第１章

量子コンピュータへの
いざない

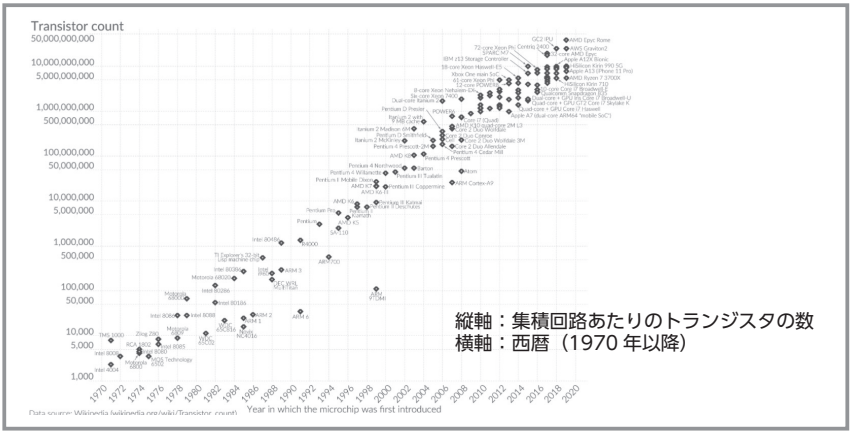
$$|\varphi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{C}^2$$

1.1　量子コンピュータへの期待

私たちが使っているコンピュータは 20 世紀に大きく発展し、普及しました。特に 20 世紀後半の発展は凄まじく、「集積回路あたりの部品数が毎年 2 倍になる」という**ムーアの法則**（Moore's law）が提唱されました[*1]。この法則は、その後「2 年で 2 倍」に修正されたものの、この法則にしたがってコンピュータの計算能力は指数関数的に向上しました。

20 世紀半ばには大きな部屋いっぱいのサイズだったコンピュータは、ムーアの法則にしたがって集積化が進み、21 世紀の現在ではスマートフォンに搭載されて持ち運びができるまでになりました。しかし、集積回路の小型化が進んだ結果、物理的な限界に達しつつあります。

図 1.1：ムーアの法則

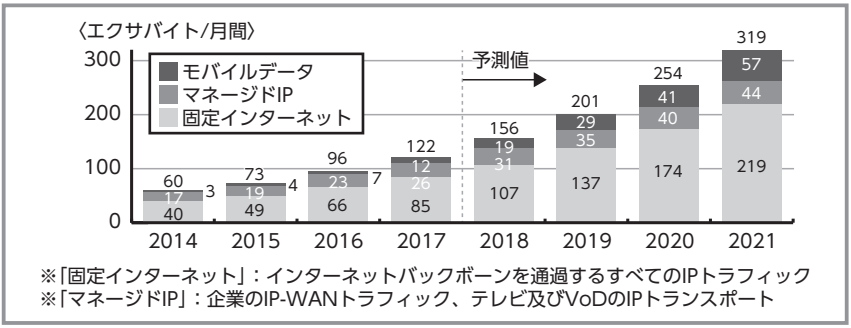


https://en.wikipedia.org/wiki/Moore%27s_law#/media/File:Moore's_Law_Transistor_Count_1970-2020.png より引用

＊1　G. E. Moore, "Cramming more components onto integrated circuits", *Electronics* 38.8 (1965).

一方で、コンピュータが発展するにつれ、人類が扱うデータ量は急激に増大しています。インターネットができた当初はテキストのように小さなデータを中心に扱っていましたが、画像や動画といった大きなサイズのデータも扱うようになりました。Web 会議の普及などにより、今後もデータ量が増大していくと推測されます。増大するデータを扱うため、ムーアの法則が鈍化しても指数関数的な計算能力の成長を期待したくなります。

図 1.2：世界のトラフィックの推移及び予測



https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r01/html/nd112110.html より引用

こうした中、指数関数的な計算能力の成長の候補となるのが**量子コンピュータ**（quantum computer）です。量子コンピュータと区別するため、従来のコンピュータを**古典コンピュータ**（classical computer）といいます。物理学では「古典力学」「量子力学」という呼び方は一般的で、区別のために「古典」と名付けられていますが、ネガティブな意味はありません。古典芸能、クラシック音楽（classical music）、古典制御など、現在も活躍している「古典」たちと同様です。

量子コンピュータでどんな計算でも速くなるわけではありませんが、古典コンピュータと比べて指数関数的に高速に計算できるアルゴリズムが発見されています。このような流れから、量子コンピュータに期待がかかっています。

1.2　量子コンピュータの歴史

量子コンピュータは、1980 年代に提唱されました。以下に、量子コンピュータの歴史を簡単にまとめました。

- 1980 年代
  - ファインマンなどにより量子コンピュータの概念が提唱される
  - ドイッチュにより理論的に定式化される（量子チューリングマシン）
- 1990 年代
  - ショアにより高速に素因数分解できる量子アルゴリズムが発見され、期待が高まる
  - ショアやスティーンにより量子誤り訂正が発見され、実現性への期待が高まる
  - イオントラップ量子ビット、超伝導量子ビットなどの量子ビットが実現される
- 2010 年代
  - 計算精度の向上、2 桁量子ビットが実現される
  - IBM 社などにより量子コンピュータの実機が公開され、インターネット経由で実行できる環境が整備される
  - 各国が予算を割き、量子コンピュータの研究が急速に発展した

量子コンピュータの研究に力を入れる研究機関や企業が増え、ここ数年、日進月歩で進んでいます。しかし、2023 年現在、実用的な計算では古典コンピュータの方が速いのが正直なところです。実用的な計算で古典コンピュータより速い量子コンピュータが完成するのはまだまだ先で、十年単位の時間がかかると思われます。

1.3 古典コンピュータと量子コンピュータの違い

1.3.1 なぜ量子コンピュータは速いのか

量子コンピュータについて学ぶにあたり、「なぜ量子コンピュータは速いのか」を知っておくと学びやすいと思います。

この問いには、さまざまな視点からの回答がありえます。「普通の人」が「使う立場」で見たときの回答としては、「指数関数的に巨大なユニタリ行列のかけ算を高速に実行できるから」と捉えるのが分かりやすいと考えています。これは標語的な説明なので、正確には注釈が必要ですが、概ねこのように捉えて構いません[*2]。また、この説明には量子力学固有の専門用語は登場しません。「ユニタリ行列」は馴染みがないかもしれませんが、特定の種類の行列のことで、第2章で説明します。

量子コンピュータの計算では行列が重要になるため、本書では行列のかけ算をしながら量子コンピュータの計算ルールを理解します。また、理解を定着させるために、実際に手を動かして計算することを重視しています。ルールに沿って実際に手を動かして慣れましょう。

1.3.2 古典ビットと量子ビット

コンピュータが情報を扱う単位である、古典ビットと量子ビットの違いについて、概念的に説明します。科学として扱うために必要な定式化は第3章以降で行います。

私たちが普段使っている古典コンピュータの情報の単位は**ビット**（bit）です。量子コンピュータのビットと区別するため、本書では**古典**

＊2 「GPUは行列計算が高速だから、画像処理が速い」くらいのイメージです。

ビット（classical bit）といいます。1 古典ビットは 0 または 1 で、2 種類の情報を表せます。2 古典ビットであれば 00、01、10、11 の 4 通りの情報を表せます（**表 1.1**）。このように、n 古典ビットでは 2^n 通りの情報を表せます。ただし、古典ビットがいくつあっても、同時に持てる値は 1 通りです。また、古典ビットは、同じ演算を行って結果を読み取ると、毎回同じ結果を得ます。

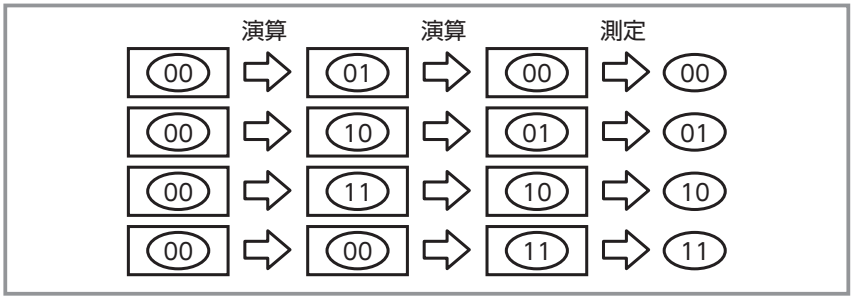
演算した結果を読み取ることを、**測定**（measurement）と呼びます。

表 1.1：古典ビット

ビット数	古典ビット
1 古典ビット	0、1
2 古典ビット	00、01、10、11

図 1.3 のように、古典コンピュータの並列計算では同時に複数の計算を行い、複数の結果を測定できます。

図 1.3：古典コンピュータの並列計算のイメージ



一方、量子コンピュータの情報の単位を**量子ビット**（quantum bit、qubit）といいます。英語では「qubit」と書かれ、「キュービット」と読みます。1 量子ビットは $|0\rangle$ または $|1\rangle$ という記号を使い、2 種類の情報を表せます。2 量子ビットであれば $|00\rangle$、$|01\rangle$、$|10\rangle$、$|11\rangle$ の 4 通り

の情報を表せます（**表 1.2**）。n 量子ビットの情報は 2^n 通りの情報を表せます。ここまでは、古典ビットも量子ビットも同じです。

表 1.2：量子ビット

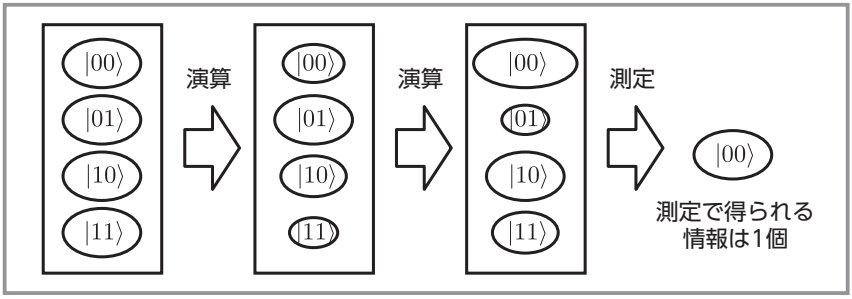
ビット数	量子ビット
1 量子ビット	$\|0\rangle$、$\|1\rangle$
2 量子ビット	$\|00\rangle$、$\|01\rangle$、$\|10\rangle$、$\|11\rangle$

しかし、量子ビットは古典ビットとは違った性質があります。実は、n 量子ビットは 2^n 通りの情報を同時に持てます。そして、量子コンピュータの演算は、この 2^n 通りの情報を 1 度に操作できる利点があります。

ただし、扱いづらい点もあります。2^n 通りの情報を同時に持てますし、同時に演算できますが、測定で得られる情報はそのうちの 1 個だけです。また、量子ビットを測定した結果は確率的に決まり、同じ操作をしても $|0\rangle$ が返ってきたり、$|1\rangle$ が返ってきます。そのうえ、測定すると量子ビットの状態も変わってしまいます。

そのため、量子コンピュータでは**図 1.4** のように、測定したときに目的の計算結果を得る確率が高くなるように演算を行います。**図 1.4** では、大きな楕円ほど測定で得られる確率が高いことをイメージしています。

図 1.4：量子コンピュータの計算イメージ



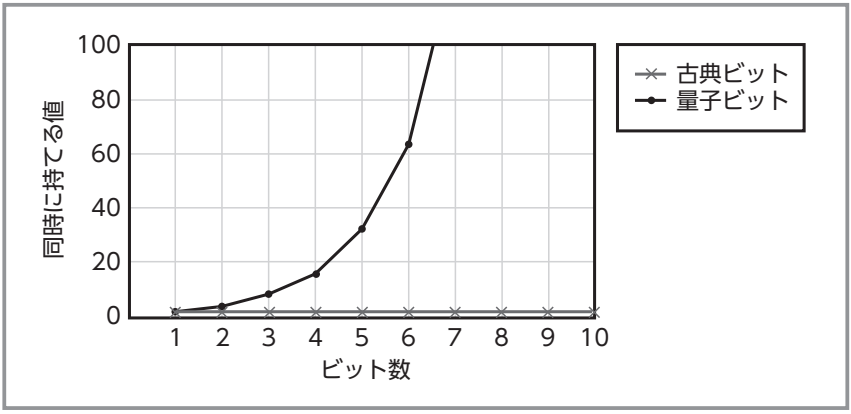
古典コンピュータの並列計算では同時に複数の計算を行って複数の結果を測定できるため、1個しか測定できない量子コンピュータとは計算のイメージが異なります。したがって、量子コンピュータで普通に計算しても高速にはならず、効果的に使うにはアルゴリズムに工夫が必要です。量子コンピュータ用のアルゴリズムを**量子アルゴリズム**（quantum algorithm）といいます。

古典ビットと量子ビットの特徴をまとめると、**表 1.3** のようになります。

表 1.3：古典ビットと量子ビットの特徴

比較対象	古典ビット	量子ビット
n ビットで表せる情報の種類	2^n 通り	2^n 通り
n ビットが同時に持てる値	1 通り	2^n 通り
測定で得られる情報の種類	1 通り	1 通り
測定で得られる値	同じ操作なら一定	同じ操作でも確率的
測定時の状態の変化	なし	あり

図 1.5：古典ビット、量子ビットが同時に持てる値



1.3.3 量子コンピュータの速さとユニタリ行列

ここでは、「指数関数的に巨大なユニタリ行列のかけ算を高速に実行できる」という量子コンピュータの特徴をもう少し掘り下げて、量子コンピュータが速い理由を説明します。

n 量子ビットで表せる情報の種類が 2^n 通りのため、数式としては 2^n 次元**ベクトル**（vector）[*3] で表せます。先ほど、量子コンピュータでは計算結果を 1 個しか測定できないと説明しましたが、ベクトルで表したときの絶対値が大きい成分ほど、その成分に対応する量子ビットを測定で得る確率が高くなります[*4]（**図 1.4**）。

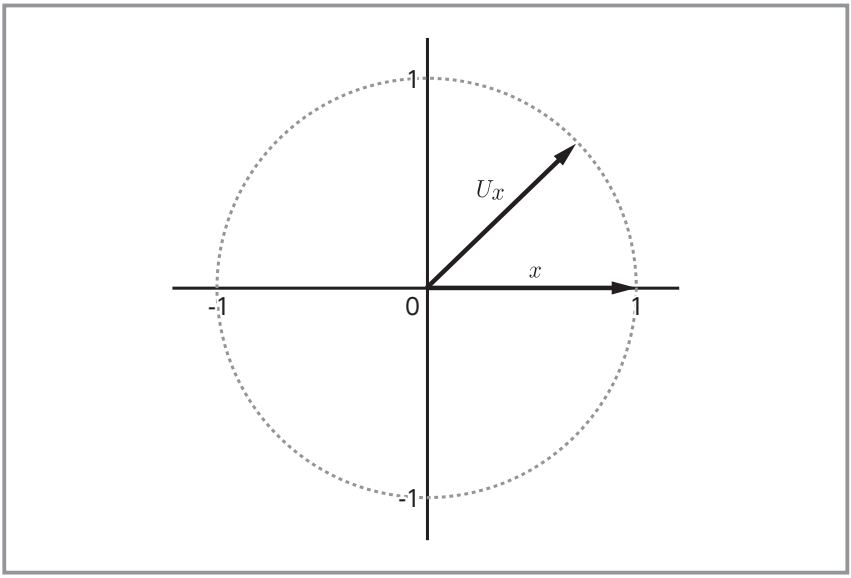
量子コンピュータによる計算は、n 量子ビットの情報（2^n 次元ベクトル）を別の n 量子ビットの情報（2^n 次元ベクトル）に変換する処理であるため、$2^n \times 2^n$ 次行列で表せます。もう少し正確にいうと、量子コンピュータで計算できる形には制約があるため、**ユニタリ行列**（unitary matrix）という種類の行列になります。ユニタリ行列は、ベクトルにかけても長さを変えない行列です[*5]。このユニタリ行列を計算すれば、古典コンピュータで量子コンピュータをシミュレーションできます。

＊3　プログラミングとしては 2^n 次元配列のイメージです。

＊4　正確には、各成分（複素数）の絶対値の 2 乗が、その成分に対応する量子ビットを測定で得る確率と一致します。詳しくは、第 3 章で説明します。

＊5　ユニタリ行列については、第 2 章や第 3 章で詳しく説明します。

図 1.6：ベクトル x にユニタリ行列 U をかけても、ベクトルの長さは変わらない。「x の長さ＝Ux の長さ」であり、この図では長さが 1 のまま変わらない。



ただし、行列のサイズが量子ビット数に対して指数関数（$= 2^n$）になるため、量子ビット数が増えると行列の計算時間やメモリ量も指数関数で増加します。情報を表すベクトルは 2^n 次元ベクトルなので、1 量子ビット増えるたびにサイズ（＝必要なメモリ量）が 2 倍になります。工夫してある程度メモリ量を圧縮できますが、それにも限界はあります。そのため、古典コンピュータによるシミュレーションは、量子ビット数が増えてくると困難になります。

一方、量子コンピュータは量子ビット数が増えても、計算時間への影響は少ないです。そのため、古典コンピュータと比べて、量子コンピュータは指数関数的に巨大なユニタリ行列を高速に計算できます。

したがって、巨大なユニタリ行列を使って効率よく計算できるアルゴリズムは、量子コンピュータで高速化できます。ここでは、概念的な説明をしましたが、第 3 章以降で数式を使って具体的に説明します。

第 2 章

量子コンピュータ入門以前

$$|\varphi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{C}^2$$

この章では量子コンピュータの計算ルールを理解するために必要な数学の記法と性質について説明します。量子コンピュータの計算で必要となる数学は、主に行列やベクトル、確率、複素数です。特に、次の 3 つは頻繁に使います。

- ブラケット記法
- ユニタリ行列
- テンソル積

第 3 章以降で、数学的に分からないことがあれば、この章に戻って確認してください。また、本書ではコンピュータの動作を回路の形式で説明します。そのため、古典コンピュータの回路（古典回路）についても説明します。本書を読み終えた後により本格的に量子コンピュータを学ばれることを想定し、本書には登場しない概念も説明しています[*1]。

はじめての概念を理解するには時間がかかる場合があります。理解できなくても、まずは飛ばして読んで構いません。必要になったときに戻ってくれれば大丈夫です。

2.1　量子コンピュータは「行列」の世界

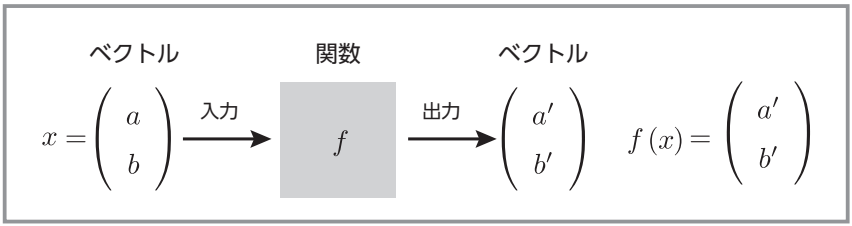
本節では、量子コンピュータの計算のイメージを説明します。用語や記法の正確な説明は次節以降で行いますので、ここではイメージができれば大丈夫です。

量子コンピュータの説明の前に、古典コンピュータの関数についてイ

＊1　たとえば、複素数について説明しますが、本書の後半で紹介する量子アルゴリズムでは複素数を利用しません。

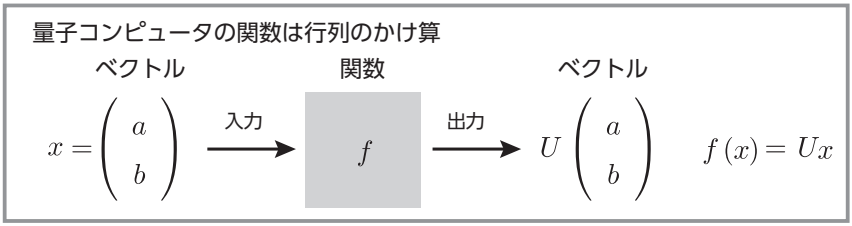
メージしてみましょう。ここで、2つの変数を入力とし、2つの変数を出力する関数をイメージしてみます。すると、**図 2.1** のように表せます。

図 2.1：古典コンピュータの関数



入力の変数は2個なので2次元の**ベクトル**（vector）[*2]で表せます。また、出力の変数も2個なので2次元のベクトルで表せます。そのため、ベクトル $\begin{pmatrix} a \\ b \end{pmatrix}$ が関数 f によって、ベクトル $\begin{pmatrix} a' \\ b' \end{pmatrix}$ になるイメージです。量子コンピュータで入力と出力の変数が2個ずつの場合は、**図 2.2** のようなイメージになります。

図 2.2：量子コンピュータの関数



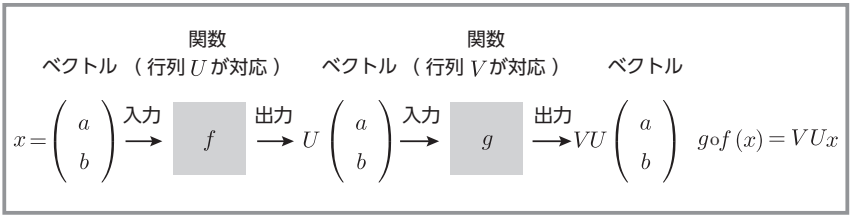
量子コンピュータの関数は、ベクトルに行列をかける形で表せます。入力の変数と出力の変数が2個なので2次元のベクトルで表せます。また、サイズが 2×2 の行列 U を入力のベクトルにかけたものが、出力になります[*3]。

＊2　ベクトルについては「2.2 行列の基本をおさらい」で説明します。

＊3　2次元のベクトルに左から 2×2 の行列をかけると、2次元のベクトルになります。そのため、出力も2次元になります。

そのため、ベクトル $\begin{pmatrix} a \\ b \end{pmatrix}$ が関数 f によって、ベクトル $U\begin{pmatrix} a \\ b \end{pmatrix}$ になるイメージです。

図 2.3：量子コンピュータの関数の組み合わせ



量子コンピュータは、ベクトルに行列を左から次々とかけることにより、計算を行います（**図 2.3**）。ベクトル $\begin{pmatrix} a \\ b \end{pmatrix}$ を関数 f に入力し、その出力をさらに関数 g に入力する場合、2×2 行列 V をかけて、最終的な計算結果はベクトル $VU\begin{pmatrix} a \\ b \end{pmatrix}$ になります。

古典コンピュータには、様々な種類の関数があります。一方、量子コンピュータは、どんなに複雑なアルゴリズムでも、行列のかけ算を繰り返しているだけです。行列の計算方法に慣れれば、怖がる必要はありません。

このイメージを持ちつつ、量子コンピュータに必要な数学を見ていきましょう。

2.2 行列の基本をおさらい

2.2.1 行列とベクトルの定義

縦と横に数を並べたものを**行列**（matrix）といいます。たとえば、次のようなものは行列です。

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 \end{pmatrix}, \quad \begin{pmatrix} 4 & 3 & 8 \\ 9 & 5 & 1 \\ 2 & 7 & 6 \end{pmatrix}$$

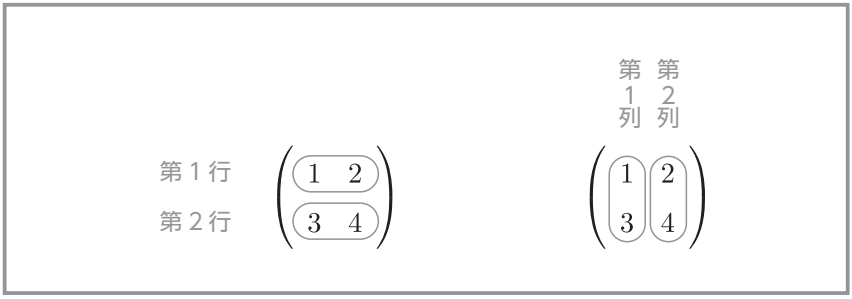
ここでは、

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

を例に、行列の用語について説明します。

まず、行列の横方向のかたまりを**行**（row）といい、上から下に向かって順に第1行、第2行…と数えます。また、行列の縦方向のかたまりを**列**（column）といい、左から右に向かって順に第1列、第2列…と数えます。さきほどの行列の場合、行と列は**図 2.4** のようになります。

図 2.4：行列の「行」と「列」



行列に並べられた数字を**成分**（entry）といい、「第○行第×列」や「(○, ×) 成分」といいます。この例では、

$$\begin{pmatrix} \text{第1行第1列} & \text{第1行第2列} \\ \text{第2行第1列} & \text{第2行第2列} \end{pmatrix}, \quad \begin{pmatrix} (1,1)\text{成分} & (1,2)\text{成分} \\ (2,1)\text{成分} & (2,2)\text{成分} \end{pmatrix}$$

となります。行列の**サイズ**（size）を表すとき、行数と列数を明示して「2 × 2行列」のようにいいます。たとえば、

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

は2 × 1行列です。

行と列の大きさが同じものを**正方行列**（square matrix）といいます。サイズが$n \times n$の場合、n次正方行列といいます。

行が1つの行列を**行ベクトル**（row vector）、列が1つの行列を**列ベクトル**（column vector）といいます。列ベクトルは単に**ベクトル**（vector）と呼ぶことが多いです。たとえば、

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \text{は行ベクトル}$$

$$\begin{pmatrix} 1 \\ 3 \end{pmatrix} \text{は列ベクトル}$$

です。ベクトルの成分の数がn個のとき、n次ベクトルといいます。

行列の各成分を添え字を使った変数で表すとき、添え字の1つ目の変数は行を表し、2つ目の変数は列を表すのが一般的です。

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

ベクトルの各成分を2乗したものの和に対して平方根を取った値を、**ベクトルの長さ**（length of vector）といいます。たとえば、ベクトル

$\begin{pmatrix} 3 \\ 4 \end{pmatrix}$ の長さは次のように計算できます。

$$\sqrt{3^2+4^2} = \sqrt{9+16} = \sqrt{25} = 5$$

ベクトルの次元が大きくなっても同様で、$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$ の長さは次のように計算できます。

$$\sqrt{1^2+2^2+3^2+4^2} = \sqrt{1+4+9+16} = \sqrt{30}$$

ベクトルのサイズ（n 次ベクトル）と長さは異なる概念ですので、混同しないように注意してください。

2.2.2　行列の和・差・積

行列の和・差

サイズが同じ行列の和（足し算）や差（引き算）を、同じ成分同士の計算として定義します。

定義

$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$ **のとき、** A **と** B **の和と差を次のように定義する。**

(1)（和） $A+B := \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{pmatrix}$

(2)（差） $A-B := \begin{pmatrix} a_{11}-b_{11} & a_{12}-b_{12} \\ a_{21}-b_{21} & a_{22}-b_{22} \end{pmatrix}$

第 3 章

量子コンピュータの基本ルール

$$|\varphi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{C}^2$$

3.1　この章で学ぶこと

第 1 章では数式を使わずに、量子コンピュータの概念やよくある誤解について説明しました。ここからは数式を使い、量子コンピュータの計算ルール（定義）について説明します。

ベクトル・行列・確率を使った数式が必要になりますが、手計算できる例を挙げますので、あまり構えずに進みましょう。量子コンピュータは、古典コンピュータとは異なる計算ルールで成り立っています。見慣れない概念が多く登場するとは思いますが、具体的に計算しながら少しずつ慣れましょう。数学的に分からない箇所があったときは、第 2 章の説明を読み返してみてください。

この章では、基本となる 1 量子ビットの計算ルールを説明します。特にユニタリ行列は、量子コンピュータの計算の基礎になる重要な概念です。1 度で理解できなかったとしても、手を動かし反復して理解を深めてください。

3.2　量子コンピュータの基礎「量子ビット」

まずは、古典ビットに対応する量子ビットや量子状態について説明します。第 1 章で説明した古典ビットと量子ビットの性質の違いを覚えているでしょうか。この章では両者の特徴について行列やベクトルを使ってより具体的に説明します。

古典ビットの 0、1 に対応する**量子ビット**（quantum bit、qubit）を $|0\rangle, |1\rangle$ と記し、次のベクトルとして定義します。

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

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

$|0\rangle$ はゼロベクトルでない点に注意してください。

$$|0\rangle \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

古典ビットの状態は 0 か 1 のどちらかでしたが、量子ビットの状態は $|0\rangle$ と $|1\rangle$ 以外もあり得ます。

定義

複素数 a, b が $|a|^2 + |b|^2 = 1$ という条件を満たすとき、

$$a|0\rangle + b|1\rangle$$

の形をしたベクトルを量子状態（quantum state）という。

$|a|$ や $|b|$ は複素数の絶対値を表します（詳しくは第 2 章を参照してください）。複素数が難しいと感じる方は、実数の絶対値（符号をプラスにしたもの）をイメージしてください。

この記法を利用することで、

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

と書けます。量子状態をベクトルの形で表したものを**状態ベクトル**（state vector）といいます。記法を変えただけですが、次の定義も同

じ量子状態を表しています。

定義

以下に定義する集合 Q の要素を量子状態（quantum state）と呼ぶ。

$$Q := \left\{ \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{C}^2 \;\middle|\; |a|^2 + |b|^2 = 1 \right\}$$

この定義は「複素数を成分とした 2 次元のベクトルで $|a|^2 + |b|^2 = 1$ を満たすもの」という意味です。

いくつか例を挙げると、次のものは量子状態です。

$$|0\rangle, \quad |1\rangle, \quad \frac{1}{\sqrt{5}}|0\rangle + \frac{2}{\sqrt{5}}|1\rangle, \quad -\frac{1}{\sqrt{3}}|0\rangle + \frac{1+i}{\sqrt{3}}|1\rangle$$

計算に慣れるため、実際に $|a|^2 + |b|^2 = 1$ を満たすことを確認してみてください。また、次のものは量子状態では**ありません**。

$$|0\rangle + |1\rangle \quad (|1|^2 + |1|^2 = 2 \neq 1 \text{ となるため})$$

量子状態は、内積を使って表すこともできます（内積については、第 2 章を参照してください）。量子状態

$$|\varphi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{C}^2$$

の内積 $\langle\varphi|\varphi\rangle$ を計算すると、

$$\langle\varphi|\varphi\rangle \;=\; \begin{pmatrix} a^* & b^* \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

$$= \quad a^*a + b^*b = |a|^2 + |b|^2$$

となります。これにより、量子状態全体集合をQとすると、次のように表せます。

$$Q = \left\{ |\varphi\rangle \in \mathbb{C}^2 \;\middle|\; \langle\varphi|\varphi\rangle = 1 \right\}$$

定義

複素数を成分とした2次元ベクトルが$\langle\varphi|\varphi\rangle = 1$という条件を満たすとき、$|\varphi\rangle$を量子状態（quantum state）という。

量子状態を表す方法はいくつかあるため、状況に応じて扱いやすい方法を使います。

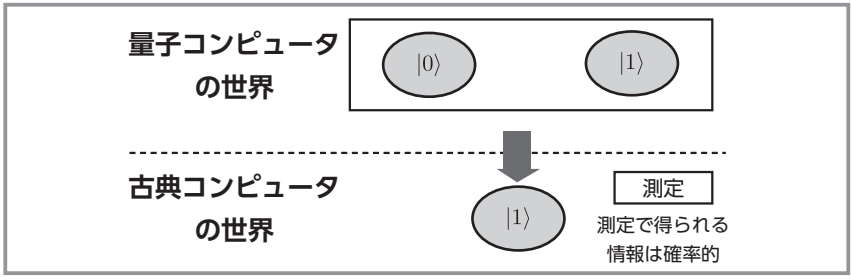
3.2.1 「重ね合わせ状態」とは？

$\frac{1}{\sqrt{5}}|0\rangle + \frac{2}{\sqrt{5}}|1\rangle$のように、係数が0でない複数の量子ビットの和で表される量子状態を**重ね合わせ状態**（superposition、superposition state）と呼びます。1量子ビットの量子状態は$|0\rangle$と$|1\rangle$という、2種類の量子ビットの重ね合わせ状態を表せます。古典ビットは重ね合わせ状態にできないため、古典ビットと量子ビットで大きく異なる点です。

3.3　計算結果を得る「測定」

計算結果を確認するために古典コンピュータが保持している古典ビットを読み取った場合、0 を読み取ると 0 が返り、1 を読み取ると 1 が返ります。そして、何回読み取っても同じ結果になります。なんだか、当たり前のことに感じますが、量子コンピュータではこれが当たり前ではありません。量子コンピュータは $\frac{1}{\sqrt{5}}|0\rangle + \frac{2}{\sqrt{5}}|1\rangle$ のように $|0\rangle$ でも $|1\rangle$ でもない状態を表せました。これを読み取ると何が返ってくるのでしょうか。実は、$|0\rangle$ が返ってくることもあれば、$|1\rangle$ が返ってくることもあります。同じ量子状態を読み取っても、何が返ってくるかは毎回異なり、実際に読み取るまで分かりません。ただし、訳が分からない状態ではなく、何が返ってくるか確率で表せます。この読み取りのことを**測定**（measurement）といいます。

図 3.1：測定のイメージ



測定のルールを具体的に定式化すると、次のようになります[*1]。

＊1　量子力学では測定値は実数です。そのため、正確には「$|0\rangle$」「$|1\rangle$」ではなく、「0」「1」が測定値になります。本書では量子ビットを測定していることを強調したい場合はケット記号の付いた「$|0\rangle$」や「$|1\rangle$」を測定値として表記します。そのため、「測定値 $|0\rangle$ を得る」と「測定値 0 を得る」は同じ意味です。

定義

量子状態を測定すると、$|0\rangle$ または $|1\rangle$ を得る。このことを「測定値 $|0\rangle$ を得る」とか単に「$|0\rangle$ を得る」という。
量子状態 $a|0\rangle + b|1\rangle$ を測定したとき、各測定値を得る確率は次の通り。

- **測定値 $|0\rangle$ を得る確率は $|a|^2$**
- **測定値 $|1\rangle$ を得る確率は $|b|^2$**

測定後の量子状態は、得られた測定値に変化する。

古典ビットは何度測定しても同じ結果が返ってきました。
次の点は、量子コンピュータと古典コンピュータで大きく異なります。

(1) 得られる測定値は確率によって変わる
(2) 測定によって状態が変化する

たとえば、

$$|\varphi\rangle = \frac{1}{\sqrt{5}}|0\rangle + \frac{2}{\sqrt{5}}|1\rangle$$

とし、$|\varphi\rangle$ を測定すると、次のようになります。

- 測定値 $|0\rangle$ を得る確率は $\left|\frac{1}{\sqrt{5}}\right|^2 = \frac{1}{5} = 0.2$ となり、測定後の量子状態は $|0\rangle$ に変化する。
- 測定値 $|1\rangle$ を得る確率は $\left|\frac{2}{\sqrt{5}}\right|^2 = \frac{4}{5} = 0.8$ となり、測定後の量子状態は $|1\rangle$ に変化する。

これらを踏まえ、古典ビットと量子ビットに関する測定の違いをまとめると、**表 3.1** のようになります。

表 3.1：古典ビットと量子ビットの測定の違い

比較対象	古典ビット	量子ビット
測定で得られる結果	同じ操作なら一定	同じ操作でも確率的に変わる
測定時の状態の変化	なし	あり

古典ビット 0 を測定して得られる値の確率分布は**図 3.2** のようになります。必ず 0 が得られるので、0 の確率は 1、その他の確率は 0 です。

量子状態 $|\varphi\rangle = \frac{1}{\sqrt{5}}|0\rangle + \frac{2}{\sqrt{5}}|1\rangle$ を測定して得られる値の確率分布は、**図 3.3** のようになります。

図 3.2：古典ビット 0 を測定して得られる値の確率分布

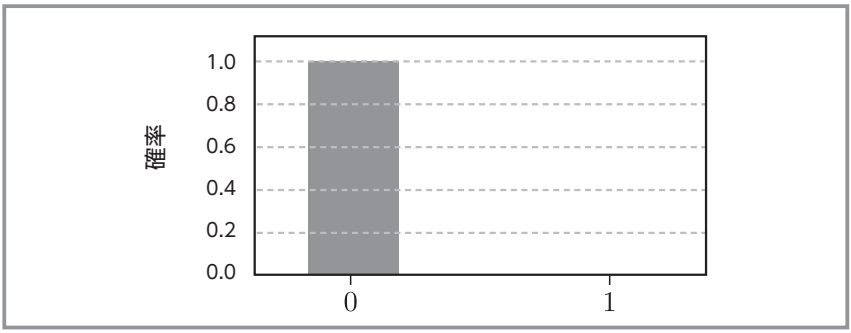
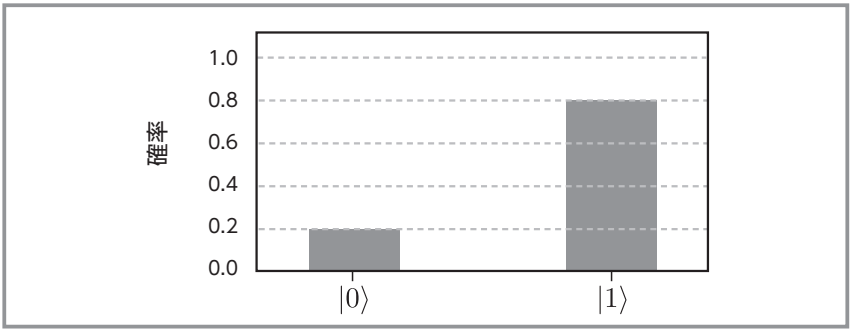


図 3.3：量子状態 $|\varphi\rangle = \frac{1}{\sqrt{5}}|0\rangle + \frac{2}{\sqrt{5}}|1\rangle$ を測定して得られる値の確率分布



一般的には、量子状態を測定するたびに異なる値が得られますが、$|0\rangle$ の場合は何度測定しても $|0\rangle$ が得られます。また、$|1\rangle$ を何度測定しても $|1\rangle$ が得られます。

このように、重ね合わせ状態になっていない $|0\rangle$ や $|1\rangle$ の測定は、古典ビットと同じように振る舞います。

次の $|\psi\rangle$ を測定すると、どうなるでしょうか。

$$|\psi\rangle = \frac{1}{\sqrt{5}}|0\rangle - \frac{2}{\sqrt{5}}|1\rangle$$

測定の定義にあてはめると、次のようになります。

- 測定値 $|0\rangle$ を得る確率は $\left|\frac{1}{\sqrt{5}}\right|^2 = \frac{1}{5} = 0.2$ となり、測定後の量子状態は $|0\rangle$ に変化する。
- 測定値 $|1\rangle$ を得る確率は $\left|-\frac{2}{\sqrt{5}}\right|^2 = \frac{4}{5} = 0.8$ となり、測定後の量子状態は $|1\rangle$ に変化する。

先ほどの $|\varphi\rangle$ とこの $|\psi\rangle$ のように、「ベクトルとしては異なるが、測定したときの振舞いは一致する」というケースがある点は注意が必要です。

3.4　ビットの状態を変化させる「量子ゲート」

量子ビットの状態について説明し、それを測定できることも分かりました。しかし、これだけでは計算できません。古典コンピュータでは、ビット演算を行うゲート（AND、OR、XOR、NOT など）を使い、ビットの状態を変化させて計算します。古典コンピュータのゲートに相当す

るものが、量子ゲートです。この量子ゲートの操作にあたるものを、ユニタリ発展といいます。

定義

量子状態$|\varphi\rangle$は、2×2ユニタリ行列Uを使って$U|\varphi\rangle$という量子状態に写る。このような量子状態の変化を<u>ユニタリ発展</u>(unitary evolution)という。

第2章で説明したように、ユニタリ行列は$U^\dagger U = I$が成り立つ行列です。量子コンピュータは、ユニタリ発展を行う回路を実装することで量子ビットの状態を変化させ、計算できます。

ユニタリ発展に利用するユニタリ行列のサイズが指数関数的に巨大になっても、量子コンピュータの計算時間への影響は古典コンピュータほどは大きくなりません。そのため、量子コンピュータは指数関数的に巨大なユニタリ発展を高速に行えます。あらゆる行列計算を高速化できる訳ではなく、ユニタリ発展による高速なアルゴリズムが発見されている場合だけ高速化できます。

量子コンピュータでよく利用するユニタリ発展については、第4章や第6章、第8章で具体的な計算例と共に紹介します。

3.4.1 量子状態をユニタリ発展させたものは、量子状態になる

ユニタリ発展した$U|\varphi\rangle$が量子状態にならないと、この定義は意味がありません。$|\varphi\rangle$を量子状態とし、Uをユニタリ行列したとき、$U|\varphi\rangle$が量子状態になることを、確認してみましょう。

$$\begin{aligned}\langle U\varphi|U\varphi\rangle &= (U\varphi)^\dagger(U\varphi) = \varphi^\dagger U^\dagger U\varphi = \varphi^\dagger I\varphi = \varphi^\dagger\varphi \\ &= \langle\varphi|\varphi\rangle = 1\end{aligned}$$

となります。これは、内積による量子状態の定義（87ページ）を満た

します。そのため、$U|\varphi\rangle$ も量子状態です。これにより、ユニタリ行列 U を量子状態から量子状態への関数と見ることができます。

3.4.2　任意のユニタリ発展は、ハードウェアとして実装可能

任意のユニタリ発展を近似的に実装できることが、理論的に分かっています。詳しい説明は本書の範囲を越えますが、**ソロベイ－キタエフの定理**（Solovay–Kitaev theorem）として知られています。興味ある方は巻末の参考文献［8］［9］［10］を参照してください。

3.4.3　ユニタリ発展の可逆性

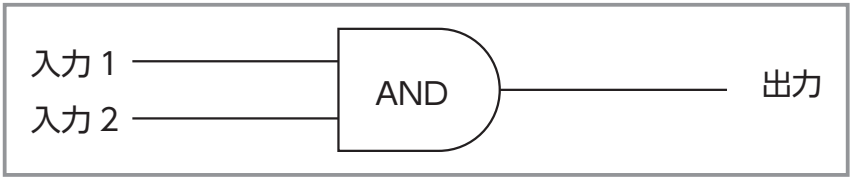
ユニタリ行列は必ず逆行列を持ちます。そのため、$U|\varphi\rangle$ を U^{-1} でユニタリ発展させると、

$$U^{-1}U|\varphi\rangle = |\varphi\rangle$$

となり、$|\varphi\rangle$ に戻ります。状態の変化に対して、元に戻す変化が可能であることを**可逆**（reversible）と言います。そのため、任意のユニタリ発展は可逆です。

量子回路の可逆性は、古典回路と大きく異なる点です。たとえば、**図 3.4** にある古典ビットの AND を考えてみましょう。

図 3.4：古典ビットの AND



AND は 2 個のビットを入力し、1 個の出力を得る操作です。すべての入力が 1 のときに出力が 1 になり、それ以外は出力が 0 になります。