プログラマの為の量子コンピュータ入門

Part 1:
関連数学と1量子ビット計算

先端IT活用推進コンソーシアム
オープンラボ

宮地直人（miyachi@langedge.jp）

Ver1.0 2019年7月25日

Part 0：イントロダクション（プロローグ）

Richard Feynman（リチャード ファイマン）

“If you think you understand quantum mechanics, you don’t understand quantum mechanics.”

「もしあなたが量子力学を理解できたと思うならば、それは量子力学を理解できていないと言うことだ。」

※ と言うことできっと私も理解できていないので間違い等ご指摘を！

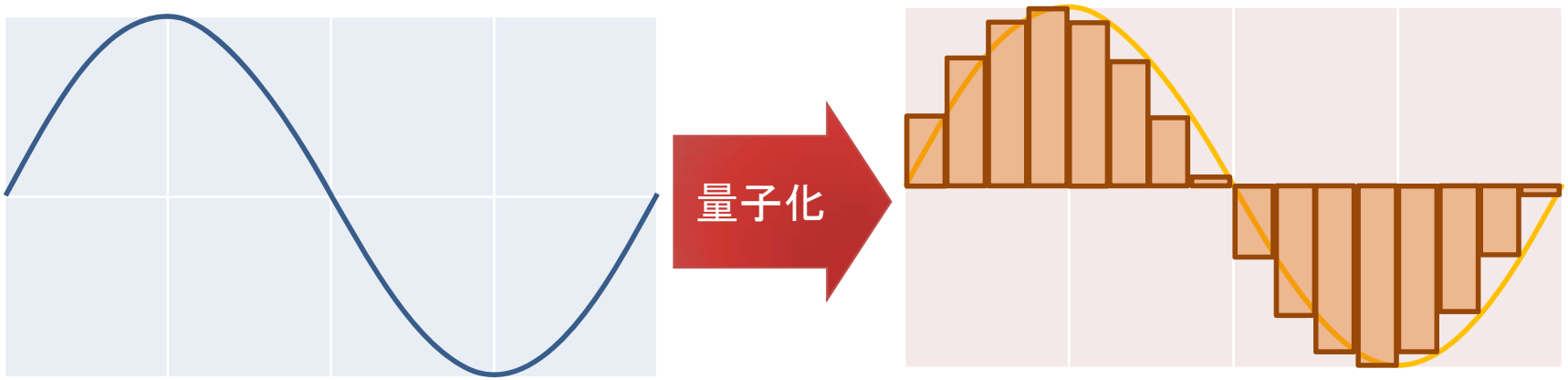
量子とは？

一般的な定義：

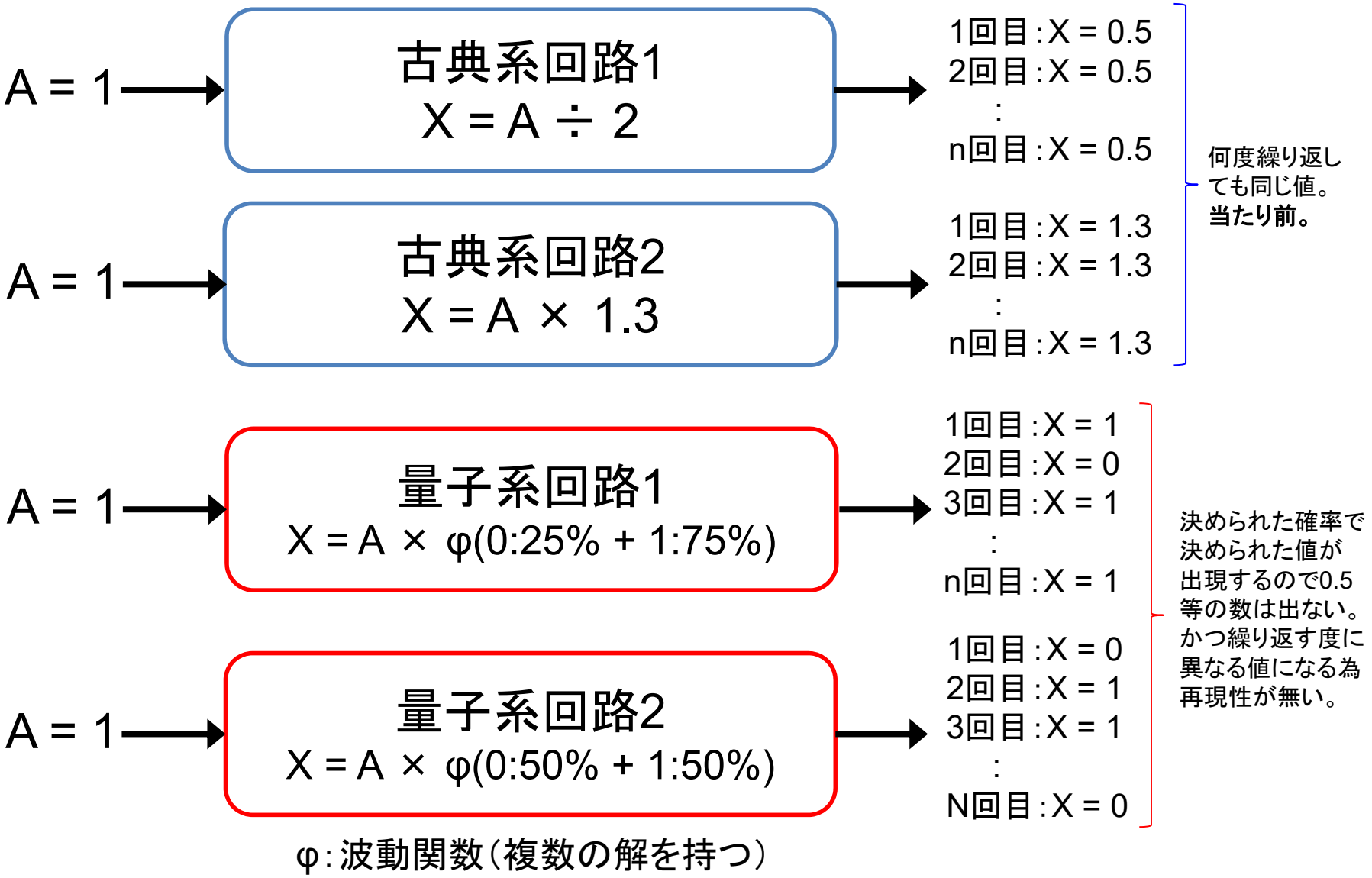
物理量を実数では無く単位量の整数倍で表す場合にその単位量を「量子」と呼ぶ。

※ 量子論/力学以外でも使われる場合がある。

例：アナログ波形のデジタル化を量子化と呼ぶ。



古典系と量子系の測定



φ : 波動関数（複数の解を持つ）

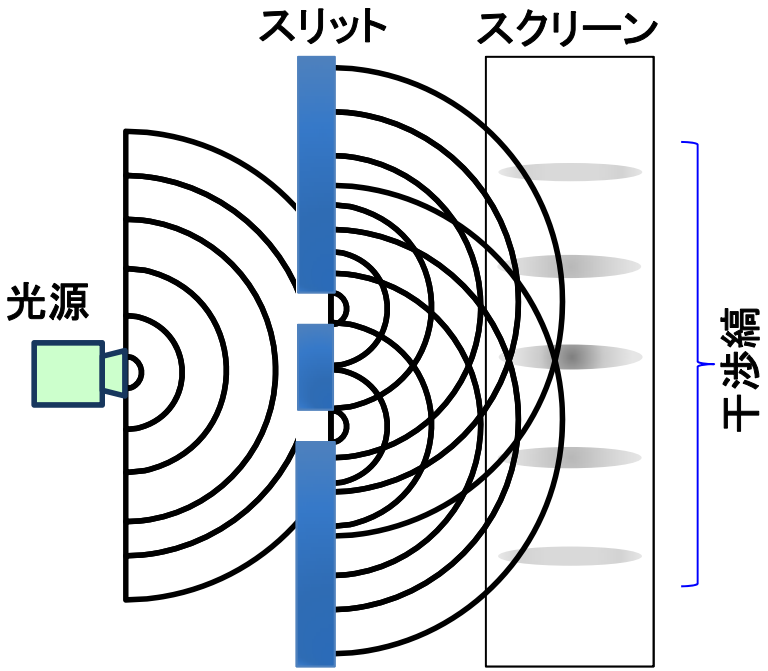
量子系(量子論の世界)

- 量子論で記述された(量子)系が取る状態。
- 古典論では全く同じ系で測定された物理量は**毎回同じ測定値(実数)を示す。**
- 量子論では全く同じ系で測定された物理量は**毎回異なった測定値を示す。**
- 量子状態の測定値は実数では無く固有値によるとびとびの値を取る(だから量子)。
- どの測定値となるかは波動関数により決まる**確率分布**にて示される(コペンハーゲン解釈)。
- 一般に原子以下のミクロな世界は量子状態。

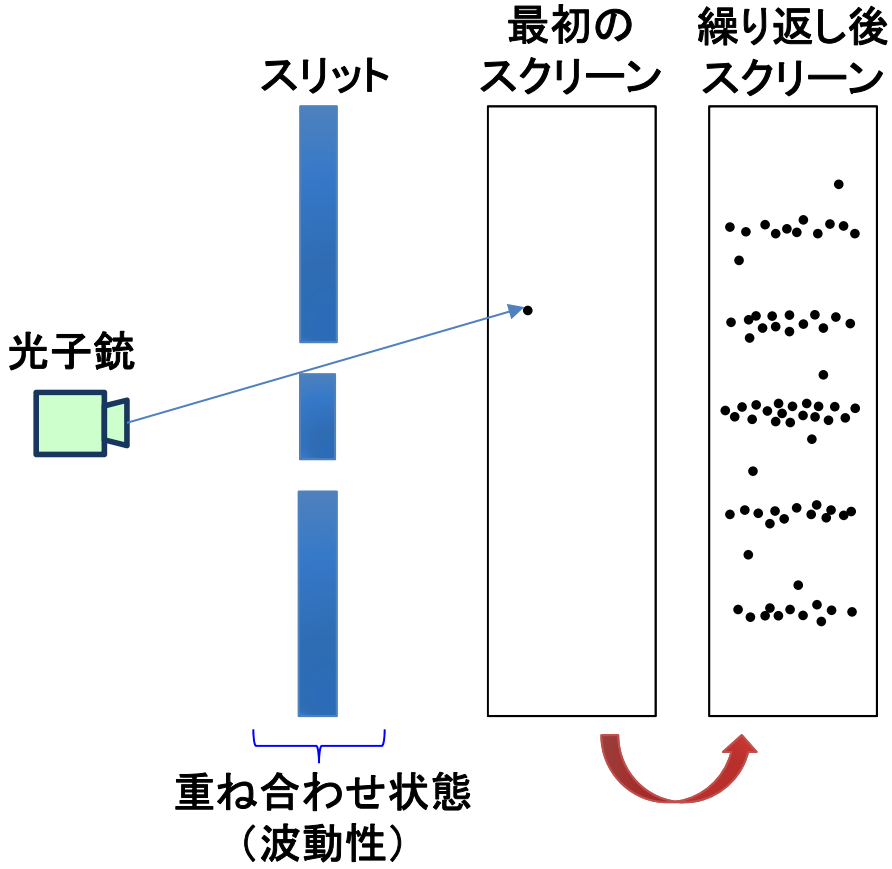
量子の対語は古典

二重スリット実験（光の波動性）

二重スリット通過後に干渉縞が表示される。
と言うことは光に波動性があることを示す。
干渉縞は同じ位相を持つ光源が2つあり、
距離により位相が強めあう場所と弱めあう
場所がある為に生じる現象である。

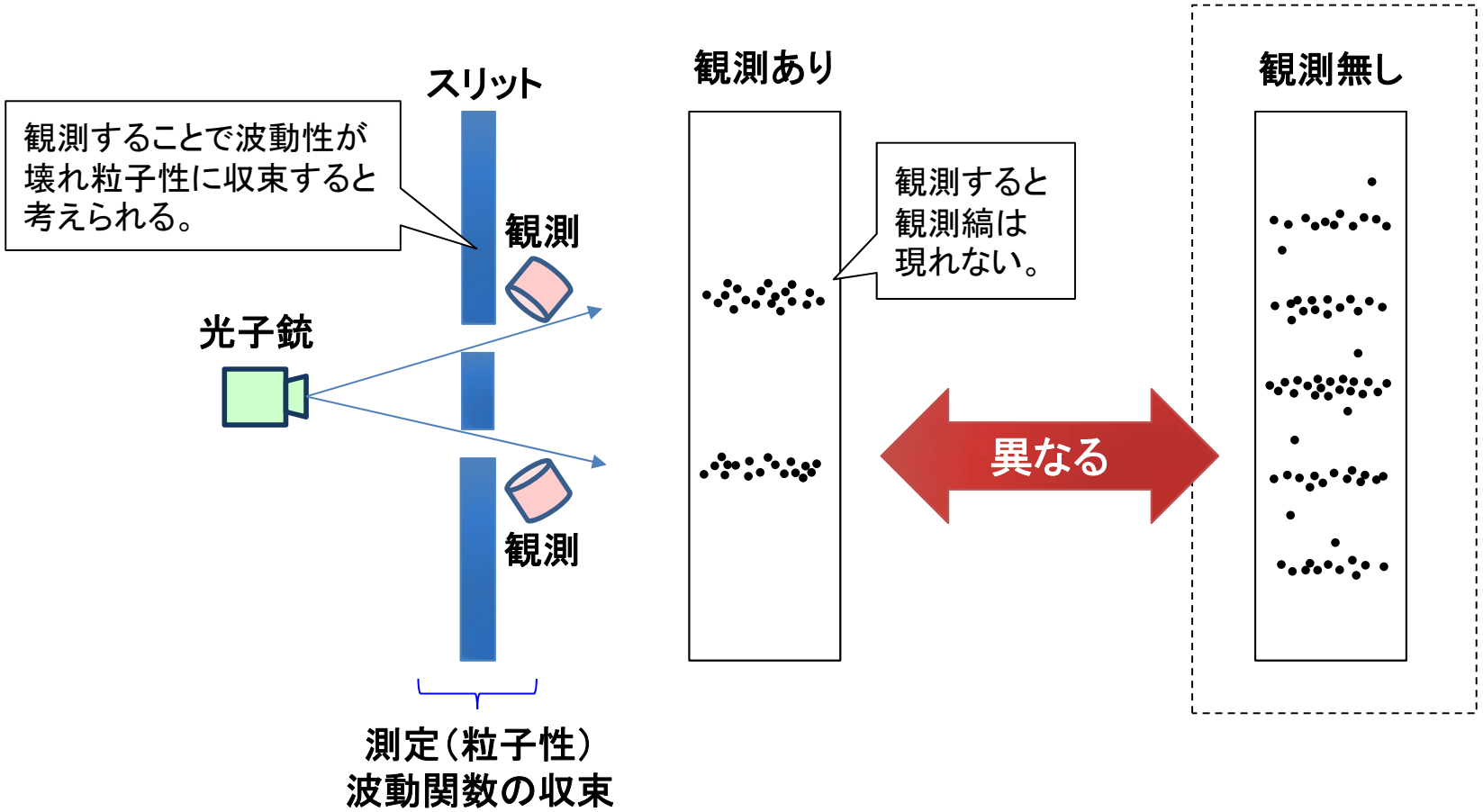


光子1個ずつ発射可能な光子銃と光子1個を
認識できるスクリーンを使う。光子1個ずつを
打ち出しても繰り返すうちに干渉縞が出てくる。
つまり光子1個でも波動性があることになる。



二重スリット実験の観測問題

観測問題：どちらのスリットを通過したかを観測することにより干渉縞が出なくなる。
コペンハーゲン解釈では観測することで量子干渉が壊れて粒子として収束していると考える。
しかし他にも解釈は存在しており（平行宇宙論等）、正確なことは分かっていない。



波動と粒子の二重性（量子重ね合わせ）

観測しなくても
時間が経つと崩壊する

観測

重ね合わせ状態（波動性）

固有値状態（粒子性）

収束

量子デコヒーレンス

複数の状態が同時に存在して
量子干渉を生じている
（コペンハーゲン解釈）

波動関数（列ベクトル）を
$|\psi>$ と表現すると

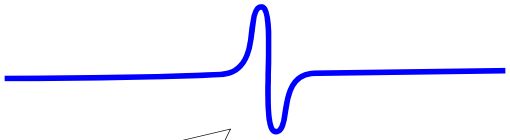
$$|\psi> = c_0|\psi_0> + c_1|\psi_1>$$
$$|c_0|^2 + |c_1|^2 = 1.0$$

c_0 と c_1 は複数数

固有状態のうち1つの状態

$|\psi_0\rangle$ または $|\psi_1\rangle$ のいずれか

波動性は失われる
複素数虚数部は意味を持たない

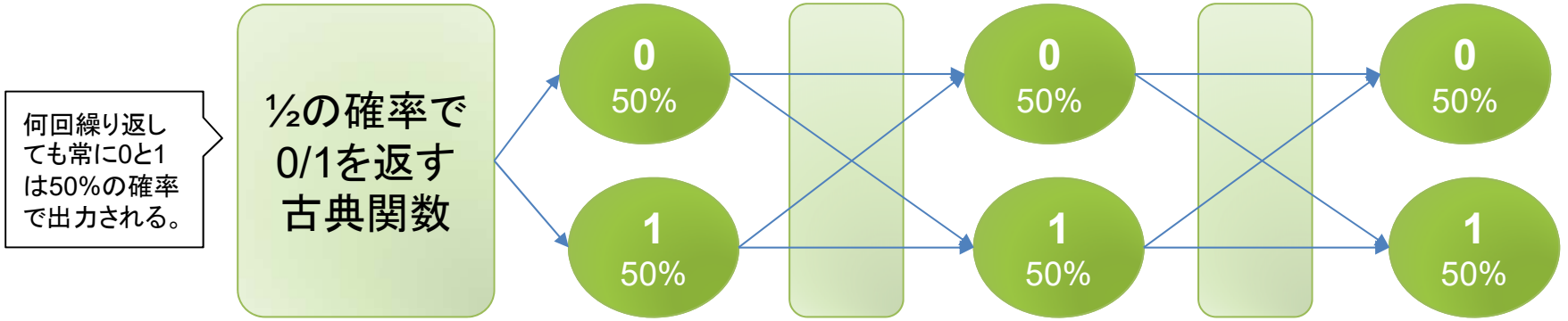


コペンハーゲン解釈では、粒子状態
は波動がパルス状になったと考える。
位置が決まると波長は定まらなくなる。

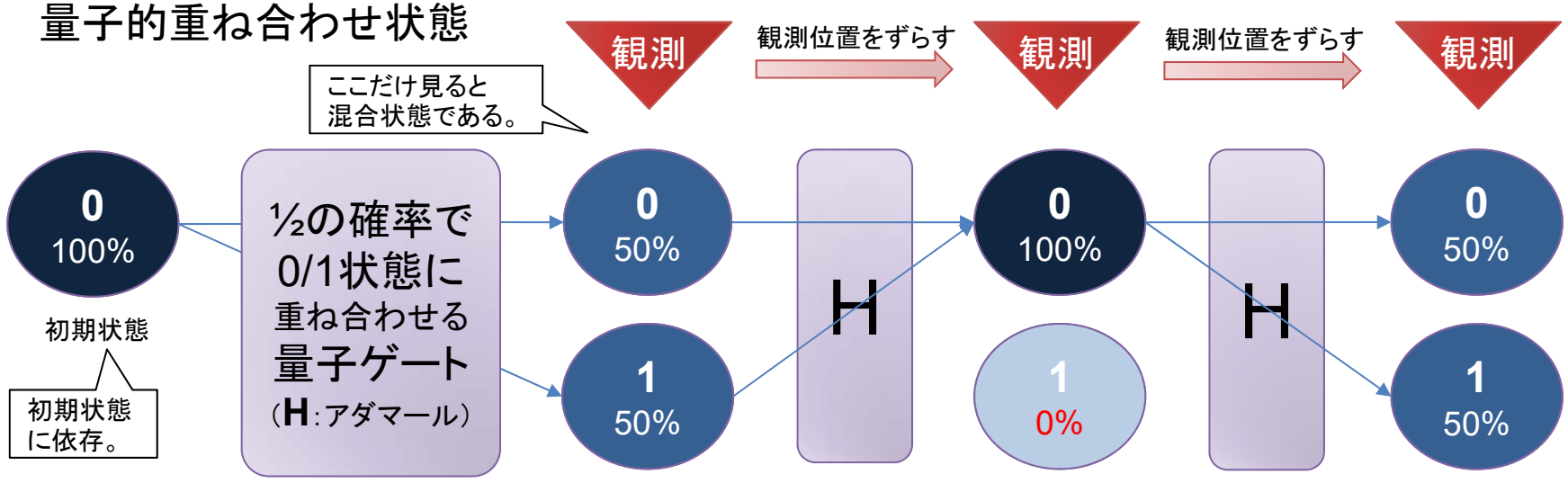
※ コヒーレンス時間（状態の量子干渉が失われるまでの時間）は短い。

量子的重ね合わせと古典的混合状態

古典的混合状態



量子的重ね合わせ状態



※ 量子には波動性がある為に確率以外の要素(波動の位相)がある。

量子ビット（重ね合わせの実現）

- 0と1の量子重ね合わせの状態が可能な単位。
- 観測により2つの固有値（0か1）に収束する。
- 物理的に実現するには幾つかの方法がある。

方式	概要	開発
超伝導量子ビット	現在主流となっている超伝導状態のシリコン回路で量子ビットを実現する方式（極低温）	IBM, Google, D-Wave
イオントラップ	捕獲したイオンをレーザーで冷却して利用（室温） 理論的には量子ビット間の全結合が可能	IonQ
量子ドット	原子10～50個で構成した微小半導体を利用（極低温?）	Intel
トポロジカル	超電導体とトポロジカル絶縁体による量子ビット（極低温?）	Microsoft
NVセンター ダイヤモンド窒素-空孔中心	ダイヤモンドの炭素を窒素に置き換えて生じる欠損部に電子を捕獲して量子ビットに利用（室温）	（研究レベル）
QNN 量子ニューラルネットワーク	光パルスを量子ビットとして利用（室温） 全結合によるアニーリング型の計算が可能（らしい）	NTT/NII/東大（ImPACT）

可逆コンピュータ

可逆コンピュータは非量子な場合には低電力になっても低性能になる為にほぼ実用化されていない。

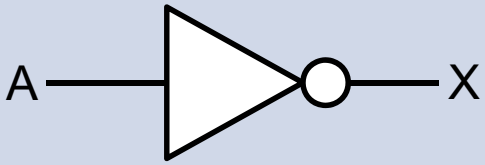
情報が失われる時にエネルギー（熱）が消費される。

可逆コンピュータが実現すると低電力化が可能となる。

IBMから1960年代に出た理論で今も研究が続いている。

※ 量子コンピュータは可逆コンピュータの一種である（詳細は次ページ）。

NOTゲート

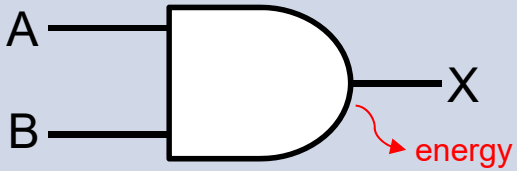


A	X
1	0
0	1

出力から入力が再現できるので可逆

可逆

ANDゲート

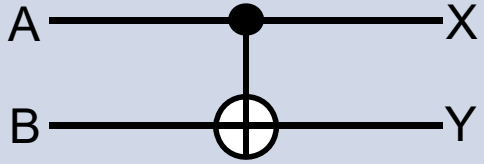


A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

出力から入力が再現できない

非可逆（量子計算では使えない）

CNOTゲート



A	B	X	Y
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Yだけ見るとXOR

可逆

なぜ量子コンピュータは可逆なのか？

量子計算:

1. 量子力学の計算にはシュレディンガー方程式を利用。
2. シュレディンガー方程式は時間に可逆な性質を持つ。

従って:

シュレディンガー方程式を使う量子計算も可逆となる必要がある。

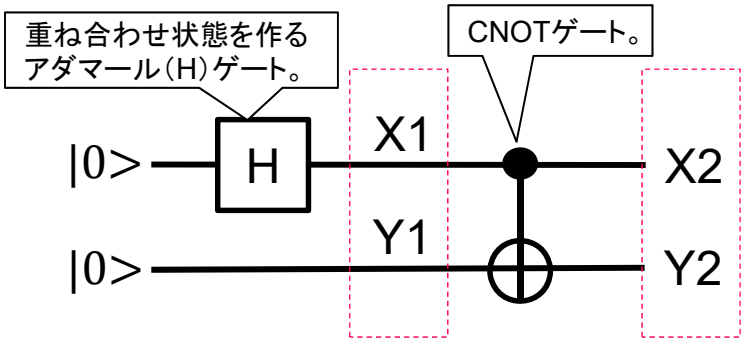
つまり:

（ゲート型）量子コンピュータは可逆コンピュータ。

※ 後ほどもう少し詳しくシュレディンガー方程式を見ます。

量子もつれ（量子エンタングルメント）

- 量子もつれは、2つの粒子（量子ビット）が互いに影響をおよぼし合い、一方を測定すると、もう一方の値が確定する現象である。
- 複数の量子ビット間を、量子もつれにより関連付けることで量子回路を構築する。以下HとCNOTによる例。



X2が0ならY2も0に、
Y2が1ならX2も1と、観測

$X1 = |0\rangle : 50\% + |1\rangle : 50\%$
$Y1 = |0\rangle$

この辺り詳しくは次回。

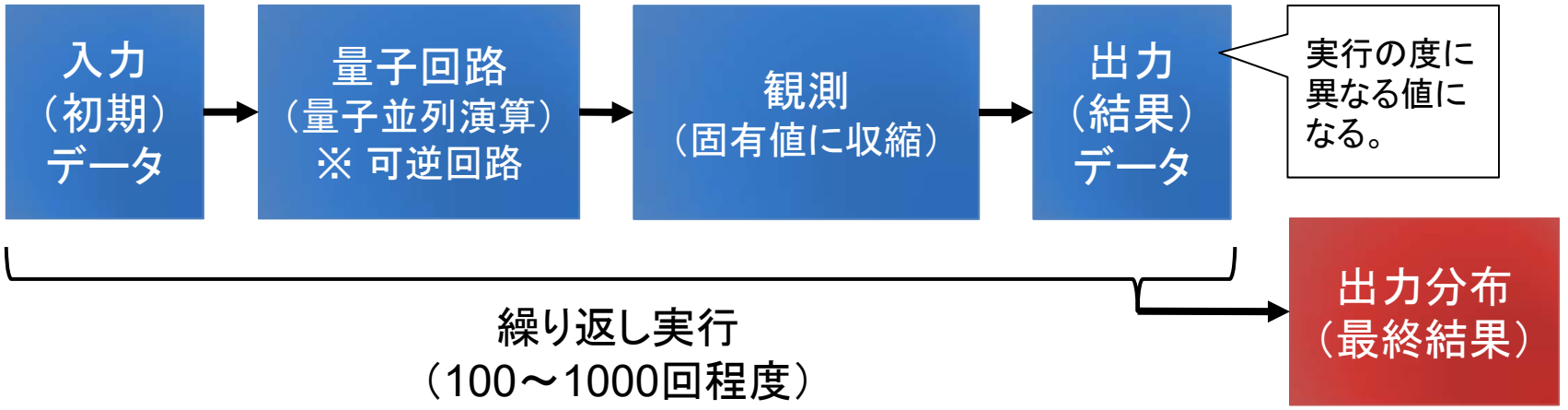
$X2 = X1 = |0\rangle : 50\% + |1\rangle : 50\%$
X1が $|0\rangle$ なら Y2も $|0\rangle$ （Y1そのまま）
X1が $|1\rangle$ なら Y2も $|1\rangle$ （Y1を反転する）

$X2Y2 = |00\rangle : 50\% + |11\rangle : 50\%$
※ $|01\rangle$ や $|10\rangle$ は 0%

量子コンピュータの概要

1. 0と1の2つの基底を持つ複数の量子ビットを利用。
2. 重ね合わせた状態のままで量子並列演算を行う。
3. 量子ビット同士を絡み合わせて量子回路を構築。
4. 重ね合わせ状態の出力を観測して固有値に収縮。
5. **繰り返し実行**し確率的な固有値の**出力分布**を得る。

→ 二重スリット実験時に光子の分布により干渉縞を確認するようなもの。



量子コンピュータの種類

量子ゲート型（狭義の量子コンピュータ）

方式：量子ゲートの量子回路による量子計算

対象問題：汎用（ただし量子アルゴリズムの範囲内）

開発企業：IBM/Google/Intel/Alibaba/Microsoft等

量子アニーリング型（正確には量子シミュレータ）

方式：イジングモデルを使った量子シミュレーション

対象問題：最適化問題特化（深層学習等への応用）

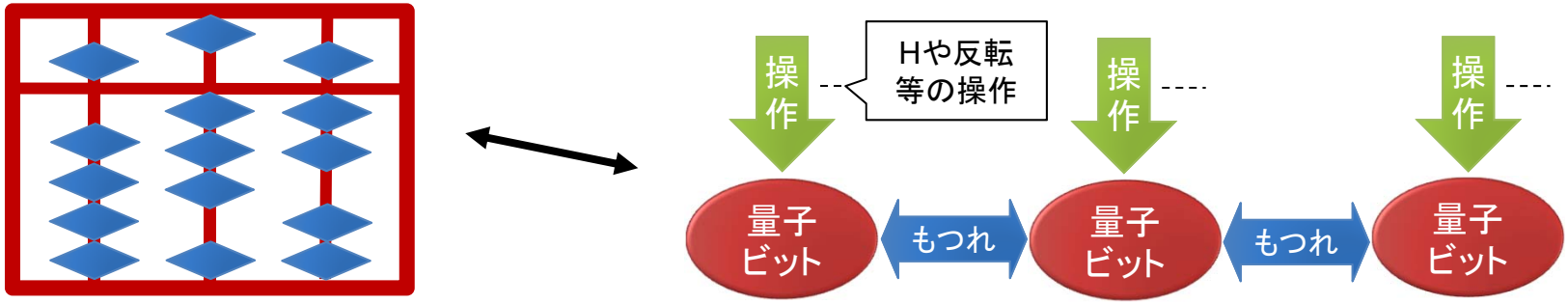
開発企業：D-Wave（非量子型では富士通と日立）

※ 非量子：富士通「デジタルアニーラ」、日立「CMOSアニーリングマシン」。

※ 他に光を使ったCIM（コヒーレントイジングマシン）もあるがここでは省略。

量子ゲート型とソロバン

ソロバンは珠(たま)を配置したハードウェアを、指で弾いて行くことで計算を進めて行く。各珠の間には桁上がり等の関連性がある。



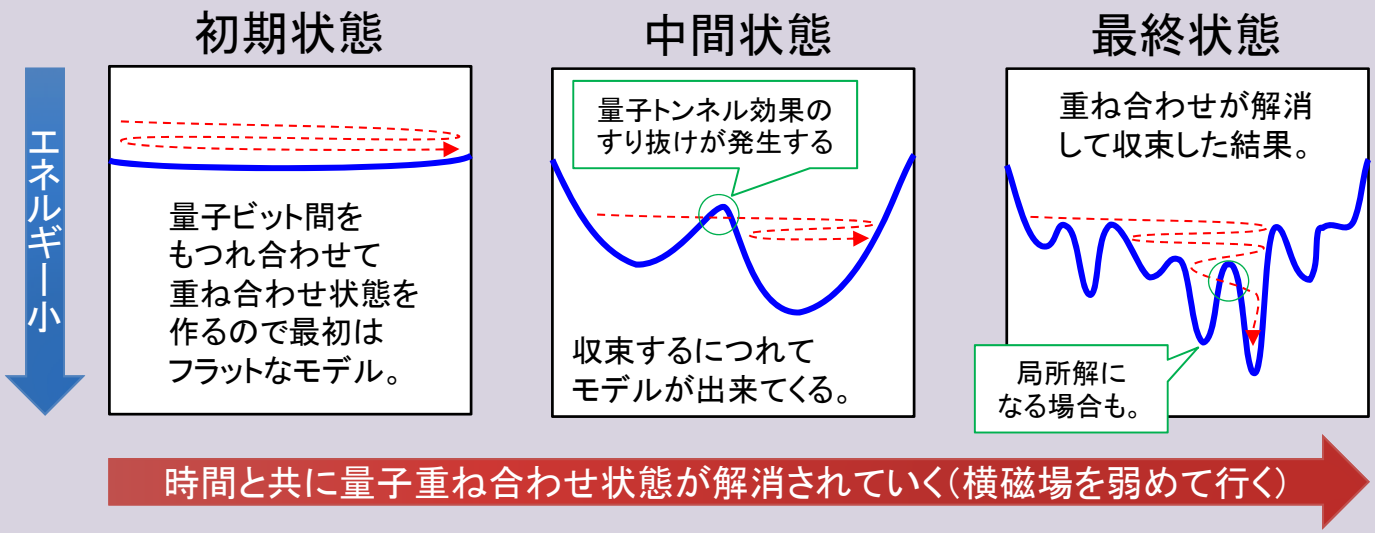
量子ゲート型は量子ビットを配置したハードウェアを、レーザーや電界で弾いて行くことで計算を進めて行く。各量子ビット間には量子もつれによる関連性がある。

ソロバンは可逆回路でもある。またソロバンは同じ操作をすれば毎回同じ値になるが量子では異なる。

量子アニーリング型の計算

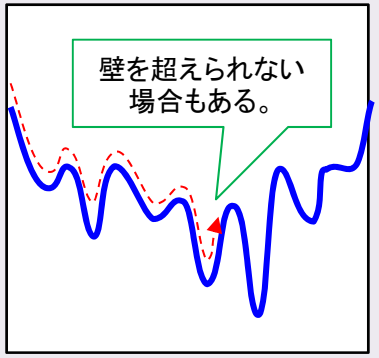
どちらも問題をイジング
モデルとして定式化する
必要があり数学が必要。

量子アニーリング



量子アニーリングでは量子ビット間の全結合(もつれ)が理想だが…

シミュレーテッドアニーリング



シミュレーテッドアニーリングは先にモデルをセットしてから最適解を求めて行く。

シミュレーテッドアニーリングは古くから使われており、現在でも富士通や日立が専用ハードウェアを使ったアニーリングを実現(例:デジタルアニーラ)している。

NISQの時代（今後5～10年程度）

Q2B: QUANTUM FOR BUSINESS 2017

物理学者 John Preskill による基調講演の論文

「*Quantum Computing in the NISQ era and beyond*」

https://arxiv.org/abs/1801.00862

NISQ: Noisy Intermediate-Scale Quantum

ノイズエラーがあり中規模量子ビット数の時代

50～数百量子ビット程度

現在は標準ハードウェアと標準ソフトウェア（アルゴリズム）を確立する時期で、特にソフトウェアは量子シミュレータを使って勉強しておくことで量子プログラミング時代に備える。

量子計算シミュレータ

富士通のデジタルアニーラや、
日立のCMOSアニーリングも、
ある意味ではシミュレータである。

今、我々に必要なもの：

エラーなし高速の量子計算シミュレータ

目的：量子アルゴリズムの学習の為

量子コンピュータの実機とシミュレータの比較：

実機（NISQ）：エラーがあり小規模（各種制限あり）のみ

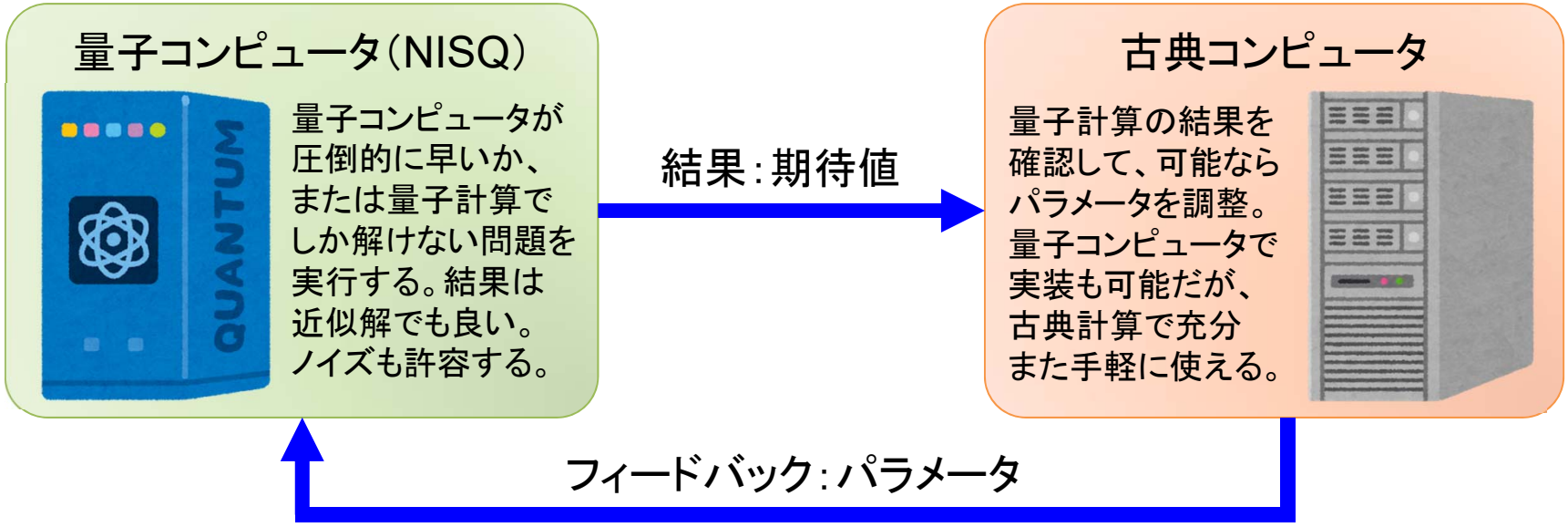
シミュレータ：エラーなしで中規模（制限少）、ただし限界あり

※ シミュレータもNISQ計算のQiskit/Cirqはその意味では向いていない。

- **Qulacs**（QunaSys社）：C++実装で高速化
- **QGATE**（個人の趣味）：GPUを使って高速化

古典量子ハイブリッドアルゴリズム

現実的な利用方法として古典コンピュータとの組合せ利用が進んでいる。
第2部で説明するショアのアルゴリズムもある意味ハイブリッド計算である。



主な用途:

- 近似最適化:QAOA(Quantum Approximate Optimization Algo)
- 基底状態探索:VQE(Variational Quantum Eigensolver)
- 機械学習:QCL(Quantum Circuit Learning)

量子コンピュータの未来予想

量子超越
（古典コンピュータでは解けない問題）

RSA暗号解読

どこかでブレークスルーを生じる可能性はゼロではないが…

2015　2018　2023～2030　2040～2060くらい（予測困難）

	NISQ *ノイズあり中規模*	*量子加速* *（Quantum Speedup）*	*万能（Universal）量子コンピュータ*
実用性 汎用性	ほぼない （一部を除き古典機の方が早い）	一部あり （古典機よりも早い分野がある）	あり （ただし得意不得意はある）
信頼性	ノイズ（エラー）あり	（部分的な）エラー訂正	エラー訂正あり
量子数	$10 \sim 10^2$	（エラー訂正あり）$10^2 \sim 10^6$	10^6（数百万）以上
利用形態	専用機	専用機・外部接続装置	単独・古典機と融合

参考：（ノイマン型）古典コンピュータの歴史

1950　1955　1970

第1世代 *真空管・試用や研究*	*第2世代* *トランジスタ・商用化*	*第3世代* *集積回路・パーソナル化*

量子計算フレームワーク（量子ゲート型）

IBMやGoogleは自社量子コンピュータを使う為の量子計算フレームワークを公開している。実機だけではなくシミュレーション機能を持っているので、量子プログラミングの勉強用として最適だが小規模の量子回路のみとなる。

項目	IBM Qiskit	Google Cirq
ロゴ	 Quantum Information Science Kit	 Cirq
構成	Terra: 量子計算の基盤部（Python） Aqua: 量子アルゴリズムのライブラリ OpenQASM: 量子低レベル中間言語	Cirq:量子計算基盤Pythonライブラリ OpenFermion: 量子化学ライブラリ
提供	オープンソース（GitHub）	オープンソース（GitHub）
取得	https://qiskit.org/ https://github.com/Qiskit	https://github.com/quantumlib/Cirq
情報	https://qiskit.org/documentation/ja/	https://cirq.readthedocs.io/en/latest/
その他	IBM Q Experience: GUI利用 ※ GUIからOpenQASMに展開し実行 https://quantumexperience.ng.bluemix.net/	2018年夏に正式公開されたライブラリ 量子コンピュータ実機はまだ使えない ※ 量子アニーリング計算も可能

量子計算フレームワーク（日本開発）

日本人や日本のベンチャーが開発した量子計算フレームワークも公開されている。まだ歴史は浅いが、後発の分使いやすさを重視しているように見える。

項目	Blueqat	OpenJij
ロゴ	Blueqat	OpenJij
構成	量子ゲート型の計算 量子アニーリング計算も可能	量子アニーリング型の計算
提供	オープンソース（GitHub）	オープンソース（GitHub）
取得	https://github.com/Blueqat	https://github.com/OpenJij
開発	株式会社 MDR https://mdrft.com/?hl=ja	株式会社 Jij https://j-ij.com/
その他	Qiskit/Cirqよりも回路記述が簡単。 前身は量子アニーリング用のWildqat。 日本語のSlackも参加可能。 勉強会・ブログ等で積極的に日本語で情報発信をしている。	2019年に本格的に公開開始した。 定式化された数式で入力可能。 日本語のチュートリアルも完備。 日本語のSlackも参加可能。 現在Windows上では動作しない。

Q-LEAP（光・量子飛躍フラッグシッププログラム）

https://www.jst.go.jp/stpp/q-leap/

文部科学省の日本国産プロジェクト。

今後10年で国内産学共同で100量子ビット（量子超越）の量子コンピュータやインフラ（開発環境）を開発する…予定だそうです。

人材育成も目的とのことですので、今後日本でもますます量子計算ができる人が求められる時代が来るでしょう。

プログラマの為の量子コンピュータ入門 目次:

Part 1: 関連数学と1量子ビット操作

- 線形代数学の基本知識
- ブラケット記法と量子計算
- ブロッホ球と1量子ビット操作（IBM Qiskit）

今回範囲

Part 2: 量子ゲート型のプログラミング

- 2量子ビット以上の量子回路（量子もつれ）
- 量子アルゴリズム（グローバ検索・フーリエ変換等）
- 量子ゲート型プログラミング（Google Cirq）

Part 3: 量子アニーリング型のプログラミング

- イジングモデルとQUBOと量子アニーリング
- 量子アルゴリズム（セールスマン巡回問題）
- 量子アニーリング型プログラミング（Blueqat他）

Part 1：関連数学と1量子ビット操作

参考：シュレディンガー方程式と波動関数

※ 理解できなくても量子プログラミングでの大きな問題にはなりません

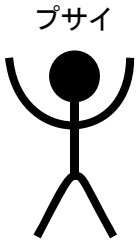
(時間依存する)シュレディンガー方程式

$\hbar$：プランク定数h / 2π

m：質量

$$i\hbar\frac{d}{dt}\psi(x,t) = H\psi(x,t) = \left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\right)\psi(x,t)$$

- $i\hbar\frac{d}{dt}$：時間から見た全エネルギー
- H：ハミルトニアン
- ψ：波動関数
- $-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}$：運動エネルギー
- $V(x)$：ポテンシャルエネルギー

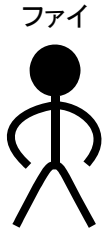


シュレディンガー方程式は、波動関数の値を求める方程式ではなく、
波動関数の式そのものを求める方程式となっている。

古典力学では粒子のエネルギーは保存される(時間依存が無い)ので、
時間発展しないシュレディンガー方程式を導くことができる。

$$E\phi(x) = H\phi(x) = \left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\right)\phi(x)$$

- $E\phi(x)$：エネルギー固有値
- $H\phi(x)$：φ:波動関数（時間依存無し）



http://scienceinoh.jp/schrodinger/

量子計算は複素数計算が必要

シュレディンガー方程式

$$i\hbar\frac{d}{dt}\psi(x,t) = H\psi(x,t)$$

左辺に虚数単位 i が出てくる。
この方程式を成立させる為には右辺の H が実数であるので、波動関数 ψ が複素数で表現されなければならい。

波動力学と行列力学

時間発展しないシュレディンガー方程式（波動力学）

$$H\varphi(x) = E\varphi(x)$$

Hはハミルトニアン式、φ(x)は波動関数（固有関数）、Eはエネルギー固有値

行列における固有値問題の式

$$Ax = \lambda x$$

Aは正方行列、xは列ベクトル（固有ベクトル）、λはスカラー値（固有値）

量子計算は行列の固有値問題として解くことができる。
これをハイゼンベルク方程式（行列力学）と呼ばれる。
この場合に波動関数は固有ベクトルとして求められる。

※ 波動力学と行列力学が数学的に同じであることはシュレディンガーにより確認されている。

量子の状態はベクトルで表現できる

量子系の状態ベクトル

n次元量子の状態 $=$
基底: $|0\rangle, |1\rangle, \ldots, |n\text{-}1\rangle$

$$\begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_{n-1} \end{pmatrix} = C_0|0\rangle + C_1|1\rangle + \ \ldots + C_{n-1}|n\text{-}1\rangle$$

ただし $|C_0|^2+|C_1|^2+|C_{n-1}|^2 = 1$

ただし C_0～C_{n-1}は複素数。

量子ビットの状態 $=$
基底: $|0\rangle$ と $|1\rangle$

$$\begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = C_0|0\rangle + C_1|1\rangle$$

ただし $|C_0|^2+|C_1|^2 = 1$

それぞれ確率50%なら: $C_0 = C_1 = \frac{1}{\sqrt{2}}$

$|0\rangle$ を得る確率は $|C_0|^2$
$|1\rangle$ を得る確率は $|C_1|^2$

1-1：線形代数学の基本知識

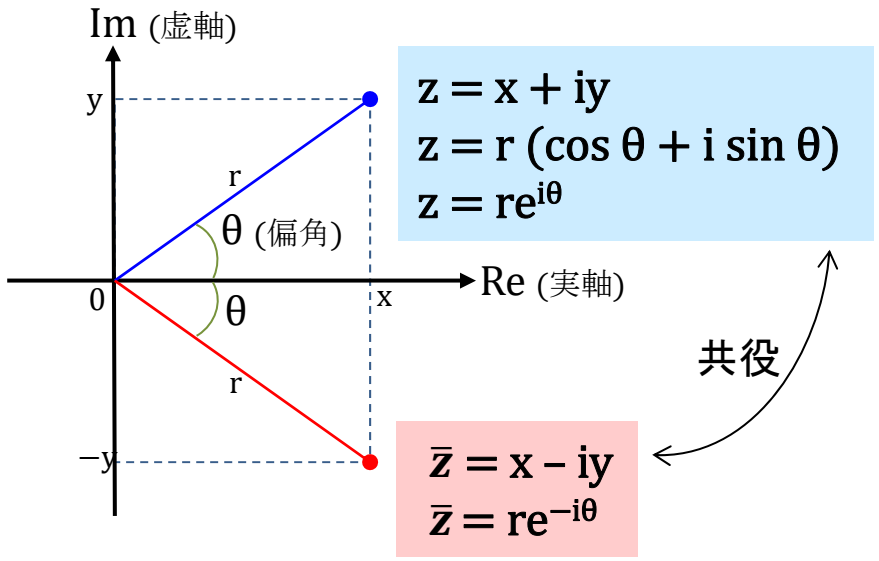
量子計算が行列演算で解けるということは、
行列演算の線形代数学の知識が必要です。

このパートは線形代数学をマスターしている人は
読み飛ばして頂いて構いません。

数学: 複素数と共役(複素共役)

名称	定義	例
実数	実在する数	$0, 2, 0.4, 1/3, \sqrt{2}$
虚数	2乗してマイナスになる数 i をつけて表す	$i = \sqrt{-1}, i3 = \sqrt{-3}, -i4 = -\sqrt{-4}$ $i^0 = 1, i^1 = i, i^2 = -1, i^3 = -i$
複素数	実数と虚数で示される数 実数部/虚数部が0も含む	$5 + i3, 2 - i4, -0.2 + i0.2,$ $-2 + i0 = -2, 0 - i3 = -3i$

複素平面における共役関係



複素共役(ふくそきょうやく)

虚数部がマイナスになっているペア値。

複素数 Z に対して $\overline{Z}$ と書く。

1. z が実数なら、$z = \bar{z}$
2. $\overline{(z_1 + z_2)} = \bar{z}_1 + \bar{z}_2$
3. $\overline{(z_1 \times z_2)} = \bar{z}_1 \times \bar{z}_2$
4. $\overline{(z_1 \div z_2)} = \bar{z}_1 \div \bar{z}_2$
5. $\bar{z}\, z = |z|^2$

数学: オイラーの公式

複素数の場合には指数関数と三角関数の相互変換が可能。

指数関数を三角関数で表す。

$$e^{i\theta} = \cos\theta + i\sin\theta$$

オイラーの公式の複素共役。

$$e^{-i\theta} = \cos\theta - i\sin\theta$$

三角関数を指数関数で表す。

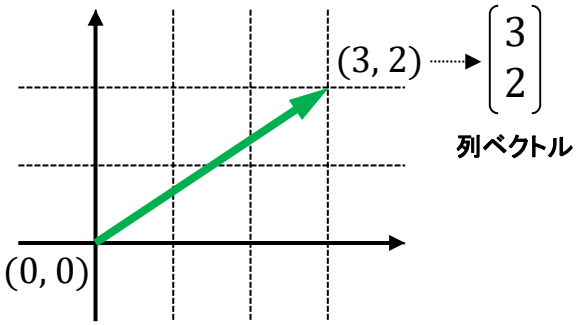
$$\sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}, \quad \cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2}$$

数学: ベクトルと基底ベクトル

ベクトル：始点/終点と向き/長さを持つ

- n次元の1行のみの行ベクトルや1列のみの列ベクトルとして扱う。
- 量子計算では原点を始点としたベクトルのみを扱う（線形変換）。
- 量子計算では座標を示す数は複素数。

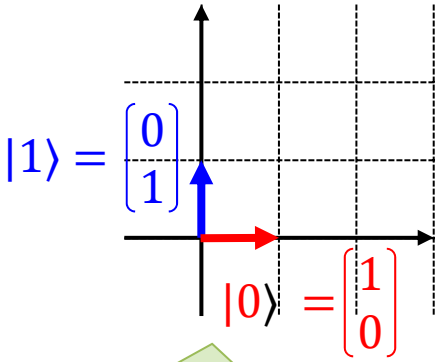
ベクトルと列ベクトル表記



基底ベクトル：基本単位となるベクトル

- 量子計算は $|0\rangle$ と $|1\rangle$ を規定ベクトルとする。「$|\psi\rangle$（ケット）」に関しては後述。
- 規定ベクトルを基準にすることで、移動や回転が表現できる（線形変換）。
- 量子計算では回転のみを利用する。

基底ベクトル



注：値は複素数
例：$1 = 1 + i0$

数学: ベクトルの演算

スカラー倍:

$$m\,(a_1\ a_2\ \dots\ a_n) = (ma_1\ ma_2\ \dots\ ma_n)$$

,

$$m\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} mb_1 \\ mb_2 \\ \vdots \\ mb_n \end{pmatrix}$$

ベクトル同士の和と差（同じ次元数同士のみ）:

$$(a_1\ a_2\ \dots\ a_n) + (b_1\ b_2\ \dots\ b_n) = (a_1{+}b_1\ a_2{+}b_2\ \dots\ a_n{+}b_n)$$

$$(a_1\ a_2\ \dots\ a_n) - (b_1\ b_2\ \dots\ b_n) = (a_1{-}b_1\ a_2{-}b_2\ \dots\ a_n{-}b_n)$$

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} a_1{+}b_1 \\ a_2{+}b_2 \\ \vdots \\ a_n{+}b_n \end{pmatrix}$$

,

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} a_1{-}b_1 \\ a_2{-}b_2 \\ \vdots \\ a_n{-}b_n \end{pmatrix}$$

数学: ベクトルの内積

$$\text{行ベクトル} \cdot \text{列ベクトル} = (a_1\ a_2\ \ldots\ a_n) \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

同じ次元間のみ
内積計算が可能

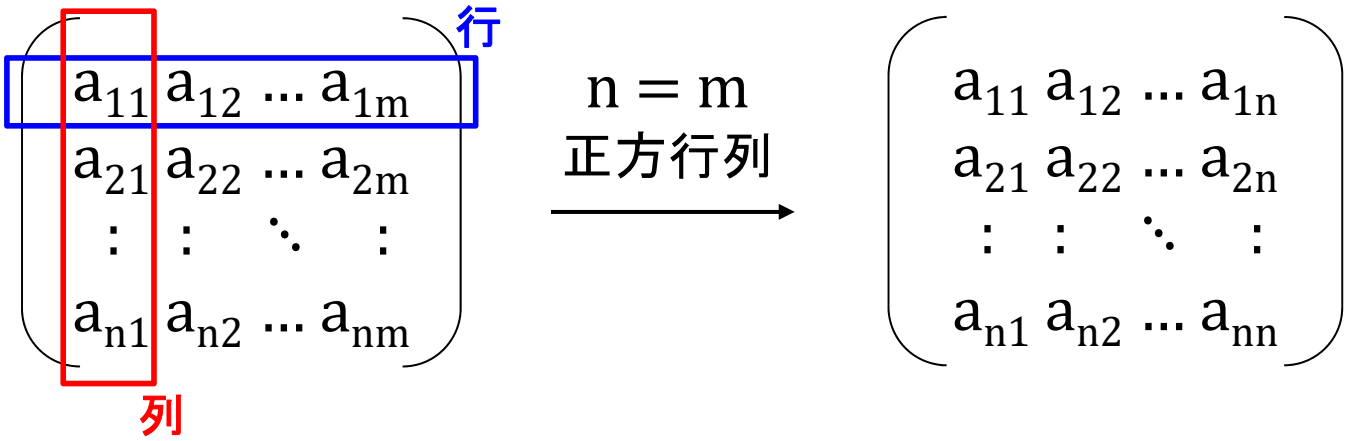
$$= a_1b_1 + a_2b_2 + \ldots + a_nb_n$$

内積の結果はベクトル
ではなくスカラー値になる
ただし複素数なので注意

$$= \sum_{j=0}^{n} a_jb_j$$

数学: 行列と正方行列

行列とは数・記号・式などを縦（列）と横（行）に並べたもの。
列数と行数が同じ場合には正方行列と呼び、量子計算においては正方行列のみを理解していれば良い。



$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \xrightarrow{n = m \text{ 正方行列}} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

ex1) 2x2 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$

ex2) 3x3 $\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}$

数学: 行列の演算

スカラー倍:

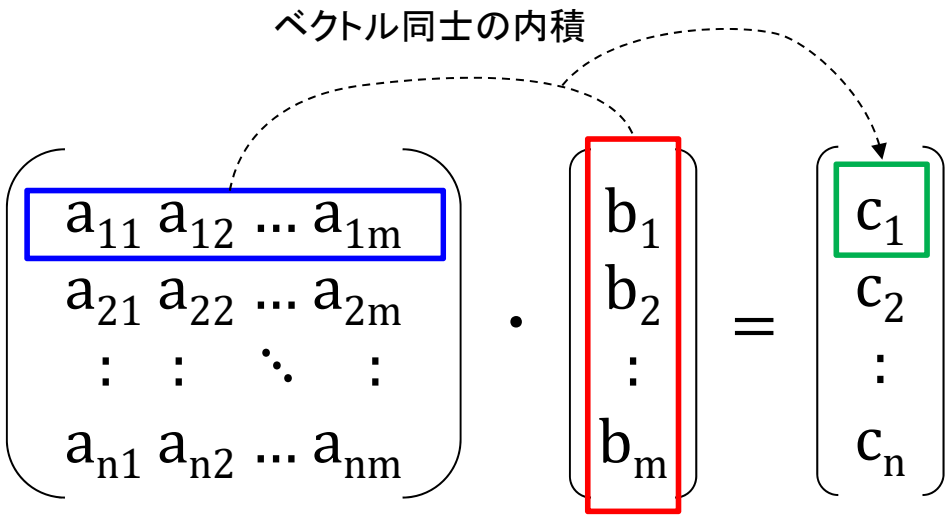
$$\lambda \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} \lambda a_{11} & \lambda a_{12} & \cdots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \cdots & \lambda a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda a_{n1} & \lambda a_{n2} & \cdots & \lambda a_{nn} \end{pmatrix}$$

行列同士の和（同じ次元数同士のみ）:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \cdots & a_{nn}+b_{nn} \end{pmatrix}$$

数学: 行列と列ベクトルの内積

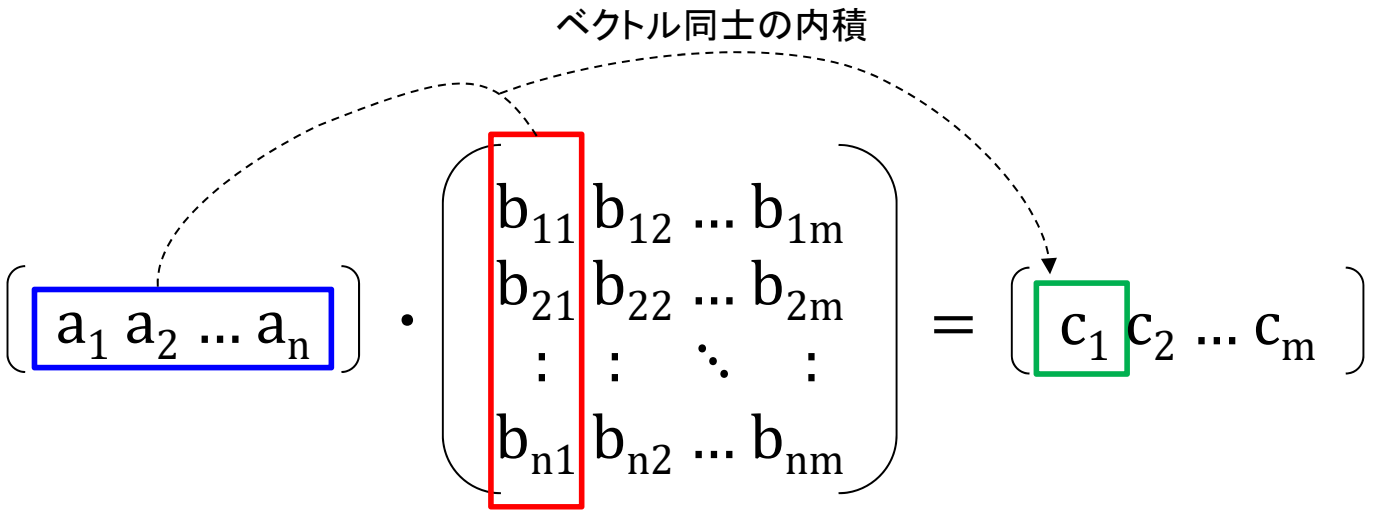
n行m列とm次の列ベクトルの内積はn次の列ベクトルになる:



$$
\begin{aligned}
c_1 &= a_{11}b_1 + a_{12}b_2 + \cdots + a_{1m}b_m \\
c_2 &= a_{21}b_1 + a_{22}b_2 + \cdots + a_{2m}b_m \\
&\vdots \\
c_n &= a_{n1}b_1 + a_{n2}b_2 + \cdots + a_{nm}b_m
\end{aligned}
\qquad \rightarrow \quad c_1 = \sum_{j=1}^{m} a_{1jbj}
$$

数学: 行ベクトルと行列の内積

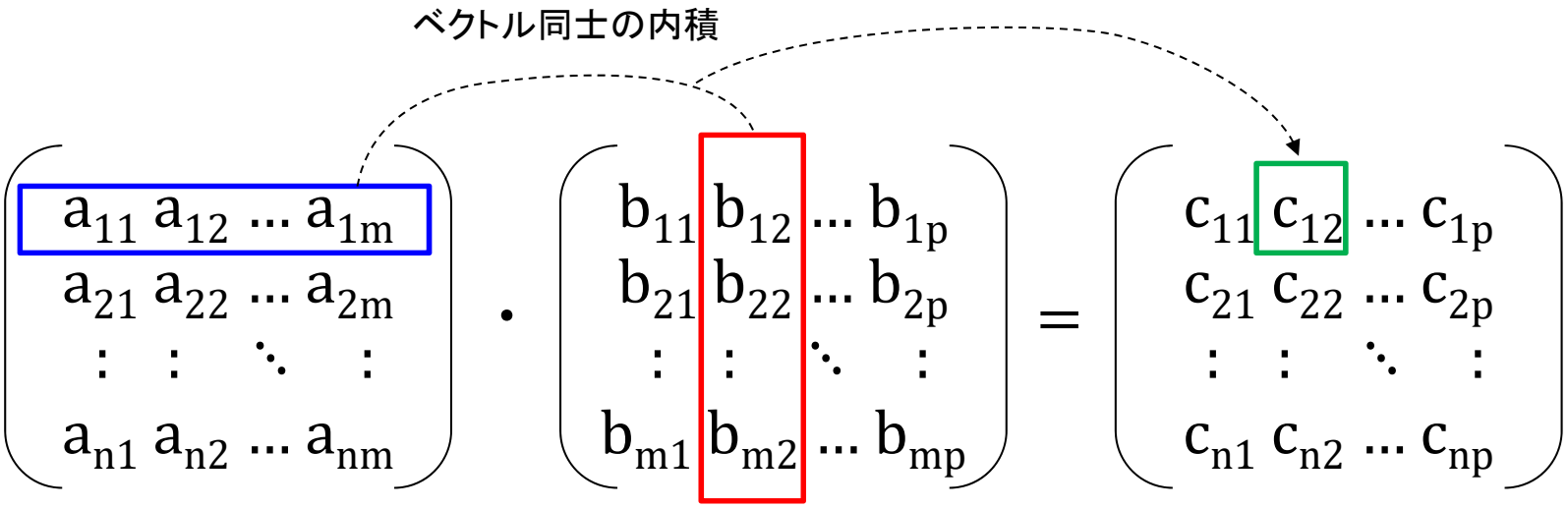
n次の行ベクトルとn行m列の内積はm次の行ベクトルになる:



$$
\begin{aligned}
c_1 &= a_1 b_{11} + a_2 b_{21} + \cdots + a_n b_{n1} \\
c_2 &= a_1 b_{12} + a_2 b_{22} + \cdots + a_n b_{n2} \\
&\vdots \\
c_m &= a_1 b_{1m} + a_2 b_{2m} + \cdots + a_n b_{nm}
\end{aligned}
\qquad \rightarrow \quad c_1 = \sum_{j=1}^{n} a_j b_{j1}
$$

数学: 行列同士の内積

n行m列とn行m列との内積はn行m列の行列になる:



$$
\begin{aligned}
c_{11} &= a_{11}b_{11} + a_{12}b_{21} + \ldots + a_{1m}b_{m1} \\
c_{12} &= a_{11}b_{12} + a_{12}b_{22} + \ldots + a_{1m}b_{m2} \\
&\vdots \\
c_{np} &= a_{n1}b_{1p} + a_{n2}b_{2p} + \ldots + a_{nm}b_{mp}
\end{aligned}
$$

数学: 単位行列

単位行列とは対角の値が1でそれ以外の値が0の正方行列:

Eはシュレディンガー方程式のエネルギー固有値Eとは異なるので注意すること。

$$\text{単位行列E} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

ex1) 2x2

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

ex2) 4x4

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

単位行列Eの性質:

正方行列Aを単位行列Eにかけると元の正方行列Aのまま

$$AE = EA = A$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

数学: 逆行列

逆行列とは元の正方行列をかけると単位行列になる正方行列:

行列Aの逆行列を A^{-1}(インバース)と書く

逆行列A^{-1}の性質:

正方行列Aに逆行列A^{-1}をかけると単位行列Eになる

$$A\,A^{-1} = A^{-1}\,A = E$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

ex) 3x3

$$\begin{pmatrix} -1 & 1 & 1 \\ 1 & 0 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & 1 & 1 \\ 1 & 0 & -1 \\ 0 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

数学: 行列式と逆行列の計算

行列式は正方行列を式としてスカラー値を得る計算ができる:

行列A

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

行列式A

$$|A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

3x3

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{21}a_{32}a_{13} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{23}a_{32}a_{11}$$

逆行列の計算:

$$A^{-1} = \frac{1}{|A|}\hat{A}$$

余因子行列:Â(エーハット)

$$\hat{A} = \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

ここではこれ以上
詳しく説明しない

数学: 固有値の計算（固有値問題1）

$$Ax = \lambda x$$

Aは正方行列、xは固有ベクトル（列ベクトル）、λは固有値（スカラー値）

固有値 λ_1 λ_2 を求める計算例:
※ 2次元の場合の固有値は2つ（重解なら1つ）。

$A = \begin{pmatrix} 1 & 2 \\ -1 & 4 \end{pmatrix}$ $\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix}$ とした場合、

$$\begin{pmatrix} 1 & 2 \\ -1 & 4 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}$$

単位行列を使って展開

$$= \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

右辺を左辺に移動

$$\begin{pmatrix} 1 & 2 \\ -1 & 4 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1-\lambda & 2 \\ -1 & 4-\lambda \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1-\lambda & 2 \\ -1 & 4-\lambda \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

この行列がゼロである必要がある

X ≠ 0 かつ Y ≠ 0

$$\begin{vmatrix} 1-\lambda & 2 \\ -1 & 4-\lambda \end{vmatrix} = 0$$

$$(1-\lambda)(4-\lambda) + 2 = 0$$

$$\lambda^2 - 5\lambda + 6 = 0$$

$$(\lambda-2)(\lambda-3) = 0$$

固有値:
$\lambda_1 = 2$
$\lambda_2 = 3$

数学: 固有ベクトルの計算（固有値問題2）

ex)

$$\begin{pmatrix} 1-\lambda & 2 \\ -1 & 4-\lambda \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

の固有値は $\lambda_1 = 2$ と $\lambda_2 = 3$ である。

この時のそれぞれの固有ベクトルを計算:

$\lambda_1 = 2$ の時:

$$\begin{pmatrix} 1-2 & 2 \\ -1 & 4-2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 & 2 \\ -1 & 2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow -x + 2y = 0$$

$-x + 2y = 0$ の固有ベクトル解:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$ （の定数倍）

$\lambda_2 = 3$ の時:

$$\begin{pmatrix} 1-3 & 2 \\ -1 & 4-3 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -2 & 2 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow -x + y = 0$$

$-x + y = 0$ の固有ベクトル解:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$ （の定数倍）

数学: 行列の対角化（固有値問題3）

$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ と、$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ から行列Pを、$P = \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix}$ とすると、

固有値λを対角化した行列と、$AP = P\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$ が成り立つ。

確認: $\underbrace{\begin{pmatrix} 1 & 2 \\ -1 & 4 \end{pmatrix}}_{A}\underbrace{\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}}_{P} = \underbrace{\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}}_{P}\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} = \begin{pmatrix} 4 & 3 \\ 2 & 3 \end{pmatrix}$

$AP = P\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$ の両辺にP^{-1}を掛けて、$P^{-1}AP = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$

が成り立つ。

数学: 転置行列と複素共役行列

行列の転置とは行と列を入れ替える操作:

行列Aの転置行列を A^T (トランスポーズ)と書く

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \qquad A^T = \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

行列の複素共役は虚数部を反転する操作:

行列Bの複素共役行列を $\overline{B}$ (バー)と書く

ex)
3x3

$$B = \begin{pmatrix} 1 & 2i & 3 \\ 4 & 5-3i & -6i \\ 7i & 8+2i & 9 \end{pmatrix} \qquad \overline{B} = \begin{pmatrix} 1 & -2i & 3 \\ 4 & 5+3i & 6i \\ -7i & 8-2i & 9 \end{pmatrix}$$

数学: 共役転置行列（または随伴行列）

共役転置行列は、行列の複素共役と転置を行う:

行列Aの共役転置行列を A^*（スター）と書く

$$A^* = \overline{A}^T$$

物理学の定義: $A^\dagger = [\,A^T\,]^*$

ex)
3x3

$$B = \begin{pmatrix} 1 & 2i & 3 \\ 4 & 5-3i & -6i \\ 7i & 8+2i & 9 \end{pmatrix} \quad B^T = \begin{pmatrix} 1 & 4 & 7i \\ 2i & 5-3i & 8+2i \\ 3 & -6i & 9 \end{pmatrix}$$

$$B^* = \overline{B}^T = \begin{pmatrix} 1 & 4 & -7i \\ -2i & 5+3i & 8-2i \\ 3 & 6i & 9 \end{pmatrix}$$

数学: 正規行列

元行列と共役転置行列を掛けて、交換法則(可換)が成り立つ場合に、その行列を正規行列と呼ぶ:

$$A^* A = A A^*$$

物理学の定義: $A^\dagger A = A A^\dagger$

ex)
3x3

$$B = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 2 \\ 2 & 0 & 0 \end{pmatrix} \qquad B^* = \begin{pmatrix} 0 & 0 & 2 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

$$B^* B = \begin{pmatrix} 0 & 0 & 2 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix} \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 2 \\ 2 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

$$B B^* = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 2 \\ 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 2 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

数学: ユニタリ行列

元行列と共役転置行列を掛けると単位行列Eになる行列:
※ 逆行列と共役転置行列が等しくなる。

正規行列

$$U^* U = U U^* = E \Rightarrow A^{-1} = A^*$$

物理学の定義: $U^\dagger U = U U^\dagger$ (ダガー) $= I$

ex)
$$A = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}$$

$$A^* A = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$A A^* = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

数学: エルミート行列

エルミート行列は元行列と共役転置行列が同じ行列:

$$A = A^* \Rightarrow \overbrace{A^* A = A A^*}^{\text{正規行列}} = A A = A^* A^*$$

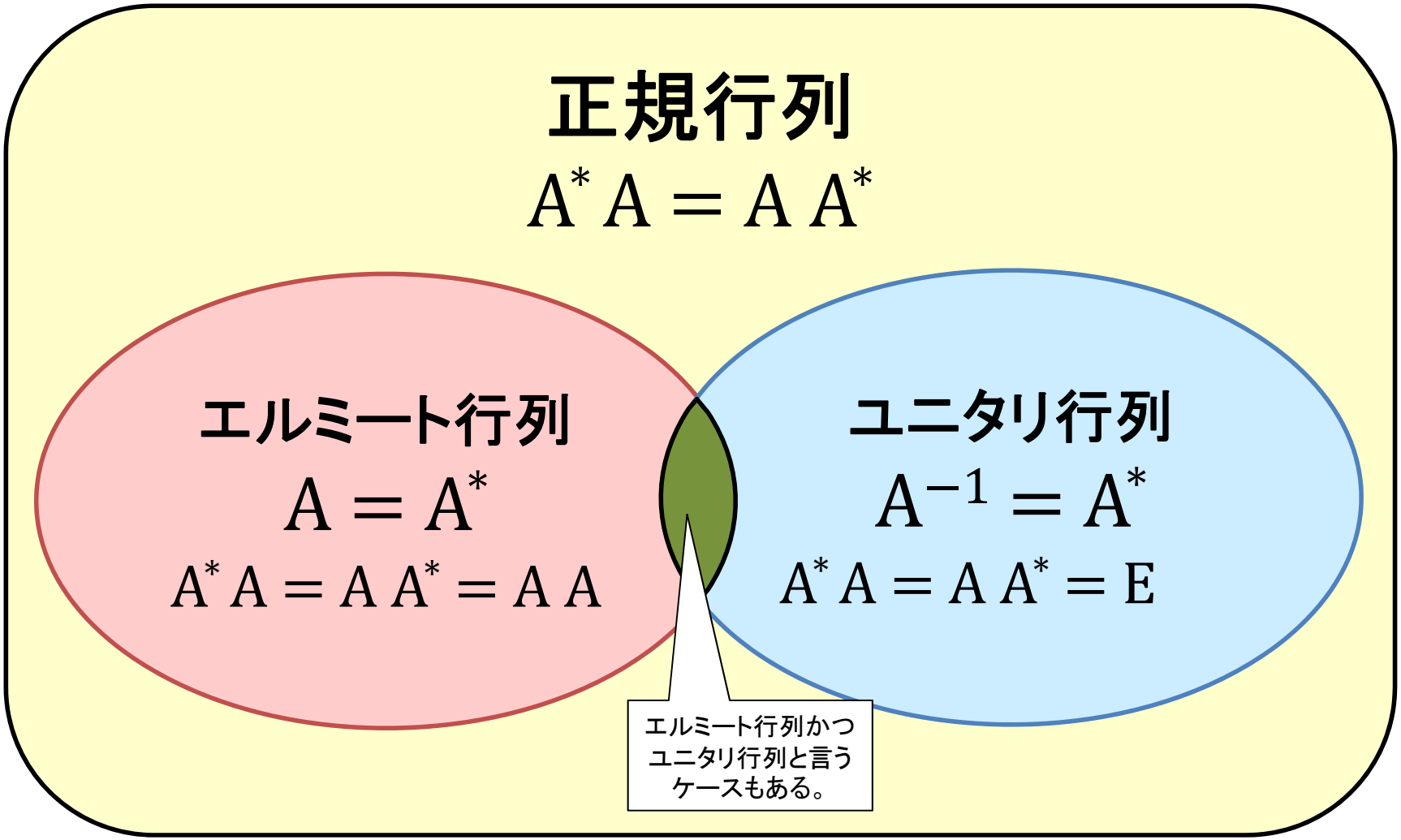
物理学の定義:行列 A のエルミート行列は $A^\dagger$ (ダガー)

ex) 2x2
$$\begin{pmatrix} 1 & 2+i \\ 2-i & 3 \end{pmatrix}$$

ex) 3x3
$$\begin{pmatrix} 1 & 2-i & 3+i \\ 2+I & 5 & 4+i \\ 3-i & 4-I & 6 \end{pmatrix}$$

エルミート行列の対角は実数になる。
エルミート行列の固有値は実数になる。

数学: 正規行列・エルミート行列・ユニタリ行列



1-2: ブラケット記法と量子計算

このパートからいよいよ量子計算関連する話になります。

と同時に数学から物理学へ話が変わります。

物理学と数学の違い

紛らわしい…特にA^*… このページ以降は物理学の 作法で記述して行きます。	物理学 （量子力学）	数学 （線形代数学）
複素共役 （虚数部の±反転）	A^* （スター）	$\overline{A}$ （バー）
複素共役転置	$A^\dagger$ （ダガー）	A^* （スター）
エルミート （自己随伴）	$A = A^\dagger$	$A = A^*$
ユニタリ	$U^{-1} = U^\dagger$	$U^{-1} = U^*$
単位行列	I （id:単位ゲート）	E

ブラケット(BraKet)記法

⟨Bra| ：ブラ(行)ベクトル
|Ket⟩ ：ケット(列)ベクトル

n次元ブラ:
(行)ベクトル $\langle\varphi| = \begin{pmatrix} d_0 & d_1 & \cdots & d_n \end{pmatrix}$,

n次元ケット:
(列)ベクトル $|\psi\rangle = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix}$

量子計算では
基底ベクトルとして
1 と 0 の2次元の
ブラケットを利用。

$\langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix}$

$\langle 1| = \begin{pmatrix} 0 & 1 \end{pmatrix}$

$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

量子計算結果の
|0⟩ と |1⟩ が重要

ブラケットベクトルの内積と外積

n次φブラ：（行ベクトル）
$$\langle\varphi| = \begin{pmatrix} d_0 & d_1 & \cdots & d_n \end{pmatrix}$$

,

n次ψケット：（列ベクトル）
$$|\psi\rangle = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix}$$

がある時に、

内積：
$$\langle\varphi|\psi\rangle = \begin{pmatrix} d_0 & d_1 & \cdots & d_n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = d_0c_0 + d_1c_1 + \cdots + d_nc_n = \sum_{j=0}^{n} d_jc_j$$

結果はスカラー

外積：
$$|\psi\rangle\langle\varphi| = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} \begin{pmatrix} d_0 & d_1 & \cdots & d_n \end{pmatrix} = \begin{pmatrix} c_0d_0 & c_0d_1 & \cdots & c_0d_n \\ c_1d_0 & c_1d_1 & \cdots & c_1d_n \\ \vdots & \vdots & \ddots & \vdots \\ c_nd_0 & c_nd_1 & \cdots & c_nd_n \end{pmatrix}$$

結果は行列

同じベクトルの内積

ブラベクトル：$\langle\varphi| = \begin{pmatrix} a_0^* & a_1^* & \dots & a_n^* \end{pmatrix}$，ケットベクトル：$|\varphi\rangle = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$

がある時に内積計算は、

$$\langle\varphi|\varphi\rangle = \begin{pmatrix} a_0^* & a_1^* & \dots & a_n^* \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$$

$$= a_0^* a_0 + a_1^* a_1 + \dots + a_n^* a_n$$

量子計算では、内積の結果が1となるように規格化されている。

$$\langle\varphi|\varphi\rangle = 1$$

※ これはベクトルの長さが1と言うことと同じ。

テンソル積（1階のテンソル積）

2つの2次元ベクトル $|\psi_1\rangle$ と $|\psi_2\rangle$ がある時、

$$|\psi_1\rangle = \begin{pmatrix} a \\ b \end{pmatrix} = a|0\rangle + b|1\rangle \qquad |\psi_2\rangle = \begin{pmatrix} c \\ d \end{pmatrix} = c|0\rangle + d|1\rangle$$

$|\psi_1\rangle$ と $|\psi_2\rangle$ のテンソル積⊗は4次元ベクトルとなる。

$$|\psi_1\rangle \otimes |\psi_2\rangle = \begin{pmatrix} a\begin{pmatrix} c \\ d \end{pmatrix} \\ b\begin{pmatrix} c \\ d \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix} = \begin{array}{l} ac|00\rangle + ad|01\rangle \\ \quad + bc|10\rangle + bd|11\rangle \end{array}$$

テンソル積の記号⊗は省略されることが多い。

$$|\psi_1\rangle \otimes |\psi_2\rangle \quad \Rightarrow \quad |\psi_1\rangle|\psi_2\rangle \quad \Rightarrow \quad |\psi_1\psi_2\rangle$$

複数量子ビットとテンソル積

複数量子ビットは
テンソル積で表せる。

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$ であるので、

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$|000\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad |001\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{...(略)...} \quad |111\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$|111\rangle = |7\rangle$
と表示する場合もある

演算子（正方行列）

ブラケットを利用した計算の演算子は正方行列で表す。
演算子をA（正方行列）とすると、ケットは左から、ブラは右から
演算子を適用して、新しいケットやブラに変換ができる。

相対関係 ---------- 複素共役転置になる

$$A|\psi\rangle = |\psi'\rangle \quad \longleftrightarrow \quad \langle\psi|A^{\dagger} = \langle\psi'|$$

ex)

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad |\psi\rangle = |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

ビット反転演算子

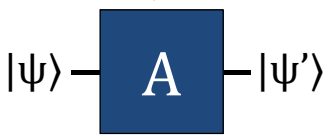
の時、

$$A|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

なので、 $|\psi'\rangle = |1\rangle$ となる。

$A|\psi\rangle = |\psi'\rangle$

量子回路的表現



ユニタリ(Unitary)演算

量子計算の演算子は全てユニタリ演算子 U となる。

$U^{\dagger} = U^{-1}$ $\left(U^{\dagger}U = UU^{\dagger} = I \right)$ の時、

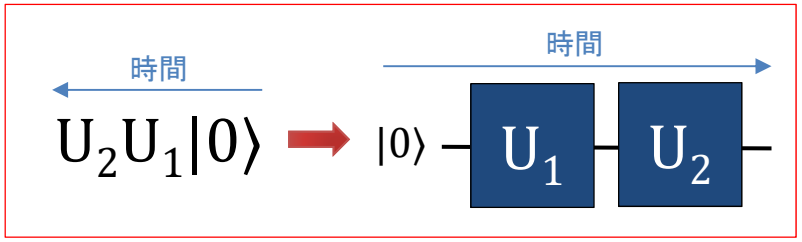
ユニタリ演算により、$U|\psi\rangle = |\psi'\rangle$ となる。

ユニタリ演算子は**回転操作**(長さは変化しない)演算である。
この為に複数のユニタリ演算子を時間的に順番に利用できる。

ex)

位相反転演算子

$$U_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad U_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$



$$U_2U_1|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle$$

マイナスは位相反転

エルミート(Hermitian)演算

エルミート行列Hは、

$$H^{\dagger} = H \quad \left(H^{\dagger}H = HH^{\dagger} = HH \right)$$ である。

2つの固有ベクトル φ_i と φ_j と、その固有値 λ_i と λ_j がある時、

式1: $H|\varphi_i\rangle = \lambda_i|\varphi_i\rangle$ と、式2: $H|\varphi_j\rangle = \lambda_j|\varphi_j\rangle$ となる。

式1の左辺から $\langle\varphi_j|$ を適用し $\langle\varphi_j|H|\varphi_i\rangle = \lambda_i\langle\varphi_j|\varphi_i\rangle$ となる。

式2を複素共役転置して式3: $\langle\varphi_j|H^{\dagger} = \langle\varphi_j|H = \lambda_j^{*}\langle\varphi_j|$ とし、

式3の右辺から $|\varphi_i\rangle$ を適用し $\langle\varphi_j|H|\varphi_i\rangle = \lambda_j^{*}\langle\varphi_j|\varphi_i\rangle$ となる。

結果: $\langle\varphi_j|H|\varphi_i\rangle = \lambda_i\langle\varphi_j|\varphi_i\rangle = \lambda_j^{*}\langle\varphi_j|\varphi_i\rangle$ より、

$(\lambda_i - \lambda_j^{*})\langle\varphi_j|\varphi_i\rangle = 0$ を得る。

$i = j$ なら $\lambda_i = \lambda_i^{*}$ で λ_i は実数、規格化より $\langle\varphi_i|\varphi_i\rangle = 1$ となる。
$i \neq j$ なら $\lambda_i - \lambda_j^{*}$ は0ではなく、$\langle\varphi_j|\varphi_i\rangle = 0$ (直交)となる。

演算子の行列表現（対角化）

エルミート演算を波動ベクトルで囲った行列を演算子の行列表現と呼ぶ。ここでは演算を $\langle\varphi_n|H|\varphi_n\rangle$ で（n=0,1,2,...,n）とすると、

$$\begin{pmatrix} H_{00} & H_{01} & \cdots & H_{0n} \\ H_{10} & H_{11} & \cdots & H_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ H_{n0} & H_{n1} & \cdots & H_{nn} \end{pmatrix} = \begin{pmatrix} \langle\varphi_0|H|\varphi_0\rangle & \langle\varphi_0|H|\varphi_1\rangle & \cdots & \langle\varphi_0|H|\varphi_n\rangle \\ \langle\varphi_1|H|\varphi_0\rangle & \langle\varphi_1|H|\varphi_1\rangle & \cdots & \langle\varphi_1|H|\varphi_n\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle\varphi_n|H|\varphi_0\rangle & \langle\varphi_n|H|\varphi_1\rangle & \cdots & \langle\varphi_n|H|\varphi_n\rangle \end{pmatrix}$$

$i = j$ なら $\lambda_i = \lambda_i^*$ で λ_i は実数、規格化より $\langle\varphi_i|\varphi_i\rangle = 1$ であり、
$i \neq j$ なら $\lambda_i - \lambda_j^*$ は0ではなく、$\langle\varphi_j|\varphi_i\rangle = 0$（直交）となるので、

$$\langle\varphi_0|H|\varphi_0\rangle = \lambda_0\langle\varphi_0|\varphi_0\rangle = \lambda_0$$
$$\langle\varphi_1|H|\varphi_1\rangle = \lambda_1\langle\varphi_1|\varphi_1\rangle = \lambda_1$$
$$\vdots$$
$$\langle\varphi_n|H|\varphi_n\rangle = \lambda_n\langle\varphi_n|\varphi_n\rangle = \lambda_n$$

$$H = \begin{pmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

エルミート演算による期待値

観測前ベクトル $|\psi\rangle$ と、観測後固有ベクトル $|\varphi_n\rangle$ と、固有値 λ_n があり、どちらのベクトルも規格化されている（長さが1）とする。
この時の期待値（観測される物理量）は $P = \langle\psi| H |\psi\rangle$ で計算可能。

$$\langle\psi| H |\psi\rangle = \begin{pmatrix} \langle\varphi_0|\psi\rangle^* & \langle\varphi_1|\psi\rangle^* & \cdots & \langle\varphi_n|\psi\rangle^* \end{pmatrix} \begin{pmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix} \begin{pmatrix} \langle\varphi_0|\psi\rangle \\ \langle\varphi_1|\psi\rangle \\ \vdots \\ \langle\varphi_n|\psi\rangle \end{pmatrix}$$

$$= \begin{pmatrix} \langle\varphi_0|\psi\rangle^* & \langle\varphi_1|\psi\rangle^* & \cdots & \langle\varphi_n|\psi\rangle^* \end{pmatrix} \begin{pmatrix} \lambda_0\langle\varphi_0|\psi\rangle \\ \lambda_1\langle\varphi_1|\psi\rangle \\ \vdots \\ \lambda_n\langle\varphi_n|\psi\rangle \end{pmatrix}$$

$$= \underbrace{\lambda_0|\langle\varphi_0|\psi\rangle|^2}_{\lambda_0\text{の期待値}} + \underbrace{\lambda_1|\langle\varphi_1|\psi\rangle|^2}_{\lambda_1\text{の期待値}} + \cdots + \underbrace{\lambda_n|\langle\varphi_n|\psi\rangle|^2}_{\lambda_n\text{の期待値}}$$

ボルンの規則

量子系の任意ベクトル ψ の物理量(オブザーバブル)の観測時に、各測定値(固有値λ_n)が取る期待値は固有値のどれかとなり、また測定値(固有値λ_n・固有ベクトルφ_n)を得る確率は $|\langle\varphi_n|\psi\rangle|^2$ となる。

確率振幅を $c_n = \langle\varphi_n|\psi\rangle$ とした時に、全固有ベクトルは完全系をなすので、以下の式が成り立つ。なお c_n は複素数である。

$$|\psi\rangle = \sum_n c_n |\varphi_n\rangle$$
$$= c_0|\varphi_0\rangle + c_1|\varphi_1\rangle + \ldots + c_n|\varphi_n\rangle$$

$$|c_0|^2 + |c_1|^2 + \ldots + |c_n|^2 = 1$$

全確率の合計は1(100%)となる

よって任意のベクトルは取りえる全ての固有ベクトルで表せる。

※ 期待値が確率振幅の二乗となることをボルンの規則と呼ぶ。

- ボルンの規則は他の方程式から導けないが実験結果と一致するので現在主流となっている計算方法である。

有限準位量子力学系（量子ビット＝2次元）

量子ビットは2つの固有ベクトル･固有値を持つ。
2つの固有（基底）ベクトルを $|0\rangle$ と $|1\rangle$ とする。

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

任意の量子ベクトル ψ に関して確率振幅 c_n を
使ってボルンの規則により以下の式が成り立つ、

確率振幅
c_0 と c_1 は
複素数

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle$$

$$|c_0|^2 + |c_1|^2 = 1$$

$$c_0 = \langle 0|\psi\rangle \qquad c_1 = \langle 1|\psi\rangle$$

重要!

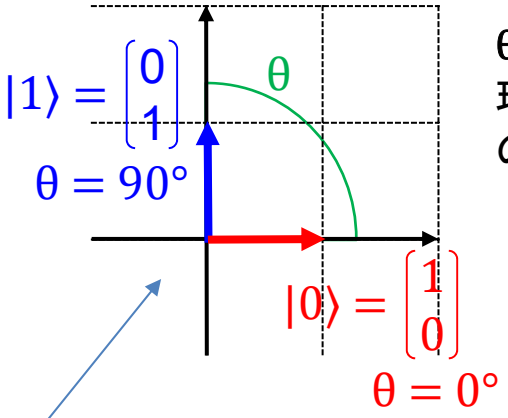
1-3: ブロッホ球と1量子ビット操作

お待たせしました！
このパートからやっと量子計算の話になります。

ここからが本題です！

量子ビットとブロッホ球

基底ベクトル



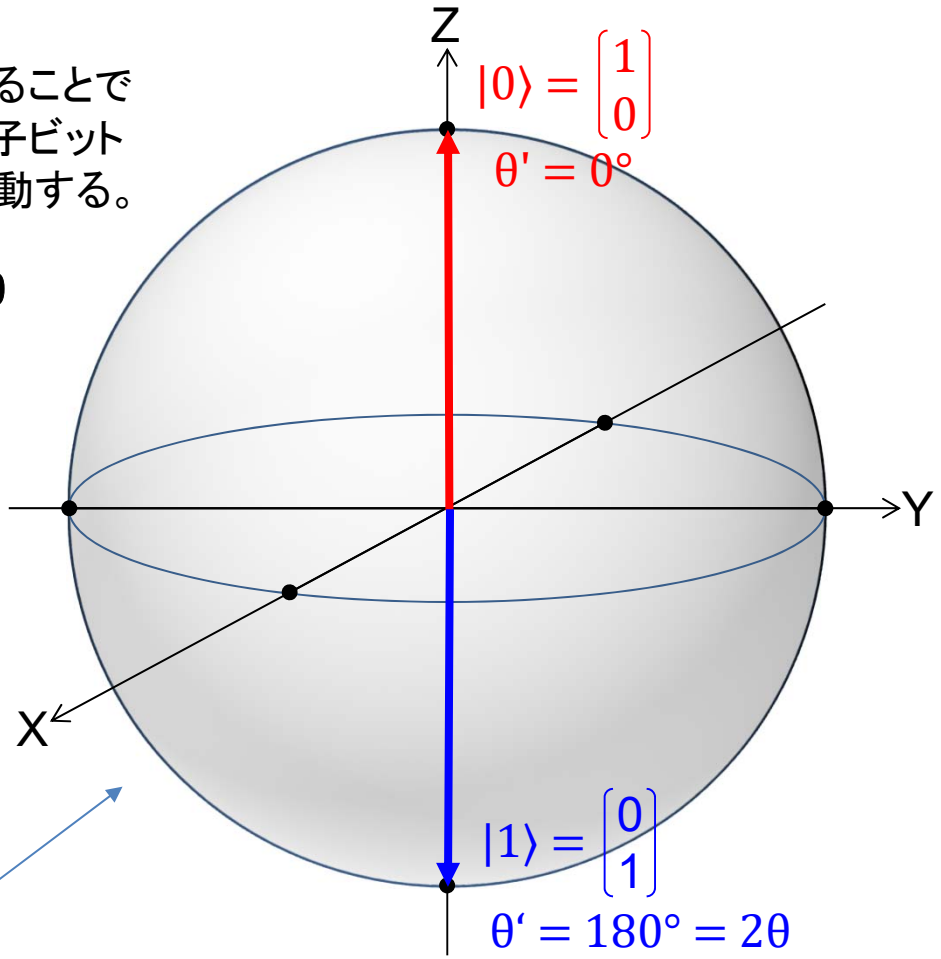
これでは上半球の部分しか
使えない…また虚数部がある
ので4次元の空間となる。
量子状態のイメージが掴めない…

θ‘＝2θ とすることで
球面上を量子ビット
の状態が移動する。

θ' = 2θ

ブロッホ球

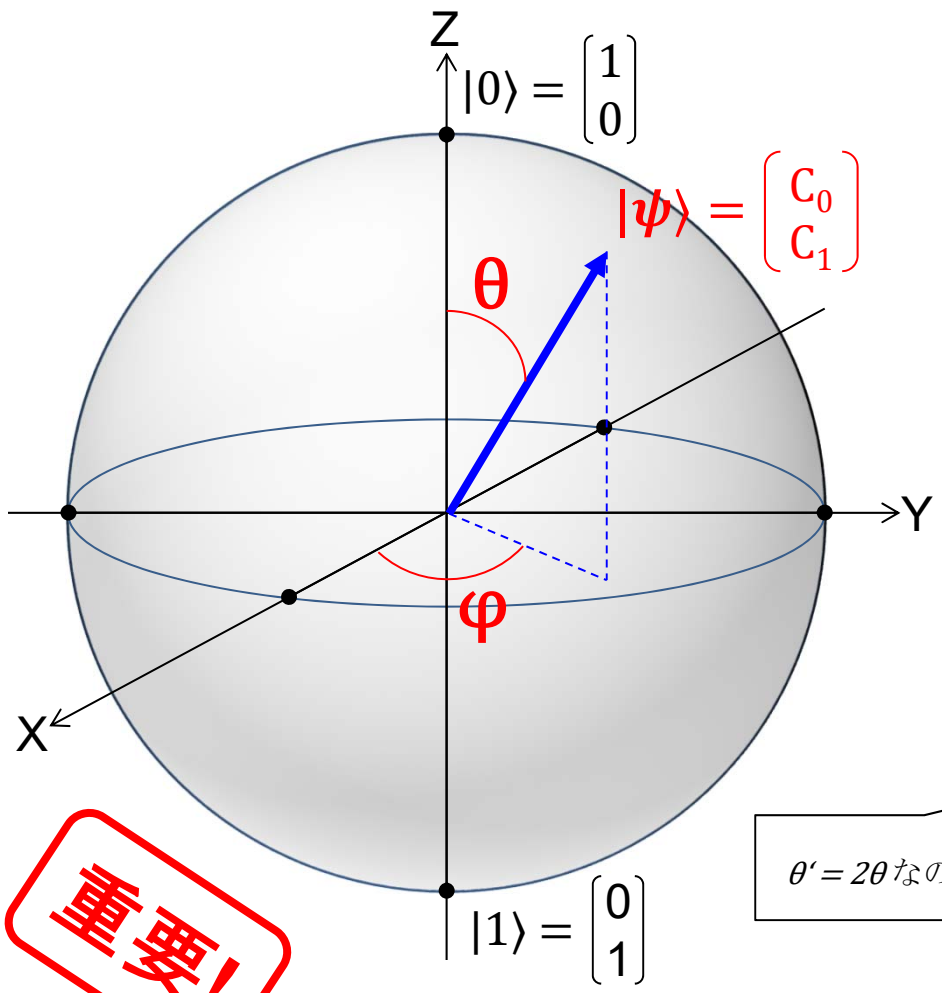
量子状態を単位球面上に表す表記法



虚数部（位相）は絶対値ではなく
干渉に関する相対値が必要。
なので|0⟩の位置をゼロとして、
|1⟩の虚数部（Z軸の回転θ）に
位相差を示し3次元空間にする。

ブロッホ球は複素ヒルベルト空間を示す。

量子ビットの存在確率と位相



重要!

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle$$

$$|c_0|^2 + |c_1|^2 = 1$$ 球面上の制約

$\|0\rangle$ 実数部	$z = cos\ \theta$
$\|0\rangle$ 虚数部	位相差を見るのでゼロ
$\|1\rangle$ 実数部	$x = cos\ \varphi\ sin\ \theta$
$\|1\rangle$ 虚数部	$y = sin\ \varphi\ sin\ \theta$

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + (cos\ \varphi + i\ sin\ \varphi)\ sin\frac{\theta}{2}|1\rangle$$

位相差

$$|\boldsymbol{\psi}\rangle = \boldsymbol{\cos\frac{\theta}{2}}|0\rangle + \boldsymbol{e^{i\varphi}\ sin\frac{\theta}{2}}|1\rangle$$

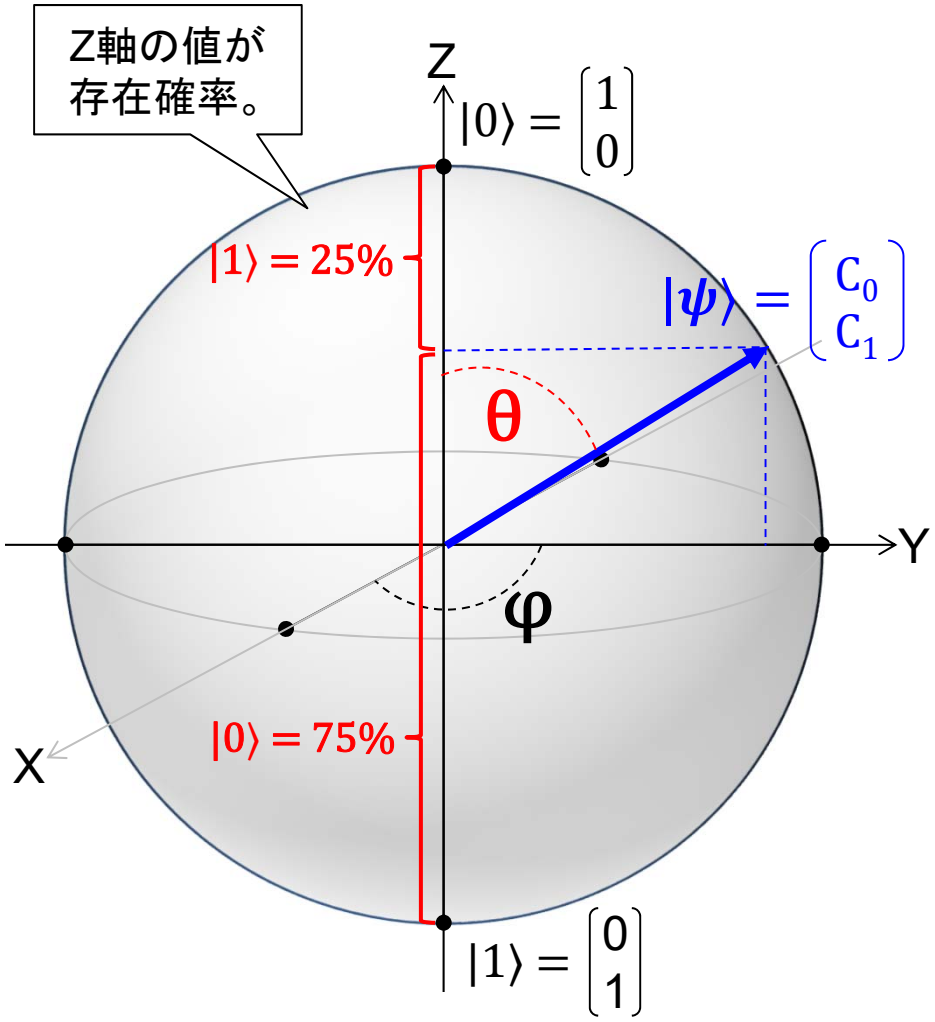
$\theta' = 2\theta$ なので $\frac{\theta}{2}$

二乗すると $|0\rangle$ の存在確立

二乗すると $|1\rangle$ の存在確立

※θは存在確率を、φは位相を示す。

存在確率の計算例（角度θのみに依存）



θ＝π/3＝60° の場合

$$|0\rangle = |\cos(\pi/3/2)|^2$$
$$= |\cos(\pi/6)|^2$$
$$= |0.86602\ldots|^2$$
$$= 0.75 = \mathbf{75\%}$$

$$|1\rangle = |\sin(\pi/3/2)|^2$$
$$= |\sin(\pi/6)|^2$$
$$= |0.5|^2$$
$$= 0.25 = \mathbf{25\%}$$

※ 位相 $e^{i\varphi}$ は虚数部なので1つの量子ビットでは影響しない。

量子ビットに対するユニタリ演算

ユニタリ行列による座標変換: $U|\psi\rangle = |\psi'\rangle$

id 恒等演算 $\mathrm{iden}(q) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

単位行列 I なので何もしない演算となる。

X ビット反転演算 $\mathrm{x}(q) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

Y 位相ビット反転演算 $\mathrm{y}(q) = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$

Z 位相反転演算 $\mathrm{z}(q) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

パウリ (Pauli) ゲート

位相のみを変換する場合は位相シフトゲートとも呼ぶ。

IBMのQiskitを使う

環境：Anaconda3（Python3.5）

以下より環境に合わせてダウンロードとインストール

https://www.anaconda.com/distribution/

ライブラリ：Qiskit（キスキット）

Windows版：Anaconda Prompt

MacOS版：ターミナル

インストール

```
pip install qiskit
```

バージョン指定インストール

```
pip install qiskit=0.10.5
```

アンインストール

```
pip uninstall qiskit
```

※ Qiskitのバージョン確認：

In:	import qiskit qiskit.__qiskit_version__
Out:	{'qiskit': '0.10.5', 'qiskit-terra': '0.8.2', 'qiskit-ignis': '0.1.1', 'qiskit-aer': '0.2.1', 'qiskit-ibmq-provider': '0.2.2', 'qiskit-aqua': '0.5.2'}

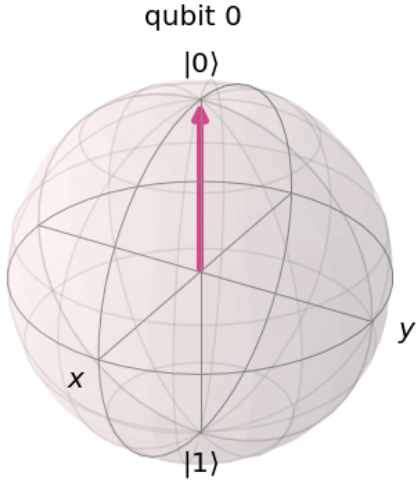
本資料のソースは 0.10.5 と表示される環境にて確認しています。

Qiskitでブロッホ球を表示する（恒等演算）

Jupiter Notebook から以下を実行(コピペで大丈夫です)。

```
from qiskit import *                              # 量子計算用
from qiskit.tools.visualization import *          # 結果表示用
backend = Aer.get_backend('statevector_simulator')  # シミュレータ指定
q = QuantumRegister(1)                            # 量子ビットを1つ用意
qc = QuantumCircuit(q)                            # 量子回路に量子ビットをセット
qc.iden(q[0])                                     # 恒等演算（初期値 |0⟩ を出力）
r = execute(qc, backend).result()                 # 回路実行して結果取得
psi = r.get_statevector(qc)                       # ステータス取得
print(psi)                                        # ステータス（ベクトル）表示
plot_bloch_multivector(psi)                       # ブロッホ球表示（※ Qiskit 0.7以降）
```

※ Qiskit 0.6 以前は plot_state(psi,"bloch") を使う。

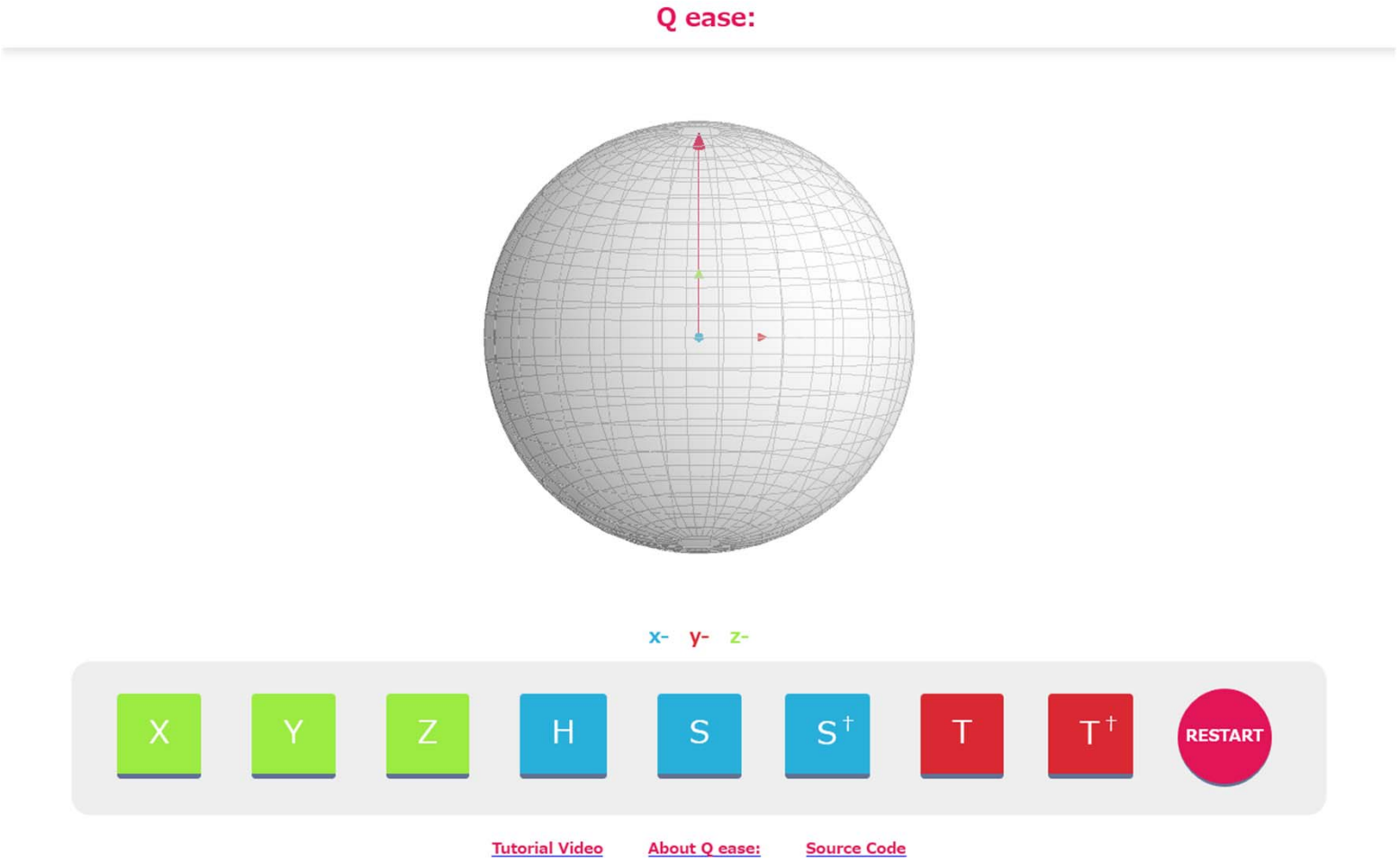


各量子ビットの初期状態は |0⟩ なので、左のように |0⟩ のブロッホ球が表示されるはず。

次ページからはプログラムの6行目の演算を変更することでユニタリ演算を確認して行く。

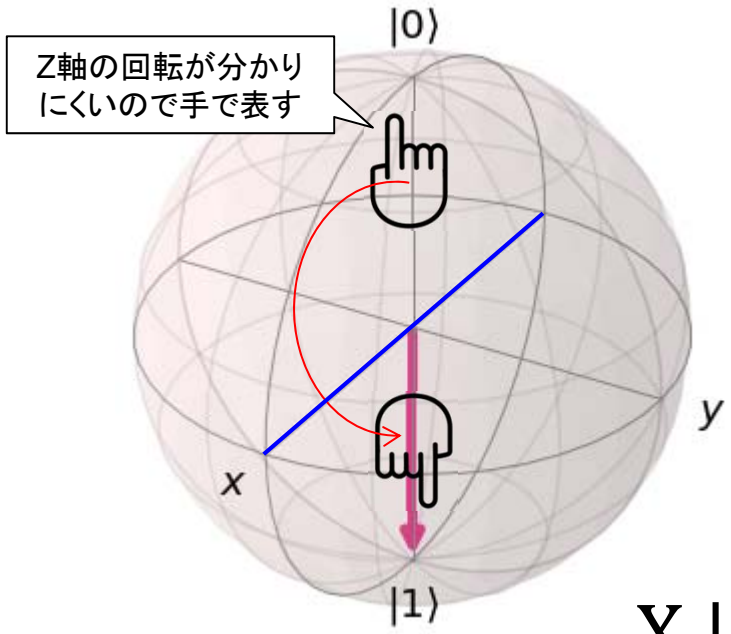
Webでブロッホ球を表示する

Try Bloch!
https://qease.herokuapp.com/bloch/try/



ビット反転演算 X ゲート

qc.x(q[0])



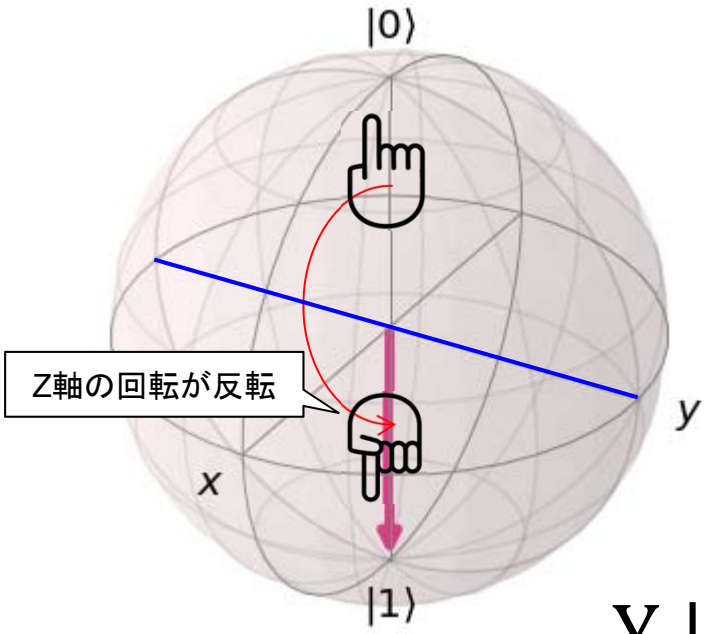
Xゲートの演算は、
ブロッホ球のX軸回りに、
180度(π)回転する。
ビットは反転するが、
位相φは変わらない。

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

位相ビット反転演算Yゲート

qc.y(q[0])



Yゲートの演算は、
ブロッホ球のY軸回りに、
180度(π)回転する。
ビットは反転し、
位相φも反転する。

虚数部は結果に影響しない

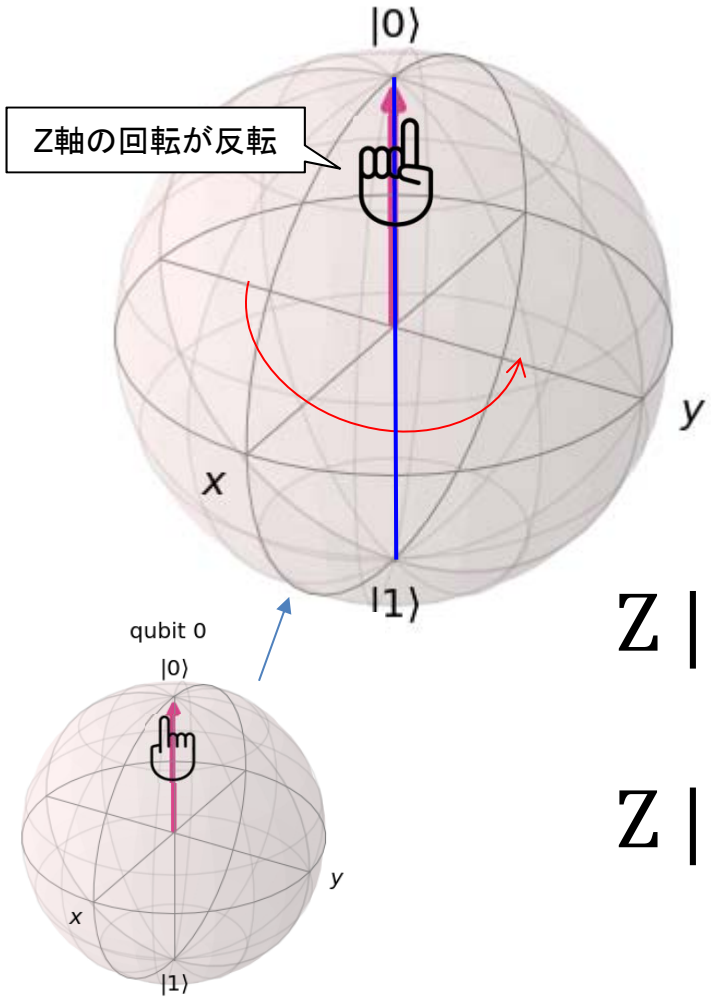
$$Y|0\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ i \end{pmatrix} = i|1\rangle$$

$$Y|1\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -i \\ 0 \end{pmatrix} = -i|0\rangle$$

マイナスが位相反転を示す

位相反転演算 Z ゲート

qc.z(q[0])



Zゲートの演算は、
ブロッホ球のZ軸回りに、
180度(π)回転する。
ビットは反転しないが、
位相φは反転する。

|0⟩の虚数部はゼロ固定

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle$$

|1⟩のマイナスが位相反転を示す

アダマール H ゲート（重ね合わせ状態の生成）

アダマールによる座標変換: $H|\psi\rangle = |\psi'\rangle$

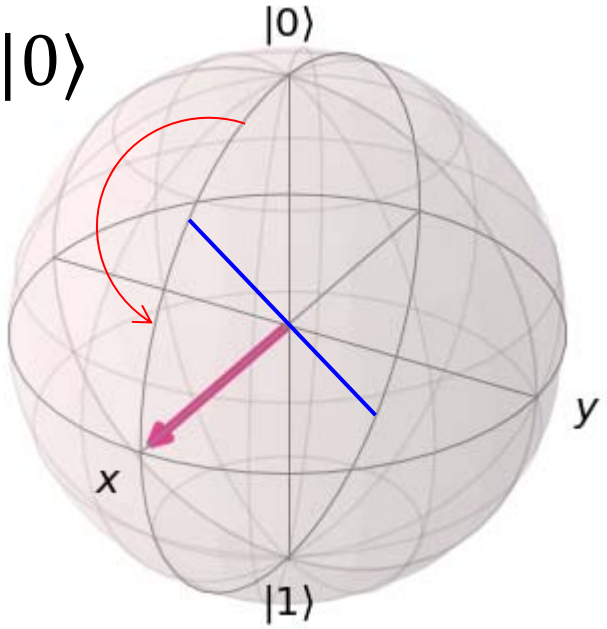
H　アダマール演算 $h(q) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Hadamard

$= \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$

qc.h(q[0])

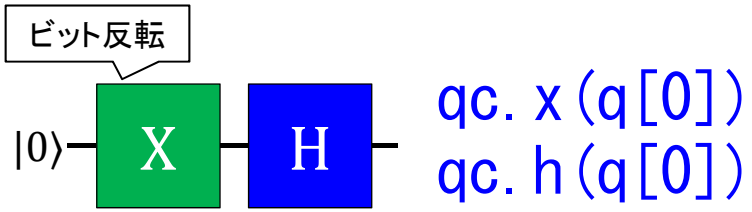
$H|0\rangle$



Hゲートの演算は、
ブロッホ球のXZ平面に、
45度傾いた軸回りに、
180度（π）回転する。

※ Y軸に90度回転する訳ではない。

|1⟩ へのアダマール H ゲート適用

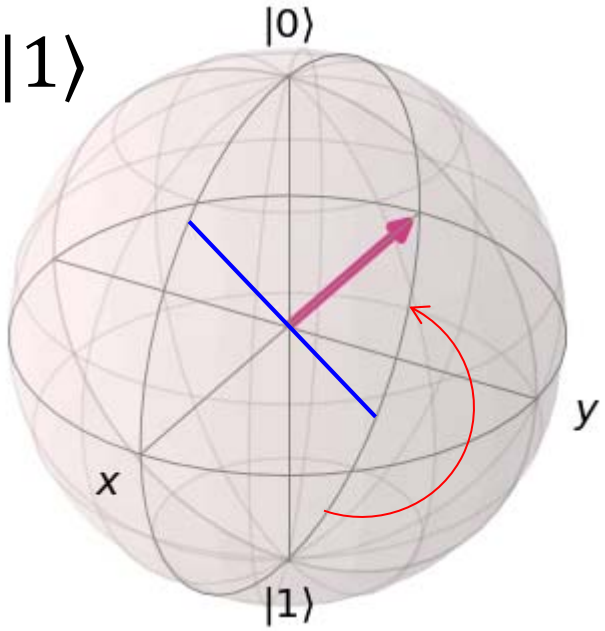



```
qc.x(q[0])
qc.h(q[0])
```

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix}$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$$

$H|1\rangle$

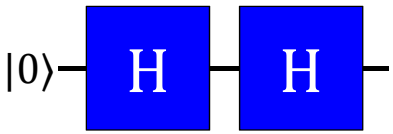


$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix}$$

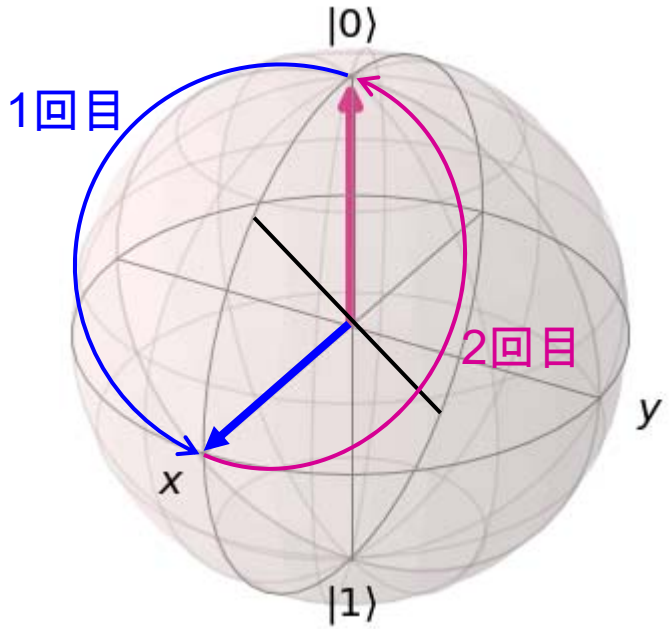
$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ -1\end{pmatrix} = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$$

$H|0\rangle$ と $H|1\rangle$ はどちらも $|0\rangle : 50\%$ と $|1\rangle : 50\%$ となり同じ確率分布になる。
ただし位相が逆になっている。

アダマール H ゲートを2回適用




```
qc.h(q[0])
qc.h(q[0])
```



$$HH = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I \text{（単位行列）}$$

$$HH|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

結論：2回アダマール演算を適用すると元のベクトルに戻る。

量子ゲートの組み合わせ

- アダマールゲートで挟むと変換が可能となる。

$HXH = Z$ ：ビット反転Xゲートを挟むと位相反転Zゲートに。

$HZH = X$ ：位相反転Zゲートを挟むとビット反転Xゲートに。

$HYH = -Y$ ：位相ビット反転Yゲートを挟むと位相のみ反転。

ex) HXHの計算

$$XH = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$H(XH) = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = Z$$

位相シフト S/S†/T/T† ゲート

Z　位相反転演算 $z(q) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

反転なので逆位相も同じ

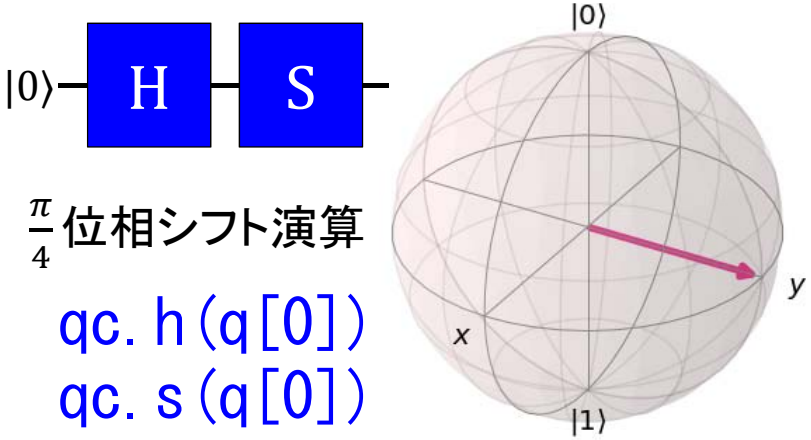
S　$\frac{\pi}{4}$ 位相シフト演算 $s(q) = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$

S†　$-\frac{\pi}{4}$ 位相シフト演算 $s^{\dagger}(q) = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}$

T　$\frac{\pi}{8}$ 位相シフト演算 $t(q) = \begin{pmatrix} 1 & 0 \\ 0 & \frac{(1+i)}{\sqrt{2}} \end{pmatrix}$

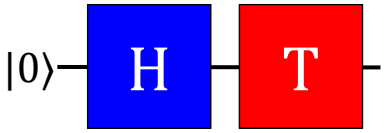
T†　$-\frac{\pi}{8}$ 位相シフト演算 $t^{\dagger}(q) = \begin{pmatrix} 1 & 0 \\ 0 & \frac{(1-i)}{\sqrt{2}} \end{pmatrix}$

重ね合わせ状態での位相シフト



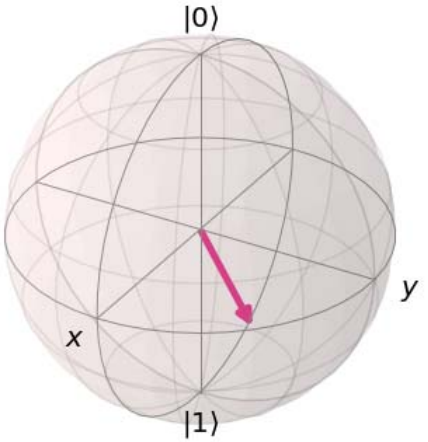
$\frac{\pi}{4}$位相シフト演算

```
qc.h(q[0])
qc.s(q[0])
```

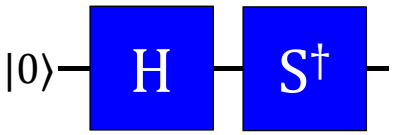



$\frac{\pi}{8}$位相シフト演算

```
qc.h(q[0])
qc.t(q[0])
```

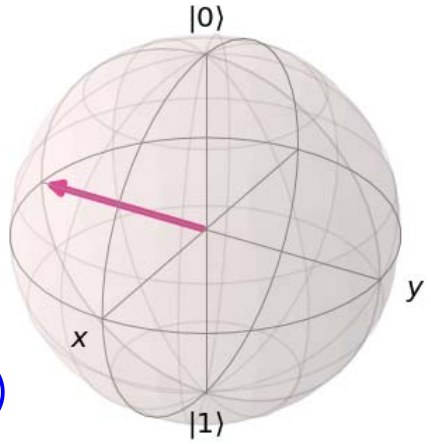


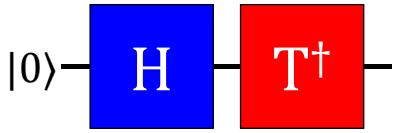
※ ダガー"†"が付くと逆方向の位相となる。



$-\frac{\pi}{4}$位相シフト演算

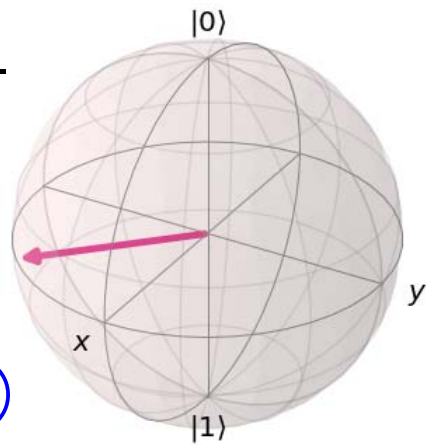
```
qc.h(q[0])
qc.sdg(q[0])
```



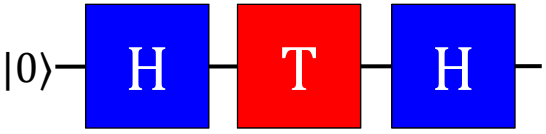


$-\frac{\pi}{8}$位相シフト演算

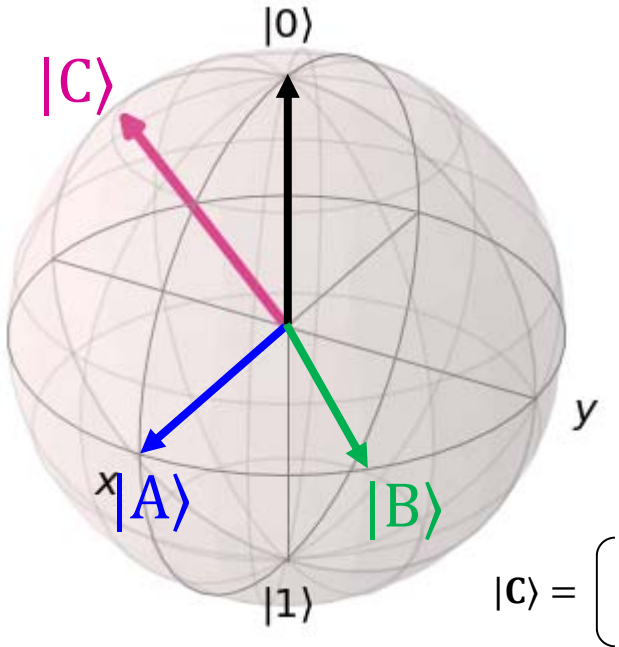
```
qc.h(q[0])
qc.tdg(q[0])
```



π/2 以外の位相シフトの意味




```
qc.h(q[0]) #|A⟩
qc.t(q[0]) #|B⟩
qc.h(q[0]) #|C⟩
```



課題：

量子演算を行う為にはブロッホ球上の任意の場所にベクトルを移動する必要がある。

解決方法：

アマダール演算と位相シフト演算を組み合わせることで任意の位置にベクトル移動することが可能となる。ただしπ/8のシフト演算だけだと取れる方向はある程度限定されてしまうが現在の量子計算では十分な精度らしい。

$$|\mathbf{C}\rangle = \begin{pmatrix} 0.85355339+0.35355339\mathrm{j} \\ 0.14644661-0.35355339\mathrm{j} \end{pmatrix} \quad \begin{aligned} |c_0|^2 &= 0.85355339 \\ |c_1|^2 &= 0.14644661 \end{aligned}$$

軸回転 Rx/Ry/Rz ゲート

実際の量子プログラミングでは、任意角度での軸回転（シフト）ゲートが必要になる。

Rx X軸回転演算 $Rx(\theta, q) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$

Ry Y軸回転演算 $Ry(\theta, q) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$

Rz Z軸回転演算 $Rz(\theta, q) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$

ユニタリ回転 U1/U2/U3 ゲート

ユニタリ演算は回転なので Rx/Ry/Rz の演算により全てのユニタリ演算ゲートが実現できる。

U_1 指定角λ演算 $u1(\lambda, q) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}$

ex) z(q) = u1(pi, q)

U_2 指定角φ,λ演算 $u2(\varphi, \lambda, q) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\varphi} & e^{i(\lambda+\theta)} \end{pmatrix}$

ex) h(q) = u2(0, pi, q)

U_3 指定角θ,φ,λ演算 $u3(\theta, \varphi, \lambda, q) =$

$$\begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\varphi}\sin\frac{\theta}{2} & e^{i(\lambda+\theta)}\cos\frac{\theta}{2} \end{pmatrix}$$

ex) x(q) = u3(pi, 0, pi, q)

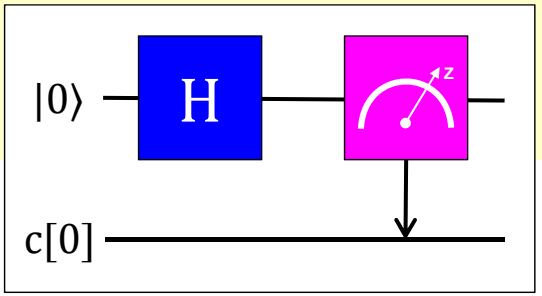
u1(λ, q) = u3(0, 0, λ, q)
u2(φ, λ, q) = u3(pi/2, φ, λ, q)

測定（量子ビットを収束させて古典ビットとして取り出す）

標準基底測定 measure(q, c)

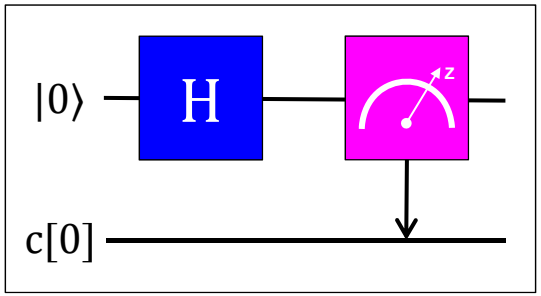
Jupiter Notebook から以下を実行(コピペで大丈夫です)。

```
from qiskit import *                              # 量子計算用
from qiskit.tools.visualization import *          # 結果表示用
backend = Aer.get_backend('qasm_simulator')  # 量子シミュレータ指定
q = QuantumRegister(1)          # 量子ビットを1つ用意
c = ClassicalRegister(1)        # 古典ビットを1つ用意
qc = QuantumCircuit(q, c)       # 量子回路に量子ビットと古典ビットをセット
qc.h(q[0])                      # 量子重ね合わせ（アダマール演算）
qc.measure(q[0], c[0])          # 量子ビットq[0]を観測し古典ビットc[0]へ
r = execute(qc, backend, shots=1000).result() # 回路を1000回実行する
rc = r.get_counts()             # 結果取得
print(rc)                       # 結果表示
plot_histogram(rc)              # ヒストグラム表示
```



測定結果の例1：Hゲート（アダマール演算）

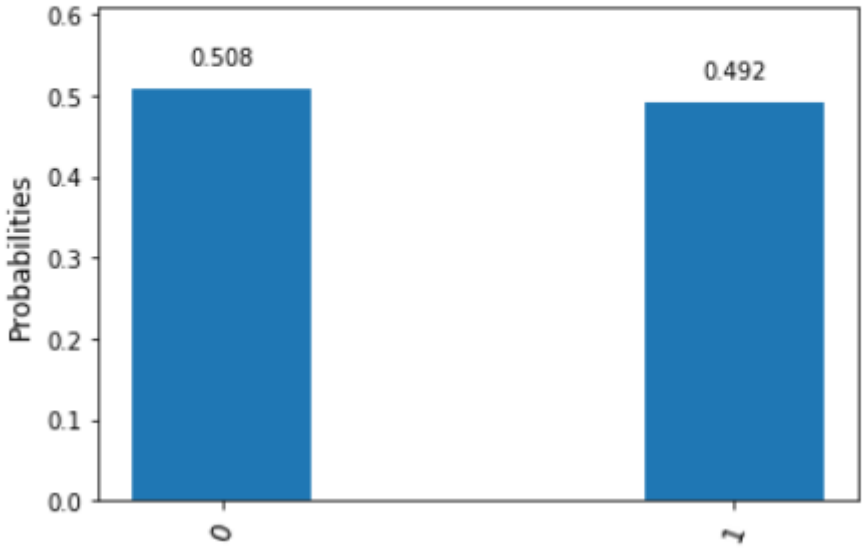
```
qc.h(q[0])
qc.measure(q[0],c[0])
```

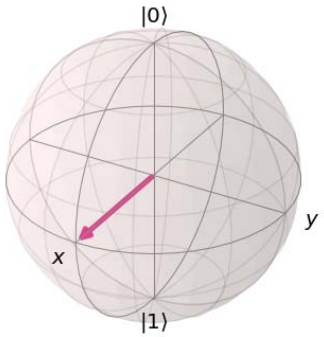



```
print(rc)                    # 結果表示
plot_histogram(rc)           # ヒストグラム表示
```

{'0': 508, '1': 492}

'0' : 500, '1': 500 となるはずだが…シミュレータでもノイズを考慮している。

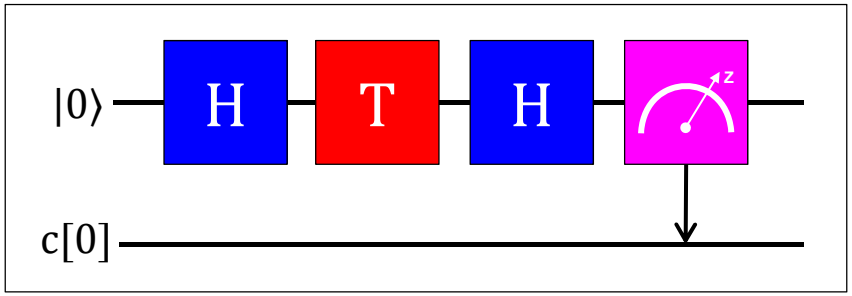




参考：ブロッホ球表示

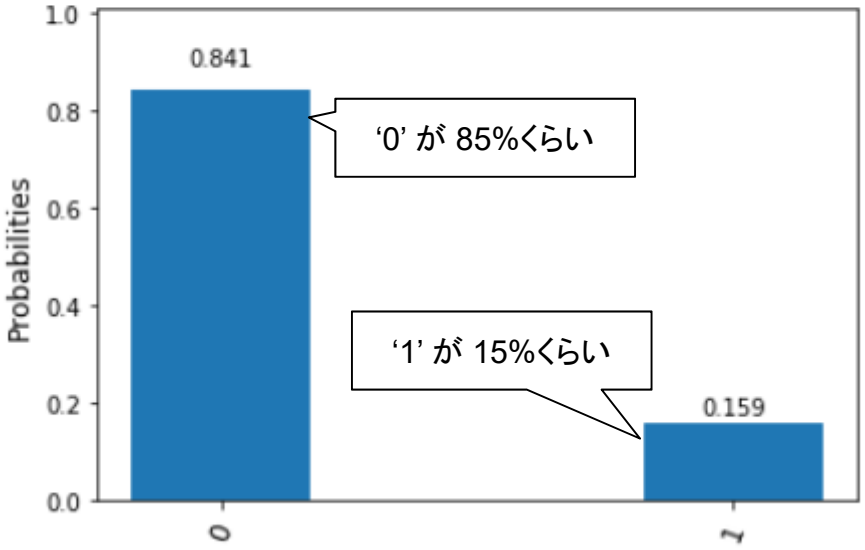
測定結果の例2：Hゲートと Tゲート

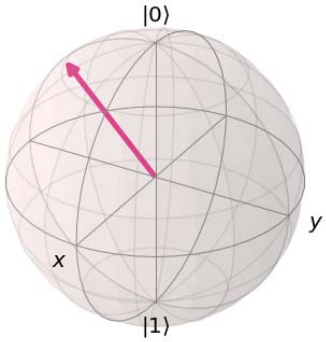
```
qc.h(q[0])
qc.t(q[0])
qc.h(q[0])
qc.measure(q[0],c[0])
```




```
print(rc)                    # 結果表示
plot_histogram(rc)           # ヒストグラム表示
```

```
{'0': 841, '1': 159}
```





参考：ブロッホ球表示

$$|\psi\rangle = \begin{pmatrix} 0.85355339+0.35355339\mathrm{j} \\ 0.14644661-0.35355339\mathrm{j} \end{pmatrix}$$

$$|c_0|^2 = (0.85355339)^2+(+0.35355339)^2 = 0.85355339 \ [85.36\%]$$

$$|c_1|^2 = (0.14644661)^2+(-0.35355339)^2 = 0.14644661 \ [14.64\%]$$

Qiskitのバックエンド

- 'statevector_simulator'
  - 重ね合わせ状態のまま、値を取得できる量子シミュレータ。
  - ノイズ無しの理論値（確率振幅）を取得できる。
  - ブロッホ球（重ね合わせ状態）の表示時に使う。
- 'qasm_simulator'
  - ノイズありの量子シミュレータ。
  - 通常の量子計算に使う標準のシミュレータ。
- 'ibmqx4' / 'ibmqx2'
  - IBM Q の量子コンピュータ実機（トークンが必要）。
  - 利用可能な量子コンピュータ毎に名前がある。

1-4：IBM Q

https://www.research.ibm.com/ibm-q/

GUIで量子計算：The IBM Q Experience

IBM Q をGUIで使う為のクラウドサービス

https://quantumexperience.ng.bluemix.net/qx/editor

サインインの為には登録（無料）が必要。
Linkedin/GitHub/Google/Twitterアカウントも利用可能
https://quantumexperience.ng.bluemix.net/qx/login

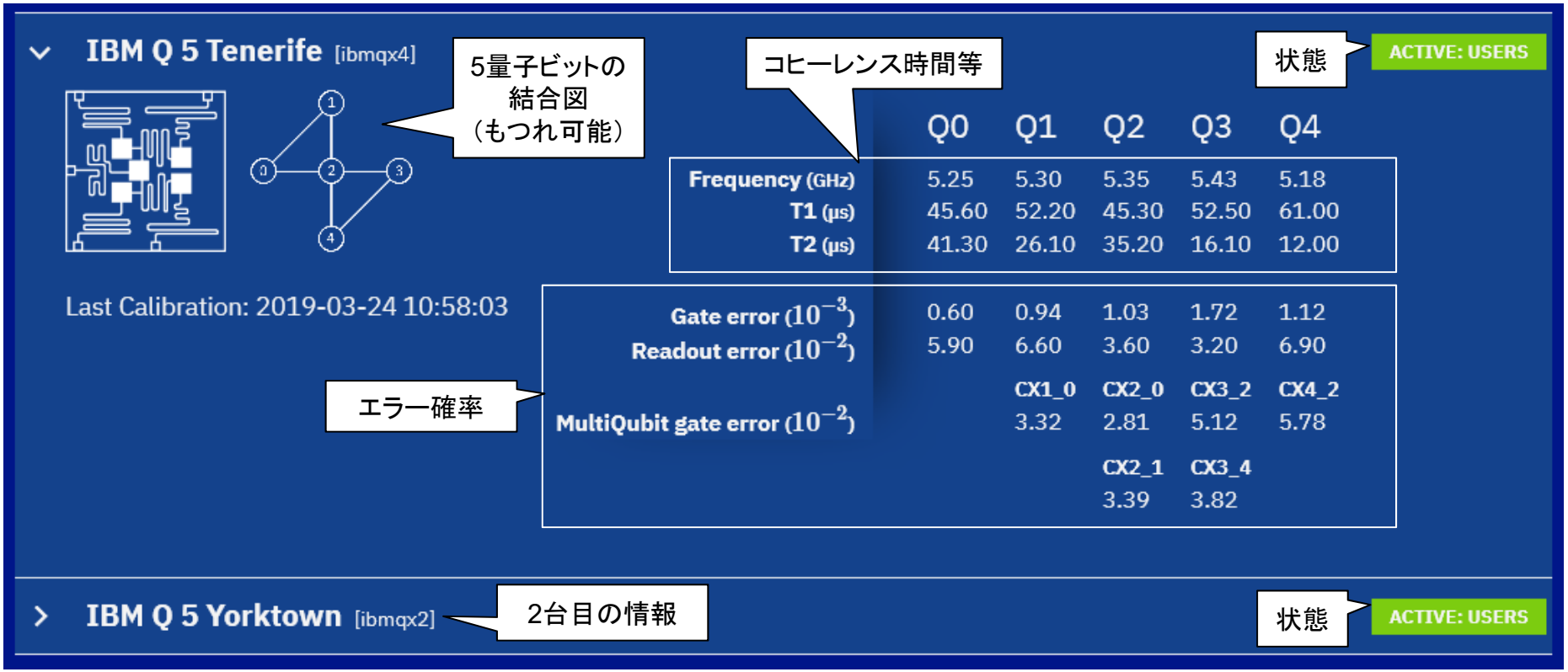
機能：

1. GUIを使って簡単な量子回路の編集が可能。
2. 実行時に以下の選択が可能：
   - A) 実機を使わない量子シミュレーション（Qiskitと同じ）
   - B) 5/16量子ビットの実機を使った量子計算（バッチ実行）
   - C) 過去に同じ量子回路の実機を使った結果の利用

※ 過去には可能だったがブロッホ球表示は非サポートになった。

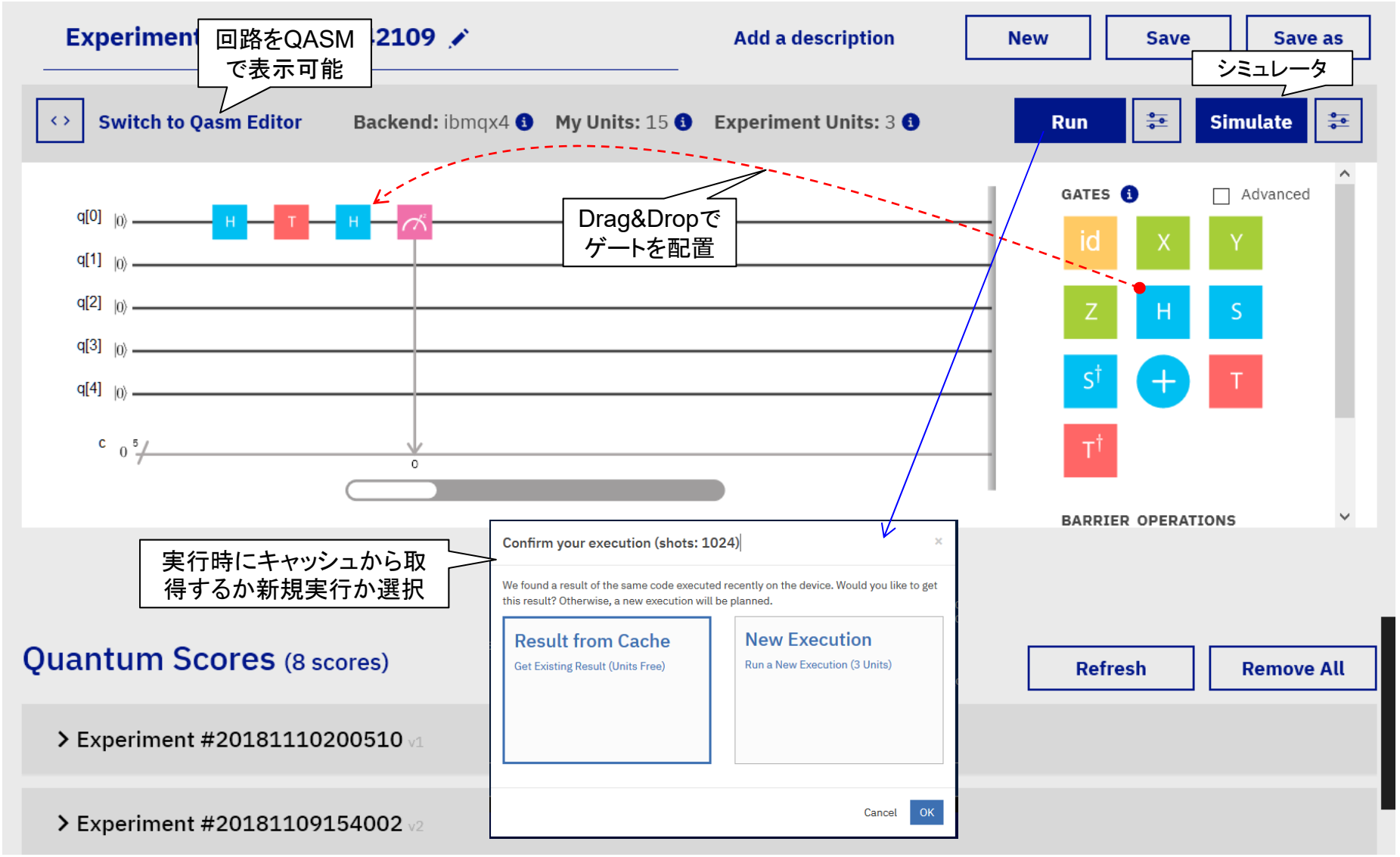
The IBM Q 公開中システム

トップに利用可能な IBM Q Experience システムのステータス表示

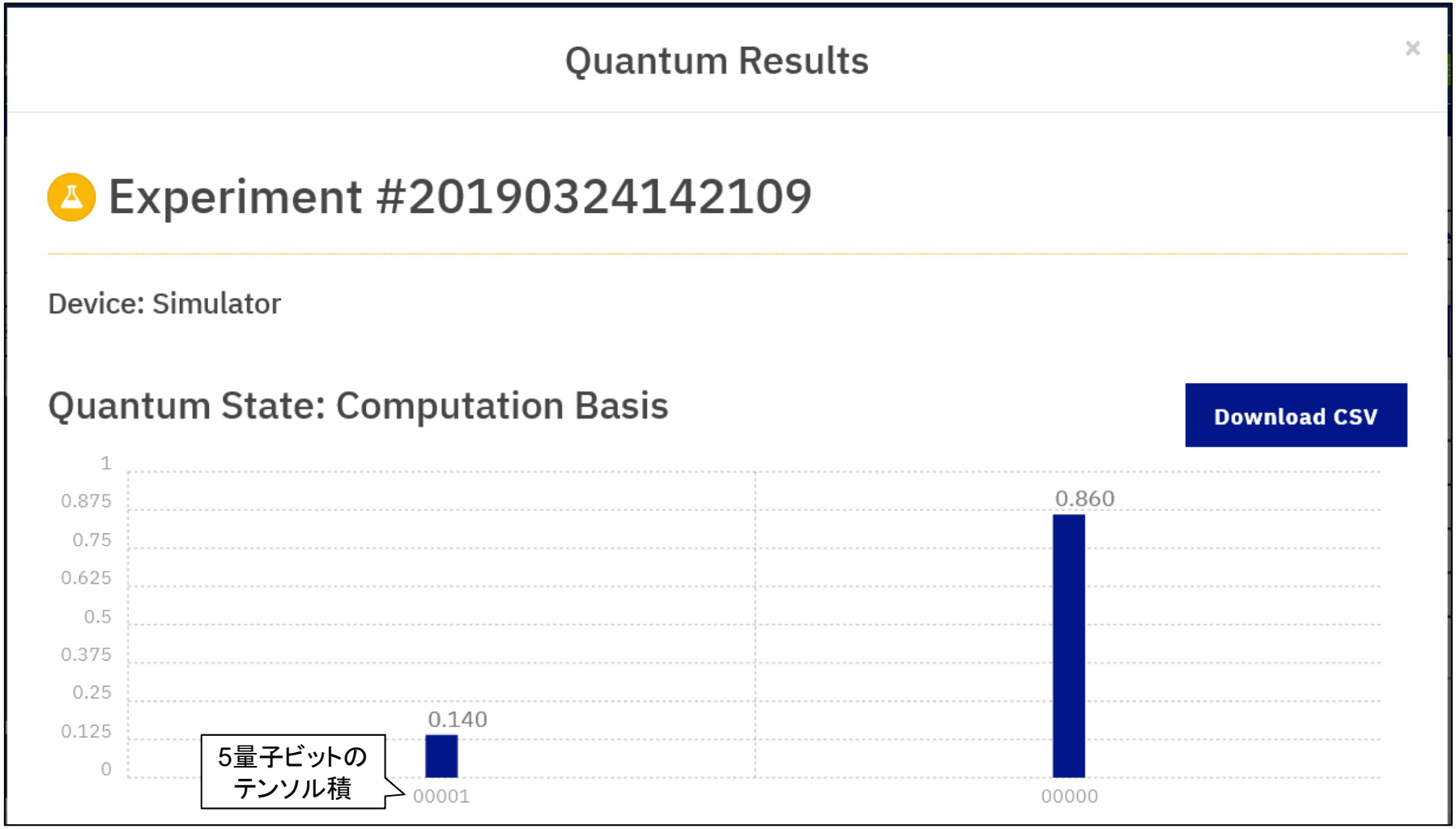


- 有償契約すれば20量子ビットのシステムも利用が可能。
- 今後新しいシステムが追加され量子ビット数も増える予定。

The IBM Q Experience 回路編集と実行

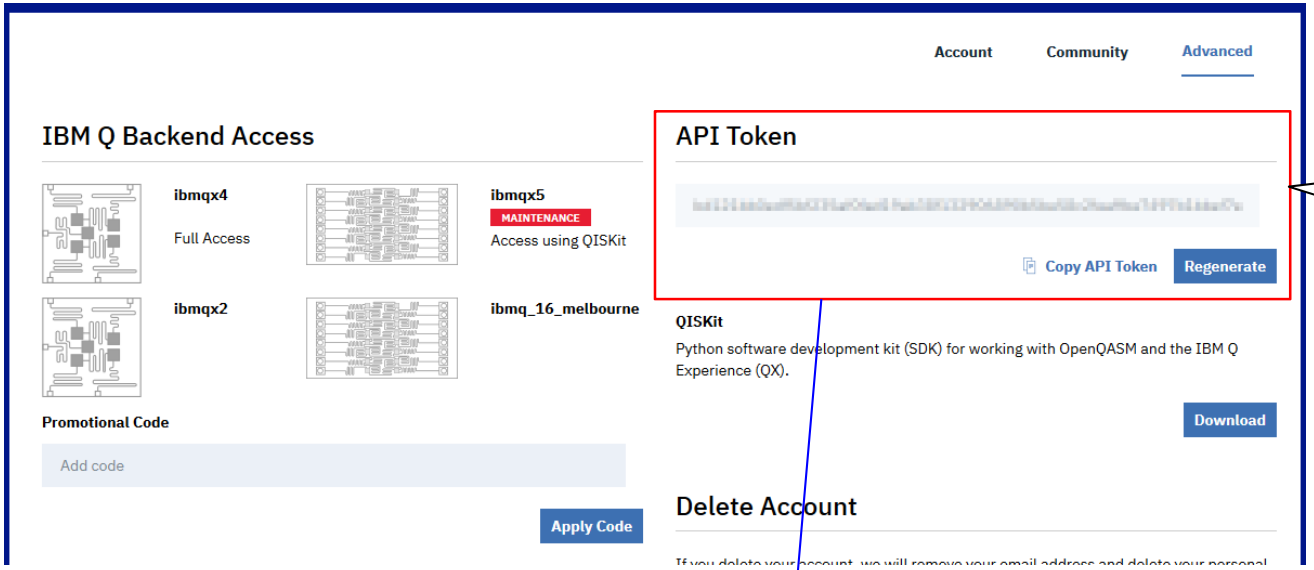


The IBM Q Experience 実行結果表示



Qiskit で IBM Q 実機を使う（準備）

https://quantumexperience.ng.bluemix.net/qx/account/advanced



The IBM Q Experience のユーザ情報からAPIトークンを取得する。

登録：1回のみ	from qiskit import IBMQ IBMQ.save_account('MY_API_TOKEN')
読み込み：セッション毎	from qiskit import IBMQ IBMQ.load_accounts()
指定：	from qiskit import IBMQ backend = IBMQ.get_backend('ibmqx4')
バージョン：	import qiskit qiskit.__version__

.qiskit

```
[ibmq]
token = MY_API_TOKEN
url = https://quantumexperience.ng.bluemix.net/api
verify = True
```

ここまでが前準備。
OKなら次ページのプログラム実行。

Qiskit で IBM Q 実機を使う(実行)

1. Pythonプログラムファイル ibmq.py を用意(ファイル名は任意)。

```
from qiskit import *                              # 量子計算用
from qiskit import IBMQ                           # 実機利用用
print("Start: Load accounts")
IBMQ.load_accounts()                              # 実機用にアカウントロード
backend = IBMQ.get_backend('ibmqx4')              # 実機指定：IBM Q 5 Tenerife
#backend = Aer.get_backend('qasm_simulator')      # 量子シミュレータ指定
q = QuantumRegister(1)          # 量子ビットを1つ用意
c = ClassicalRegister(1)        # 古典ビットを1つ用意
qc = QuantumCircuit(q, c)       # 量子回路に量子ビットと古典ビットをセット
qc.h(q[0])                      # 量子重ね合わせ（アダマール演算）
qc.measure(q[0], c[0])          # 量子ビットq[0]を観測し古典ビットc[0]へ
print("Run: Start")
r = execute(qc, backend, shots=1000).result() # 回路を1000回実行する
print("Run: End:")
rc = r.get_counts()             # 結果取得
print(rc)                       # 結果表示
```

この行を有効にするとシミュレータ実行

この部分は実機でもシミュレータでも同じです。

2. Anaconda Prompt から ibmq.py を実行。

```
(base) >python ibmq.py
Start: Load accounts
Run: Start
Run: End:
{'1': 490, '0': 510}
```

Run: Start 後に数分かかる場合があります(キュー実行の為)

IBM Q 実機での実行結果！

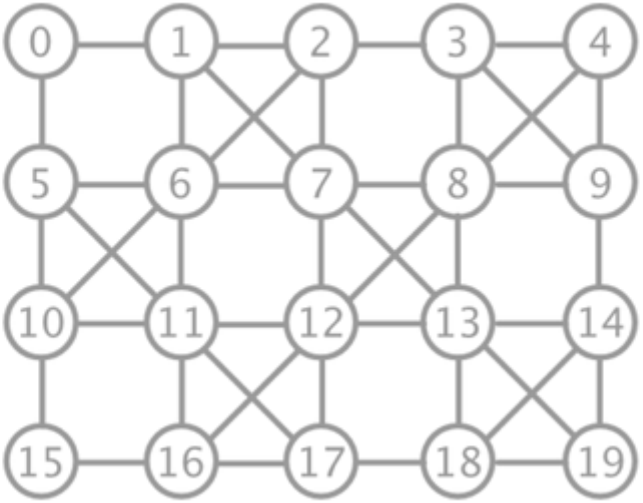
注:世界中から1台の量子コンピュータを使うので利用には注意を!!

IBM Q の 20量子ビット機

IBM Q 20 Tokyo

20

qubits



20量子ビットの結合図
やはり接続に制限があるが
5量子ビットより複雑だ

※ IBM Q Network で利用可能

Availability & status

For IBM Q clients

● Online

Last calibration occured

2019-03-24 8:14:51 pm

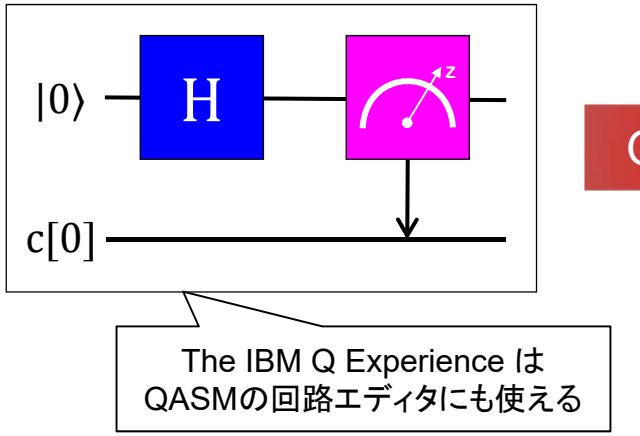
Average measurements

Frequency (GHz)	4.97
T1 (μs)	81.75
T2 (μs)	51.07
Gate error (10^{-3})	1.88
Readout error (10^{-2})	7.44

https://www.research.ibm.com/ibm-q/technology/devices/

参考：QASM（OpenQASM 2.0）

IBMが公開している量子回路アセンブラ言語。
Cirq（Google）、Q#（MS）、Blueqat（MDR）等のほぼ全ての量子プラットホームがサポートしているので相互運用ができる。
Visual Studio Code用の拡張も公開されている。
https://marketplace.visualstudio.com/items?itemName=qiskit.qiskit-vscode



QASM

```
include "qelib1.inc";
qreg q[1];
creg c[1];
h q[0];
measure q[0] -> c[0];
```

Part 1：エピローグ

プログラマの為の量子コンピュータ入門 目次:

Part 1: 関連数学と1量子ビット操作

- 線形代数学の基本知識
- ブラケット記法と量子計算
- ブロッホ球と1量子ビット操作（IBM Qiskit）

Part 2: 量子ゲート型のプログラミング

- 2量子ビット以上の量子回路（量子もつれ）
- 量子アルゴリズム（グローバ検索・フーリエ変換等）
- 量子ゲート型プログラミング（Google Cirq）

次回!

Part 3: 量子アニーリング型のプログラミング

- イジングモデルとQUBOと量子アニーリング
- 量子アルゴリズム（セールスマン巡回問題）
- 量子アニーリング型プログラミング（Blueqat他）

Part 1: 関連数学と1量子ビット操作 まとめ…

現在の量子プログラミングはまだアセンブラ言語を使っている状態です。(昔のワンボードマイコンか？)

量子ビット(ゲート)の動作を理解する為に数学が必要になるので Part 1 では基本を勉強しました。

将来量子アルゴリズムがAPI化されることでアセンブラの知識が無くても使えるようになるでしょう。

でも基本から覚えておくことは損にはなりません。
それに動作を理解することは楽しくもあります！

次回 Part 2 では複数量子ビット操作と更には、量子アルゴリズムを勉強します。是非ご参加を！

http://scienceinoh.jp/schrodinger/