

1 古典ビット

従来の古典的なコンピュータでは、それが取り扱う情報の基本単位をビット(bit)と呼ぶ。1つのビットは0か1の状態を取ることが定義されており、一般に n 個のビットは 2^n 通りの状態をとる。

(ビット表現の例)

2個のビットならば、00、01、10、11の4($=2^2$)通りの状態をとる。

3個のビットならば、000、001、010、011、100、101、110、111の8($=2^3$)通りの状態をとる。

このように表現される複数のbit状態を、従来の古典的なコンピュータでは対象とする情報と一対一対応させている。つまり、一度に表現できる情報は一つだけで、演算にも時間がかかってしまう。

従来の古典的なコンピュータで利用しているという意味で、ビットは古典ビットと呼ばれる。

2 量子ビット

一方、量子コンピュータではそれが取り扱う情報の基本単位を量子ビット(Qubit)と呼ぶ。

量子コンピュータでは光子や磁子などの微小粒子がもつ量子状態を応用して演算するため、量子状態を複素ベクトルとして表現し、その状態に対応させた状態ベクトルを量子ビットとしている。

光子などの微小粒子
の何らかしらの量子状態



数学的な処理により
複素ベクトルに表現

量子ビット(状態ベクトル)

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

量子ビットのイメージ

量子ビットを古典ビットに対応させた場合、ブラケット記法[$|\square\rangle$ シラバス 7-1-2]を用いれば以下のように表現できる。

古典ビット 0 \rightarrow 量子ビット $|0\rangle$ (0ケットと呼ぶ)

古典ビット 1 \rightarrow 量子ビット $|1\rangle$ (1ケットと呼ぶ)

量子ビットは古典ビットと異なり状態ベクトルであるため、状態ベクトルの線形結合(数学的な足し合わせ操作)により、以下のような新しい状態ベクトルを表現でき、新たな量子ビットとして表現ができる。

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

※ α 、 β は複素数かつ $|\alpha|^2 + |\beta|^2 = 1$ (規格化条件)を満たす。

このように複数の量子ビットが線形結合している状態は重ね合わせ状態(superposition)と呼ばれ、一つの状態で複数の情報をもてる性質があるため、量子コンピューティングは膨大な情報量を効率よく演算できる。

3 関連キーワード

量子とは[$|\square\rangle$ シラバス 2]、量子重ね合わせ[$|\square\rangle$ シラバス 3-2]

4 参考文献

[量子コンピュータとは何か？\(三井情報株式会社\)](#)

1 1量子ビットの標準基底

1量子ビットは2次元の複素ベクトル空間上の(状態)ベクトルで表現。2次元の複素ベクトル空間には2つの基底ベクトル $\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ が存在する。

量子ビット $|0\rangle, |1\rangle$ に対応し2つの基底ベクトルが対応し、これらベクトルを標準基底と呼ぶ。標準基底は正規直交基底を構成している。

量子コンピュータの世界では計算基底状態(computational basis state)とも呼ばれる。

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

2 1量子ビットの状態ベクトル

量子ビット同士の線形結合で表現しなおすと、1量子ビットの状態ベクトルは以下で表現できる。

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha + 0 \\ 0 + \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

※ α, β は複素数かつ $|\alpha|^2 + |\beta|^2 = 1$ を満たす。

3 2量子ビットの標準基底

1量子ビットの時と同じよう2量子ビットは $2^2(=4)$ 次元の複素ベクトル空間上の(状態)ベクトルで表現できる。

2^2 次元の複素ベクトル空間には4つの基底ベクトルが存在し、量子ビット $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ に対応する。

$$|00\rangle := \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |01\rangle := \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |10\rangle := \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, |11\rangle := \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

4 2量子ビットの状態ベクトル

2量子ビットの状態ベクトルは以下で表現できる。

$$\begin{aligned} |\psi\rangle &= c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle \\ &= c_0 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + c_1 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + c_3 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c_0+0+0+0 \\ 0+c_1+0+0 \\ 0+0+c_2+0 \\ 0+0+0+c_3 \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \end{aligned}$$

※ c_0, c_1, c_2, c_3 は複素数かつ $|c_0|^2 + |c_1|^2 + |c_2|^2 + |c_3|^2 = 1$ を満たす。

5 n量子ビットの標準基底

n量子ビットは 2^n 次元の複素ベクトル空間上の(状態)ベクトルで表現。

2^n 次元の複素ベクトル空間には 2^n 個の基底ベクトルが存在し、n量子ビット $|00\dots 00\rangle, |00\dots 01\rangle, \dots, |11\dots 10\rangle, |11\dots 11\rangle$ に対応する。

$$|00\dots 00\rangle := \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, |00\dots 01\rangle := \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \dots, |11\dots 10\rangle := \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}, |11\dots 11\rangle := \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

6 n量子ビットの状態ベクトル

n量子ビットの状態ベクトルは以下で表現できる。

$$\begin{aligned} |\psi\rangle &= c_0|00\dots 00\rangle + c_1|00\dots 01\rangle + \dots + c_{2^{n-2}}|11\dots 10\rangle + c_{2^{n-1}-1}|11\dots 11\rangle \\ &= c_0 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + c_1 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \dots + c_{2^{n-2}} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} + c_{2^{n-1}-1} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} c_0+0+\dots+0+0 \\ 0+c_1+\dots+0+0 \\ \vdots \\ 0+0+\dots+c_{2^{n-2}}+0 \\ 0+0+\dots+0+c_{2^{n-1}-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{2^{n-2}} \\ c_{2^{n-1}-1} \end{pmatrix} \end{aligned}$$

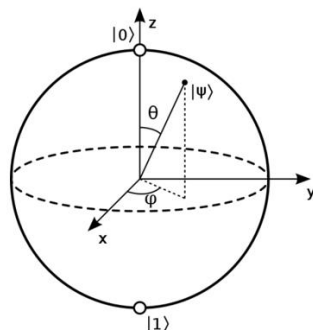
$= \sum_{i=0}^{2^n-1} c_i |(i\text{番目の}2\text{進数})\rangle$ ※ $c_0, c_1, \dots, c_{2^{n-2}}, c_{2^{n-1}-1}$ は複素数かつ $\sum_{i=0}^{2^n-1} |c_i|^2 = 1$ を満たす。

7 参考文献

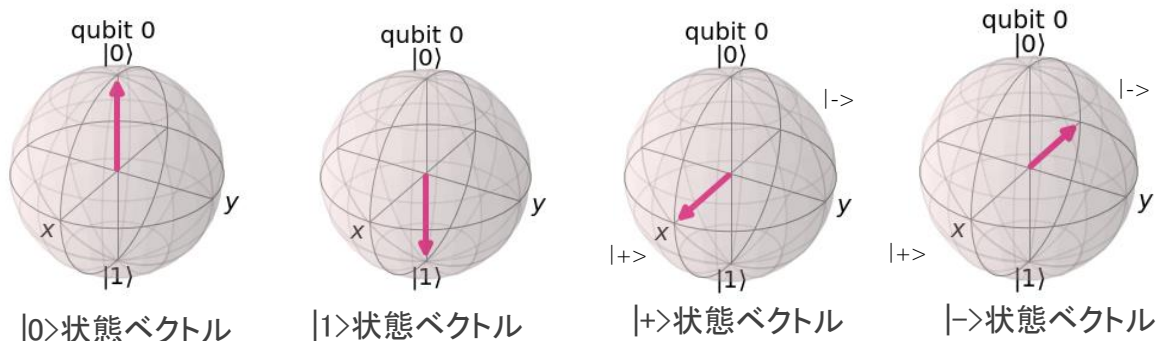
1 Bloch球

量子ビットはあくまで量子状態の数学的表現に過ぎないのでイメージがしづらい。そこで、1量子ビットの状態ベクトルを可視化する手段として、Bloch球と呼ばれるものがある。

Bloch球は1量子ビットの状態ベクトル $|\psi\rangle$ を複素三角関数で表現したもので、3次元単位球面において、z軸の正の方に $|0\rangle$ 、負の方に $|1\rangle$ を置き、z軸から θ 回転され、x軸から反時計回りに ϕ 回転した点として以下のように可視化できる。



1量子ビットの種々の状態ベクトルがBloch球では以下のように可視化される。



2 ブラケット記法

Dirac方程式で有名な大物理学者Paul Adrien Maurice Diracが考案したベクトルの内積記法の一つ。 \langle, \rangle (bracket記号)と $|$ (縦線)の組み合わせでベクトルを表現する。

列ベクトル $\begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$ は量子の状態ベクトルを表している。複素数 a_0, a_1 を用いた表現は以下のとおり。 $|\psi\rangle$ (ψ ケットベクトル) と呼ぶ。

$$|\psi\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$

行ベクトル $(a_0 \ a_1)$ は、物理的に特に意味をなさないことが多いがケットベクトルの複素共役を表していることが多い。複素数 a_0, a_1 を用いた表現は以下のとおり。以下のとおり $\langle\psi|$ (ψ ブラベクトル) と呼ぶ。

$$\langle\psi| = (|\psi\rangle)^\dagger = (a_0^* \ a_1^*)$$

3 関連キーワード

量子ビット表現・表示[シラバス 6-3-1]

4 参考文献

[ブロッホ球 - 量子力学 - EMANの物理学](#)
[ブラケット記法 - 量子力学 - EMANの物理学](#)

1 Bloch球の導出

- まず1量子ビットの状態ベクトル $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ について考える。
 - 複素数($z = a + bi$)の絶対値は $|z| = \sqrt{a^2 + b^2}$ と実数で表現できるので、 $|\alpha|, |\beta|$ は非負の実数。
 - 規格化条件($|\alpha|^2 + |\beta|^2 = 1$)より $0 \leq \theta \leq \frac{\pi}{2}$ を満たす θ を用いて、 $|\alpha| = \cos \theta, |\beta| = \sin \theta$ と置く。
 $\theta = 0$ の場合、 $|\alpha|^2 + |\beta|^2 = (\cos 0)^2 + (\sin 0)^2 = 1 + 0 = 1$
 - $\theta = \frac{\pi}{2}$ の場合、 $|\alpha|^2 + |\beta|^2 = \left(\cos \frac{\pi}{2}\right)^2 + \left(\sin \frac{\pi}{2}\right)^2 = 0 + 1 = 1$
 - $\theta = \frac{\pi}{4}$ の場合、 $|\alpha|^2 + |\beta|^2 = \left(\cos \frac{\pi}{4}\right)^2 + \left(\sin \frac{\pi}{4}\right)^2 = \left(\frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2} + \frac{1}{2} = 1$
- 次に複素指数関数と三角関数との関係式であるオイラーの公式($e^{i\phi} = \cos \phi + i \sin \phi$)を考える。
- オイラーの公式に $\cos \theta, \sin \theta$ をそれぞれ掛けてみる。
 $e^{i\phi} \cos \theta = (\cos \phi + i \sin \phi) \cos \theta = \cos \phi \cos \theta + i \sin \phi \cos \theta \dots \textcircled{1}$
 $e^{i\phi} \sin \theta = (\cos \phi + i \sin \phi) \sin \theta = \cos \phi \sin \theta + i \sin \phi \sin \theta \dots \textcircled{2}$
- それぞれの右辺に着目して、複素数の絶対値が計算できそう。
 $\textcircled{1}$ の右辺から、 $\sqrt{(\cos \phi \cos \theta)^2 + (\sin \phi \cos \theta)^2} = \sqrt{\cos^2 \theta (\cos^2 \phi + \sin^2 \phi)}$
 $= \sqrt{\cos^2 \theta} = \cos \theta$
 $|\alpha| = \cos \theta$ より、複素数 $\alpha = e^{i\phi_\alpha} \cos \theta (0 < \phi_\alpha < 2\pi)$ と表現できた。
 $\textcircled{2}$ の右辺から、 $\sqrt{(\cos \phi \sin \theta)^2 + (\sin \phi \sin \theta)^2} = \sqrt{\sin^2 \theta (\cos^2 \phi + \sin^2 \phi)}$
 $= \sqrt{\sin^2 \theta} = \sin \theta$
 $|\beta| = \sin \theta$ より、複素数 $\beta = e^{i\phi_\beta} \sin \theta (0 < \phi_\beta < 2\pi)$ と表現できた。

- 1量子ビットの状態ベクトル $|\psi\rangle$ について改めて表現し直すと以下の式になる。

$$\begin{aligned} |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle \\ &= e^{i\phi_\alpha} \cos \theta |0\rangle + e^{i\phi_\beta} \sin \theta |1\rangle \\ &= e^{i\phi_\alpha} \left(\cos \theta |0\rangle + \frac{e^{i\phi_\beta}}{e^{i\phi_\alpha}} \sin \theta |1\rangle \right) \\ |\psi\rangle &= e^{i\phi_\alpha} \left(\cos \theta |0\rangle + e^{i(\phi_\beta - \phi_\alpha)} \sin \theta |1\rangle \right) \end{aligned}$$

- ここで簡単にするために、 $\phi_\alpha = \lambda, \phi_\beta - \phi_\alpha = \phi, \theta$ を $\frac{\theta}{2}$ に文字を置き換える。
 $0 < \phi_\alpha, \phi_\beta < 2\pi$ としていたので、 $0 < \lambda, \phi < 2\pi$ となり、また θ の範囲を $0 \leq \theta \leq \pi$ とする。

$$|\psi\rangle = e^{i\lambda} \left(\cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \right) \dots \textcircled{1}$$

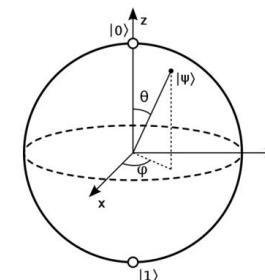
※変数 ϕ を位相、変数 λ をグローバル位相と呼ぶ。

- ここで3次元単位球面上の点を考えると、グローバル位相のある $e^{i\lambda}$ は点の位置には影響せず無視できるので、実効的に式は以下のとおりになる。

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \dots \textcircled{2}$$

※ $0 < \phi < 2\pi, 0 \leq \theta \leq \pi$

- 最後に、3次元単位球面において、z軸の正の方に $|0\rangle$ 、負の方に $|1\rangle$ を置き、z軸から θ 回転され、x軸から反時計回りに ϕ 回転した点 $|\psi\rangle$ を式に対応させると、下図のように θ, ϕ の範囲から単位球面上の点が表現できる。



2 基底ベクトル

Bloch球ではX、Y、Z軸上に基底ベクトルが可視化されます。以下それぞれ軸についての基底ベクトルを確認する。

Z軸上の基底ベクトル

Bloch球のZ軸上の点 $(\theta, \phi) = (0, 0), (\pi, 0)$ を確認する。

②式に $(\theta, \phi) = (0, 0), (\pi, 0)$ を代入。

$(\theta, \phi) = (0, 0)$ の場合

$$\begin{aligned} |\psi\rangle &= \cos 0 |0\rangle + e^{i0} \sin 0 |1\rangle \\ &= 1|0\rangle + 1 \cdot 0|1\rangle = |0\rangle \end{aligned}$$

$(\theta, \phi) = (\pi, 0)$ の場合、

$$\begin{aligned} |\psi\rangle &= \cos \frac{\pi}{2} |0\rangle + e^{i0} \sin \frac{\pi}{2} |1\rangle \\ &= 0|0\rangle + 1 \cdot 1|1\rangle = |1\rangle \end{aligned}$$

Z軸上の基底ベクトルは $|0\rangle$ 、 $|1\rangle$ で、Z基底と呼ばれる。

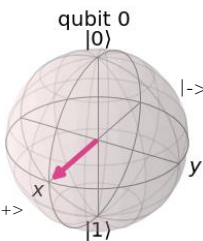
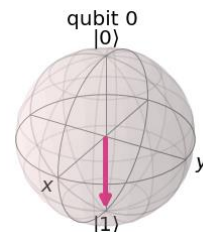
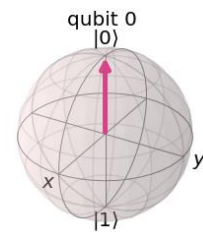
X軸上の基底ベクトル

Bloch球のX軸上の点 $(\theta, \phi) = (\frac{\pi}{2}, 0), (\frac{\pi}{2}, \pi)$ を確認する。

②式に $(\theta, \phi) = (\frac{\pi}{2}, 0), (\frac{\pi}{2}, \pi)$ を代入。

$(\theta, \phi) = (\frac{\pi}{2}, 0)$ の場合

$$\begin{aligned} |\psi\rangle &= \cos \frac{\pi}{2} |0\rangle + e^{i0} \sin \frac{\pi}{2} |1\rangle = \cos \frac{\pi}{4} |0\rangle + e^{i0} \sin \frac{\pi}{4} |1\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) := |+\rangle \end{aligned}$$



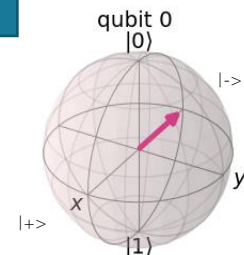
$(\theta, \phi) = (\frac{\pi}{2}, \pi)$ の場合

$$\because e^{i\pi} = \cos \pi + i \sin \pi = -1 + i \cdot 0 = -1$$

$$\begin{aligned} |\psi\rangle &= \cos \frac{\pi}{2} |0\rangle + e^{i\pi} \sin \frac{\pi}{2} |1\rangle = \cos \frac{\pi}{4} |0\rangle + e^{i\pi} \sin \frac{\pi}{4} |1\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) := |-\rangle \end{aligned}$$

X軸上の基底ベクトルは $|+\rangle$ 、 $|-\rangle$ で、X基底と呼ばれる。

また $|+\rangle$ 、 $|-\rangle$ はH(アダマール)ゲートと呼ばれる特殊な量子ゲートによっても生成されることからアダマール基底とも呼ばれる。



Y軸上の基底ベクトル

Bloch球のY軸上の点 $(\theta, \phi) = (\frac{\pi}{2}, \frac{\pi}{2}), (\frac{\pi}{2}, \frac{3\pi}{2})$ を確認する。

②式に $(\theta, \phi) = (\frac{\pi}{2}, \frac{\pi}{2}), (\frac{\pi}{2}, \frac{3\pi}{2})$ を代入。

$(\theta, \phi) = (\frac{\pi}{2}, \frac{\pi}{2})$ の場合

$$\begin{aligned} \because e^{i\frac{\pi}{2}} &= \cos \frac{\pi}{2} + i \sin \frac{\pi}{2} \\ &= 0 + i \cdot 1 = i \end{aligned}$$

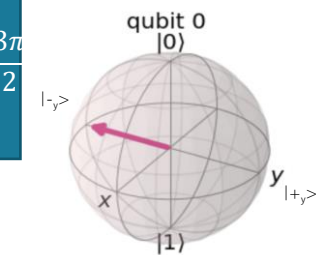
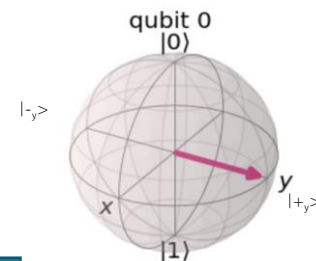
$$\begin{aligned} |\psi\rangle &= \cos \frac{\pi}{4} |0\rangle + e^{i\frac{\pi}{2}} \sin \frac{\pi}{4} |1\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle + i \frac{1}{\sqrt{2}} |1\rangle = \frac{1}{\sqrt{2}} (|0\rangle + i|1\rangle) \\ &:= |+_y\rangle \text{ or } |+_i\rangle \end{aligned}$$

$(\theta, \phi) = (\frac{\pi}{2}, \frac{3\pi}{2})$ の場合

$$\begin{aligned} \because e^{i\frac{3\pi}{2}} &= \cos \frac{3\pi}{2} + i \sin \frac{3\pi}{2} \\ &= 0 - i \cdot 1 \\ &= -i \end{aligned}$$

$$\begin{aligned} |\psi\rangle &= \cos \frac{\pi}{4} |0\rangle + e^{i\frac{3\pi}{2}} \sin \frac{\pi}{4} |1\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle - i \frac{1}{\sqrt{2}} |1\rangle = \frac{1}{\sqrt{2}} (|0\rangle - i|1\rangle) \\ &:= |-_y\rangle \text{ or } |-_i\rangle \end{aligned}$$

Y軸上の基底ベクトルは $|+_y\rangle$ 、 $|-_y\rangle$ で、Y基底と呼ばれる。



3 参考文献

IBM Quantumで学ぶ量子コンピュータ - 秀和システム

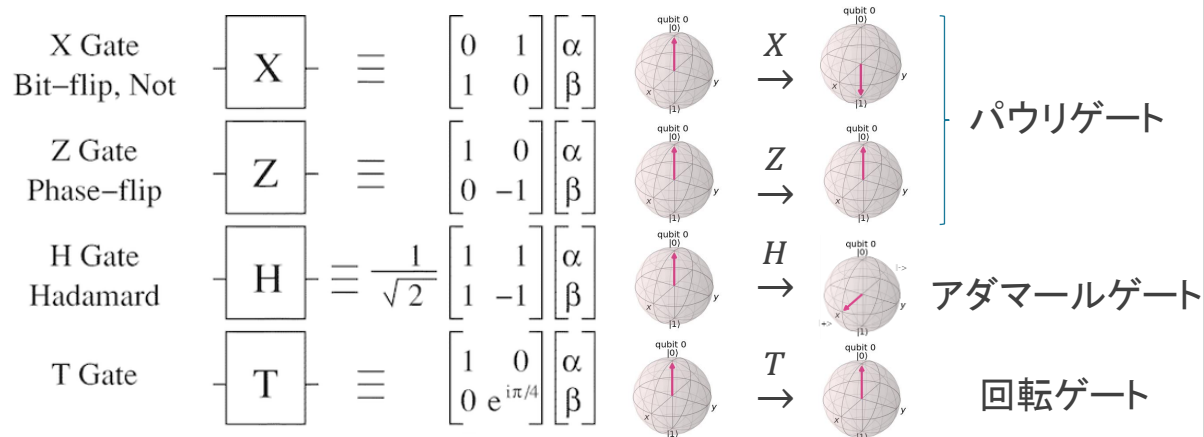
1 単一量子ビットゲート

従来のコンピュータでは古典ビット情報を含んだ電気的な信号をトランジスタなどの半導体で構成される論理回路(ゲート)に通すことでその状態を変化させて計算処理している。

量子コンピュータも同様に量子ゲートとよばれる装置に量子ビット情報を通すことで状態ベクトルを変化させることができる。

一般的に量子ゲートは数学的には行列として表現される。

例えば量子ビットゲートには次のようなゲートが定義されている。



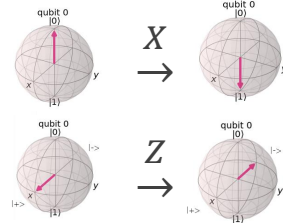
上記のような量子ゲートは単一の量子ビットに作用し、その状態ベクトルを回転させる効果を持つ。

量子コンピュータは基本的には単一量子ゲートを駆使して量子ビットの状態を変化させることで高速で大容量な演算処理を実現している。

2 パウリゲート

単一量子ゲートの中では、Bloch球の各軸を基準に180° 回転させるXゲート、Zゲートなどがある。これはパウリゲートとも呼ばれそれぞれx軸回転、z軸回転に対応している。

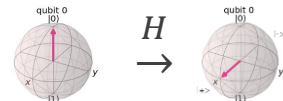
$$\begin{array}{l} \text{X Gate} \\ \text{Bit-flip, Not} \end{array} \quad \boxed{X} \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \beta|0\rangle + \alpha|1\rangle$$

$$\begin{array}{l} \text{Z Gate} \\ \text{Phase-flip} \end{array} \quad \boxed{Z} \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha|0\rangle - \beta|1\rangle$$


Xゲートについては $|0\rangle$ を $|1\rangle$ に、 $|1\rangle$ を $|0\rangle$ へ反転させることができるので、従来のコンピュータで置き換えるとNOTゲートに対応している。

3 アダマールゲート

単一量子ゲートで最もユニークなゲートとして、H(アダマール)ゲートがある。これは単一量子ビットを $|0\rangle$ と $|1\rangle$ に重ね合わせ状態に変化させることができ、様々な量子アルゴリズムで利用されている。

$$\begin{array}{l} \text{H Gate} \\ \text{Hadamard} \end{array} \quad \boxed{H} \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$


4 関連キーワード

量子回路[ シラバス 6-3-2]

5 参考文献

[単一量子ビットゲート - Qiskit](#)

1 Xゲートの作用

Xゲートの作用は、単一量子ビットをBloch球上の状態ベクトルを考えたときに、X軸周りに 180° (π ラジアン)だけ回転させる。

数学的に定義されるXゲートの作用の行列が、電子スピンの振る舞いに関するパウリスピン行列(スピン演算子)のX方向成分と一致しているためパウリゲートとも呼ばれる。

また、 $|0\rangle$ 、 $|1\rangle$ を反転させる作用があるため、古典ビットの論理回路に対応しBit-flipやNOTゲートなど呼ばれることがある。

(数学的な定義)

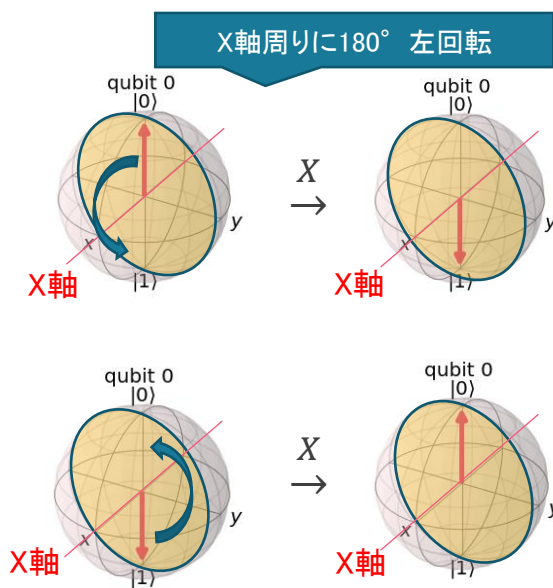
Xゲートは右の行列Xで定義されている。行列Xは $|0\rangle$ 、 $|1\rangle$ などの単一量子ビットをX軸周りに 180° (π ラジアン)だけ回転させる。

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ = |0\rangle\langle 1| + |1\rangle\langle 0|$$

(計算例)

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

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



2 Yゲートの作用

Yゲートの作用は、単一量子ビットをBloch球上の状態ベクトルを考えたときに、Y軸周りに 180° (π ラジアン)だけ回転させる。

数学的に定義されるYゲートの作用の行列が、電子スピンの振る舞いに関するパウリスピン行列(スピン演算子)のY方向成分と一致しているためパウリゲートとも呼ばれる。

また、Xゲートのように $|0\rangle$ 、 $|1\rangle$ を反転させる作用を持ちますが $|1\rangle$ について $-$ の位相変化を与える。

(数学的な定義)

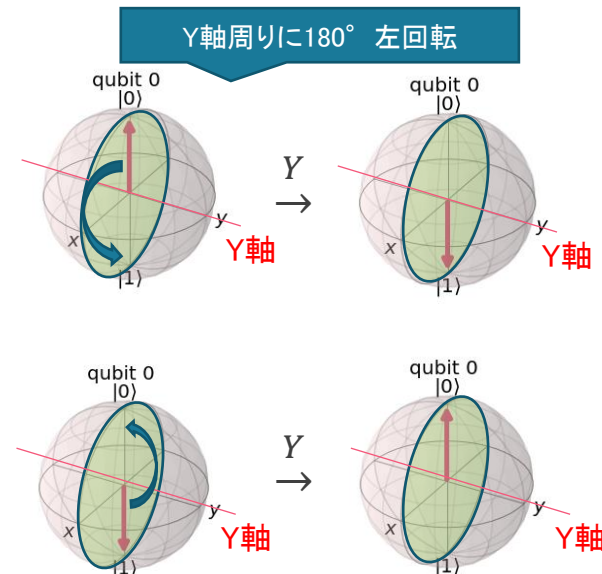
Yゲートは右の行列Yで定義されている。行列Yは $|0\rangle$ 、 $|1\rangle$ などの単一量子ビットをY軸周りに 180° (π ラジアン)だけ回転させる。

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \\ = -i|0\rangle\langle 1| + i|1\rangle\langle 0|$$

(計算例)

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

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



3 Zゲートの作用

Zゲートの作用は、単一量子ビットをBloch球上の状態ベクトルを考えたときに、Z軸周りに 180° (π ラジアン)だけ回転させる。

数学的に定義されるZゲートの作用の行列が、電子スピンの振る舞いに関するパウリスピ行列(スピン演算子)のZ方向成分と一致しているためパウリゲートとも呼ばれる。

また回転作用以外にも、 $|1\rangle$ について $-$ の位相変化を与える。

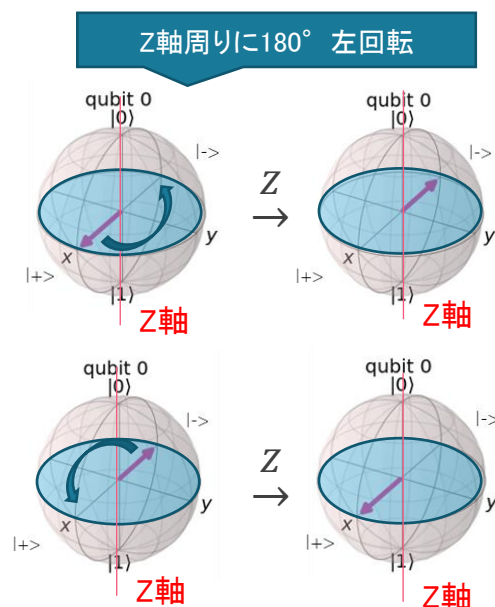
(数学的な定義)

Zゲートは右の行列Zで定義されている。行列Zは $|+\rangle$ 、 $|-\rangle$ などの単一量子ビットをZ軸周りに 180° (π ラジアン)だけ回転させる。

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

(計算例)

$$\begin{aligned} Z|+\rangle &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = |-\rangle \\ Z|-\rangle &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = |+\rangle \end{aligned}$$



4 Pゲートの作用

Pゲートは別名位相ゲートとも呼ばれ、その作用は、単一量子ビットをBloch球上の状態ベクトルを考えたときに、Z軸周りに ϕ ラジアンだけ回転させる。 ϕ は実数であり回転角の変数であるので、任意のZ軸回転を作用させることができる。

Pゲートが $\phi = \pi$ の場合、前述のZゲートと同値であるので、ZゲートはPゲートの特殊ケースであると言える。

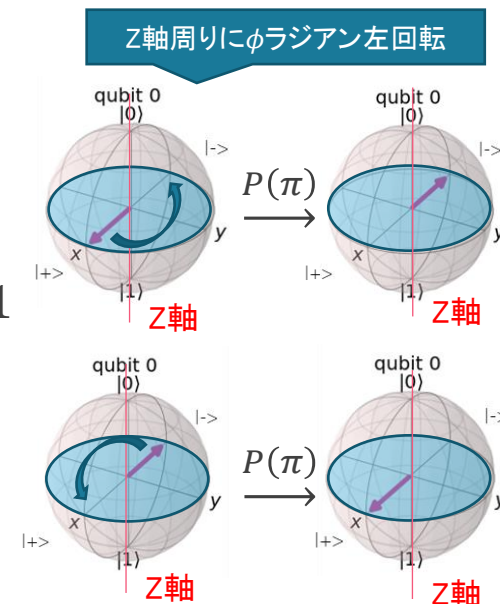
(数学的な定義)

Pゲートは右の行列Pで定義されている。行列Pは $|+\rangle$ 、 $|-\rangle$ などの単一量子ビットをZ軸周りに ϕ ラジアンだけ回転させる。

$$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$

(計算例)

$$\begin{aligned} P(\pi)|+\rangle &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = |-\rangle \quad \because e^{i\pi} = -1 \\ P(\pi)|-\rangle &= \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = |+\rangle \quad \because -e^{i\pi} = 1 \end{aligned}$$



5 Hゲートの作用

H(アダマール)ゲートの作用は、単一量子ビットをBloch球上の状態ベクトルを考えたときに、 $|0\rangle$ または $|1\rangle$ をZ軸からX軸へと向かう中間の 45° を軸とする面に対して 180° 回転させる。これは $|0\rangle$ および $|1\rangle$ の重ね合わせ状態として $|+\rangle$ または $|-\rangle$ へ変化することを意味している。また、 $|+\rangle$ または $|-\rangle$ への変化は、 $|0\rangle$ と $|1\rangle$ の重ね合わせ状態として確率的に表現される。(等確率)

(数学的な定義)

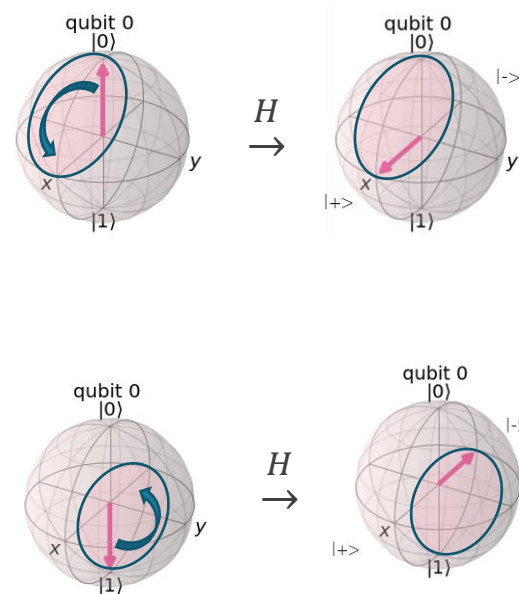
Hゲートは右の行列Hで定義されている。行列Hは $|0\rangle$ または $|1\rangle$ をZ軸からX軸へと向かう中間の 45° を軸とする面に対して 180° 回転させる。

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

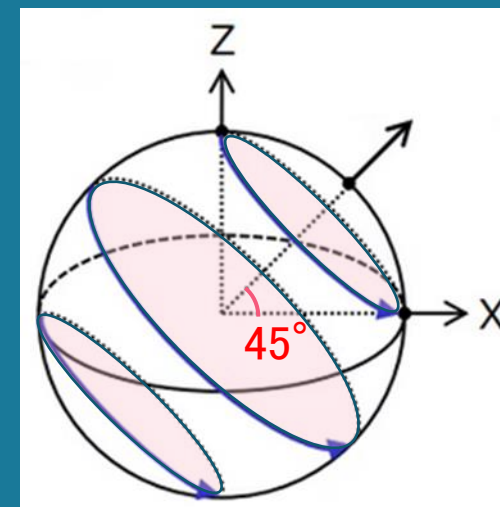
(計算例)

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \\ &= |+\rangle \end{aligned}$$

$$\begin{aligned} H|1\rangle &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \\ &= |-\rangle \end{aligned}$$



Z軸からX軸へと向かう中間の 45° を軸とする面に対して 180° 回転させます。



6 Iゲートの作用

Iゲートは、Identityゲートともよばれ”何もしない”作用がある。数学的には単位行列で定義されており、計算でよく利用される。
また実際のハードウェアを検討する際に、「何もしない」または「なし」の操作を指定する場合に利用される。

(数学的な定義)

Iゲートは右の行列Iで定義されている。
これは単位行列と同じように表現される。
Zゲートなど 180° (π ラジアン)回転させるゲートが2回作用すると元に戻る性質を表現できる。

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = ZZ$$

7 Sゲートの作用

Sゲートの作用は、単一量子ビットをBloch球上の状態ベクトルを考えたときに、Z軸周りに 90° ($\frac{\pi}{2}$ ラジアン)だけ回転させる。
また、Sゲートを2回作用させた場合、Zゲートと同じになるので、 \sqrt{Z} ゲートとも呼ばれる。

(数学的な定義)

Sゲートは右の行列Sで定義されている。
Pゲートの $\phi = \frac{\pi}{2}$ の場合と同値。

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix} = P\left(\frac{\pi}{2}\right)$$

(計算例)

$$SS|+\rangle = Z|+\rangle$$

8 Tゲートの作用

Tゲートの作用は、単一量子ビットをBloch球上の状態ベクトルを考えたときに、Z軸周りに 45° ($\frac{\pi}{4}$ ラジアン)だけ回転させる。
また、Tゲートを2回作用させた場合、 \sqrt{Z} ゲート(=Sゲート)と同じになるので、 $\sqrt[4]{Z}$ ゲートとも呼ばれる。

(数学的な定義)

Tゲートは右の行列Tで定義されている。
Pゲートの $\phi = \frac{\pi}{4}$ の場合と同値。

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} = P\left(\frac{\pi}{4}\right)$$

(計算例)

$$TT|+\rangle = S|+\rangle \quad TTTT|+\rangle = SS|+\rangle = Z|+\rangle$$

9 Iゲート、Sゲート、Tゲートの関係

Iゲート、Sゲート、Tゲートについては以下の関係でまとめられる。

$$I = ZZ = SSSS = TTTTTTTT$$

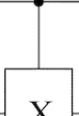
10 参考文献

[単一量子ビットゲート - Qiskit](#)

1 複数量子ビットゲート


複数量子ゲートは複数の量子ビットに作用し、数学的に行列として表現できる。例えば複数量子ビットゲートには次のようなゲートが定義されている。

Controlled Not
Controlled X
CNot



$$\equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = a|00\rangle + b|01\rangle + d|10\rangle + c|11\rangle$$

Swap

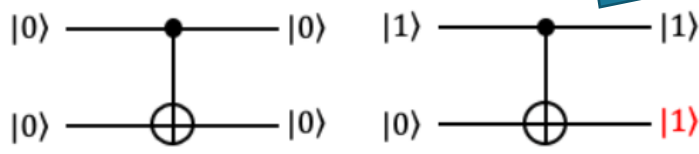
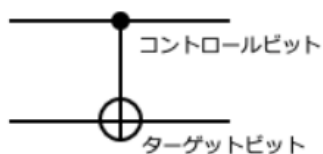


$$\equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = a|00\rangle + c|01\rangle + b|10\rangle + d|11\rangle$$

2 CNOTゲート

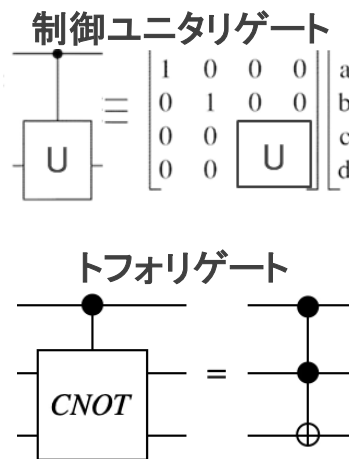
このゲートは条件付きゲートであり、1つ目の量子ビット（コントロール）が $|1\rangle$ の場合に2つ目の量子ビット（ターゲット）にXゲートを適用する。従来のコンピュータで置き換えるとXORゲートに対応している。

CNOTゲート



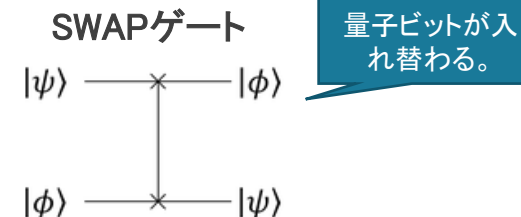
3 制御ユニタリゲート

CNOTゲートを一般化した呼び名で、コントロール量子ビットとターゲット量子ビットがある量子ゲート全般を制御ユニタリゲートと呼ぶ。ターゲット量子ビットに量子ゲートUを施すような制御ユニタリゲートを一般にCUゲートと呼び、Uに種々の回転ゲートが対応する。また、U自身にCNOTゲートも置くことができ、これはCCNOTゲートやトフォリゲートなどと呼ばれ、古典コンピュータ上のANDやNANDゲートに対応する。



4 SWAPゲート

このゲートは2量子ビットの1つ目の量子ビットと2つ目の量子ビット交換する。量子コンピュータ上で情報を移動する必要があるときに重要となる。



5 関連キーワード

量子回路でのアルゴリズム表現[] シラバス 6-4-1

6 参考文献

[複数量子ビットともつれ状態 - Qiskit](#)

1 量子ビットの重ね合わせと表現の限界

複数量子ビットの重ね合わせの状態は、以下のようにテンソル積（積状態）で表現される。

$$|a\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}, \quad |b\rangle = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$
$$|ba\rangle = |b\rangle \otimes |a\rangle = \begin{bmatrix} b_0 \times \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \\ b_1 \times \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} b_0 a_0 \\ b_0 a_1 \\ b_1 a_0 \\ b_1 a_1 \end{bmatrix}$$

しかし、このようなテンソル積で表現できない複数量子ビットの重ね合わせがあり、その状態を「もつれ（状態）」またはエンタングルと呼ぶ。

例えば以下の2つの量子ビットの状態を考える。

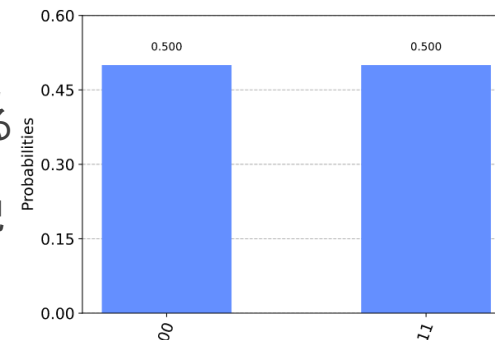
$|0\rangle \otimes |+\rangle = |0+\rangle$ を生成し、CNOTゲートを作用された場合は、以下の状態の量子ビットが得られる。

$$\text{CNOT}|0+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

この状態を前述のテンソル積で表現した場合、この状態を満たす $|a\rangle$ 、 $|b\rangle$ が存在しないためテンソル積の積状態で表現できず、上記の量子ビットの状態はもつれ状態といえる。

この状態を測定すると、50%の確率で $|00\rangle$ の状態が観測され、50%の確率で $|11\rangle$ の状態が確認できる。

尚、観測で一番上の桁の量子ビットを観測して $|1\rangle$ が得られた場合、結合しているもう片方の量子ビットの状態も $|1\rangle$ に決まるため、このようなもつれ状態は20世紀初期には「不気味な遠隔作用」と呼ばれていた。



もつれ状態は、このような不思議な作用を持つため量子テレポーテーションやさまざまなアルゴリズムに応用されている。

2 有名なもつれ

よく知られているもつれ状態として、Bell状態、GHZ状態の他、W状態、Telecloning状態、Cluster状態などさまざまなもつれ状態が確認されている。

3 関連キーワード

テンソル積[ シラバス 8-1-1]

4 参考文献

[複数量子ビットともつれ状態 - Qiskit](#)

1 位相キックバック

前述した制御ユニタリゲートを使った量子演算の重要なテクニックとして、位相キックバックがある。

位相キックバックは制御ユニタリゲートの固有値をコントロール量子ビットの位相(係数)に反映させる操作である。

位相キックバックの利点は2つある。

(利点)

1. コントロール量子ビットを補助量子ビットとみなせば、ターゲット量子ビットの状態を変えずに制御ユニタリゲートの固有値情報を補助量子ビットに取り出しできる。
2. 制御ユニタリゲートをうまく選べば、コントロール量子ビットの位相を取り出すだけで、ターゲット量子ビットの固有ベクトルの固有値を推定することができる。

このような利点から位相キックバックは、アダマールテストやShorのアルゴリズムなど様々な量子アルゴリズムに応用されている。

(例: アダマールテスト ※以下は計算途中の一部)

$$|0\rangle|\psi\rangle \xrightarrow{H_0} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle$$

$$\xrightarrow{CU_{0,1}} \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle U|\psi\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\lambda}|1\rangle)|\psi\rangle$$

制御ユニタリゲートの固有値がコントロール量子ビットの位相に反映

2 位相キックバックの操作条件

位相キックバックをするためには、以下の条件で制御ユニタリゲートを作用させる必要がある。

(条件)

1. コントロール量子ビットを重ね合わせ状態で入力する。

例) アダマールゲートを作用させて重ね合わせを作る。

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

2. ターゲット量子ビットを制御ユニタリゲートの固有ベクトルとする。

例) ターゲット量子ビット $|\psi\rangle$ にユニタリ演算 U を作用させたときに固有値 $e^{i\lambda}$ を取ることができるとする。

$$U|\psi\rangle = e^{i\lambda}|\psi\rangle$$

左記の例: アダマールテストでは上記の条件で $|0\rangle$ はコントロール量子ビット, $|\psi\rangle$ はターゲット量子ビットの $|0\rangle|\psi\rangle$ に制御ユニタリゲート CU を作用させている。

3 関連キーワード

アルゴリズム[ シラバス 6-4-2]

4 参考文献

[位相キックバックの特徴と重要性を5分で理解する - Investor D](#)

1 量子プログラミング

量子プログラミングは、量子コンピュータを実行するための量子計算に基づいた操作を記述すること。

ここでは量子ゲートコンピュータ[]シラバス 3-2]を扱うためのオープンソースであるQiskitを例に実際の実装の手順を紹介する。

環境設定について[]詳説-1]参照

2 QiskitでBell状態を作成してみよう

①まず、量子回路を用意する。
00状態が用意される

②量子ゲート操作を行う
アダマールゲートと
CNOTゲートでBell状態を作る
その他のゲートについて[]詳説-

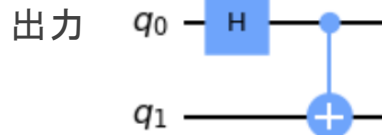
③回路を描写して確認

```
from qiskit import QuantumCircuit
qc = QuantumCircuit(2)
```

```
qc.h(0)
qc.cx(0,1)
qc.draw('mpl')
```

2量子ビットの回路を作成
量子ビットの初期値は0

0以外の値に
初期化可能[]詳説-3]



2つ目の引数に古典ビット数を指定
QuantumCircuit(2,2)

もしくはレジスタを作成して、引数にレジスタを指定する事もできる

```
qr=QuantumRegister(3,'q')
cr=ClassicalRegister(2,'c')
QuantumCircuit(cr,qr)
```

‘q’ と命名した3量子ビット
‘c’ と命名した2古典ビット

3 作成した回路を測定して実行してみよう

④-1シミュレータでの実行

```
qc.measure([0,1],[0,1])
backend = BasicAer.get_backend('qasm_simulator')
execute(qc, backend, shots=1024).result().get_counts()
```

測定[]詳説-5]

実機のような動作をする
シミュレータ[]詳説7]

1024回実行

出力 {'00': 521, '11': 503}

00状態が521回、11状態503
測定される

4 実行結果を確認してみよう

④-2実機での実行

```
qc.measure([0,1],[0,1])
backend=IBMQ.get_provider().get_backend('ibmq_quito')
execute(qc, backend, shots=1024).result().get_counts()
```

実機[]詳説8]
を指定

出力 {'00': 489, '01': 101, '10': 72, '11': 362}

qasm simulatorと比べてエラーが
多く理論値以外の結果も検出される

5 参考文献

- [Qiskit Textbook](#)
- 湊雄一郎,比嘉恵一郎,永井隆太郎,加藤拓己.「IBMQで学ぶ量子コンピュータ」.2021-3-10.秀和システム.Chapter3 01-03

1 環境設定

①IBM Quantum Experienceのアカウント作成

1. IBM Q Experience (<https://quantum-computing.ibm.com>) を開き、Sign in to IBM Quantum の下に表示された各アイコンからログイン可能



2. もしくCreate an IBMid account. をクリックして新しいIBM IDを作成する



②実行環境の用意

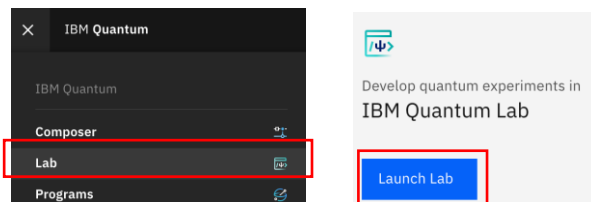
QiskitはPythonのフレームワークであるためPythonの環境を用意し

```
pip install qiskit
```

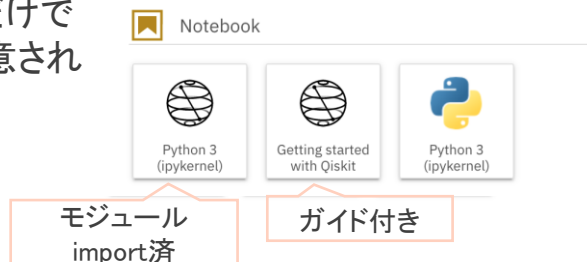
を実行してQiskit をインストールする

IBM Quantum上のQuantum LabではJupyter Notebook環境が用意されておりPythonの環境構築なしで使用する事ができる

トップページのLaunch Lab
もしくは左のタブメニューLab
をクリック

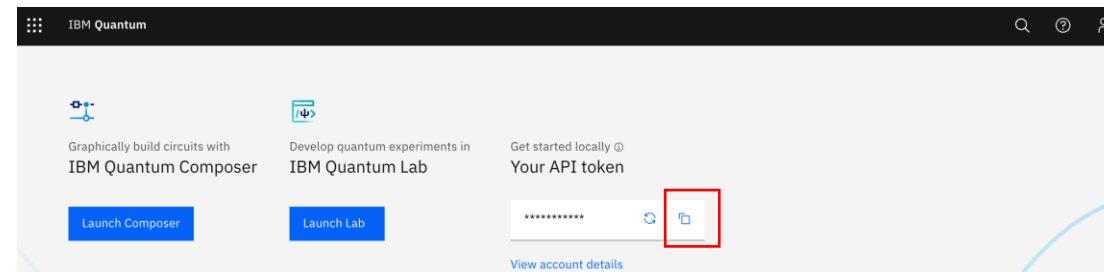


通常のJupyter Notebookだけでなく初心者用のガイドも用意されている



③APIの取得

IBM Q Experience にログインし、Your API token の枠内の右のアイコンをクリックして API をコピーする



④認証情報の保存

下記コードの 'my_token' にコピーした API を貼り付けて実行し認証情報を保存する

```
from qiskit import IBMQ
IBMQ.save_account('my_token')
```

また、2回目以降は下記コード実行により保存された情報を読み込むことができる

```
IBMQ.load_account()
```

2 初期化

量子ビットのデフォルトの初期値は $|0\rangle$ に設定される。
initializeを使用して、これを任意の状態に初期化できる。
ただし、正規化されていないベクトルを与えるとエラーとなる。

例えば2ビットの量子回路の上から2番目の量子ビットを $|1\rangle$ に初期化

```
circuit = QuantumCircuit(2)
circuit.initialize([0,1], 1)
```

$|1\rangle$ の状態ベクトル

初期化するビット

3 量子ゲート

1量子ビットゲート

qc.x(0) Xゲート
qc.y(0) Yゲート
qc.z(0) Zゲート
qc.h(0) Hゲート

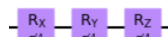


qc.s(0) Sゲート
qc.sdg(0) S†ゲート
qc.t(0) Tゲート
qc.tdg(0) T†ゲート



qc.p(np.pi/4,0) Pゲート

qc.rx(np.pi/4,0) Rxゲート
qc.ry(np.pi/4,0) Ryゲート
qc.rz(np.pi/4,0) Rzゲート



複数量子ビットゲート

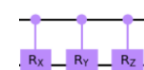
qc.cx(0,1) Controlled X
qc.cy(0,1) Controlled Y
qc.cz(0,1) Controlled Z



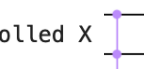
qc.cp(np.pi/4,0,1) Controlled P



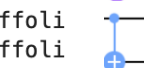
qc.crx(np.pi/4,0,1) Controlled Rx
qc.cry(np.pi/4,0,1) Controlled Ry
qc.crz(np.pi/4,0,1) Controlled Rz



qc.ccx(0,1,2) Controlled Controlled X
qc.toffoli(0,1,2) Toffoli



qc.mct([0],1) Multiple Control Toffoli
qc.mct([0,1],2) Multiple Control Toffoli



qc.swap(0,1) Swap
qc.cswap(0,2,3) Controlled Swap



4 測定

重ね合わせの量子ビットが与えられていても複数の情報を直接得ることはできない。測定をすることで0、1のいずれかの状態に定まった値のみ得ることができる。

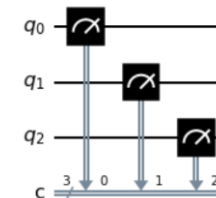
また、測定によって得られる値は確率的に決まる。この確率は基底との内積によって定義される。QiskitはZ基底での測定を行う。

量子ビットと値を書き込む古典ビットを指定して測定

```
qc = QuantumCircuit(3,3)
qc.measure([0,1,2],[0,1,2])
qc.draw(output='mpl')
```

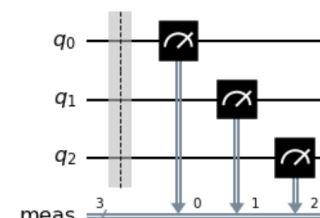
量子ビット

古典ビット



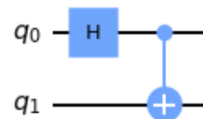
古典ビットを用意しなくても測定可能

```
qc=QuantumCircuit(3)
qc.measure_all()
qc.draw(output='mpl')
```



5 シミュレーション

Qikitの土台となるQiskit Terraのシミュレータには下記の3つが存在し、図の回路のBell状態を測定した時のそれぞれの結果を下記に示す



▪ statevector simulator 理論上の量子回路の出力状態ベクトルを計算する

```
backend = BasicAer.get_backend('statevector_simulator')
execute(qc, backend).result().get_statevector(qc)
```

出力 array([0.70710678+0.j, 0. +0.j, 0. +0.j, 0.70710678+0.j])

▪ unitary simulator 理論上の量子回路のと等価なユニタリ変換行列を計算する

```
backend = BasicAer.get_backend('unitary_simulator')
execute(qc, backend).result().get_unitary()
```

出力 array([[0.70710678+0.00000000e+00j, 0.70710678-8.65956056e-17j,
 0. +0.00000000e+00j, 0. +0.00000000e+00j],
 [0. +0.00000000e+00j, 0. +0.00000000e+00j,
 0.70710678+0.00000000e+00j, -0.70710678+8.65956056e-17j],
 [0. +0.00000000e+00j, 0. +0.00000000e+00j,
 0.70710678+0.00000000e+00j, 0.70710678-8.65956056e-17j],
 [0.70710678+0.00000000e+00j, -0.70710678+8.65956056e-17j,
 0. +0.00000000e+00j, 0. +0.00000000e+00j]])

▪ qasm simulator 実機を模範したノイズを伴うシミュレータ

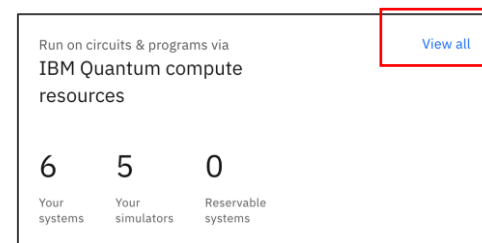
```
qc.measure([0,1],[0,1]) 状態のカウント数を出力する
backend = BasicAer.get_backend('qasm_simulator')
execute(qc, backend, shots=1024).result().get_counts()
```

出力 {'00': 521, '11': 503}

6 実機

Qiskitでは無料で実行可能な実機を提供している。使い方としてはまず、%qiskit_backend_overview もしくは IBM Quantum Experienceにアクセスして実行可能な実機を確認する。

IBM Quantum Experienceのトップページの右図のView allから確認可能



```
from qiskit import IBMQ
IBMQ.load_account()

qc.measure([0,1],[0,1])
backend=IBMQ.get_provider().get_backend('ibmq_quito')
execute(qc, backend, shots=1024).result().get_counts()
```

使用する実機名を入力して実行

出力 {'00': 489, '01': 101, '10': 72, '11': 362}

7 参考文献

- [IBM Q Experience](#)
- 湊雄一郎,比嘉恵一郎,永井隆太郎,加藤拓己.「IBMQで学ぶ量子コンピュータ」.2021-3-10.秀和システム
- [Quantum Native Dojo 3-2. QiskitとIBM Q Experienceの使い方](#)
- [Qiskit Textbook](#)

1 量子アルゴリズム

アルゴリズムは問題を解くために、したがうべき計算などの手順を記したものである。

アルゴリズムはコンピュータのアーキテクチャなどに依存しない形式で表現されることが多いが、量子コンピュータで実行されることを意識して作られたアルゴリズムを量子アルゴリズムという。

2 量子アルゴリズムの特徴と代表例

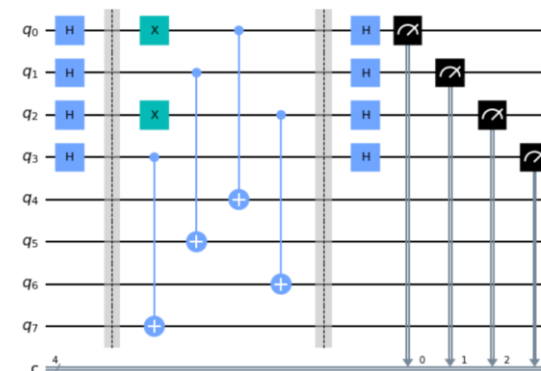
量子アルゴリズムは、重ね合わせや干渉などの量子計算の特徴を生かしたさまざまなものが提案されている。

古典コンピュータで実行されることを意図してつくられたアルゴリズムと比較し、計算回数などで有利となるような問題について、量子計算で解くアルゴリズムが提案されている。代表的な例としては大きな数の素因数分解を行うShor(ショア)のアルゴリズムなどがある。

3 量子アルゴリズムの表現と実装

量子アルゴリズムは、一般的には量子回路という量子ビット、古典ビットとそれらに対して作用する量子ゲート等からなる図で表現される。実装は、OpenQASMというアセンブラ形式の言語などでなされる。

一例として、右図に「Simonのアルゴリズム」という、ある種の未知数を求めるアルゴリズムを示した。



量子回路の例

```
'OPENQASM 2.0;
include "qelib1.inc";
qreg q[8];
creg c[4];
h q[0];
h q[1];
h q[2];
h q[3];
barrier q[0],q[1],q[2],q[3],q[4],q[5],q[6],q[7];
cx q[0],q[4];
x q[0];
cx q[0],q[4];
cx q[0],q[7];
x q[0];
```

QASMによる実装例

4 参考文献

Andrew W. Cross, Lev S. Bishop, John A. Smolin, Jay M. Gambetta,
Open Quantum Assembly Language,
<https://arxiv.org/abs/1707.03429>

1 アルゴリズム

アルゴリズムは、「算法」とも言い、何らかの問題を解くための計算手順を表したものである。
古典コンピュータでは情報が2進の古典ビットで表されるが、そのビットに対して必要な論理演算を行うことで、アルゴリズムの実行を実現している。

2 量子アルゴリズム

量子コンピュータの特徴を生かし、何らかの問題を解くための手順を表したものを量子アルゴリズムという。
量子コンピュータでは、量子ビット、量子ゲートなどの素子を使って量子回路を構成し、アルゴリズムを実装することが一般的である。
また、量子回路を構成するためのプログラミング言語やライブラリなどもある。

量子コンピュータでは、 n 個の量子ビットを使って、 2^n パターンの古典ビットの状態を重ね合わせて表現することができ、干渉効果を用いて、問題の答を求めることができる。

古典アルゴリズムでは繰り返して計算を行う必要があるものが、量子アルゴリズムでは、重ね合わせと干渉を用いることにより、その計算回数を大幅に削減することができる。

一般的には古典アルゴリズムと比較し、計算回数で指数的な縮減ができることを狙って量子アルゴリズムが考案・開発されることが多い。

量子状態が必要な時間保持され、ノイズ等による誤りがない理想的な条件のもとでの量子アルゴリズムでの処理は、古典コンピュータの場合と同様、ハードウェアのビット数やメモリの容量、それらの処理速度に依存する。現時点では量子状態の保持時間や誤り訂正などの制約があり、こうした制約を考慮してアルゴリズムの実装などが行われる。

3 参考文献

藤井啓祐、驚異の量子コンピュータ 宇宙最強マシンへの挑戦、岩波書店、2019.

1 基本的な3種類のアルゴリズム

ここでは基本的な以下の3種類のアルゴリズムについて述べる。Deutsch-Jozsaのアルゴリズム、Bernstein-Vaziraniのアルゴリズム、Simonのアルゴリズムは、あらかじめ量子回路に組み込まれたオラクルと呼ばれる関数について、関数のタイプの識別をしたり未知数を求めるアルゴリズムである。

オラクルはいわゆる中身が分からないブラックボックスである。 n 次元の2進ビット列の入力から m 次元の2進ビット列を出力するような関数を想定しているが、3種類のアルゴリズムで想定されているオラクルはそれぞれ異なる。例えば入力のビット列と未知数の2を法とする内積を出力するものなどである。

これらのアルゴリズムでは、アダマールゲートを使って重ね合わせ状態を作り、オラクルを呼び出して、その結果について再びアダマールゲートを作用させ、測定することにより必要な情報を得る。なお、Simonのアルゴリズムは量子計算後、得られた結果をもとに、古典コンピュータ上で連立1次方程式を解く必要がある。各アルゴリズムの計算量などを比較したものを右の表に示した。

2 Shorのアルゴリズム

Shor(ショア)のアルゴリズムは、量子計算で素因数分解を行うアルゴリズムである。

素因数分解を行う整数 n について、古典アルゴリズムでは計算回数が n の指数関数に比例する。(右上へ続く)

3種類のアルゴリズムの計算量などの比較

アルゴリズム	対象とする問題	古典的計算量	量子的計算量	漸近的なスピードアップ
Deutsch-Jozsa	定数型とバランス型関数の判別	$2^{n-1} + 1$	1	指数
Bernstein-Vazirani	ドット積の未知数を求める	n	1	多項式
Simon	XORマスクの未知数を求める	$O(2^{n/2})$	$O(n)$	指数

(左下より)

量子アルゴリズムでは n のべき関数と対数の積に比例する回数となり、大きな数の素因数分解では量子アルゴリズムの方が圧倒的に少ない計算回数で素因数分解を行うことができる。

Shorのアルゴリズムは、量子フーリエ変換、量子位相推定など複数のアルゴリズムなどから構成されている。

3 参考文献

Thomas G. Wong, Introduction to Classical and Quantum Computing, Rooted Grove, 2022.

<https://www.thomaswong.net/introduction-to-classical-and-quantum-computing-1e3p.pdf>

本資料の著作権は、日本アイ・ビー・エム株式会社（IBM Corporationを含み、以下、IBMといいます。）に帰属します。

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMまたはセッション発表者は責任を負わないものとします。本資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものでもありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴは、米国やその他の国におけるInternational Business Machines Corporationの商標または登録商標です。他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点でのIBMの商標リストについては、ibm.com/trademarkをご覧ください。