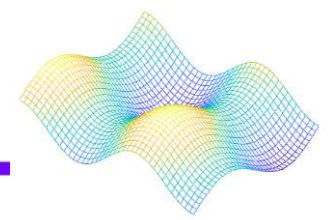


実際に熱状態の適切な近似値が準備されたことを確認するため、ハイゼンベルクハミルトニアン of 行列指数を計算した

0に近いほど、2つの画像は類似していることから、熱状態の近似値を構築できたことが示唆された

# シミュレーション結果





編集: PennyLaneプログラム履修編集委員会

お疲れ様でした!