量子数理教育

～数学的アプローチによる量子人材育成～

入門編

九州大学マス・フォア・インダストリ研究所

落合 啓之　北川 宜稔

内閣府総合科学技術・イノベーション会議の研究開発とSociety 5.0との橋渡しプログラム（BRIDGE）「量子人材教育エコシステムの開発と試行」
九州大学受託契約研究「量子数理教育」令和7年度 研究成果

BRIDGE

Institute of Mathematics for Industry
Kyushu University

九州大学
KYUSHU UNIVERSITY

量子数理教育
～数学的アプローチによる量子人材育成～
入門編

前書き

本書は、内閣府 BRIDGE 施策のうち、「量子人材教育エコシステムの開発と試行」(QST(国立研究開発法人量子科学技術研究開発機構) が研究推進法人) の中の、九州大学マス・フォア・インダストリ研究所が担当する、「量子数理教育〜数学的アプローチによる量子人材育成〜」 の初年度 (2025 年度) の成果物としての、量子数理教育向けテキストである。

近年、ゲート型量子コンピュータの社会実装の可能性が日に日に高まってきていると言われているが、そのアルゴリズムの作成は、通常、物理学としての量子力学に関する一定の知識を前提としている。このため、理工系の学部において体系的に量子力学を含む物理学の教育を受けた者以外にとっては、入口の部分で難しさを感じる可能性が高い。

他方で、良く知られているいくつかのゲート型量子コンピュータ固有のアルゴリズムには、数学の世界において有名な定理や特徴が用いられている。体系的に数学を学んだ者以外にとっては、この部分にも難しさを感じるかもしれない。また、体系的に数学を学んだ者にとっても、実は、大学における基礎的な数学教育ですでに学んだことがアルゴリズムの中に含まれていることに、後になって気付くかもしれない。

そこで、本テキストでは、物理学 (量子力学) 的なアプローチを最小限に留め、できるだけ数学的な観点から、アルゴリズム作成の考え方を整理することにより、物理学を十分理解できていない者でも、アルゴリズムについて考察できるような環境作りに配慮した。これは、古典コンピュータのアルゴリズムにおいても、コンピュータ内部の物理的動きを全く知らなくても、コンピュータ言語と基本的な論理さえ分かれば、プログラムが書けることに対応している。

また、現在、複数のゲート型量子コンピュータの方式間での開発競争が行われているが、いずれの方式が選ばれるにせよ、最終的には、ゲート型量子コンピュータ固有のアルゴリズムを作成していくことが重要であり、そのための数学的な素養を養っておくことが不可欠であることにも対応している。

九州大学マス・フォア・イノベーション卓越大学院プログラムでは、2024 年度後期より、量子数理教育に関連する講義を試験的に行っている。本講義では、数学的知識の量子コンピュータアルゴリズムへの応用を最初から意識し、良く知られているアルゴリズムから入り、それらにどのような数学的要素が活用されているかという順番で説明を行った。これは、数学系の授業でよく行われている、基礎から積み上げていく形式の講義構成とは全く異なるものであるが、必要十分

な数学的知識をより効率的に習得できるという点ではメリットがあると考えている。

本テキストは、これまでの講義の内容から Shor の素因数探索アルゴリズムに関連する部分を抽出し、入門編として再編したものである。講義では詳しく説明できなかった部分や演習問題について補足している。また、関連する数学的な知識も大幅に加筆している。講義ではいくつかのアルゴリズムを逍遥したが、この冊子では Shor のアルゴリズムに焦点を当てて、量子計算やアルゴリズムの特徴や、関連する量子以外の技術の数学的な側面をまとめることとした。Grover などのそのほかのアルゴリズムは、来年度以降の続編に譲る。

本テキストは、基本的には、大学の教養課程の数学教育 (線形代数、解析など) を履修した者を対象にしているが、より多くの若い世代に関心を持ってもらうため、ゲート型量子コンピュータに興味があれば、一定の準備さえすれば、意欲的な高校生にも理解できるよう配慮した。

国内外の各種調査によれば、早ければ 2030 年代半ばには、ゲート型量子コンピュータの本格的な利用が開始され、人口の約 1 割弱 (日本では約 1 千万人) の人達が、何等かの形で量子技術に係ることになると予測されている。

新しい時代は、もうすぐ目の前に迫っている。量子コンピュータが社会の発展を支え、それに係る研究者、技術者、エンジニアが大いに活躍する時代となることを願うばかりである。

2026 年 2 月

九州大学マス・フォア・インダストリ研究所 教授 落合啓之

本文の構成

本テキスト (入門編) の構成は以下のとおりである。

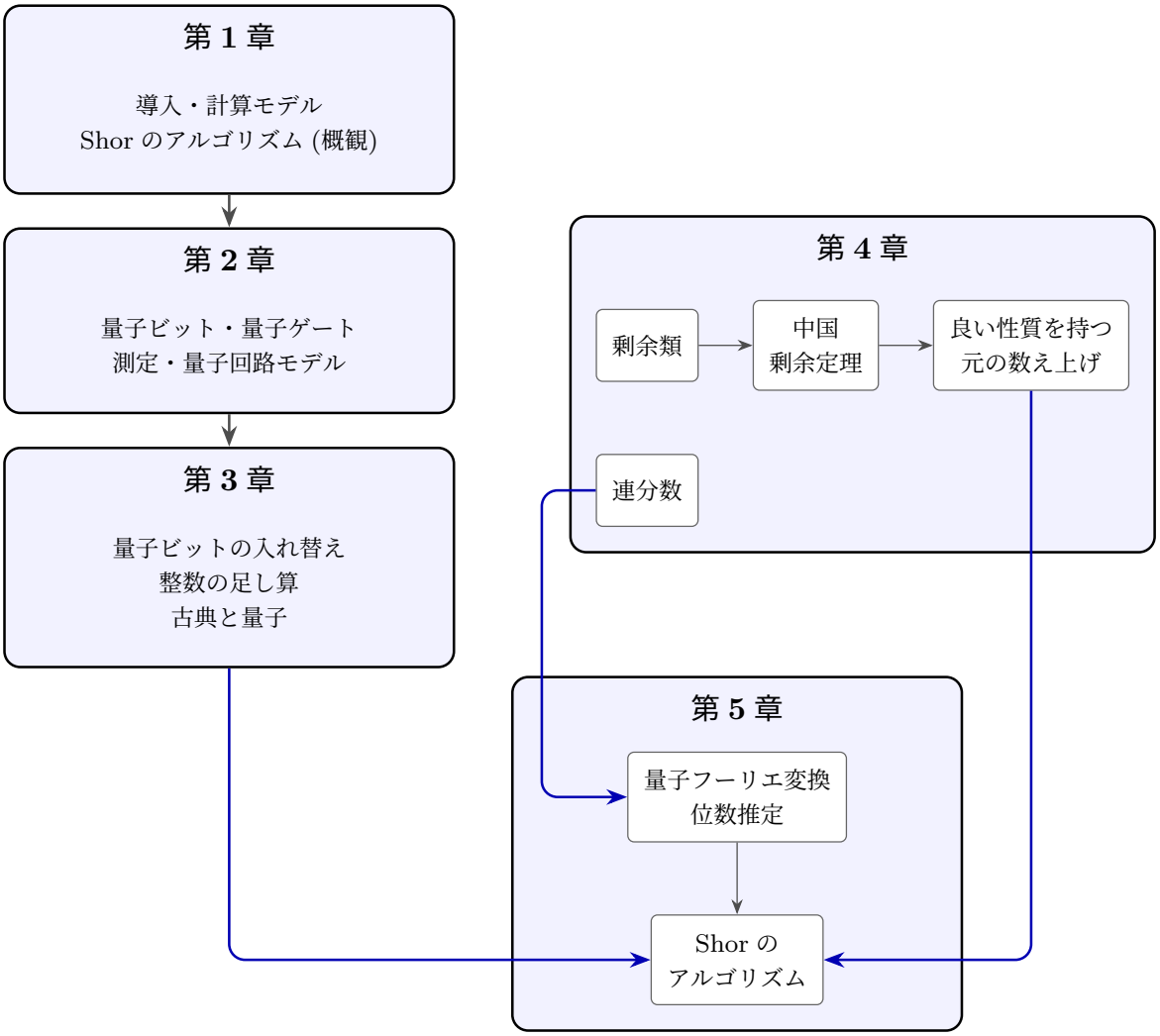
1 章では計算モデルについて概観し、Shor のアルゴリズムを具体例を交えながら解説する。ここでは多くの用語は無定義のまま進め、Shor のアルゴリズムを理解するというよりも、どのような道具が使われているかを眺めることを主題としている。

2 章では本テキストの目的の一つである量子回路モデルを導入する。量子ビット、量子ゲート、測定といった基本的な用語、および量子回路モデルにおける計算とは何かを定義する。量子回路モデルの定義に必要なベクトル空間やテンソル積などの線形代数の知識も、それらが必要になった時点で導入している。

その後の 3 章では、量子回路モデルでどのような計算ができるかについてまとめる。量子ビットの入れ替え、整数の足し算、量子ビットを増やす・減らす、古典コンピュータとの関係など、量子回路モデルにおける基本的な操作と初学者が疑問に思いやすい話題を扱っている。

4 章の内容は完全に数学の範疇であり、Shor のアルゴリズムと情報理論で必要な整数論の結果を扱っている。主な話題は $\mathbb{Z}/N\mathbb{Z}$ と連分数展開の性質であり、多くの整数論の専門書でも扱われている一般的な内容である。4.4 節の内容は Shor のアルゴリズム固有の話題であるが、中国剰余定理の応用でもあるのでこの章に配置している。

5 章では再び Shor のアルゴリズムに立ち返り、量子回路モデルでの実現方法を解説する。また、Shor のアルゴリズムで中心的な役割を果たす位数推定および量子フーリエ変換を扱う。


第 1 章
導入・計算モデル
Shor のアルゴリズム (概観)
第 2 章
量子ビット・量子ゲート
測定・量子回路モデル
第 3 章
量子ビットの入れ替え
整数の足し算
古典と量子
第 4 章
剰余類
中国
剰余定理
良い性質を持つ
元の数え上げ
連分数
第 5 章
量子フーリエ変換
位数推定
Shor の
アルゴリズム

各章の構成と依存関係

目次

前書き iii

本文の構成 v

謝辞 vii

記号表 xi

第 1 章 イントロダクション: Shor のアルゴリズム 1
1.1 アルゴリズムと計算モデル 1
1.2 Shor のアルゴリズム 4
1.3 $N = 21, q = 2^7, x = 2$ の場合 9

第 2 章 量子回路モデル 13
2.1 数ベクトルとベクトル空間 13
2.2 量子ビット 16
2.3 量子ビットの記法 21
2.4 測定 22
2.5 量子ゲート 23
2.5.1 量子ゲートと線形写像 24
2.5.2 X ゲート、Y ゲート、Z ゲート 26
2.5.3 Hadamard ゲート 27
2.5.4 CNOT ゲート 27
2.5.5 Toffoli ゲート 28
2.5.6 S,T ゲート、位相ゲート 29
2.5.7 テンソル積 30
2.6 量子回路モデルと量子回路図 33

第 3 章 量子回路モデルで何ができるか 37
3.1 簡単な操作 37
3.1.1 量子ビットの入れ替え 37
3.1.2 制御ゲートにする 38
3.1.3 制御ビットを増やす 41
3.1.4 足し算 44
3.2 量子もつれ状態 46
3.3 量子ビットを減らす、増やす 49
3.3.1 量子ビットを増やす 50
3.3.2 量子ビットを減らす 51
3.4 古典と量子 51
3.5 量子ビットはコピーできない 55

第 4 章 初等整数論 57
4.1 剰余と位数 57
4.2 中国剰余定理 66
4.3 N と互いに素な整数の割合 69
4.4 良い性質を持つ元の個数 72
4.5 連分数と Legendre の定理 77

第 5 章 Shor のアルゴリズム 87
5.1 アルゴリズムの再考 87
5.2 $f(j)$ の計算 90
5.3 フーリエ変換 92
5.3.1 指数関数とフーリエ変換 93
5.3.2 離散フーリエ変換 93
5.3.3 確率の評価 100
5.4 量子フーリエ変換 108
5.5 位数推定まとめ 114

参考文献 117

索引 118

記号表

数学全般

(記号) $\coloneqq$ (式)	(記号) を (式) で定義する。

集合

$\mathbb{Z}$	整数全体の集合。
$\mathbb{Q}$	有理数全体の集合。
$\mathbb{R}$	実数全体の集合。
$\mathbb{C}$	複素数全体の集合。
$\lvert X\rvert$	集合 X の元の個数。
$X \times Y$	集合 X, Y の直積集合。順序対 (x, y) $(x \in X, y \in Y)$ 全体の集合。

複素数

$\sqrt{-1}$	虚数単位。本テキストでは i は使わず $\sqrt{-1}$ で統一する。
$\exp(\sqrt{-1}x)$	指数関数 $e^{\sqrt{-1}x} = \cos(x) + \sqrt{-1}\sin(x)$。

線形代数

$\mathbb{C}^n$	複素数成分の n 次元数ベクトル全体の集合。
I_n, I	単位行列、または恒等写像。サイズ n が明らかなときは省略する。
$\det(X)$	正方行列 X の行列式。
X_{ij}	行列 X の (i, j) 成分。
${}^tX$	行列 X の転置。$({}^tX)_{ij} = X_{ji}$。
$\overline{X}$	複素数成分の行列 X の複素共役。$\overline{X}_{ij} = \overline{X_{ij}}$。
X^*	複素数成分の行列 X の随伴 $X^* = {}^t\overline{X}$。

整数論

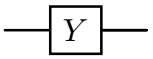
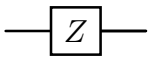
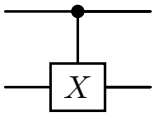
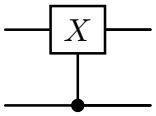
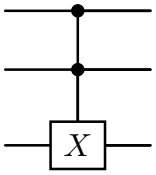
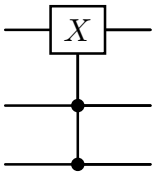
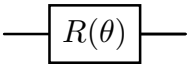
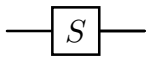
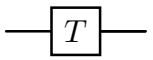
$\gcd(a, b)$	整数 a, b の最大公約数。

$\mathrm{lcm}(a, b)$	整数 a, b の最小公倍数。
$a \bmod N$	整数 a を正の整数 N で割った余り。$0 \leq (a \bmod N) < N$ を満たす。
$[a]_N, [a]$	N を法とする a の剰余類。
$\mathbb{Z}/N\mathbb{Z}$	N を法とする剰余類全体の集合。 $\mathbb{Z}/N\mathbb{Z} = \{[0]_N, [1]_N, \ldots, [N-1]_N\}$
$(\mathbb{Z}/N\mathbb{Z})^\times$	$\mathbb{Z}/N\mathbb{Z}$ の可逆元全体の集合。
$\phi(n)$	n と互いに素な 1 以上 n 以下の整数の個数。Euler のトーシェント関数。
$\lfloor x \rfloor$	x 以下の最大の整数。$\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$ を満たす。

量子情報理論

$\{0,1\}^n$	n 個の $\{0,1\}$ の直積集合。長さ n の $0,1$ の列 (n ビット) 全体の集合とみなす。
$\|0\rangle, \|1\rangle, a\|0\rangle + b\|1\rangle$	1 量子ビットの状態。
$\|v\rangle, \|w\rangle$ など	量子ビットの状態を表す記号。$(\mathbb{C}^2)^{\otimes n}$ の元。
$a \oplus b$	$0 \oplus 0 = 1 \oplus 1 = 0, 0 \oplus 1 = 1 \oplus 0 = 1$ を満たす和。排他的論理和 (XOR)、あるいは mod 2 での和。

量子ゲート

名前	数式中の記号と作用	回路図での記法
X ゲート	$X\left\|0\right\rangle = \left\|1\right\rangle, X\left\|1\right\rangle = \left\|0\right\rangle$	
Y ゲート	$Y\left\|0\right\rangle = \sqrt{-1}\left\|1\right\rangle, Y\left\|1\right\rangle = -\sqrt{-1}\left\|0\right\rangle$	
Z ゲート	$Z\left\|0\right\rangle = \left\|0\right\rangle, Z\left\|1\right\rangle = -\left\|1\right\rangle$	
Hadamard ゲート	$H\left\|0\right\rangle = \frac{1}{\sqrt{2}}(\left\|0\right\rangle + \left\|1\right\rangle), H\left\|1\right\rangle = \frac{1}{\sqrt{2}}(\left\|0\right\rangle - \left\|1\right\rangle)$	
CNOT ゲート	$CX_{1\to 2}\left\|j_1 j_2\right\rangle = \left\|j_1, j_1 \oplus j_2\right\rangle$ (j_1 が 1 の場合のみ j_2 を反転)	
	$CX_{2\to 1}$ (量子ビットの 2 番目が制御ビットで 1 番目が標的ビットの場合)	
Toffoli ゲート	$CCX_{1,2\to 3}\left\|j_1 j_2 j_3\right\rangle = \left\|j_1 j_2, (j_1 \cdot j_2) \oplus j_3\right\rangle$ ($j_1 = j_2 = 1$ の場合のみ j_3 を反転)	
	$CCX_{2,3\to 1}$ (量子ビットの 2, 3 番目が制御ビットで 1 番目が標的ビットの場合)	
位相ゲート	$R(\theta)\left\|0\right\rangle = \left\|0\right\rangle, R(\theta)\left\|1\right\rangle = e^{\sqrt{-1}\theta}\left\|1\right\rangle$	
S ゲート	$S\left\|0\right\rangle = \left\|0\right\rangle, S\left\|1\right\rangle = \sqrt{-1}\left\|1\right\rangle$ $S = R(\pi/2)$	
T ゲート	$T\left\|0\right\rangle = \left\|0\right\rangle, T\left\|1\right\rangle = e^{\sqrt{-1}\pi/4}\left\|1\right\rangle$ $T = R(\pi/4)$	

第 1 章

イントロダクション: Shor のアルゴリズム

この章では、量子アルゴリズムの導入として Shor のアルゴリズムについて紹介します。あまり多くのことを導入せずに大雑把に扱いつつ、量子アルゴリズムがベクトルと行列の操作によって記述されることをみます。入門編の目標である Shor のアルゴリズムの理解へ向けて、何が必要になるのかを概観していきましょう。

1.1 アルゴリズムと計算モデル

現在様々な量子コンピュータの実現が考案されていますが、どれも複雑な物理現象を利用しています。一方、量子コンピュータで利用される量子計算や量子アルゴリズムは、現実の量子コンピュータの仕組みや物理現象とは独立に議論できるもので、数学的には行列とベクトルの演算によって記述できます。これはちょうど、古典的なコンピュータ (いわゆる普通のコンピュータ) の仕組みを知らなくても、ユークリッドの互除法やガウスの掃き出し法といったアルゴリズムを論じることができるのと同じです。本テキストでは、現実の量子コンピュータがどのような理論の下で動いているかについては一切触れず、量子計算理論、特に量子アルゴリズムと量子誤り訂正について数学的な視点から解説します。

本テキストの主題の一つである**アルゴリズム**というのは、決められた操作とルールの下で問題を解く手順を記述したもののことを指します。数学における「解法」も一種のアルゴリズムといえ、例えば筆算の手順や鶴亀算、あるいは最大公約数を求めるユークリッドの互除法、連立一次方程式を解くガウスの掃き出し法はアルゴリズムの例になります。掛け算の筆算の場合、許される操作は足し算と一桁同士の掛け算と思えます。筆算は桁数が大きくなるほど計算が大変になりますが、仮に任意の桁数の掛け算が一瞬で計算できる人がいるなら、その人にとって掛け算は常

に 1 回の操作で計算できる問題といえるでしょう。このように、どのような操作を許すかを変えると同じ問題でも効率的に解けるかどうかが変わります。古典計算と量子計算による問題を解く効率の違いは、このルールの違いにあります。

古典・量子にかかわらず計算理論では、基本的な操作を積み上げて複雑なアルゴリズムを実現します。例えば古典的なコンピュータでは、0, 1 の列に AND, NOT を組み合わせた操作を繰り返すことで、足し算や掛け算、あるいはより複雑なアルゴリズムを実現していきます。許される基本的な操作とルールを数学的に記述したものを、**計算モデル**といいます。例えば、古典的な計算モデルとして、チューリングマシンやラムダ計算がよく知られています。

計算モデルを考えるメリットの一つとして、アルゴリズムを数学的に扱い、その効率を理論的に評価できることが挙げられます。アルゴリズムを用いて問題を解く際に基本的な操作を何回用いるかを数えることで、問題を解くまでの計算時間などを見積もることができるのです。各計算モデルにおいてどのような問題が効率的に解けるかは計算量理論における重要な問題の一つで、未解決問題も多く残されています。有名な $P = NP$ 問題は、このような未解決問題の一つです。

本テキストでは量子計算を扱いますが、量子ゲートを用いた**量子回路モデル**と呼ばれる計算モデルのみを用います。この計算モデルは、概ね以下のように記述されます。ここで出てくる線形代数の言葉については 2 章で改めて復習します。

簡易的な量子回路モデル

(1) **ゲート操作**: 特別なベクトル ${}^t(1, 0, 0, \ldots, 0)$ から始めて、**量子ゲート**と呼ばれる数種類の簡単なユニタリ行列 (やそれらのテンソル積) を掛けていく。

(2) **測定**: (1) で得られたベクトル $\boldsymbol{a} = {}^t(a_1, a_2, \ldots, a_n) \in \mathbb{C}^n$ に対して、$1, 2, \ldots, n$ を以下の確率で出力する

$$k \in \{1, 2, \ldots, n\} \text{ の出る確率は} \frac{|a_k|^2}{|a_1|^2 + |a_2|^2 + \cdots + |a_n|^2}.$$

測定で k が出た場合、ベクトルは ${}^t(0, 0, \ldots, 0, \overset{k}{1}, 0, \ldots, 0)$ (k 番目だけが 1 のベクトル) に変化する。

量子回路モデルの厳密な定義は 2 章で行います。ここでは大雑把に、行列を掛ける操作とベクトルから数値を得る操作と考えてもらってかまいません。(2) の測定が特に量子計算における特徴的な操作になります。

例 1.1.1. $\boldsymbol{a} = \frac{1}{\sqrt{6}}{}^t(1, -1, 2, 0)$ の場合、測定の出力が $k = 1, 2, 3, 4$ となる確率はそれぞれ表 1.1 のようになる。

k	1	2	3	4
確率	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{2}{3}$	0

表 1.1: 確率分布

例 1.1.2. X ゲートと呼ばれる量子ゲートは

$$\begin{pmatrix} a \\ b \end{pmatrix} \overset{\text{X ゲート}}{\longmapsto} \begin{pmatrix} b \\ a \end{pmatrix}$$

という第 1 成分と第 2 成分を入れ替える操作で、$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ という行列で与えられる。例えば、以下のように $\frac{1}{\sqrt{5}}\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ というベクトルに X ゲートを掛けると、測定時の 1 と 2 の出力される確率が入れ替わる。

$$\frac{1}{\sqrt{5}}\begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad \overset{\text{X ゲート}}{\longmapsto} \quad \frac{1}{\sqrt{5}}\begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

出力	確率
1	1/5
2	4/5

出力	確率
1	4/5
2	1/5

これらの例から分かるように、最終的なベクトル $\boldsymbol{a}$ によって測定でどんな数値が出力されやすいかが変わります。また、(1) のゲート操作で量子ゲートの組み合わせを変えると、$\boldsymbol{a}$ は変化し測定でどの数値が出やすいかという確率も変動します。もし、量子ゲートを適切に選んで問題の答えや答えにつながる数値が出やすいようにできれば、それを利用して問題を解くことができる、というのが基本的なアイデアです。

問題を解く手順　問題が与えられる。

(a) 与えられた問題に応じてゲート操作で掛けていく量子ゲートの列を決める。
(b) (a) で作った行列の列を ${}^t(1, 0, 0, \ldots, 0)$ に掛けていく。
(c) 最終的に得られたベクトルを測定して出力された数値から答えを計算する。
(d) 最後のステップは確率的なので、うまくいかない場合は (b)(あるいは (a)) に戻って繰り返す。

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \overset{\substack{\text{ゲート操作} \\ \text{行列 } U_m \ldots U_2 U_1 \text{ を掛ける}}}{\longmapsto} \boldsymbol{a} \overset{\text{測定}}{\longmapsto} 1, 2 \ldots, n \text{ のどれか } (\boldsymbol{a} \text{ で決まる確率で出力})$$

$\longmapsto$ 答えを得る、またはやり直し

注意することとして、私たちは最終的なベクトル $\boldsymbol{a}$ を直接知ることはできません[*1]。知ることができるのは、$\boldsymbol{a}$ を測定して得られる数値のみです。$\boldsymbol{a}$ 自体に問題の答えに関する情報が含まれていたとしても、測定によってそれが取り出せるかは別問題です。

例 1.1.3. 問題が与えられ、量子ゲートの列を以下の性質が成り立つように選べたとする。

最終的に得られたベクトル $\boldsymbol{a}$ の負の成分の場所が問題の答えになる。

最終的に $\boldsymbol{a} = \frac{1}{\sqrt{5}}{}^t(1,1,1,-1,1)$ というベクトルが得られた場合、4 が答えのはずだが、測定すると $1,2,3,4,5$ が同じ確率で出力される。$\boldsymbol{a}$ には答えの情報があるにもかかわらず、測定すると一切の情報が失われてしまっている。

1.2　Shor のアルゴリズム

ここからしばらくは Shor のアルゴリズムと呼ばれるアルゴリズムを題材に、量子アルゴリズムへの理解を深めていきます。**Shor(ショア) のアルゴリズム**は 1994 年に Shor により考案された、合成数の素因数分解を求めるための量子アルゴリズムです。

問題 1 (素因数分解)**.** 合成数の非自明な約数を一つ出力せよ。

入力　合成数 N

出力　1 と N 以外の N の約数

合成数とは素数でも 1 でもない正の整数のことでした。別の言い方をすると、合成数とは 1 より大きい二つの整数の積で表されるような整数のことです。例えば $N = 15$ の場合なら、3 または 5 が出力すべき答えになります。以降では、$1, N$ 以外の N の約数のことを非自明な約数と呼ぶことにします。

通常、素因数分解という言葉は N を素数の積で表すことを指しますが、問題 1 ではそうなっていません。しかし、N の非自明な約数 M が見つかれば、$N = M \cdot (N/M)$ とより小さい自然数の積に分解することができ、$M, N/M$ に対して繰り返し問題 1 を解くことで N の素因数分解が求まります。古典的なコンピュータでも、自然数の割り算、掛け算、素数かどうかの判定は比較的高速に行えるので、N/M を計算したり、素因数分解が完了したかの判定はそちらに任せることができます。したがって、問題 1 を高速に解くことが主題となります。

[*1] 古典的なコンピュータで行列とベクトルの積を実際に計算して求めることはできますが、たいてい膨大な時間がかかります。また、それができる問題なら量子コンピュータを使う意味がありません。

■単純なアルゴリズム N の約数を見つける最も簡単な方法は、2 から $N-1$ までの各自然数で N を割り切れるか試すというものです。この方法ではおよそ N 回の割り算が必要になります。さらに考えると、$N = M \cdot N/M$ の場合、$M \leq \sqrt{N}$ または $N/M \leq \sqrt{N}$ の少なくとも一方が成り立つので、2 から $\sqrt{N}$ までの自然数で割り切れるか試せばよいことが分かります。この方法だとおよそ $\sqrt{N}$ 回の割り算で十分になります。

RSA 暗号では数百桁という巨大な合成数が用いられており、素因数分解されると暗号が解読されてしまいます。ここで述べた $\sqrt{N}$ 回割り算を行うアルゴリズムでは、このような巨大な自然数を素因数分解するには途方もない時間がかかります。より高速なアルゴリズムも知られていますが、それでも古典的なコンピュータでは巨大な自然数を素因数分解するには多くの時間がかかるというのが現状です。

■Shor のアルゴリズム Shor のアルゴリズムを用いると素因数分解を比較的少ないゲート数で解くことができます。アルゴリズムの手順を順にみていきましょう。ここでは具体的な量子ゲートには触れず、それらを掛けて得られる行列やベクトルだけに注目します。具体例として $N = 15$ の場合に各ステップがどうなるかを計算しています。

手順 1.

- 自然数 $x(1 < x < N)$ を選ぶ。
- $\gcd(x, N) \neq 1$ ならそれが非自明な約数を与えるので終了。
- 自然数 q を、2 のべき乗で N^2 より大きくなるように取る。

運よく $\gcd(x, N) \neq 1$ となると即座に非自明な約数を見つけることができます。以下の手順では x が N と互いに素な場合を処理していくことになります。

$N = 15$ の例の場合に、$x = 2$, $q = 2^5 = 32$ を取っておきましょう。$q > N^2$ ではないですが、具体的な計算を書くために小さく取っています。

手順 2.

- 写像 f: 非負整数の集合 $\to \{0, 1, \ldots, N-1\}$ を $f(j) = x^j \bmod N$ で定める。ただし、$a \bmod N$ は整数 a を N で割った余りを意味する。

$N = 15, x = 2$ の場合の f の値は表 1.2 のようになります。sin 関数のように周期的になっている様子が分かります。

手順 3.

- 初期ベクトルにゲート操作を行って、$A = \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} E_{j,f(j)}$ という q 行 N 列の行列を表すベクトルを作る。ただし、行列単位 $E_{i,j}$ は (i, j) 成分が 1 でそれ以外が 0 の行列である。

j	0	1	2	3	4	5	6	7	8	9	$\cdots$
x^j	1	2	4	8	16	32	64	128	256	512	$\cdots$
$x^j \bmod N$	1	2	4	8	1	2	4	8	1	2	$\cdots$

表 1.2: $N = 15, x = 2$ の場合の $f(j)$ の値

ここでは

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

のようにベクトルの成分を長方形に並び替えることで行列とみなすと考えてください。あとでベクトルの測定を行いますが、そのときも行列に合わせて

$$1 \leftrightarrow (0,0), \quad 2 \leftrightarrow (0,1), \quad 3 \leftrightarrow (1,0), \quad 4 \leftrightarrow (1,1)$$

のように測定結果の数値 (今の場合は $1, 2, 3, 4$) を座標に変換して計算を行います。計算の都合で、行列の左上が $(0,0)$、右下が $(1,1)$ になるように座標を 1 ずらしています。

$N = 15, x = 2, q = 32$ の場合、行列 $\sqrt{q}A$ は (1.1) のようになります。各行 (横方向) には 1 がただ一つあり、各列 (縦方向) は周期的に 1 と 0 が並んでいます。1 がある列の周期の長さの 4 は $x^4 - 1$ が N で割り切れるという性質を反映しています。

ちなみに A を測定すると、ペア $(j, f(j))$ がランダムに出力されますが、$f(j)$ 自体は古典的なコンピュータでも容易に計算でき、約数とも直接結びつかないためあまり意味はありません。A に行列を掛けてうまく周期を抽出することが次のステップの目標になります。

$$\sqrt{q}A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & \cdots \end{array} \\ \left(\begin{array}{cccccccccccc} 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & \\ 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & \\ 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & \\ 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & \underline{\mathbf{1}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ & & & & & \vdots & & & & & & & \end{array}\right) \end{array} \tag{1.1}$$

手順 4. (量子フーリエ変換)

- q 次正方行列 $B = (b_{k,j})_{0 \le k,j < q}$ を

$$b_{k,j} = \exp\left(-\frac{2\pi\sqrt{-1}}{q} jk\right)$$

  により定義[*2]し、B を複数の量子ゲートを使って表す。
- ゲート操作を行い最終的なベクトル[*3] BA を得る。

$\exp(x) = e^x$ ですが、x に純虚数を入れたものについては定義 2.5.4 で復習します。ここで作った行列 B は以下の性質を満たし、$\frac{1}{\sqrt{q}}B$ はユニタリ行列と呼ばれる種類の行列になります。

$$B\overline{B} = q \cdot I_q, \quad B = {}^tB.$$

ただし、I_q は $q \times q$ の単位行列です。厳密には $\frac{1}{\sqrt{q}}B$ の方を量子ゲートを使って作るのですが、ここでは定数倍は無視することにします。測定結果にも影響はありません。

B はある種のフーリエ変換を表す行列で、B を掛けることによって行列 A の列の周期を検出することができます。

手順 5. (測定)

- BA を測定すると、ある成分の位置 (k, l) が出力される。

手順 3 で述べたように、最終的なベクトルを行列とみなしているので、出力される数字を座標 (k, l) と解釈しています。重要なポイントは、周期の情報を持っているような k の値が出やすく、そうでない k は出にくいということです。

$N = 15, x = 2, q = 32$ の場合、$\sqrt{q}BA$ は以下のようになります。0 の成分は省略して空白にしています。この行列を測定すると、$k = 0, 8, 16, 24$, $l = 1, 2, 4, 8$ の組み合わせ 16 通りのどれかが等確率で得られます。

$$\sqrt{q}BA = \begin{array}{c} \\ 0 \\ 8 \\ 16 \\ 24 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 4 & 8 \end{array} \\ \begin{pmatrix} 8 & 8 & 8 & 8 \\ 8 & -8\sqrt{-1} & -8 & 8\sqrt{-1} \\ 8 & -8 & 8 & -8 \\ 8 & 8\sqrt{-1} & -8 & -8\sqrt{-1} \end{pmatrix} \end{array}$$

[*2] 5.3.2 節の記号に合わせて k を行、j を列にしています。

[*3] 今の場合は行列と思っています。

手順 6. (連分数)

- 連分数展開を使い、$\frac{k}{q}$ をよく近似するような既約分数 $\frac{k'}{r}$ で、$r < N$ かつ r が最大になるものを見つける。

高い確率で、この r が手順 3 で述べた A の列の周期を与え、$x^r \bmod N = 1$ となります。良い状況では $\frac{k}{q}$ を約分をするだけで分母が N 未満の既約分数が得られますが、そうでない場合には近似を行う必要があります。

例えば、$N = 21, q = 2^7 = 128, k = 21$ の場合を考えると、

$$\frac{21}{128} = \frac{1}{6} - \frac{1}{384} \fallingdotseq \frac{1}{6} = \frac{k'}{r}$$

となり $r = 6$ を得ます。このような良い近似値は

$$\frac{128}{21} = 6 + \frac{2}{21}, \quad \frac{21}{2} = 10 + \frac{1}{2},$$
$$\frac{21}{128} = \cfrac{1}{6 + \cfrac{1}{10 + \cfrac{1}{2}}} \fallingdotseq \frac{10}{61} \fallingdotseq \frac{1}{6}$$

という風に連分数を途中で打ち切ることで求められます。連分数の打ち切り方は複数ありますが、$21/128, 10/61$ は分母が N 以上なので無視します。連分数の性質については 4.5 節で詳しく扱います。

$N = 15, x = 2, q = 32$ の場合、$k = 0, 8, 16, 24$ なので対応する $\frac{k}{q}$ と r は表 1.3 のようになります。この場合は近似を考えず、単に約分するだけで分母を N 未満にできます。

k	0	8	16	24
$\frac{k}{q}$	$\frac{0}{1}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{3}{4}$
r	1	4	2	4

表 1.3: $N = 15, x = 2, q = 32$ の場合

手順 7.

- $x^r - 1$ が N で割り切れない場合は、周期が正しく求まらなかったので手順 3 に戻る。
- r が奇数のときはうまくいかない x なので、手順 1 に戻り x を取り直す。
- r が偶数のときは、高い確率で $\gcd(x^{r/2} - 1, N)$ と $\gcd(x^{r/2} + 1, N)$ が N の非自明な約数を与える。そうでないなら、うまくいかない x なので手順 1 に戻り x を取り直す。ここで、$\gcd(a, b)$ は a と b の最大公約数を意味する。

手順 6 で得られた r が $f(j)$ の周期なら $x^r - 1 = (x^{r/2} - 1)(x^{r/2} + 1)$ が N で割り切れる、ということを利用して N の約数を求めています。x によっては r が奇数であったり $x^{r/2} + 1$ が N で割り切れてしまい、非自明な約数が得られないこともあります。一方で、定理 4.4.8 で示すように、半数以上の x はうまく約数が求まるものになっているので、高確率でこのステップは成功します。

$N = 15, x = 2, q = 32$ の場合を考えましょう。手順 6 で得られた r は $1, 2, 4$ のどれかで、$x^r - 1$ が N で割り切れるものは 4 のみです。$r = 1, 2$ が出てきた場合は、手順 3 からやり直して正しい周期を求め直します。$r = 4$ の場合は、

$$x^{r/2} - 1 = 3, \quad x^{r/2} + 1 = 5$$

となり、gcd を取るまでもなく約数の $3, 5$ が得られています。

手順 6,7 は古典的なコンピュータでも高速に計算できるため、手順 5 までが量子コンピュータが必要になる部分です。N が 2^{2024} のように大きな場合、行列 A, B は非常に巨大なものになり、古典的なコンピュータのメモリに載せることすら不可能です。また、行列 A を作るには $f(j)$ を $0 \leq j < q$ の範囲すべてで計算する必要がありますが、これも古典的なコンピュータでは膨大な計算になります。行列 B を掛ける操作は高速フーリエ変換という手法で高速に計算できることが知られていますが、それでも古典的なコンピュータでは $q \log(q)$ 回程度の演算が必要になります。量子ビットや重ね合わせの状態を使うことで、このような計算に必要なビット数やゲート操作の回数を大幅に減らせる、というのが古典と量子の大きな違いになります。

ここまで Shor のアルゴリズムの全体像を述べてきました。しかし、多くの言葉を定義なしに使ってきましたし、行列 A, B を量子ゲートでどのように作るか、周期の性質など触れなかったことも多くあります。この先の章では、Shor のアルゴリズムを題材に数学的な用語と量子回路モデルを導入し、アルゴリズムの理解を目指します。どのようなことが必要になるかをまとめておきましょう。

- 素数や合成数の性質と $x^j \bmod N$ の周期
- ベクトルや行列の性質
- 量子回路モデル
- (量子) フーリエ変換

1.3　$N = 21, q = 2^7, x = 2$ の場合

前節の $N = 15, q = 2^5, x = 2$ の例では測定の確率分布が分かりやすかったり、手順 6 で近似しなくても周期が求まるという良い状況になっていました。これは周期が q を割り切るという特別な

性質[*4]を持つような N だったためで、一般にはもう少し複雑になります。$N = 21, q = 2^7, x = 2$ の場合にどうなるかをまとめておきます。

手順 2 で考えた $f(j) = x^j \bmod N$ の表をこの場合にも書いておきましょう。

j	0	1	2	3	4	5	6	7	8	9	10	$\cdots$
x^j	1	2	4	8	16	32	64	128	256	512	1024	$\cdots$
$x^j \bmod N$	1	2	4	8	16	11	1	2	4	8	16	$\cdots$

表 1.4: $N = 21, x = 2$ の場合の $f(j)$ の値

この場合もやはり f は周期的になっており、その周期は 6 です。手順 3 で作る行列 A も $N = 15$ の場合と似たような形になり、各列はすべて 0 であるか周期が 6 になるように 0 と 1 が並びます。行列が複雑なため BA の具体的な形は省略しますが、手順 5 で BA を測定すると図 1.1 の確率分布で k が出力されます。表 1.5 は確率が高い k だけを抜き出したものです。

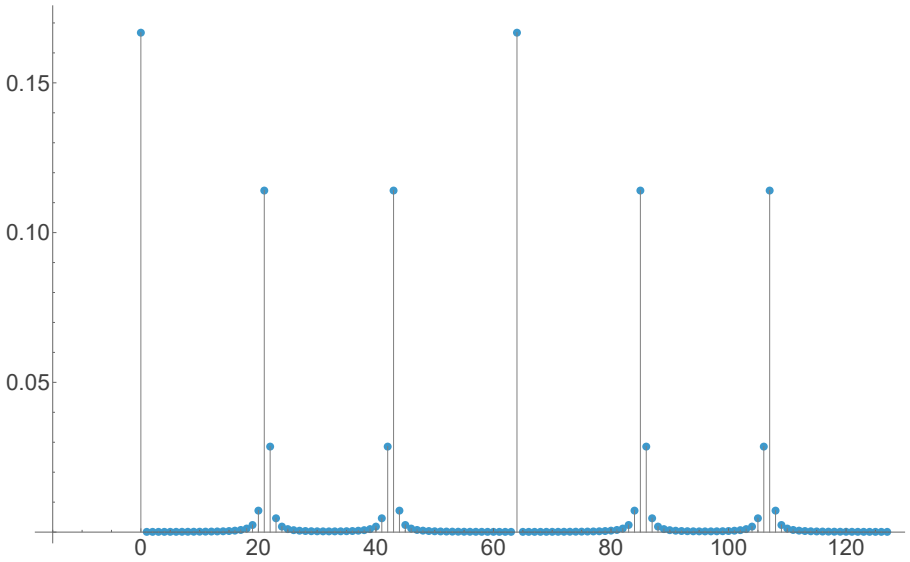


図 1.1: $N = 21, x = 2, q = 128$ の場合の確率分布

[*4] $15 = 3 \cdot 5$ はフェルマー素数と呼ばれる特別な素数の積になっています。

現れる確率	k
16.7%	0, 64
11.4%	21, 43, 85, 107
3.0%	22, 42, 86, 106
⋮	⋮

表 1.5: 測定結果として現れる k とその確率。例えば $k = 64$ はおよそ 16.7% で現れる

表にある各 k に対して手順 6 を実行して r を求めます。$k = 0$ の場合は $r = 1$ となり、これは周期ではないので手順 3 からやり直しです。$k = 64$ の場合は $\frac{k}{q} = \frac{1}{2} = \frac{k'}{r}$ となり $r = 2$ を得ますが、これも周期ではありません。$k = 21$ の場合、

$$\frac{k}{q} = \frac{21}{128} = \frac{1}{6 + \dfrac{1}{10 + \dfrac{1}{2}}} \fallingdotseq \frac{10}{61} \fallingdotseq \frac{1}{6}$$

となり、分母が N 未満になるような近似が欲しいので、$\frac{10}{61}$ は除外します。よって、この場合は $r = 6$ となります。同様に

$$k = 22 \rightsquigarrow \frac{k}{q} = \frac{22}{128} = \frac{11}{64} = \frac{1}{5 + \frac{1}{1 + \frac{1}{4 + \frac{1}{2}}}} \fallingdotseq \frac{1}{6} \quad \rightsquigarrow r = 6$$

$$k = 43 \rightsquigarrow \frac{k}{q} = \frac{43}{128} = \frac{1}{2 + \frac{1}{1 + \frac{1}{42}}} \fallingdotseq \frac{1}{3} \quad \rightsquigarrow r = 3$$

$$k = 42 \rightsquigarrow \frac{k}{q} = \frac{42}{128} = \frac{21}{64} = \frac{1}{3 + \frac{1}{21}} \fallingdotseq \frac{1}{3} \quad \rightsquigarrow r = 3$$

$$k = 85 \rightsquigarrow \frac{k}{q} = \frac{85}{128} = \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{42}}}} \fallingdotseq \frac{2}{3} \quad \rightsquigarrow r = 3$$

$$k = 86 \rightsquigarrow \frac{k}{q} = \frac{86}{128} = \frac{43}{64} = \frac{1}{1 + \frac{1}{2 + \frac{1}{21}}} \fallingdotseq \frac{2}{3} \quad \rightsquigarrow r = 3$$

$$k = 107 \rightsquigarrow \frac{k}{q} = \frac{107}{128} = \frac{1}{1 + \frac{1}{5 + \frac{1}{10 + \frac{1}{2}}}} \fallingdotseq \frac{5}{6} \quad \rightsquigarrow r = 6$$

$$k = 106 \rightsquigarrow \frac{k}{q} = \frac{106}{128} = \frac{53}{64} = \frac{1}{1 + \frac{1}{4 + \frac{1}{1 + \frac{1}{4 + \frac{1}{2}}}}} \fallingdotseq \frac{5}{6} \rightsquigarrow r = 6$$

となります。連分数を途中で打ち切る方法は何通りかありますが、分母が N 未満で最大になるものを選びます。ここでチェックした k/q たちが、$1/6, 2/6, 3/6, 4/6, 5/6$ のどれかと近い分数になっていることが読み取れます。

$r = 3$ は $x^r - 1$ が N で割り切れず f の周期にならないので手順 3 からやり直しです。$r = 6$ の場合は正しい周期が得られており、

$$\gcd(x^{r/2} - 1, N) = \gcd(7, 21) = 7,$$
$$\gcd(x^{r/2} + 1, N) = \gcd(9, 21) = 3$$

と確かに非自明な約数 $3, 7$ が得られます。

始めに $x = 4$ を選んだ場合、$f(j) = x^j \bmod N$ は表 1.6 のようになり、周期は 3 になります。この場合は、手順 7 で正しい周期が得られていたとしても、周期が奇数になるので手順 1 に戻り x を選び直します。また、$x = 20$ を選んだ場合、$20^2 \bmod 21 = 1$ なので周期は偶数の 2 になりますが、$20^{2/2} - 1 = 19$, $20^{2/2} + 1 = 21$ となり $x^{r/2} + 1$ が N で割り切れてしまい、手順 7 で非自明な約数が求まりません。この場合も、手順 1 に戻り x を選び直します。

j	0	1	2	3	4	5	6	$\cdots$
x^j	1	4	16	64	256	1024	4096	$\cdots$
$x^j \bmod N$	1	4	16	1	4	16	1	$\cdots$

表 1.6: $x = 4$ の場合の $f(j)$ の値

このように x の取り方によって、周期 r が正確に求まっていたとしても約数が求まらないこともあります。一方、手順 7 がうまくいくような x は表 1.7 の 2 行目で、N と互いに素な $1 < x < N$ のうち半数以上で約数が求まります。表 1.7 では、下 2 行がうまくいかない x になっていますが、それほど多くはありません。以上で、$N = 21$ の場合でも高い確率で N の非自明な約数が見つかることが分かりました。1 回で駄目だったとしても、数回繰り返せばほぼ確実に非自明な約数が求まるといえます。

x の種類	x のリスト
$\gcd(x, N) \neq 1$	$3, 6, 7, 9, 12, 14, 15, 18$
手順 7 でうまく約数が求まる	$2, 8, 10, 11, 13, 19$
f の周期が奇数	$4, 16$
f の周期が偶数かつ $(x^{r/2} + 1) \bmod N = 0$	$5, 17, 20$

表 1.7: $N = 21$ の場合の x の分類

第 2 章

量子回路モデル

この章では、線形代数の用語を復習しつつ量子回路モデルを厳密に定義します。1章で述べたように量子回路モデルはベクトルと行列の積だけで定義できます。一方で、数ベクトルのままでは量子ゲートや測定の記述が複雑になるので、量子ビット (qubit) という単位を導入します。

2.1 数ベクトルとベクトル空間

量子情報理論では線形代数が中心的な役割を果たします。必要な用語は適宜導入しますが、ここでは最も基本的な数ベクトル空間とその基底について復習します。

数を縦に並べ括弧を付けたものを**数ベクトル** (列ベクトル、縦ベクトル) といいます。例えば、

$$\begin{pmatrix}1\\0\end{pmatrix}, \quad \begin{pmatrix}1\\2\\-1\end{pmatrix}, \quad \begin{pmatrix}a_1\\a_2\\\vdots\\a_n\end{pmatrix} \tag{2.1}$$

などが数ベクトルになります。ここで、$a_1, \ldots, a_n$ は実数や複素数ですが、すべて実数なら実数成分の (または $\mathbb{R}$ 上の) 数ベクトル、複素数なら複素数成分の (または $\mathbb{C}$ 上の) 数ベクトルと呼ばれます。量子情報理論では基本的に複素数成分の数ベクトルを扱います。文章中だと縦に長いベクトルは見づらいので、${}^t(a_1, a_2, \ldots, a_n)$ と書くこともあります。

縦に何個の数を並べているかを数ベクトルの次元と呼びます。例えば、(2.1) のベクトルは左か

ら 2 次元、3 次元、n 次元の数ベクトルとなります。数ベクトルの和とスカラー倍が

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

$$c \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} ca_1 \\ ca_2 \\ \vdots \\ ca_n \end{pmatrix}$$

$$(a_1, \ldots, a_n, b_1, \ldots, b_n, c \in \mathbb{C})$$

によって定義され、結合法則や分配法則が成り立ちます。和は同じ次元の数ベクトルに対してのみ定義されます。

定義 2.1.1 (数ベクトル空間)**.** 複素数成分の n 次元数ベクトル全体の集合を $\mathbb{C}^n$ と表し、$\mathbb{C}$ 上の n 次元**数ベクトル空間**と呼ぶ。実数成分の n 次元数ベクトル全体の集合を $\mathbb{R}^n$ と表し、$\mathbb{R}$ 上の n 次元数ベクトル空間と呼ぶ。

以下、$\mathbb{C}^n$ について述べますが、$\mathbb{R}^n$ でも $\mathbb{C}$ を $\mathbb{R}$ に置き換えればまったく同じことがいえます。$\mathbb{C}^n$ を具体的に書くと

$$\mathbb{C}^n = \left\{ \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \middle| \, a_1, \ldots, a_n \in \mathbb{C} \right\}$$

となります。もちろん $\mathbb{C}^n$ は集合ですが、それだけではなく付加的な構造、和とスカラー倍という演算が備わっています:

$$\boldsymbol{v}, \boldsymbol{w} \in \mathbb{C}^n \Rightarrow \boldsymbol{v} + \boldsymbol{w} \in \mathbb{C}^n,$$
$$\boldsymbol{v} \in \mathbb{C}^n, c \in \mathbb{C} \Rightarrow c \cdot \boldsymbol{v} \in \mathbb{C}^n.$$

$\mathbb{C}^n$ のように和とスカラー倍を備えた集合のことをベクトル空間と呼びます。

定義 2.1.2 (ベクトル空間)**.** 集合 V に和とスカラー倍 (複素数倍) が備わっており、結合法則、分配法則などが成り立つとき、V を $\mathbb{C}$ 上の**ベクトル空間**と呼ぶ。また、ベクトル空間の元を**ベクトル**と呼ぶ。

例えば複素数成分の $n \times m$ 行列全体の集合を $M_{n,m}(\mathbb{C})$ と表すと、行列の和とスカラー倍に関して $M_{n,m}(\mathbb{C})$ は $\mathbb{C}$ 上のベクトル空間になります。ベクトル空間としてみると、$M_{n,m}(\mathbb{C})$ と

$\mathbb{C}^{nm}$ は nm 個の複素数の並べ方が違うだけで本質的には同じものです。しかし、数を直線状ではなく長方形に並べることで、行列の積という構造が分かりやすくなっています。

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}}_{n \times m \text{ 行列}} \longleftrightarrow \underbrace{\begin{pmatrix} a_{11} \\ a_{12} \\ \vdots \\ a_{ij} \\ \vdots \\ a_{nm} \end{pmatrix}}_{nm \text{ 次元数ベクトル}}$$

あるいは $n \times m$ 行列は n 次元数ベクトルを m 個並べたものだ、という構造を持っているとも思えます。次の節で導入する量子ビットも同様で、テンソル積というものを使い、数の並べ方を工夫することで「n 量子ビット」が「1 量子ビット」の積み重ねであるという構造を表現できるようになります。そのため数ベクトル空間という枠組みから外に出て一般のベクトル空間で考える必要が出てきます。

V が $\mathbb{C}$ 上のベクトル空間なら、例えば

$$(\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w}=\boldsymbol{u}+(\boldsymbol{v}+\boldsymbol{w}), \quad a\cdot(\boldsymbol{v}+\boldsymbol{w})=a\boldsymbol{v}+a\boldsymbol{w} \quad (\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\in V, a\in\mathbb{C})$$

などいくつかの関係式が成り立つことを要請しますが、それらについてはここでは深入りしません。本テキストで現れるベクトル空間は、数ベクトル空間か次の節で出てくるテンソル積しかなく、結合法則や分配法則などの基本的な等式は複素数の和と積の性質からすぐに分かるものになっているからです。したがって、ベクトル空間といっても $\mathbb{C}^n, M_{n,m}(\mathbb{C})$ やそれに近い具体的な集合を考えていると思って構いません。

線形代数では、いくつかのベクトルをスカラー倍して足すという操作が頻出します。この操作に名前を付けておきましょう。

定義 2.1.3 (線形結合)**.** V をベクトル空間とする。いくつかのベクトル $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k \in V$ に対して、

$$a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2 + \cdots + a_k\boldsymbol{v}_k \quad (a_1, \ldots, a_k \in \mathbb{C})$$

というスカラー倍の和のことを $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$ の**線形結合**という。

例 2.1.4. $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ は $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ と $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ の線形結合で表せる。実際、

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix} - 2\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

となっている。

ベクトルたちの関係として重要なものに基底があります。例えば、$\begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{C}^2$ は

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

のように $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ と $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ の線形結合で一通りに表せます。あるいは $\begin{pmatrix} a \\ b \\ c \end{pmatrix} \in \mathbb{C}^3$ は

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

のように $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ と $\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ と $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ の線形結合で一通りに表せます。同様のことは $\mathbb{C}^4, \mathbb{C}^5$ や一般に $\mathbb{C}^n$ でも成り立ちます。

定義 2.1.5 (標準基底、基底)**.** $\boldsymbol{e}_i$ を第 i 成分のみ 1 でそれ以外が 0 の n 次元数ベクトルとする。$\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n$ という列を $\mathbb{C}^n$ の**標準基底**と呼ぶ。

V をベクトル空間とし、$\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in V$ とする。任意の $\boldsymbol{v} \in V$ が $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ の線形結合で一通りに表せるとき、$\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ という列を V の**基底**と呼ぶ。このような基底が存在するとき、V の**次元**は n であるといい、$\dim_{\mathbb{C}}(V) = n$ と定義する。

$\boldsymbol{e}_1$ や $\boldsymbol{e}_2$ 一つ一つのことではなく、$\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ をまとめて標準基底と呼んでいます。定義から、標準基底は基底になっていて $\dim_{\mathbb{C}}(\mathbb{C}^n) = n$ が成り立ちます。$M_{2,2}(\mathbb{C})$ なら

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

が基底になり、$\dim_{\mathbb{C}}(M_{2,2}(\mathbb{C})) = 4$ となります。同様に $M_{n,m}(\mathbb{C})$ の基底が具体的に作れ、$\dim_{\mathbb{C}}(M_{n,m}(\mathbb{C})) = nm$ が分かります。$M_{2,2}(\mathbb{C})$ の基底としてはほかにも

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

のようなものもあり、用途によって使い分けられます。

2.2　量子ビット

整数を書くとき現代では多くの場合 10 進数を使います。0 から 9 を並べて

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, \ldots$$

のように様々な数を表現します。あるいは、学校の全クラスを表記するときに、学年とアルファベットを並べた

$$1A, 1B, 1C, 1D, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D$$

といった表記を用いることもあります。このような少しの数や記号を並べて大きな数を表記する方法を位取り記数法と呼びます。基本的な記号だけで多くのものを表現できることと、単なる数ではなく構造化されたデータ (数列や文字列) として扱えるようになるという長所があります。

情報理論では扱いの容易さから $0, 1$ の 2 種類の数を並べる位取り記数法 (2 進法) を用いることが多くあります。

2 進	000	001	010	011	100	101	110	111
10 進	0	1	2	3	4	5	6	7

そのような文脈では先頭の 0 たちは省略せずに書き、同じ桁数になるようにします。2 進数の 1 桁のことを 1 **ビット** (bit, binary digit) と呼びます。$0, 1$ を 2 個並べた $00, 01, 10, 11$ を 2 ビット、n 個なら n ビットといいます。数式で扱いやすくするために n ビットの整数を、長さ n の $0, 1$ の数列とみなすことも多くあります。その場合、直積集合 $\{0, 1\}^n$ で長さ n の $0, 1$ の数列全体を表します。

2 進法で表すメリットとして、非常に簡単な操作の組み合わせで複雑な操作が実現できるということがあります。例えば 2 進数の足し算の筆算を行う場合、1 桁の三つの数 (0 か 1) の和だけできれば、それを組み合わせることで何桁でも計算できるようになります。$0, 1$ と単純な処理だけなので理論的にも現実的にも扱いやすくなっています。

量子でも同様で、$0, 1$ を基本単位にして様々な操作を考えていきます。ベクトル $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ と $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ をそれぞれ $|0\rangle, |1\rangle$ と表記しましょう。そうすると $|0\rangle, |1\rangle$ は $\mathbb{C}^2$ の基底になっていて 2 次元の数ベクトルは

$$\begin{pmatrix} a \\ b \end{pmatrix} = a\,|0\rangle + b\,|1\rangle \quad (a, b \in \mathbb{C})$$

と一通りに表せます。$a = b = 0$ の場合の零ベクトルを除いた、このようなベクトルを $|0\rangle$ と $|1\rangle$ が**重ね合わせ**になった状態だと思い、1 **量子ビット** (qubit, quantum binary digit) の状態と呼びます[*1]。$|0\rangle, |1\rangle$ は $\mathbb{C}^2$ の標準基底の別表記に過ぎず、1 量子ビットは単なる 2 次元の数ベクトルにほかなりません。

$|\cdot\rangle$ は Dirac により導入された記法で、内積 $\langle x \mid y\rangle$ の左右を分解した右側という気持ちが込められています。括弧を英語でブラケット (bracket) と呼ぶことにちなんで、$\langle x|$ をブラ (bra)、$|y\rangle$ をケット (ket) と呼びます。ブラも重要ですが、入門編ではケットのみを扱います。

[*1] 厳密には長さ 1 のベクトルのみを考えます。

2 量子ビットも同様に

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

$$a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle = \begin{pmatrix}a\\b\\c\\d\end{pmatrix}$$

を満たす記号として定義することができます。この定義でも問題ないのですが、単に数ベクトルを書き換えただけで $00, 01, 10, 11$ という 2 進数の長所が生かされていません。2 進数は 1 ビット $(0, 1)$ の列と思え、各桁ごとに操作ができるというのが長所でした。それに合わせて、$|01\rangle$ などを 1 量子ビットの積 $|0\rangle\,|1\rangle$ として捉え、2 量子ビットを 1 量子ビットの積の和として扱えるようにしましょう。例えば

$$\underset{\text{第 1 量子ビット}}{(|0\rangle + |1\rangle)} \times \underset{\text{第 2 量子ビット}}{|0\rangle} = |00\rangle + |10\rangle$$

のような計算ができる記号 (積) が必要になります。一般の場合は後にして、まずは 2 量子ビットに対応する場合だけを定義しましょう。

定義 2.2.1 ($\mathbb{C}^2$ と $\mathbb{C}^2$ のテンソル積)**.** $\mathbb{C}^2 \otimes \mathbb{C}^2$ を以下の性質を満たす集合とする。

(1) $\mathbb{C}^2 \otimes \mathbb{C}^2$ はベクトル空間である。

(2) $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{C}^2$ に対して、$\boldsymbol{v} \otimes \boldsymbol{w} \in \mathbb{C}^2 \otimes \mathbb{C}^2$ という元が定まり、

$$\begin{aligned}(a\boldsymbol{v}_1 + b\boldsymbol{v}_2) \otimes \boldsymbol{w} &= a(\boldsymbol{v}_1 \otimes \boldsymbol{w}) + b(\boldsymbol{v}_2 \otimes \boldsymbol{w}),\\ \boldsymbol{v} \otimes (a\boldsymbol{w}_1 + b\boldsymbol{w}_2) &= a(\boldsymbol{v} \otimes \boldsymbol{w}_1) + b(\boldsymbol{v} \otimes \boldsymbol{w}_2)\\ &\qquad (\forall a, b \in \mathbb{C}, \boldsymbol{v}, \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{w}, \boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{C}^2)\end{aligned}$$

という分配法則が成り立つ。(複数の成分の線形性、**多重線形性**[*2]と呼ばれる。)

(3) $\mathbb{C}^2 \otimes \mathbb{C}^2$ の元は

$$a\begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} + b\begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} + c\begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} + d\begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} \quad (a, b, c, d \in \mathbb{C})$$

という形でただ一通りに表せる。

このとき、$\mathbb{C}^2 \otimes \mathbb{C}^2$ を $\mathbb{C}^2$ と $\mathbb{C}^2$ の**テンソル積**と呼ぶ。また、$\boldsymbol{v} \otimes \boldsymbol{w}$ を $\boldsymbol{v}$ と $\boldsymbol{w}$ のテンソル積と呼ぶ。

[*2] 二つの成分の場合は特別に双線形性と呼ばれます。

条件 3 は $\mathbb{C}^2 \otimes \mathbb{C}^2$ の元は (a, b, c, d) という複素数の四つ組と一対一に対応しており、和とスカラー倍も実質的に $\mathbb{C}^4$ のものと同じといっています:

$$a\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}+b\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}+c\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}+d\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}\longleftrightarrow\begin{pmatrix}a\\b\\c\\d\end{pmatrix}.$$

$\mathbb{C}^2 \otimes \mathbb{C}^2$ と $\mathbb{C}^4$ はベクトル空間としては実質的に同じですが、条件 2 により $\mathbb{C}^2 \otimes \mathbb{C}^2$ に追加の構造が備わっています。これにより、$\mathbb{C}^2 \otimes \mathbb{C}^2$ の元を単なるベクトル以上の、二つの数ベクトルの積の和と思えるようになります。例えば、$\otimes$ の分配法則からベクトルのテンソル積は

$$\begin{pmatrix}a\\b\end{pmatrix}\otimes\begin{pmatrix}c\\d\end{pmatrix}=ac\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}+ad\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}+bc\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}+bd\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}$$
$$\longleftrightarrow\begin{pmatrix}ac\\ad\\bc\\bd\end{pmatrix}$$

と具体的に書くことができ、$\mathbb{C}^4$ の元 ${}^t(ac, ad, bc, bd)$ と対応づきます。しかし、4 次元数ベクトルとして書いてしまうと、二つの数ベクトルの積になっているという構造が見えづらくなってしまいます。テンソル積 $\mathbb{C}^2 \otimes \mathbb{C}^2$ を導入することで、ベクトルとして扱いつつ積構造も捉えやすくなっています。

さて量子ビットの話に戻りましょう。$|0\rangle \otimes |1\rangle$ などを $|01\rangle$ のように表すことにします。そうすると

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

と対応し、

$$\mathbb{C}^2 \otimes \mathbb{C}^2 = \{a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle \mid a, b, c, d \in \mathbb{C}\}$$

となります。$\mathbb{C}^2 \otimes \mathbb{C}^2$ の元、つまり

$$a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle \quad (a, b, c, d \in \mathbb{C})$$

を $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ が重ね合わさった状態だと思い、2 量子ビットの状態と呼びます。1 量子ビットの場合と同じく、零ベクトルは除くことにしましょう。$\otimes$ の分配法則から

$$(|0\rangle + |1\rangle) \otimes |1\rangle = |01\rangle + |11\rangle$$

のような式が成り立ち、1 量子ビットを二つ掛けることで 2 量子ビットが得られるという欲しかった性質が成り立っています。

3 量子ビット、4 量子ビットなども同様で、テンソル積する個数を $3, 4, \ldots$ と増やしていくだけで定義できます。例えば 3 量子ビットの状態とは

$$\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle$$

のようなもので、加えて $|0\rangle \otimes |1\rangle \otimes |0\rangle = |010\rangle$ のような式が成り立つように記号を定義します。

定義 2.2.2 (k 個の $\mathbb{C}^2$ のテンソル積)**.** $(\mathbb{C}^2)^{\otimes k} = \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$ (k 個) を以下の性質を満たす集合とする。$\boldsymbol{e}_1, \boldsymbol{e}_2$ を $\mathbb{C}^2$ の標準基底とする。

(1) $(\mathbb{C}^2)^{\otimes k}$ はベクトル空間である。

(2) $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k \in \mathbb{C}^2$ に対して、$\boldsymbol{v}_1 \otimes \cdots \otimes \boldsymbol{v}_k \in (\mathbb{C}^2)^{\otimes k}$ という元が定まり多重線形性が成り立つ。(定義 2.2.1 の条件 2 を参照。)

(3) $(\mathbb{C}^2)^{\otimes k}$ の 2^k 個のベクトル

$$\begin{aligned}&\boldsymbol{e}_1 \otimes \boldsymbol{e}_1 \otimes \cdots \otimes \boldsymbol{e}_1 \otimes \boldsymbol{e}_1, \quad \boldsymbol{e}_1 \otimes \boldsymbol{e}_1 \otimes \cdots \otimes \boldsymbol{e}_1 \otimes \boldsymbol{e}_2, \quad \boldsymbol{e}_1 \otimes \boldsymbol{e}_1 \otimes \cdots \otimes \boldsymbol{e}_2 \otimes \boldsymbol{e}_1,\\ &\cdots,\\ &\boldsymbol{e}_2 \otimes \boldsymbol{e}_2 \otimes \cdots \otimes \boldsymbol{e}_1 \otimes \boldsymbol{e}_2, \quad \boldsymbol{e}_2 \otimes \boldsymbol{e}_2 \otimes \cdots \otimes \boldsymbol{e}_2 \otimes \boldsymbol{e}_1, \quad \boldsymbol{e}_2 \otimes \boldsymbol{e}_2 \otimes \cdots \otimes \boldsymbol{e}_2 \otimes \boldsymbol{e}_2\end{aligned}$$

は基底である。つまり、$(\mathbb{C}^2)^{\otimes k}$ の任意のベクトルは、このベクトルたちの線形結合でただ一通りに表せる。

2 量子ビットの場合と同様に、$|0\rangle \otimes |1\rangle \otimes |1\rangle$ などを $|011\rangle$ のように表すことにします。定義 2.2.2 の (3) の条件中の $\boldsymbol{e}_1 \otimes \boldsymbol{e}_1 \otimes \boldsymbol{e}_1$ などは $|0\rangle \otimes |0\rangle \otimes |0\rangle = |000\rangle$ などに対応することに注意しましょう。$\boldsymbol{e}_1 = |0\rangle, \boldsymbol{e}_2 = |1\rangle$ なので、添え字と $0, 1$ の対応は 1 だけずれています。この表記では定義 2.2.2 の (3) の基底は

$$|00\cdots000\rangle, |00\cdots001\rangle, |00\cdots010\rangle, |00\cdots011\rangle, \cdots, |11\cdots101\rangle, |11\cdots110\rangle, |11\cdots111\rangle$$

と表せ、2 進数が小さい順に 0 から並んでいることが分かります。また、$(\mathbb{C}^2)^{\otimes 3}$ の元は

$$a|000\rangle + b|001\rangle + c|010\rangle + d|011\rangle + e|100\rangle + f|101\rangle + g|110\rangle + h|111\rangle \quad (a, b, \ldots, h \in \mathbb{C})$$

という線形結合で一通りに表せます。$(\mathbb{C}^2)^{\otimes 4}, (\mathbb{C}^2)^{\otimes 5}, \ldots$ と k を増やしていっても、項の数が $2^4, 2^5, \ldots$ と指数関数的に増えていき具体的に書くのは大変ですが、同じような線形結合で表せます。$(\mathbb{C}^2)^{\otimes k}$ の零ベクトル以外の元を k 量子ビットの状態と呼びましょう。

2.3 量子ビットの記法

$|\cdot\rangle$ にはいくつかの慣習的な記法があるので紹介しておきましょう。文字 a, b が $0, 1$ のいずれかを表す場合に、$|a0\rangle$ や $|ab\rangle$ のように書くことがあります。例えば、任意の $a \in \{0, 1\}$ に対して

$$|a\rangle \otimes (|0\rangle + |1\rangle) = |a0\rangle + |a1\rangle$$

となる、といった使い方をします。a, b の部分に複雑な式を書きたいときもありますが、その場合には $|a, a \oplus b\rangle$ のようにどこからどこまでが一つの式なのかが分かるように , で区切ります。

また、$(\mathbb{C}^2)^{\otimes k}$ の元を量子ビットの状態と思う場合には、$v, \boldsymbol{v}, \vec{v}$ のようなベクトルを表す記号ではなく、$|v\rangle$ と書くことがあります。$\vec{v}$ のように $|v\rangle$ 全体で一つのベクトルとして扱います。ただし、$|v\rangle \otimes |w\rangle = |v, w\rangle$ のように、さも v や w という何かがあるような表記をすることもあるので注意してください。

同じ量子ビットを n 個テンソル積する場合、$|v\rangle^{\otimes n}$ のように指数で表記します。例えば、$|0\rangle^{\otimes n} = |00\cdots0\rangle$ となります。$|0\rangle^{\otimes n}$ や $|1\rangle^{\otimes n}$ は、$|0_n\rangle, |0^n\rangle, |1_n\rangle, |1^n\rangle$ のように書くこともあります。

n 量子ビットは 1 量子ビットを n 個テンソル積したものの和ですが、左端を 1 番目とするか、右端を 1 番目とするかは文献によります。本テキストでは左端、つまり $|i_1 i_2 \cdots i_n\rangle$ の i_1 の量子ビットを**第 1 量子ビット**と呼ぶことにします。

アルゴリズムを考える際、単なる $0, 1$ の列というだけでなく、いくつかの意味の塊に分けられると便利です。$\otimes$ を省略しない部分を残すことで、区切りを表すことにします。例えば、

$$|00010000\rangle = |0001\rangle \otimes |0000\rangle = |000\rangle \otimes |100\rangle \otimes |00\rangle$$

のように同じ元でも区切りを変えることで意味を伝えやすくできます。一つ目は全体を一塊と捉えています。二つ目は最初の 4 量子ビットと後ろの 4 量子ビットに分けており、三つ目は $3, 3, 2$ 量子ビットの塊に分けています。このような $\otimes$ の記法は数学的には以下のように定式化されます。

定義 2.3.1. n, m を正の整数とする。$\otimes\colon (\mathbb{C}^2)^{\otimes n} \times (\mathbb{C}^2)^{\otimes m} \to (\mathbb{C}^2)^{\otimes n+m}$ を以下の性質を満たすただ一つの写像として定義する。

(1) $\otimes$ は双線形性を持つ:

$$\begin{aligned}(a|v_1\rangle + b|v_2\rangle) \otimes |w\rangle &= a(|v_1\rangle \otimes |w\rangle) + b(|v_2\rangle \otimes |w\rangle),\\ |v\rangle \otimes (a|w_1\rangle + b|w_2\rangle) &= a(|v\rangle \otimes |w_1\rangle) + b(|v\rangle \otimes |w_2\rangle)\\ &\qquad (\forall a, b \in \mathbb{C}, |v\rangle, |v_1\rangle, |v_2\rangle \in (\mathbb{C}^2)^{\otimes n}, |w\rangle, |w_1\rangle, |w_2\rangle \in (\mathbb{C}^2)^{\otimes m}).\end{aligned}$$

(2) 任意の $i_1, i_2, \ldots, i_n, j_1, j_2, \ldots, j_m \in \{0, 1\}$ に対して

$$|i_1 i_2 \cdots i_n\rangle \otimes |j_1 j_2 \cdots j_m\rangle = |i_1 i_2 \cdots i_n j_1 j_2 \cdots j_m\rangle$$

が成り立つ。

定義の条件を満たす写像がただ一つ存在することの証明は省略します。三つのテンソル積の場合、$(|u\rangle \otimes |v\rangle) \otimes |w\rangle$ のようにどの順番で積を取るかを括弧で明示すべきですが、結合法則 $(|u\rangle \otimes |v\rangle) \otimes |w\rangle = |u\rangle \otimes (|v\rangle \otimes |w\rangle)$ が成り立つため、括弧を省略して $|u\rangle \otimes |v\rangle \otimes |w\rangle$ と表します。文献によっては $\otimes$ を省略して $|0001\rangle\,|0000\rangle$ のように表し、$|0001\rangle$ のような量子ビットの一塊を**レジスタ**と呼びます。

2 進法ではなく 10 進法で書いたほうが分かりやすい場面も多くあります。その場合は、

$$|000\rangle = |0\rangle\,, \quad |001\rangle = |1\rangle\,, \quad |010\rangle = |2\rangle\,, \quad |011\rangle = |3\rangle$$

のように $|\cdot\rangle$ の中にそのまま 10 進法で書きます。この書き方だけだと何量子ビットか分からなくなりますが、量子ビット数が重要でなかったり文脈から明らかだったりするため、あまり問題にはなりません。$\mathbb{C}^{2^k}$ の標準基底を $e_1, e_2, \ldots, e_{2^k}$ とすると、$|0\rangle, |1\rangle, \ldots, \left|2^k-1\right\rangle \in (\mathbb{C}^2)^{\otimes k}$ と綺麗[*3]に一対一対応し、量子ゲートを行列で表す場合などに便利になります。

2.4　測定

量子ビットを定義したので、1 章で述べた測定について詳しくみていきましょう。測定には様々な方法がありますが、本テキストでは標準基底 $|0\rangle^{\otimes n}, \ldots, |1\rangle^{\otimes n}$ による測定と呼ばれるものだけを考えます。以下の定義では 2 進法では見づらいので 10 進法で表記しています。

定義 2.4.1 (測定)**.** n 量子ビットの状態 $\sum_{i=0}^{2^n-1} a_i\,|i\rangle \in (\mathbb{C}^2)^{\otimes n}$ を**測定**すると、$0, 1, \ldots, 2^n-1$ のいずれかが以下の確率で出力される:

$$i \in \{0,1,2,\ldots,2^n-1\} \text{ が測定値になる確率} = \frac{|a_i|^2}{|a_0|^2 + |a_1|^2 + \cdots + |a_{2^n-1}|^2}.$$

測定値が i の場合、測定後の状態は $|i\rangle$ に変化する。

量子回路モデルでは、量子ビットの状態 $\sum_i a_i\,|i\rangle$ そのものを直接知ることはできず、測定を行うことでしか情報を取り出せないことに注意してください。また、測定は量子ビットの状態を $|0\rangle, \ldots, |2^n-1\rangle$ のいずれかにする非可逆な操作であることに注意しましょう。

例 2.4.2. $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ を測定すると 0 と 1 がそれぞれ 1/2 の確率で出力される。$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ を測定すると 00 と 11 がそれぞれ 1/2 の確率で出力される。

[*3] 1 ずれていますが、標準基底を e_1 から始めることが多いという慣例のためで、e_0 から始めればぴったり一致します。

例 2.4.3. $\frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)$ は

$$\frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

なので、測定すると $00, 01, 10, 11$ がそれぞれ $1/4$ の確率で出力される。

この例では、測定値の 1 ビット目は $0, 1$ が等確率で現れ、2 ビット目も $0, 1$ が等確率に現れかつ 1 ビット目とは独立になっています。第 1 量子ビット $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ と第 2 量子ビット $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ をそれぞれ独立に測定したような結果になっており、状態が因数分解されたような形で表せていることを反映しています。

命題 2.4.4. $|v\rangle, |w\rangle$ を 1 量子ビットの状態とする。$i \in \{0, 1\}$ とすると、$|v\rangle \otimes |w\rangle$ を測定したときに 1 ビット目が i になる確率は $|v\rangle$ を測定したときに i が出力される確率と等しい。2 ビット目に関しても同様である。さらに、測定値の 1 ビット目と 2 ビット目の結果は独立である。

証明. $|v\rangle = a_0 |0\rangle + a_1 |1\rangle$, $|w\rangle = b_0 |0\rangle + b_1 |1\rangle$ と表す。このとき、

$$\begin{aligned} |v\rangle \otimes |w\rangle &= (a_0 |0\rangle + a_1 |1\rangle) \otimes (b_0 |0\rangle + b_1 |1\rangle) \\ &= a_0 b_0 |00\rangle + a_0 b_1 |01\rangle + a_1 b_0 |10\rangle + a_1 b_1 |11\rangle \end{aligned}$$

となる。$i, j \in \{0, 1\}$ とし $|v\rangle \otimes |w\rangle$ を測定すると、ij が出力される確率[*4]は

$$\frac{|a_i|^2 |b_j|^2}{|a_0 b_0|^2 + |a_0 b_1|^2 + |a_1 b_0|^2 + |a_1 b_1|^2} = \frac{|a_i|^2}{|a_0|^2 + |a_1|^2} \cdot \frac{|b_j|^2}{|b_0|^2 + |b_1|^2}$$

となる。これは、$|v\rangle$ を測定して i が出力される確率と、$|w\rangle$ を測定して j が出力される確率の積になっている。ここから主張が従う。 □

命題 2.4.4 の帰結として、$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ が $|v\rangle \otimes |w\rangle$ と表せないことが分かります。実際、$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ の測定値は 00 または 11、つまり 1 ビット目と 2 ビット目が常に等しくなり、測定値の 1 ビット目と 2 ビット目は独立ではありません。

ここではすべての量子ビットを同時に測定するという操作だけを定義しましたが、続編で導入する部分測定と呼ばれる特定の量子ビットのみを測定する操作もあります。部分測定を使うと命題 2.4.4 をより一般の状態に拡張することもできます。

2.5 量子ゲート

量子回路モデルでは、量子ゲート (行列) を掛けることで量子ビット (ベクトル) を操作します。ここでは、量子ゲートのうち、X ゲート、Y ゲート、Z ゲート、Hadamard ゲート、CNOT ゲート、Toffoli ゲート、位相ゲートを扱います。

[*4] $i \times j$ ではなく i, j を並べた 2 ビットの数です。

2.5.1　量子ゲートと線形写像

X ゲートと呼ばれる量子ゲートは

$$|0\rangle \xmapsto{\text{X ゲート}} |1\rangle\,,$$
$$|1\rangle \xmapsto{\text{X ゲート}} |0\rangle\,,$$
$$\alpha\,|0\rangle + \beta\,|1\rangle \xmapsto{\text{X ゲート}} \alpha\,|1\rangle + \beta\,|0\rangle$$

のように、1 量子ビットの $|0\rangle$ と $|1\rangle$ を入れ替えるような操作です。一般に、量子ゲートは n 量子ビットの状態を与えると n 量子ビットの状態を返すようなもの、すなわち $(\mathbb{C}^2)^{\otimes n}$ から $(\mathbb{C}^2)^{\otimes n}$ への写像です。X ゲート、Y ゲート、Z ゲートなどは 1 量子ビットに対する操作で、それぞれ特別な 2×2 行列を掛ける写像で与えられます。まずは、X ゲートを例に量子ゲートがどういうものかを解説します。

1 量子ビットは $\mathbb{C}^2$ の元のことで、$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ と定義したことを思い出しましょう。2×2 行列 $A = \begin{pmatrix}a & b\\c & d\end{pmatrix}$ をベクトルに掛けるという演算は

$$A\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \begin{pmatrix}a\alpha + b\beta\\c\alpha + d\beta\end{pmatrix}$$

という式で定義されていました。特に、$|0\rangle$ と $|1\rangle$ に掛けると

$$A\,|0\rangle = \begin{pmatrix}a & b\\c & d\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}a\\c\end{pmatrix} = a\,|0\rangle + c\,|1\rangle\,,$$
$$A\,|1\rangle = \begin{pmatrix}a & b\\c & d\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}b\\d\end{pmatrix} = b\,|0\rangle + d\,|1\rangle$$

となり、行列を掛けるという写像は、量子ビットの状態を変化させる操作と思うことができます。何かを変化させる操作と思っている場合、行列を掛けることを行列を**作用**させる、という言い方をします。

X ゲートは 2×2 行列 $X = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$ を掛けるという写像で、量子ビットの記号で書くと

$$X\,|0\rangle = |1\rangle\,, \quad X\,|1\rangle = |0\rangle\,,$$
$$X(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\,|1\rangle + \beta\,|0\rangle \quad (\alpha, \beta \in \mathbb{C})$$

という性質を満たします。つまり、X ゲートを 1 量子ビットの状態に作用させると $|0\rangle$ と $|1\rangle$ が入れ替わります。

行列を掛ける写像なので分配法則 (線形性)

$$X(a\left|0\right\rangle + b\left|1\right\rangle) = aX\left|0\right\rangle + bX\left|1\right\rangle$$

が成り立ちます。この性質から、X ゲートは基底 $\left|0\right\rangle, \left|1\right\rangle$ の行き先だけで一通りに決まることに注意しましょう。量子ゲートはすべて、X ゲートのように行列を掛ける写像 (とそのテンソル積) で表せ、$\left|0\right\rangle, \left|1\right\rangle$ や $\left|00\right\rangle, \left|01\right\rangle, \ldots$ などの行き先だけで完全に決まります。線形代数の言葉でこのような写像は線形写像と呼ばれます。

定義 2.5.1 (線形写像)**.** $T\colon V \to W$ をベクトル空間の間の写像とする。T が

$$T(a\boldsymbol{v} + b\boldsymbol{w}) = aT(\boldsymbol{v}) + bT(\boldsymbol{w}) \quad (\forall a, b \in \mathbb{C}, \boldsymbol{v}, \boldsymbol{w} \in V)$$

を満たすとき、T を**線形写像**という。行列の記法を真似て、線形写像 S, T の合成を $S \cdot T$ や ST と表し、$T(\boldsymbol{v})$ を $T \cdot \boldsymbol{v}$ や $T\boldsymbol{v}$ と表す。

命題 2.5.2. V, W をベクトル空間とし、$\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in V$ を V の基底とする。任意の $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n \in W$ に対して、$T\boldsymbol{v}_i = \boldsymbol{w}_i$ $(\forall i)$ を満たす線形写像 $T\colon V \to W$ がただ一つ存在する。

証明. V の元は $a_1\boldsymbol{v}_1 + \cdots + a_n\boldsymbol{v}_n$ $(a_1, \ldots, a_n \in \mathbb{C})$ と一通りに表せるので、T を

$$T(a_1\boldsymbol{v}_1 + \cdots + a_n\boldsymbol{v}_n) = a_1\boldsymbol{w}_1 + \cdots + a_n\boldsymbol{w}_n$$

によって定義すればよい。線形性とただ一つであることの証明は省略する。 □

$\left|0\right\rangle^{\otimes n}, \cdots, \left|1\right\rangle^{\otimes n}$ は $(\mathbb{C}^2)^{\otimes n}$ の基底なので、これらの行き先を決めれば線形写像はただ一つに定まります。$n = 1$ の場合は $\left|0\right\rangle, \left|1\right\rangle$ の行き先で、$n = 2$ の場合は $\left|00\right\rangle, \left|01\right\rangle, \left|10\right\rangle, \left|11\right\rangle$ の行き先で決まります。$\mathbb{C}^n$ や $(\mathbb{C}^2)^{\otimes n}$ は無限集合ですが、線形写像は有限個のデータだけで決まってしまうというのがよい点の一つです。

命題 2.5.2 から $\mathbb{C}^2$ から $\mathbb{C}^2$ への線形写像はすべて 2×2 行列を掛ける写像で表すことができます。

定理 2.5.3. $T\colon \mathbb{C}^2 \to \mathbb{C}^2$ を線形写像とする。このとき、複素数成分の 2×2 行列 A がただ一つ存在して、

$$T\boldsymbol{v} = A\boldsymbol{v} \quad (\forall \boldsymbol{v} \in \mathbb{C}^2)$$

が成り立つ。

証明. $T\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}a\\c\end{pmatrix}, T\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}b\\d\end{pmatrix}$ $(a, b, c, d \in \mathbb{C})$ と表す。$A \coloneqq \begin{pmatrix}a & b\\c & d\end{pmatrix}$ と置くと、

$$A\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}a\\c\end{pmatrix} = T\begin{pmatrix}1\\0\end{pmatrix},$$

$$A\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}b\\d\end{pmatrix} = T\begin{pmatrix}0\\1\end{pmatrix}$$

となる。$\alpha, \beta \in \mathbb{C}$ とすると

$$\begin{aligned} A\begin{pmatrix}\alpha\\\beta\end{pmatrix} &= \alpha A\begin{pmatrix}1\\0\end{pmatrix} + \beta A\begin{pmatrix}0\\1\end{pmatrix} && \text{(行列の積の線形性)} \\ &= \alpha T\begin{pmatrix}1\\0\end{pmatrix} + \beta T\begin{pmatrix}0\\1\end{pmatrix} \\ &= T\begin{pmatrix}\alpha\\\beta\end{pmatrix} && \text{(T の線形性)} \end{aligned}$$

となる。したがって、$T\boldsymbol{v} = A\boldsymbol{v}$ $(\forall \boldsymbol{v} \in \mathbb{C}^2)$ が成り立つ。定理の条件を満たす A がただ一つであることは、A が二つのベクトル $T\begin{pmatrix}1\\0\end{pmatrix}, T\begin{pmatrix}0\\1\end{pmatrix}$ を横に並べたものにならなければいけないことから従う。 □

以後、行列そのものと行列を掛ける写像を区別せずに考えることにします。

2.5.2　X ゲート、Y ゲート、Z ゲート

X, Y, Z ゲートはそれぞれ

$$X = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}, \quad Y = \begin{pmatrix}0 & -\sqrt{-1}\\\sqrt{-1} & 0\end{pmatrix}, \quad Z = \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}$$

という行列で与えられます。命題 2.5.2 に従って、$|0\rangle, |1\rangle$ の行き先を表にしておきましょう。

入力	出力
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$

表 2.1: X ゲート

入力	出力
$\|0\rangle$	$\sqrt{-1}\,\|1\rangle$
$\|1\rangle$	$-\sqrt{-1}\,\|0\rangle$

表 2.2: Y ゲート

入力	出力
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$-\,\|1\rangle$

表 2.3: Z ゲート

X ゲートは $|0\rangle$ と $|1\rangle$ を入れ替える操作です。0 を偽、1 を真に対応させると、この操作は否定 (NOT, $\neg$) に相当します。Z ゲートは $|1\rangle$ の符号を反転させる操作です。Z ゲートを作用させた直後に測定を行っても結果にはなんら影響はありませんが、Z ゲートの後にほかの操作を組み合わせることで意味を持ちます。

$Y = \sqrt{-1}XZ$ なので、Y ゲートは Z ゲートの後に X ゲートを作用させたものと実質的に同じです。量子ゲートはすべて線形写像なので、スカラー倍するという操作は最終的な測定の結果には何も影響がないことに注意してください。ほかにも X, Y, Z ゲートの間には

$$X^2 = Y^2 = Z^2 = I \text{ (単位行列)},$$

$$XY = -YX = \sqrt{-1}Z, \quad YZ = -ZY = \sqrt{-1}X, \quad ZX = -XZ = \sqrt{-1}Y$$

という関係式が成り立ちます。この関係式の対称性は、1 量子ビットを球面[*5]上の点と対応させたときに、X, Y, Z ゲートがそれぞれ x, y, z 軸中心の π 回転になっている事実を反映しています。

2.5.3 Hadamard ゲート

X, Y, Z ゲートだけでは $|0\rangle, |1\rangle$ の行き先は $|0\rangle, |1\rangle$ の定数倍にしかならず、複雑な操作は実現できません。特に量子ビット特有の $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ のような重ね合わせの状態を作ることができません。**Hadamard(アダマール) ゲート**によりこのような状態を作れるようになります。Hadamard ゲートは H で表され行列

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

で与えられます。$|0\rangle, |1\rangle$ に作用させると

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

となります。

H は X, Z と

$$H^2 = I, \quad HXH = Z, \quad HZH = X$$

という関係があり、Hadamard ゲートを使うことで X と Z を相互に入れ替えることができます。また、この関係は X を H で対角化したものが Z である、という言い方もできます。

2.5.4 CNOT ゲート

ここまでの量子ゲートはすべて 1 量子ビットに対するものでした。CNOT ゲートと次の Toffoli ゲートは複数の量子ビットの操作で、条件分岐のような複雑な計算が可能になります。

CNOT ゲートは 2 量子ビットに対する操作で、以下の条件を満たす線形写像 CX で与えられます:

$$CX|00\rangle = |00\rangle,$$
$$CX|01\rangle = |01\rangle,$$
$$CX|10\rangle = |11\rangle,$$

[*5] Bloch 球と呼ばれます。Bloch 球については続編で扱います。

$$CX\,|11\rangle = |10\rangle\,.$$

CNOT ゲートは第 1 量子ビットをみて 0 なら何もしない、1 なら第 2 量子ビットの $0,1$ を反転させる、という操作になります。あるいは、第 1 量子ビットが 1 の場合だけ第 2 量子ビットに X を作用させる量子ゲートということもできます。これは $a,b\in\{0,1\}$ を使って

$$CX\,|ab\rangle = |a, a\oplus b\rangle \tag{2.2}$$

と一つの式で表せます。$\oplus$ は $0\oplus 0 = 1\oplus 1 = 0$, $1\oplus 0 = 0\oplus 1 = 1$ で定義される和で**排他的論理和** (**XOR**) や $\mathbb{Z}/2\mathbb{Z}$ の和として知られています。$|00\rangle, \ldots, |11\rangle$ に対しては単なる場合分けのような操作になりますが、Hadamard ゲートと組み合わせて重ね合わせになっている状態に作用させるとより複雑な状態を作り出すことができます。

CNOT や CX の C は controlled(制御) を意味します。単なる NOT(X ゲート) ではなく、第 1 量子ビットでコントロールされている NOT という意味です。この第 1 量子ビットを**制御ビット**と呼び、変化するほうのビットを**標的ビット**と呼びます。制御ビットと標的ビットを明示したい場合には、$CX_{1,2}$ や $CX_{1\to 2}$ のように一つ目の添え字に制御ビットの場所を書きます。役割を入れ替えて第 2 量子ビットを制御ビットとするような CNOT ゲートを考えることもでき、第 2 量子ビットを制御ビットにする場合は $CX_{2\to 1}$ と表記します。$CX_{2\to 1}$ の作用は

$$CX_{2\to 1}\,|ab\rangle = |a\oplus b, b\rangle\,,\quad (a,b\in\{0,1\})$$

で与えられます。

2.5.5　Toffoli ゲート

Toffoli(トフォリ) ゲートは 3 量子ビットに対する操作で、以下の条件を満たす線形写像 CCX で与えられます:

$$CCX\,|ab0\rangle = \begin{cases} |ab0\rangle & (a\cdot b = 0) \\ |ab1\rangle & (a = b = 1) \end{cases},$$
$$CCX\,|ab1\rangle = \begin{cases} |ab1\rangle & (a\cdot b = 0) \\ |ab0\rangle & (a = b = 1) \end{cases}$$
$$(a,b\in\{0,1\}).$$

Toffoli ゲートは第 $1,2$ 量子ビットが両方 1 の場合だけ第 3 量子ビットの $0,1$ を反転させるという、CNOT ゲートの制御ビットが 2 つに増えたようなゲートになっています。制御ビットが 2 つあるという意味で、CCNOT ゲートと呼ばれることもあります。

CNOT のように一つの式で表すと

$$CCX\,|abc\rangle = |ab, (a\cdot b)\oplus c\rangle\quad (a,b,c\in\{0,1\}) \tag{2.3}$$

となります。特に、$c = 0$ の場合には積 ab を計算するゲートと思うことができます。さらに 0 を偽、1 を真に対応させると $a \cdot b$ は「a かつ b」(AND, $\wedge$) なので、AND を計算するゲートと思うこともできます。

CNOT ゲートと同じように、制御ビットの場所を明示したい場合は $CCX_{12,3}$ や $CCX_{1,2\to 3}$ のように書きます。例えば、第 $2,3$ 量子ビットを制御ビットにする場合は $CCX_{2,3\to 1}$ となります。

2.5.6 S,T ゲート、位相ゲート

ここまで紹介してきた量子ゲートはすべて 2 回作用させると元に戻るという性質を持っています。ここでは、そうではない量子ゲートである S, T ゲートと位相ゲートを紹介しましょう。

まずは、位相ゲートに必要な指数関数 $\exp(\sqrt{-1}x)$ を定義します。

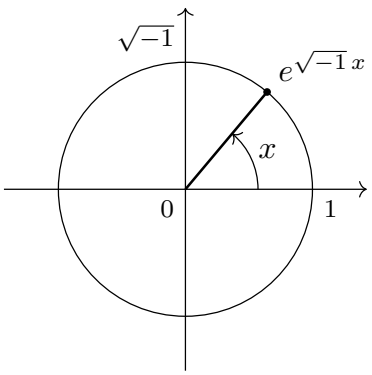
定義 2.5.4 (純虚数に対する指数関数)**.** $x \in \mathbb{R}$ に対して

$$e^{\sqrt{-1}x} := \cos(x) + \sqrt{-1}\sin(x)$$

と定義する。一般に e^z を $\exp(z)$ とも表す。

実際には、複素数 $z = x + \sqrt{-1}y$ に対して $e^z = e^x e^{\sqrt{-1}y}$ という式で指数関数の定義をさらに拡張できますが、本テキストでは使わないのでこれ以上深入りはしません。

定義式から、$e^{\sqrt{-1}x}$ は複素数平面で単位円周上にあり、実軸の正の向きとのなす角が x(ラジアン) になっています。特に、三角関数の性質 $\cos^2(x) + \sin^2(x) = 1$ から $|e^{\sqrt{-1}x}| = 1$ が成り立ちます。



普通の指数関数 e^x は指数法則 $e^{x+y} = e^x e^y$ を満たしますが、$e^{\sqrt{-1}x}$ も同様の性質を満たします。証明は両辺を三角関数の加法定理を使って比較するだけなので省略します。

定理 2.5.5 (指数法則)**.** $x, y \in \mathbb{R}$ とすると、$e^{\sqrt{-1}(x+y)} = e^{\sqrt{-1}x} e^{\sqrt{-1}y}$ となる。

位相ゲート (位相シフトゲート、位相回転ゲートなど) は $\theta \in \mathbb{R}$ に対して、

$$R(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{\sqrt{-1}\theta} \end{pmatrix}$$

という行列で定義されます。θ を決めるごとに一つの量子ゲートが定まるので、位相ゲートと呼ばれるゲートは無限個存在します。$R(\theta)$ を $|0\rangle, |1\rangle$ に作用させると

$$R(\theta)|0\rangle = |0\rangle, \quad R(\theta)|1\rangle = e^{\sqrt{-1}\theta}|1\rangle$$

となり、$|1\rangle$ を絶対値 1 の複素数倍する量子ゲートになっています。Z ゲートと同様、$R(\theta)$ を作用させるだけでは測定には何の影響も与えません。文献によっては

$$P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{\sqrt{-1}\theta} \end{pmatrix}, \quad R_z(\theta) = \begin{pmatrix} e^{-\sqrt{-1}\theta/2} & 0 \\ 0 & e^{\sqrt{-1}\theta/2} \end{pmatrix} = e^{-\sqrt{-1}\theta/2} P(\theta)$$

という記号を用いることも多いので注意してください。

特別な θ に対する $R(\theta)$ には重要なものがいくつかあり固有の名前がついています。すでに定義した Z ゲートと、S,T ゲートです:

$$Z = R(\pi) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad S = R(\pi/2) = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{-1} \end{pmatrix}, \quad T = R(\pi/4) = \begin{pmatrix} 1 & 0 \\ 0 & e^{\sqrt{-1}\pi/4} \end{pmatrix}.$$

$e^{\sqrt{-1}\pi/4}$ は 1 の 8 乗根で $(e^{\sqrt{-1}\pi/4})^2 = \sqrt{-1}$ を満たします。よって、

$$S^2 = Z, \quad T^2 = S$$

が成り立ちます。この関係式のため $S = \sqrt{Z}, T = \sqrt{S} = \sqrt[4]{Z}$ と表されることもあります。また、$S^4 = Z^2 = I, T^8 = S^4 = I$ となっています。ほかにも、

$$\begin{gathered} SXS^{-1} = Y, \quad SYS^{-1} = -X, \quad SZS^{-1} = Z, \\ R(\theta_1)R(\theta_2) = R(\theta_1 + \theta_2) = R(\theta_2)R(\theta_1) \quad (\forall \theta_1, \theta_2 \in \mathbb{R}) \end{gathered} \tag{2.4}$$

という関係式が成り立ちます。特に、Z, S, T は可換 ($ZS = SZ, ST = TS, \ldots$) になります。

2.5.2 節で Z ゲートはある種の z 軸中心 π 回転を意味すると述べましたが、位相ゲート $R(\theta)$ も z 軸中心の θ 回転という意味があります。これを踏まえると $SXS^{-1} = Y, SYS^{-1} = -X$ という関係式は、z 軸中心に $\pi/2$ 回転すると、x 軸中心の回転と y 軸中心の回転が入れ替わると解釈できます。

2.5.7　テンソル積

ここまで扱った量子ゲートは $1, 2, 3$ 量子ビットに対する操作でした。ゲートを組み合わせて複雑な操作を実現するために、n 量子ビットの特定のビットに X,Y,Z のような量子ゲートを作用さ

せる、という操作を定義します。ここで量子ビットをテンソル積を使って定義したことが活きてきます。

X ゲートは 1 量子ビットに対する操作で、$|0\rangle$ と $|1\rangle$ を入れ替えるものでした。同じような操作で、2 量子ビットの 1 番目の $0, 1$ を入れ替え、2 番目には何もしないというものを考えてみましょう。対応する線形写像を U と置いて、基底の行き先を考えると

$$\begin{aligned} U|00\rangle &= |10\rangle = (X|0\rangle) \otimes |0\rangle, \\ U|01\rangle &= |11\rangle = (X|0\rangle) \otimes |1\rangle, \\ U|10\rangle &= |00\rangle = (X|1\rangle) \otimes |0\rangle, \\ U|11\rangle &= |01\rangle = (X|1\rangle) \otimes |1\rangle \end{aligned}$$

となります。このような U の存在は命題 2.5.2 によって保障されていることに注意しましょう。

U の定義は、$a, b \in \{0, 1\}$ を使って、まとめて

$$|ab\rangle \mapsto (X|a\rangle) \otimes (I|b\rangle)$$

と表せます。さらに、$|v\rangle, |w\rangle \in \mathbb{C}^2$ という重ね合わせの状態に対しても、線形性を使って計算すると

$$U(|v\rangle \otimes |w\rangle) = (X|v\rangle) \otimes (I|w\rangle)$$

という式が成り立ちます。つまり、線形写像 U は、第 1 量子ビットには X で作用して、第 2 量子ビットには恒等写像 I で作用するようなものであると分かりました。

同様の方法で、例えば第 1 量子ビットには Z で作用して第 $2, 3$ 量子ビットには CX で作用するような線形写像が存在することなども分かります。一般には以下の線形写像のテンソル積を使って定式化することができます。

定理 2.5.6. $T\colon (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes n}$ と $S\colon (\mathbb{C}^2)^{\otimes m} \to (\mathbb{C}^2)^{\otimes m}$ を線形写像とする。このとき、

$$U(\boldsymbol{v} \otimes \boldsymbol{w}) = T(\boldsymbol{v}) \otimes S(\boldsymbol{w}) \quad (\forall \boldsymbol{v} \in (\mathbb{C}^2)^{\otimes n}, \boldsymbol{w} \in (\mathbb{C}^2)^{\otimes m})$$

を満たす線形写像 $U\colon (\mathbb{C}^2)^{\otimes n+m} \to (\mathbb{C}^2)^{\otimes n+m}$ がただ一つ存在する。ただし、条件式中の $\otimes$ は定義 2.3.1 で与えたものである。

証明. $(\mathbb{C}^2)^{\otimes n+m}$ の元は $\boldsymbol{v} \otimes \boldsymbol{w}$ という形の元の和で表されるので、条件を満たす U は存在すれば一つしかない。存在することを示そう。

$(\mathbb{C}^2)^{\otimes n}$ と $(\mathbb{C}^2)^{\otimes m}$ に対する定義 2.2.2 の条件 3 の基底を、それぞれ $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{2^n}$ と $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{2^m}$ と置く。$\boldsymbol{v}_i \otimes \boldsymbol{w}_j$ $(1 \leq i \leq 2^n,\ 1 \leq j \leq 2^m)$ を並べたものは $(\mathbb{C}^2)^{\otimes n+m}$ の基底になるので、命題 2.5.2 より、

$$U(\boldsymbol{v}_i \otimes \boldsymbol{w}_j) = T(\boldsymbol{v}_i) \otimes S(\boldsymbol{w}_j) \quad (1 \leq \forall i \leq 2^n, 1 \leq \forall j \leq 2^m) \tag{2.5}$$

を満たす線形写像 $U\colon (\mathbb{C}^2)^{\otimes n+m} \to (\mathbb{C}^2)^{\otimes n+m}$ が存在する。U が定理の条件を満たしていることを示せばよい。

$\boldsymbol{v} \in (\mathbb{C}^2)^{\otimes n}, \boldsymbol{w} \in (\mathbb{C}^2)^{\otimes m}$ とする。基底の定義 (定義 2.1.5) から $\boldsymbol{v}, \boldsymbol{w}$ は先ほどの基底の線形結合で

$$\boldsymbol{v} = \sum_{i=1}^{2^n} a_i \boldsymbol{v}_i, \quad \boldsymbol{w} = \sum_{j=1}^{2^m} b_j \boldsymbol{w}_j$$

と表せる。よって、

$$\begin{aligned}
U(\boldsymbol{v} \otimes \boldsymbol{w}) &= U\left(\left(\sum_{i=1}^{2^n} a_i \boldsymbol{v}_i\right) \otimes \left(\sum_{j=1}^{2^m} b_j \boldsymbol{w}_j\right)\right) \\
&= U\left(\sum_{i,j} a_i b_j \boldsymbol{v}_i \otimes \boldsymbol{w}_j\right) && (\otimes \text{ の双線形性}) \\
&= \sum_{i,j} a_i b_j U(\boldsymbol{v}_i \otimes \boldsymbol{w}_j) && (U \text{ の線形性}) \\
&= \sum_{i,j} a_i b_j T(\boldsymbol{v}_i) \otimes S(\boldsymbol{w}_j) && (2.5) \text{ より} \\
&= \left(\sum_i a_i T(\boldsymbol{v}_i)\right) \otimes \left(\sum_j b_j S(\boldsymbol{w}_j)\right) && (\otimes \text{ の双線形性}) \\
&= T(\boldsymbol{v}) \otimes S(\boldsymbol{w}) && (T, S \text{ の線形性})
\end{aligned}$$

となり、確かに U は欲しい条件を満たしている。 □

定義 2.5.7. 定理 2.5.6 で得られた線形写像 U を $T \otimes S$ と表し、T と S のテンソル積と呼ぶ。

命題 2.5.8. $A, B\colon (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes n}$ と $C, D\colon (\mathbb{C}^2)^{\otimes m} \to (\mathbb{C}^2)^{\otimes m}$ を線形写像とする。このとき、以下が成り立つ。

(1) $(A \otimes C)(B \otimes D) = AB \otimes CD$
(2) $(A \otimes I)(I \otimes C) = A \otimes C = (I \otimes C)(A \otimes I)$
(3) A, C が全単射である場合、$(A \otimes C)^{-1} = A^{-1} \otimes C^{-1}$

ここで、I は適切な集合 ($(\mathbb{C}^2)^{\otimes n}$ または $(\mathbb{C}^2)^{\otimes m}$) 上の恒等写像とする。

証明. (1) を示せば (2),(3) はその特別な場合として示せるので、(1) を示す。$\boldsymbol{v} \in (\mathbb{C}^2)^{\otimes n}, \boldsymbol{w} \in (\mathbb{C}^2)^{\otimes m}$ とする。線形性から、両辺の $\boldsymbol{v} \otimes \boldsymbol{w}$ での値が等しいことを示せばよい。実際、

$$(A \otimes C)(B \otimes D)(\boldsymbol{v} \otimes \boldsymbol{w}) = (A \otimes C)(B\boldsymbol{v} \otimes D\boldsymbol{w}) = AB\boldsymbol{v} \otimes CD\boldsymbol{w}$$

$$(AB \otimes CD)(\boldsymbol{v} \otimes \boldsymbol{w}) = AB\boldsymbol{v} \otimes CD\boldsymbol{w}$$

となり、$(A \otimes C)(B \otimes D) = AB \otimes CD$ が分かる。 □

2 つの線形写像のテンソル積を定義しましたが、三つ以上は $(S \otimes T) \otimes U$ のように 2 つのテンソル積を繰り返すことで定義できます。$(S \otimes T) \otimes U$ と $S \otimes (T \otimes U)$ のように括弧を付ける順序が複数ありますが、どれも同じ線形写像になることが示せます。したがって、以下では括弧を付けず $S \otimes T \otimes U$ などと表すことにしましょう。

例えば、$X \otimes I$ は第 1 量子ビットの $0, 1$ を反転させるゲート、$I \otimes X$ は第 2 量子ビットの $0, 1$ を反転させるゲート、$I \otimes I \otimes X = I^{\otimes 2} \otimes X$ は 3 量子ビットの 3 番目の $0, 1$ を反転させるゲート、となります。より一般に、$I^{\otimes a-1} \otimes X \otimes I^{\otimes n-a}$ は n 量子ビットの a 番目だけに X を作用させるゲートになります。以上で特定のビットのみに X, Y, Z, H などの量子ゲートを作用させるという操作が、線形写像のテンソル積で実現できることが分かりました。

具体的な計算例として、$|00\rangle$ からスタートして量子ゲートを作用させて複雑に「もつれた」状態を作れることをみましょう。$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ だったことを思い出して、$|00\rangle$ に $H \otimes I$ を作用させると

$$\begin{aligned}(H \otimes I)\,|00\rangle &= (H\,|0\rangle) \otimes |0\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \\ &= \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)\end{aligned}$$

となります。これに $CX_{1\to 2}$ を作用させると、第 1 量子ビットが 1 の場合に第 2 量子ビットが反転するので

$$\begin{aligned}CX_{1\to 2} \cdot (H \otimes I)\,|00\rangle &= \frac{1}{\sqrt{2}}(CX_{1\to 2}\,|00\rangle + CX_{1\to 2}\,|10\rangle) \\ &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\end{aligned}$$

となります。2.4 節の最後でみたように、この状態は $|v\rangle \otimes |w\rangle$ という形では書けず、第 1 量子ビットと第 2 量子ビットが互いに関係しあった状態になっています。

2.6　量子回路モデルと量子回路図

これで量子計算モデルの一つである量子回路モデルを定義する準備ができました。本来ならどのような実装や理論を考えるかで使ってよい量子ゲートの集合を固定しますが、本テキストでは基本的な量子ゲートはすべて使ってよいものと単純化しておきます。具体的には 2.5 節で述べた

X,Y,Z ゲート、Hadamard ゲート、CNOT ゲート、Toffoli ゲート、位相ゲート、それから恒等写像 I を含めたこれらゲートのテンソル積です。

定義 2.6.1 (量子回路モデル)**. 量子回路モデル**における計算とは、初期状態 (n 量子ビットの状態 $|0\rangle^{\otimes n}$) を準備し、2.5 節で述べた量子ゲートを有限個作用させ、最後に測定を行い測定値を得る一連の操作をいう。

量子回路全体の初期状態は基本的に $|0\rangle^{\otimes n}$ としますが、回路の途中やアルゴリズムを説明する際にはそれ以外の状態を入力として考えることもあります。その場合、どのような状態を入力とするかは都度、説明を与えるか、以下で紹介する量子回路図で明示することにします。

X, Y, Z, H, CX, CCX は 2 回行うと元に戻るので、写像としては全単射になっています。また、位相ゲートも $R(\theta)R(-\theta) = I$ という関係式から全単射であることが分かります。したがって、量子回路モデルで行える量子ビットの操作は、測定を除けばすべて可逆になっており、逆の操作で戻すことができます。特に、例えば

$$U|ij\rangle = |i0\rangle \quad (\forall i, j \in \{0, 1\})$$

のような、第 2 量子ビットを強制的に 0 にする線形写像 U は単射ではないので、量子ゲートの組み合わせで実現できません。量子ゲートによる操作が可逆であるというのも量子回路モデルの大きな特徴の一つです。

命題 2.6.2. 量子回路モデルにおいて、初期状態 $|0\rangle^{\otimes n}$ に量子ゲートの列 $U_1, \ldots, U_r$ を順に作用させるとする。

(1) 線形写像 $U_r \cdots U_1$ は全単射[*6]である。
(2) 最終状態 $U_r \cdots U_1 |0\rangle^{\otimes n}$ も n 量子ビットの状態である。

証明. 上で述べたとおり基本的な量子ゲートはすべて全単射である。よって、それらの合成である $U_r \cdots U_1$ も全単射である。また各 U_i の入力と出力は同じ量子ビット数を持つので $U_r \cdots U_1 |0\rangle^{\otimes n}$ も n 量子ビットの状態である。 □

アルゴリズムを線形写像を並べた式で表現するのは厳密な計算や証明には適していますが、直感的な全体を見渡すような理解には適していません。量子アルゴリズムを視覚的に表現する**量子回路図**を紹介しましょう。まず、前節の最後に計算した 2 量子ビットに $H \otimes I$ と $CX_{1\to 2}$ を作用させる操作は以下の回路図で表現できます。

[*6] 続編でユニタリという、より強い性質が成り立つことをみます。

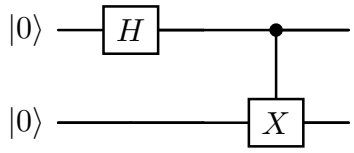


図 2.1: $CX_{1\to 2}\cdot(H\otimes I)\,|00\rangle$ を表す回路図

図は文章と同じように左から右の順でみていきます。横線一本が 1 量子ビットを表していて、上から第 1、第 2... 量子ビットという順番になっています。図 2.1 の場合、入力の $|00\rangle$ が左端に書かれています。1 量子ビットに作用する量子ゲートは X, Y, Z, H などの線形写像を表す記号を四角で囲んで表現します。例えば、始めに Hadamard ゲートを第 1 量子ビットに作用させているので、一番上の横線に量子ゲートを表す記号である、四角で囲んだ H を書いています。その後に CNOT ゲートを第 1 量子ビットを制御ビットとして作用させています。CNOT や Toffoli ゲートの制御ビットは黒丸で表し、標的ビットの場所に X ゲートと同じ記号を書きます。

図 2.2 は入力が $|v\rangle\otimes|00\rangle$ (ただし $|v\rangle$ は 2 量子ビットの状態) で、第 $1, 4$ 量子ビットを制御ビット、第 2 量子ビットを標的ビットにして Toffoli ゲートを作用させています。CNOT や Toffoli ゲートの標的部分をどのように図示するかにはいくつか書き方があり、例えば図 2.3 のようなものがあります。本テキストでは X ゲートと同じように書くことにします。

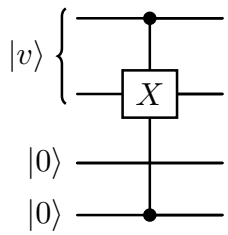


図 2.2: $CCX_{1,4\to 2}(|v\rangle\otimes|00\rangle)$ を表す回路図

図 2.3: CNOT の標的ビットの表記

出力を明示的に書く場合は、量子ビットを表す線の右に出力を書きます。図 2.4 は、入力 $|01\rangle$ に $X\otimes Z$ を作用させると結果が $|1\rangle\otimes(-|1\rangle)$ になるという図です。一般には入力や出力が $|v\rangle\otimes|w\rangle$ のように量子ビットごとに分けて表せないことがあるので、いつでも図 2.4 のように入出力を書き込めるというわけではありません。

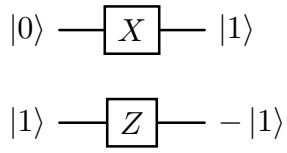


図 2.4: $(X \otimes Z)\,|01\rangle = |1\rangle \otimes (-\,|1\rangle)$ を表す回路図

X, Y, Z などのよく使われる量子ゲート以外にも、それらを組み合わせて作った線形写像や、その場限りで定義された線形写像を量子ゲートとして使うこともあります。その場合には線形写像の記号をそのまま四角の中に書きます。図 2.5 は量子ゲート U を $1, 2$ 番目に、量子ゲート U' を $3, 4$ 番目に作用させています。$U \otimes I$ と $I \otimes U'$ は作用させる順番を交換しても、あるいは同時に作用させて $U \otimes U'$ としても同じ結果になるので、縦に並べて書いています。

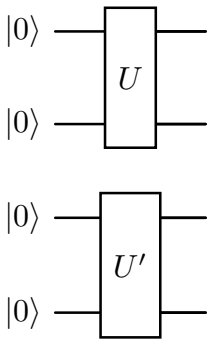


図 2.5: $(U \otimes U')\,|0000\rangle$ を表す回路図

図 2.6 は測定を表す記法です。今のところ部分測定は扱っておらず、必ず最後にすべての量子ビットを測定するので書く意味はあまりありません。続編で部分測定を扱うときに詳しく解説します。

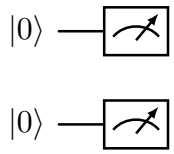


図 2.6: 測定

第 3 章

量子回路モデルで何ができるか

この章では量子回路モデルに慣れるため、整数の和や量子ビットの複製を例に、どんな操作ができ、あるいはできないのかを解説します。また、古典的なコンピュータで計算可能なものは、量子回路モデルでも計算可能であることを示します。

3.1 簡単な操作

前章で量子回路モデルを定義しました。しかし、それだけではいったい何が計算できるのか、Shor のアルゴリズムのような複雑な計算ができるのか、直感的には分かりづらいと思います。この節ではビットの入れ替えや足し算のような操作が、基本的な量子ゲートの組み合わせで実現できることをみます。ただし、3.2 節や 3.5 節で述べるような量子ビット特有の性質のために、工夫が必要な部分も出てきます。

3.1.1 量子ビットの入れ替え

2 量子ビットの 1 番目と 2 番目の量子ビットを入れ替える操作は CNOT を 3 回使用した図 3.1 の回路で実現できます。例えば、$|01\rangle$ に作用させると

$$
\begin{aligned}
|01\rangle &\xmapsto{CX_{1\to 2}} |01\rangle && \text{(第 1 量子ビットが 0 なのでそのまま)} \\
&\xmapsto{CX_{2\to 1}} |11\rangle && \text{(第 2 量子ビットが 1 なので第 1 量子ビットを反転)} \\
&\xmapsto{CX_{1\to 2}} |10\rangle && \text{(第 1 量子ビットが 1 なので第 2 量子ビットを反転)}
\end{aligned}
$$

となり、確かに 1 番目と 2 番目の量子ビットを入れ替えています。同様に、基底のほかの元に作用させた場合は、

$$
|00\rangle \xmapsto{CX_{1\to 2}} |00\rangle \xmapsto{CX_{2\to 1}} |00\rangle \xmapsto{CX_{1\to 2}} |00\rangle ,
$$

$$|10\rangle \xmapsto{CX_{1\to2}} |11\rangle \xmapsto{CX_{2\to1}} |01\rangle \xmapsto{CX_{1\to2}} |01\rangle,$$
$$|11\rangle \xmapsto{CX_{1\to2}} |10\rangle \xmapsto{CX_{2\to1}} |10\rangle \xmapsto{CX_{1\to2}} |11\rangle$$

となります。等式で式変形するなら

$$CX_{1\to2}CX_{2\to1}CX_{1\to2}|11\rangle = CX_{1\to2}CX_{2\to1}|10\rangle = CX_{1\to2}|10\rangle = |11\rangle$$

のようになります。

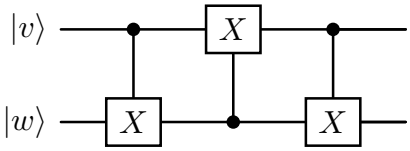


図 3.1: 第 1 量子ビットと第 2 量子ビットを入れ替える回路

この線形写像を $\mathrm{SWAP}_{1,2}$ と表すと、$a|0\rangle + b|1\rangle, c|0\rangle + d|1\rangle \in \mathbb{C}^2$ に対して

$$\begin{aligned}\mathrm{SWAP}_{1,2}((a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle)) &= \mathrm{SWAP}_{1,2}(ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle) \\ &= ac|00\rangle + ad|10\rangle + bc|01\rangle + bd|11\rangle \\ &= (c|0\rangle + d|1\rangle) \otimes (a|0\rangle + b|1\rangle)\end{aligned}$$

となり確かに 1 番目と 2 番目の量子ビットを入れ替えています。したがって、より一般に $|v_1\rangle \otimes |w_1\rangle + \cdots + |v_r\rangle \otimes |w_r\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$ という元に対しても線形性から

$$\mathrm{SWAP}_{1,2}(|v_1\rangle \otimes |w_1\rangle + \cdots + |v_r\rangle \otimes |w_r\rangle) = |w_1\rangle \otimes |v_1\rangle + \cdots + |w_r\rangle \otimes |v_r\rangle$$

となり、量子ビットの順序を入れ替えることができています。

最も量子ビット数が少ない場合で計算しましたが、n 量子ビットでも考え方は何も変わりません。例えば、n 量子ビットの 1 番目と n 番目を入れ替える場合は

$$\mathrm{SWAP}_{1,n} = CX_{1\to n}CX_{n\to1}CX_{1\to n}$$

という線形写像になります。複数ペアの量子ビットを入れ替える場合でも SWAP を複数作用させるだけで交換できます。

定理 3.1.1. n 量子ビットの i 番目と j 番目を入れ替える操作は三つの CNOT ゲートで実現できる。

3.1.2　制御ゲートにする

CX と表された CNOT ゲートは第 1 量子ビットが 0 なら何もせず、1 なら第 2 量子ビットを反転させる (=X を作用させる) という操作でした。そのため、CNOT ゲートは、制御ビット付き

の X ゲートという意味で制御 X ゲートのように呼ばれることもあります。同様に、Y, Z, H, S, T ゲートなどを制御ビット付きにしたものを考えてみます。このような制御ビット付きのゲートを基本的なゲートに含めてしまうという方法もありますが、ここでは既存の量子ゲートの組み合わせで作ってみましょう。

CNOT の X の部分を Z に置き換えたもの、つまり第 1 量子ビットが 1 のときのみ第 2 量子ビットに Z を作用させるゲート CZ を考えます。このゲートは図 3.2 の回路で実現することができます。式で表すと

$$CZ = (I \otimes H)CX_{1\to 2}(I \otimes H)$$

となります。$a \in \{0, 1\}$ とし、$Z = HXH, H^2 = I$ という関係式があったことを思い出して計算してみましょう。$CX_{1\to 2}$ は第 1 量子ビットが 0 の場合は $I \otimes I$ になり、1 の場合は $I \otimes X$ になると考えると、

$$\begin{aligned}
CZ\,|0a\rangle &= (I \otimes H)CX_{1\to 2}(I \otimes H)\,|0a\rangle \\
&= (I \otimes H)(I \otimes I)(I \otimes H)\,|0a\rangle \\
&= (I \otimes H^2)\,|0a\rangle \\
&= |0a\rangle && (H^2 = I), \\
CZ\,|1a\rangle &= (I \otimes H)CX_{1\to 2}(I \otimes H)\,|1a\rangle \\
&= (I \otimes H)(I \otimes X)(I \otimes H)\,|1a\rangle \\
&= (I \otimes HXH)\,|1a\rangle \\
&= |1\rangle \otimes (Z\,|a\rangle) && (HXH = Z)
\end{aligned}$$

となり、確かに第 1 量子ビットが 1 の場合だけ第 2 量子ビットに Z が掛けられています。あるいは、直接的に

$$\begin{aligned}
CZ\,|1a\rangle &= (I \otimes H)CX_{1\to 2}(I \otimes H)\,|1a\rangle \\
&= (I \otimes H)CX_{1\to 2}(|1\rangle \otimes (H\,|a\rangle)) \\
&= (I \otimes H)(|1\rangle \otimes (XH\,|a\rangle)) \\
&= |1\rangle \otimes (HXH\,|a\rangle) \\
&= |1\rangle \otimes (Z\,|a\rangle)
\end{aligned}$$

としても計算できます。図 3.2 のように CZ も CX と同じような記法で表し、この記法は制御ビットをもつ量子ゲート全般にも用いることにします。

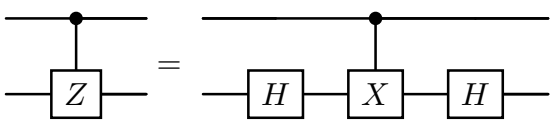


図 3.2: CZ の記号と回路

同様に、(2.4) で述べた関係式 $SXS^{-1} = Y$ を使うと、CY は図 3.3 の回路図で実現できます。ここで S^{-1} に関しては、$S^{-1} = S^3$ または $S^{-1} = R(3\pi/2)$ を使えば既存の量子ゲートで実現できます。実現の仕方は一通りではなく、ほかにも $ZX = \sqrt{-1}Y$ という関係式を使う方法もあります。

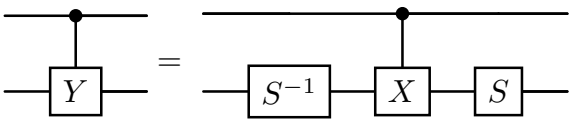


図 3.3: CY の回路

$\theta \in \mathbb{R}$ として、$R(\theta)$ の制御ビット付きのゲート $CR(\theta)$ を作ってみましょう。

$$XR(-\theta/2)X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{-\sqrt{-1}\theta/2} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} e^{-\sqrt{-1}\theta/2} & 0 \\ 0 & 1 \end{pmatrix}$$

という式から $R(\theta/2)XR(-\theta/2)X = e^{-\sqrt{-1}\theta/2}R(\theta)$ が分かります。ここから任意の $a \in \{0,1\}$ に対して

$$\begin{aligned} &((I \otimes R(\theta/2))CX_{1\to 2}(I \otimes R(-\theta/2))CX_{1\to 2})(|0\rangle \otimes |a\rangle) = |0\rangle \otimes |a\rangle\,, \\ &((I \otimes R(\theta/2))CX_{1\to 2}(I \otimes R(-\theta/2))CX_{1\to 2})(|1\rangle \otimes |a\rangle) = e^{-\sqrt{-1}\theta/2}\,|1\rangle \otimes R(\theta)\,|a\rangle \end{aligned}$$

となります。第 1 量子ビットが 1 の場合のスカラー倍 $e^{-\sqrt{-1}\theta/2}$ を消せれば $CR(\theta)$ と一致します。よって、最後に $R(\theta/2) \otimes I$ を作用させれば $CR(\theta)$ が得られます。まとめると図 3.4 のような回路図になります。

図 3.4 では一見、$R(\theta/2)$ が第 1 量子ビットに作用して制御ビットが変化してしまっているように見えます。しかし、この操作は第 1 量子ビットが 1 の場合に第 2 量子ビットを $e^{\sqrt{-1}\theta/2}$ 倍するという操作と等しくなっており、制御ビット付きのスカラー倍とみなせます:

$$\begin{aligned} &R(\theta/2)(|0\rangle \otimes |a\rangle) = |0\rangle \otimes |a\rangle\,, \\ &R(\theta/2)(|1\rangle \otimes |a\rangle) = e^{\sqrt{-1}\theta/2}\,|1\rangle \otimes |a\rangle = |1\rangle \otimes e^{\sqrt{-1}\theta/2}\,|a\rangle \quad (\forall a \in \{0,1\}). \end{aligned}$$

$\otimes$ の双線形性から、スカラー倍は全体に掛けてもどの成分に掛けても同じ結果になることに注意してください。

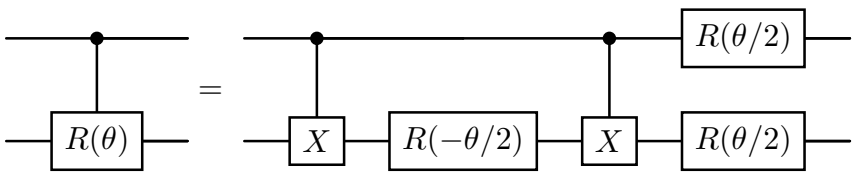


図 3.4: $CR(\theta)$ の回路

制御 Hadamard ゲートは少し複雑ですが、$H = SHTXT^{-1}HS^{-1}$ という関係式を使うことで同様に実現できます。この関係式から

$$CH = (I \otimes SHT)CX_{1\to 2}(I \otimes T^{-1}HS^{-1})$$

となります。細かい計算は省略します。関係式は $H = (SHT)X(SHT)^{-1}$ と書き直せ、H と X が相似な行列であることを意味しています。

3.1.3 制御ビットを増やす

制御ビットの個数は CNOT が一つ、Toffoli ゲートが二つでした。これを三つ、四つと増やせるかという自然な疑問が思い浮かびます。この問題について考えてみましょう。

三つの制御ビットがある場合、つまり $1,2,3$ 番目のビットがすべて 1 なら 4 番目のビットを反転させるという操作を考えます。4 量子ビットを与えると 4 量子ビットを返す線形写像を作りたいのですが、それだと少し複雑なので一時的に計算結果を保存しておくための 1 量子ビットを追加します。このような作業用の余分な量子ビットは**補助ビット**、英語で **ancilla** (アンシラ) と呼ばれます。例えば、足し算を筆算で計算するとき、繰上りを一時的に保存しておく必要がありますが、そのような情報を保持する場所として補助ビットを使います。

4 量子ビットに補助ビット一つ追加して 5 量子ビットで考え、$a,b,c,d \in \{0,1\}$ に対して

$$\begin{aligned} C^3X(|abc\rangle \otimes |d\rangle \otimes |0\rangle) &= \begin{cases} |abc\rangle \otimes |d\rangle \otimes |0\rangle & (a\cdot b\cdot c = 0) \\ |abc\rangle \otimes |d\oplus 1\rangle \otimes |0\rangle & (a\cdot b\cdot c = 1) \end{cases} \\ &= |abc\rangle \otimes |d \oplus (abc)\rangle \otimes |0\rangle \end{aligned}$$

を満たす線形写像 C^3X を作りましょう。ここで、$\oplus$ は $0\oplus 0 = 1\oplus 1 = 0,\ 1\oplus 0 = 0\oplus 1 = 1$ を満たす和でした。$a\cdot b\cdot c$ と $d\oplus(abc)$ の abc は普通の整数の積です。つまり、C^3X は $a=b=c=1$ のときだけ d の $0,1$ を反転しています。

結論から言うと、C^3X は図 3.5 の回路で実現できます。実際に $a,b,c,d \in \{0,1\}$ として、(2.3) を思い出し計算すると

$$\begin{aligned} CCX_{3,5\to 4}CCX_{1,2\to 5}(|abc\rangle \otimes |d\rangle \otimes |0\rangle) &= CCX_{3,5\to 4}(|abc\rangle \otimes |d\rangle \otimes |a\cdot b\rangle) \\ &= |abc\rangle \otimes |d\oplus(abc)\rangle \otimes |a\cdot b\rangle\,, \\ CCX_{1,2\to 5}(|abc\rangle \otimes |d\oplus(abc)\rangle \otimes |a\cdot b\rangle) &= |abc\rangle \otimes |d\oplus(abc)\rangle \otimes |0\rangle \end{aligned}$$

となり、欲しい性質を満たしてくれています。最初に $CCX_{1,2\to 5}$ を作用させて補助ビットに $a\cdot b$ を書き込んで、次の $CCX_{3,5\to 4}$ でその計算結果を利用して $a\cdot b\cdot c$ を第 4 量子ビットに足しています。この二つのゲートだけで計算できているように見えますが、補助ビットに計算過程の残骸が残っているので、最後の $CCX_{1,2\to 5}$ で補助ビットを 0 にリセットしています。

リセットすることで補助ビットを再利用できるという面もありますが、リセットしないとより本質的な問題が生じます。最後のステップの必要性については 3.3 節で詳しく扱います。

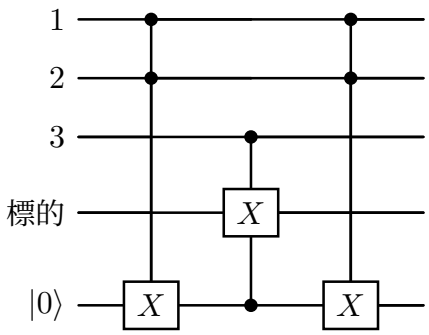


図 3.5: C^3X の回路

C^3X を一般化して C^nX というゲートを作りましょう。C^4X は 1〜4 番目のビットが 1 のときだけ、5 番目のビットを反転します。C^4X を C^3X と同じように作ろうとすると、補助ビットが途中で足りなくなってしまいます。補助ビットをさらに増やすというのが簡単な解決策で、C^3X のアイデアをそのまま適用すれば C^4X も作れます。ここでは、そうではなく補助ビット一つで計算する方法を紹介します。

まずは図 3.6 の回路を考えます。$|a_1a_2a_3a_4a_5\rangle \otimes |0\rangle$ に各 Toffoli ゲートを作用させたあと量子ビットがどのように変化するかを計算してみましょう。第 4, 5 量子ビットは関係ないので省略すると

	第 1	第 2	第 3	補助ビット (第 6)
	a_1	a_2	a_3	0
$\xrightarrow{CCX_{2,3\to6}}$	a_1	a_2	a_3	a_2a_3
$\xrightarrow{CCX_{1,6\to2}}$	a_1	$a_2 \oplus a_1a_2a_3$	a_3	a_2a_3
$\xrightarrow{CCX_{2,3\to6}}$	a_1	$a_2 \oplus a_1a_2a_3$	a_3	$a_1a_2a_3$

となります。$\oplus$ は同じものを足すと 0 になることに注意してください。制御ビットである第 2 量子ビットが変化してしまっていますが、補助ビットに $a_1a_2a_3$ が計算できています。後は C^3X の場合と同じように、$CCX_{4,6\to5}$ を作用させれば、$a_1, \ldots, a_4$ が 1 の場合だけ a_5 が反転します。

	第 1	第 2	第 3	第 4	標的 (第 5)	補助ビット (第 6)
	a_1	$a_2 \oplus a_1a_2a_3$	a_3	a_4	a_5	$a_1a_2a_3$
$\xrightarrow{CCX_{4,6\to5}}$	a_1	$a_2 \oplus a_1a_2a_3$	a_3	a_4	$a_5 \oplus (a_1a_2a_3a_4)$	$a_1a_2a_3$

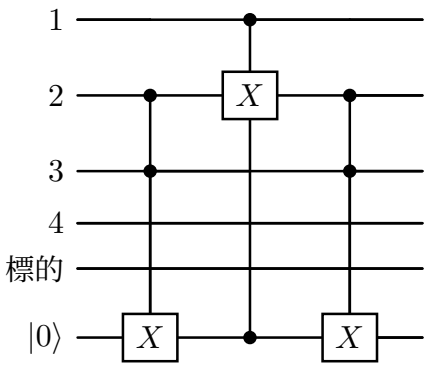


図 3.6: C^4X の回路の前半

第 2 量子ビットと補助ビットが変化してしまっているので元の値に戻してあげましょう。量子ゲートは全単射な線形写像なので、逆の操作を行ってやれば完全に元に戻すことができます。ただ、完全に初期状態に戻したいわけではなく第 5 量子ビットだけは変化させたままにしたいという状況です。ここで、最後に行った $CCX_{4,6\to5}$ では第 5 量子ビットだけが変化してそれ以外は変化していないこと、逆に図 3.6 の回路では第 5 量子ビットは無関係であること、を使いましょう。この状況なら、図 3.6 の逆の操作を行えば、第 5 量子ビットを動かさずほかのビットを初期状態に戻せるというわけです。

同じ Toffoli ゲートを 2 連続で作用させると元に戻るので、Toffoli ゲートの逆写像は Toffoli ゲートそのものです。つまり、単純に回路の左右を反転すれば逆の操作になります。図 3.6 の回路は左右対称なので反転しても変わりません。したがって、図 3.7 の回路図で C^4X が実現できます。

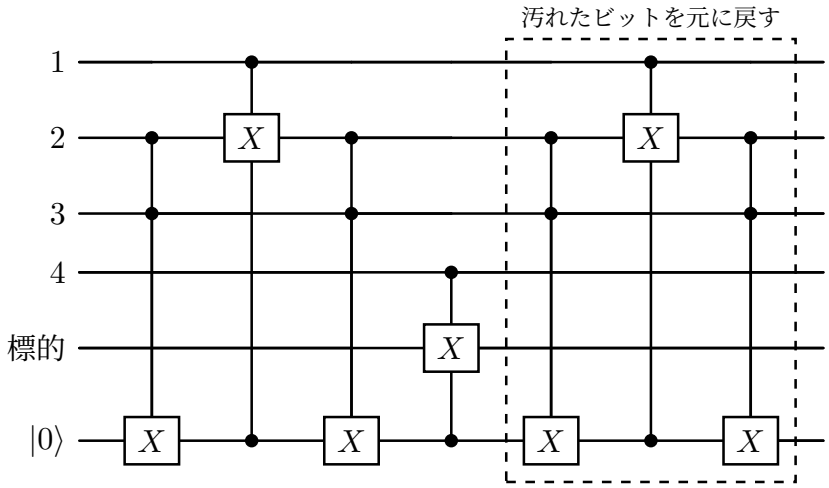


図 3.7: C^4X の回路の全体

C^4X を作るときには補助ビット一つでは足りず、制御ビットの一部を補助ビットのように使

いました。本来変化しないはずのビットを汚してしまっても、計算後に逆の操作を行うことで元に戻せるというトリックが肝でした。$C^5X, C^6X, \ldots, C^nX$ と制御ビットを増やしていっても、Toffoli ゲートだけを使って大体 n に比例する個数のゲート数と補助ビット一つで実現できます。

定理 3.1.2. C^nX は Toffoli ゲートと補助ビット一つを用いて実現でき、必要なゲートの個数は n の一次関数で抑えられる。

3.1.4　足し算

3.4 節で述べるように、量子回路モデルでは古典的なコンピュータで計算できることは同様に計算できます。ここでは、その具体例として足し算を計算する回路を考えてみます。

2 進法で 2 桁の整数二つの足し算ができれば、それを拡張していくことで任意の桁の足し算が行えます。そのため、2 桁同士の足し算だけを扱うことにします。計算結果は最大で $11 + 11 = 110$ の 3 桁ですが、拡張することを見越し 3 桁目は繰り上がりとして別枠で扱います。

問題を量子ビットを使って定式化しておきましょう。初期状態を $|a_2a_1\rangle \otimes |b_2b_1\rangle \otimes |00\rangle$ として、

$$U(|a_2a_1\rangle \otimes |b_2b_1\rangle \otimes |00\rangle) = |a_2a_1\rangle \otimes |b_2'b_1'\rangle \otimes |c_20\rangle\,,$$
$$c_2b_2'b_1' = a_2a_1 + b_2b_1, \quad (2 \text{ 進法})$$

となるような線形写像 U を基本的な量子ゲートの積で作ります。a_2a_1, b_2b_1 が入力で、それを足した結果 $c_2b_2'b_1'$ を $3, 4, 5$ 番目のビットに上書きします。最後のビットは 1 桁目の繰り上がりを一時的に保存しておくための補助ビットで、最終的には 0 にリセットされます。例えば $a_2a_1 = 10, b_2b_1 = 11$ の場合、$c_2b_2'b_1' = 101$ なので、

$$U(|10\rangle \otimes |11\rangle \otimes |00\rangle) = |10\rangle \otimes |01\rangle \otimes |10\rangle$$

となります。

まずは繰り上がりだけを計算して、そのあと大きな桁から和を計算します。この順番にすることで、補助ビットを 0 にリセットする処理が分かりやすくなります。繰り上がりの計算は 1 桁目でも 2 桁目でも基本的に変わらないのでまとめて考えましょう。$|x\rangle \otimes |a\rangle \otimes |b\rangle \otimes |0\rangle$ を初期状態とします。x が前の桁からの繰り上がり、a, b は a_i, b_i の添え字を省略したもので、最後のビットに $a + b + x$ の繰り上がりを保存します。

繰り上がりが発生するのは、

- a, b の両方が 1
- a, b の片方のみが 1 かつ $x = 1$

という 2 パターンあり、両方が同時に成立することはありません。前者は Toffoli ゲートを使えば検出できるので、後者について考えます。a, b の片方のみが 1 という条件は $a \oplus b = 1$ という条件

に書き換えられるので、CNOT ゲートで $a \oplus b$ を計算しておけば、$a \oplus b = 1$ かつ $x = 1$ という条件は Toffoli ゲートで検出できます。まとめると、図 3.8 が繰り上がりを計算する回路図になります。$a \oplus b$ の結果を保持しておくために一度第 3 量子ビットを $a \oplus b$ に変更し、計算が終わった後に元の値に戻しています。1 桁目の場合だけは前の桁の繰り上がりがない (実質 $x = 0$) ので、最初の Toffoli ゲートだけが必要でそれ以降は不要になります。

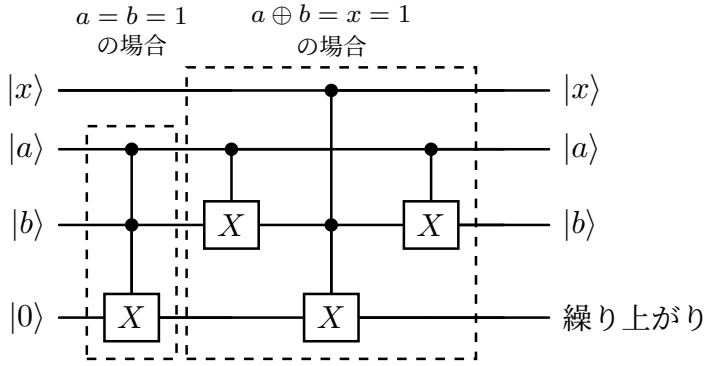


図 3.8: 繰り上がりの計算回路

次は各桁の和を計算する処理です。先ほどと同様に初期状態を $|x\rangle \otimes |a\rangle \otimes |b\rangle$ とします。x が前の桁からの繰り上がり、a, b は a_i, b_i の添え字を省略したものです。$a + b + x$ の 1 桁目が計算したいものですが、これは $a \oplus b \oplus x$ と等しくなります。$\oplus$ は CNOT ゲートで計算できるので、図 3.9 のように二つ組み合わせれば $a \oplus b \oplus x$ が計算できます。実際に計算してみると

$$CX_{1\to3}CX_{2\to3}\,|x, a, b\rangle = CX_{1\to3}\,|x, a, a \oplus b\rangle = |x, a, a \oplus b \oplus x\rangle$$

となります。

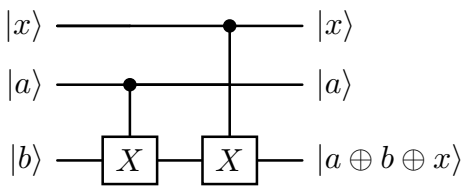


図 3.9: 1 桁の和の回路

以上の量子ゲートを組み合わせて 2 桁の整数 a_2a_1, b_2b_1 の和を計算する回路を作ります。回路図が大きくなって分かりづらいので、どのように組み合わせるかだけを述べましょう。$|a_2a_1\rangle \otimes |b_2b_1\rangle \otimes |00\rangle$ が初期状態でした。最後の二つのビットに c_2, c_1 と名前を付けておきます。

(1) 図 3.8 の回路で $a_1 + b_1$ の繰り上がりを c_1 に計算する
(2) 図 3.8 の回路で $a_2 + b_2 + c_1$ の繰り上がりを c_2 に計算する

(3) 図 3.9 の回路で $a_2 + b_2 + c_1$ の 1 桁目を b_2 のビットに上書きする
(4) (1) の回路を左右逆[*1]にして c_1 を 0 にリセットする
(5) 図 3.9 の回路で $a_1 + b_1$ の 1 桁目を b_1 のビットに上書きする

(5) で第 4 量子ビット (b_1 の場所) が変化してしまうので、変化する前に c_1 を 0 にリセットする処理 (4) を挟んでいます。同様に回路を増やしていけば n 桁同士の和を計算する回路も構成できます。この方法だと、2 進法で n 桁同士の和を計算するためには、繰り上がりを管理する n 個の補助ビットが必要になります。

最後に、重ね合わせの状態

$$\begin{aligned}|v\rangle &= \frac{1}{\sqrt{2}}(|01\rangle \otimes |01\rangle + |10\rangle \otimes |10\rangle) \otimes |00\rangle \\ &= \frac{1}{\sqrt{2}}|01\rangle \otimes |01\rangle \otimes |00\rangle + \frac{1}{\sqrt{2}}|10\rangle \otimes |10\rangle \otimes |00\rangle\end{aligned}$$

を初期状態として和を計算するとどうなるかをみておきましょう。和を計算する回路から得られる線形写像を T としておきます。T は線形写像なので、$|v\rangle = |v_1\rangle + |v_2\rangle$ となっているなら $T|v\rangle = T|v_1\rangle + T|v_2\rangle$ が成り立ちます。したがって、

$$T|v\rangle = \frac{1}{\sqrt{2}}|01\rangle \otimes |10\rangle \otimes |00\rangle + \frac{1}{\sqrt{2}}|10\rangle \otimes |00\rangle \otimes |10\rangle$$

となります。一つ目の項は $01+01=10$ を、二つ目の項は $10+10=100$ を意味しており、2 通りの足し算を並列に計算できているようにみえます。一方、この状態を測定すると、$|01\rangle \otimes |10\rangle \otimes |00\rangle$ と $|10\rangle \otimes |00\rangle \otimes |10\rangle$ が等確率で出力され、$01+01$ か $10+10$ の結果がランダムに得られます。量子ビットの状態なら並列に計算できるということは正しいですが、実際に私たちが知ることのできる測定値は、単純にそのまま測定した場合、並列計算した結果のどれかがランダムに出力されることに注意してください。

3.2　量子もつれ状態

量子特有の現象の一つに量子もつれというものがあります。すでに命題 2.4.4 などでみたように、二つ以上の量子ビットが非自明に関係しあっているという現象です。まずは 2 量子ビットの場合に定義しましょう。

定義 3.2.1 (2 量子ビットの量子もつれ)**.** 2 量子ビットの状態は、$|v\rangle \otimes |w\rangle$ ($|v\rangle, |w\rangle \in \mathbb{C}^2$) と表せないとき、**量子もつれ状態** (**エンタングル状態**, entangled state) であるといわれる。あるいは、その状態はもつれている、エンタングルしているなどという。

[*1] 1 桁目なので実際には Toffoli ゲート一つしかなく、反転の必要はありません。

例 3.2.2.

$$\frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle), \quad \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

は前者は量子もつれ状態ではないが、後者は量子もつれ状態である。

命題 2.4.4 で示したように、2 量子ビットの状態 $|u\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$ が $|u\rangle = |v\rangle \otimes |w\rangle$ と表せているなら、$|u\rangle$ を測定した結果は確率を込めて $|v\rangle$ を測定した結果と $|w\rangle$ を測定した結果を並べたものに等しくなります。つまり、もつれていない 2 量子ビットの状態は、1 番目と 2 番目のビットがある意味独立していると思えます。逆に、量子もつれ状態であれば、1 番目と 2 番目のビットの間には非自明な関係があるといえます。先ほどの例 $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ の場合、測定すると $00, 11$ が等確率で出力されますが、測定値の 1 ビット目と 2 ビット目が常に等しいという関係があります。

命題 3.2.3. $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$ が量子もつれ状態でないことと、$ad - bc = 0$ は同値である。

証明. まず、$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$ が量子もつれ状態でないと仮定して $ad - bc = 0$ を示す。仮定から、

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle)$$

となる $\alpha, \beta, \gamma, \delta \in \mathbb{C}$ が存在する。右辺を展開して係数を比較すると $a = \alpha\gamma, b = \alpha\delta, c = \beta\gamma, d = \beta\delta$ となり、$ad - bc = \alpha\gamma\beta\delta - \alpha\delta\beta\gamma = 0$ を得る。

$ad - bc = 0$ を仮定して逆を示そう。量子ビットの状態の定義から a, b, c, d の少なくとも一つは 0 ではない。$a \neq 0$ の場合を考える。必要ならスカラー倍を行って $a = 1$ としてよい。このとき $d = bc$ となるので、

$$\begin{aligned} a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle &= |00\rangle + b|01\rangle + c|10\rangle + bc|11\rangle \\ &= (|0\rangle + c|1\rangle) \otimes (|0\rangle + b|1\rangle) \end{aligned}$$

となり、量子もつれ状態ではないことが分かる。

$a = 0$ の場合、$bc = ad = 0$ なので、b または c の少なくとも一方は 0 である。$b = 0$ の場合は、

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle = c|10\rangle + d|11\rangle = |1\rangle \otimes (c|0\rangle + d|1\rangle)$$

となり、$c = 0$ の場合も同様に

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle = (b|0\rangle + d|1\rangle) \otimes |1\rangle$$

となりいずれも量子もつれ状態ではない。 □

命題 3.2.3 の帰結として、ほとんどの $\mathbb{C}^2 \otimes \mathbb{C}^2$ の元が量子もつれ状態であることが分かります。ランダムに a, b, c, d を選んだ場合に $ad - bc = 0$ が成り立つことはほとんどないからです。数学的には、$\mathbb{C}^2 \otimes \mathbb{C}^2$ の元はテンソル積 $\boldsymbol{v} \otimes \boldsymbol{w}$ で表される方がまれというわけです。

量子もつれ状態かどうかは 1 量子ビットの量子ゲートでは変化しません。実際、可逆な 2×2 行列 U に対して

$$(U \otimes I)(|v\rangle \otimes |w\rangle) = (U\,|v\rangle) \otimes |w\rangle\,,$$
$$(U^{-1} \otimes I)(|v\rangle \otimes |w\rangle) = (U^{-1}\,|v\rangle) \otimes |w\rangle$$

となるので、量子もつれ状態か否かは量子ゲート U を作用させる前後で一致します。特に、もつれていない状態に対して 1 量子ビットに作用する量子ゲートを繰り返すだけでは、第 1 および第 2 量子ビットをそれぞれ独立に操作しているのと変わりなく、2 量子ビットを扱っている意味がなくなります。そのため、CNOT や Toffoli ゲートは量子もつれ状態を作り出せるという意味でも重要になります。実際、CNOT ゲートは 1 量子ビットに作用する量子ゲートから作ることはできません。

命題 3.2.4. CNOT ゲートは 2×2 行列のテンソル積 $A \otimes B$ では表せない。

証明. もつれていない状態 $\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$ に CNOT ゲートを作用させると量子もつれ状態になる:

$$CX_{1\to 2}\left(\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)\right) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

もし CNOT ゲートが $A \otimes B$ の形で表されるなら、命題の直前の議論から、量子もつれ状態か否かはゲートを作用させる前後で変わらない。これは矛盾なので、CNOT ゲートは 2×2 行列のテンソル積 $A \otimes B$ では表せない。 □

量子もつれ状態では、第 1 量子ビットへの操作と第 2 量子ビットへの操作が一致することがありえます。$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ を例にいくつか計算してみましょう。

$$\begin{aligned}(X \otimes I) \cdot \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) &= \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) \\ &= (I \otimes X) \cdot \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \end{aligned} \tag{3.1}$$

$X \otimes I$ と $I \otimes X$ を作用させるとどちらも同じ状態になります。第 1 量子ビットに X ゲートを作用させたにもかかわらず、第 2 量子ビットに X ゲートを作用させたようにもみえるのです。X ゲート以外でも例えば Z ゲートと Hadamard ゲートなら

$$(Z \otimes I) \cdot \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

$$
\begin{aligned}
&= (I \otimes Z) \cdot \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \\
(H \otimes I) \cdot \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) &= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |1\rangle \right) \\
&= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle) \\
&= (I \otimes H) \cdot \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)
\end{aligned}
$$

と同様の性質が成り立ちます。ほかの量子もつれ状態でも同じことが成り立つかというと、そういうわけでないので注意してください。

$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ はこのように特に良い性質を持つ量子もつれ状態で、**Bell 状態 (EPR ペア)** という特別な名前がついています。ほかにも

$$
\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \quad \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \quad \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)
$$

も Bell 状態と呼ばれます。これらは二つの量子ビットが最大限にもつれ合った状態とみなせ、量子情報理論の様々な場所で重要な役割を果たします。

定義 3.2.5 (量子もつれ状態)**.** $n+m$ 量子ビットの状態を前半 n 量子ビットと後半 m 量子ビットに分けて考えている状況とする。$n+m$ 量子ビットの状態 $|v\rangle$ が $|v_1\rangle \otimes |v_2\rangle$ $(v_1 \in (\mathbb{C}^2)^{\otimes n}, v_2 \in (\mathbb{C}^2)^{\otimes m})$ と表せないとき、$|v\rangle$ は**量子もつれ状態**であるといわれる。

2 より大きなビット数の場合には、量子ビットをどのように分割して考えているかによって量子もつれ状態かどうかが変わります。つまり、$|v\rangle$ の前半 n 量子ビットと後半 m 量子ビットがもつれた状態という意味で使っていることに注意してください。例えば、$\frac{1}{\sqrt{2}}|0\rangle \otimes (|00\rangle + |11\rangle)$ は 1 量子ビット +2 量子ビットだと思うともつれておらず、2 量子ビット +1 量子ビットだと思うともつれています。$\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ はどのような分割をしても量子もつれ状態になっています。

3.3 量子ビットを減らす、増やす

3.1.3 節で作った Toffoli ゲートの制御ビットを一つ増やした量子ゲート C^3X や 3.1.4 節で作った和を計算する回路では、入力を管理するための量子ビットのほかに、補助ビットと呼ばれる計算途中の情報を管理するための量子ビットが必要でした。複雑な計算を行う過程で C^3X や和の回路を使いたいと思うと、一時的に量子ビットを増やして和などの計算が終わったら余分な量子ビットを減らす、ということをしたくなります。例えば図 3.10 のように複雑な計算の途中に C^3X を使いたい場合、C^3X に使われる補助ビットは本質的ではないので省略して書くのが自然に思えます。

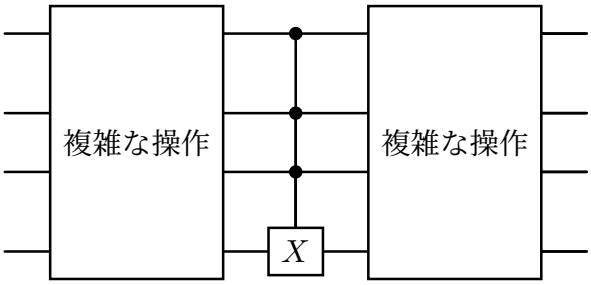


図 3.10: C^3X の補助ビットの横線は省略したい

前提として、定義 2.6.1 の量子回路モデルでは初期状態の量子ビットの数と量子ゲートを作用させた後の量子ビットの数は一致していなければいけません。したがって、途中で一時的に量子ビットを増やしたり減らしたりするのは本来やってはいけない操作になります。しかし、それでは複雑なアルゴリズムを記述するのに不便なので、量子ビットを実質的に増やしたり減らしたりできるということを述べましょう。

3.3.1　量子ビットを増やす

量子ビットを増やす操作、つまり

$$(\mathbb{C}^2)^{\otimes n} \ni |v\rangle \mapsto |v\rangle \otimes |0\rangle \in (\mathbb{C}^2)^{\otimes n+1}$$

という写像ですが、これを実現するのは簡単です。計算途中で付け加えたのではなく、初めから $|0\rangle$ という補助ビットがあったと思えばよいのです。図 3.11 のように C^3X 以外の計算で使われなかったとしても、最初から用意していれば途中で追加する必要はなくなります。また、複雑な計算の途中で補助ビットを使っていたとしても、C^3X の直前で $|0\rangle$ にリセットされていれば、そのまま補助ビットとして再利用できます。

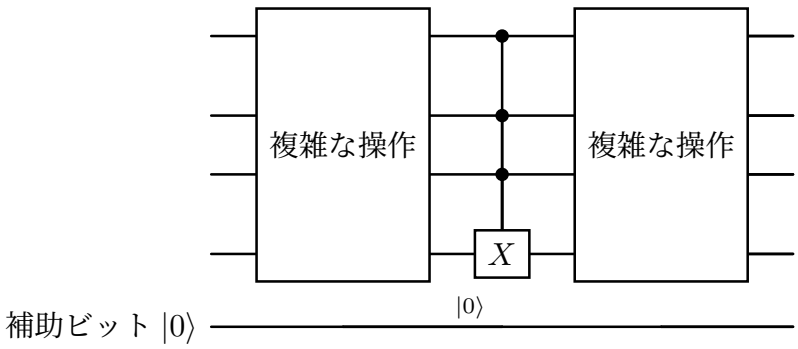


図 3.11: 補助ビットは最初からあったと思う

3.3.2 量子ビットを減らす

量子ビットを減らしたいと思った場合には、量子もつれ状態が問題になります。もし、$|v\rangle \otimes |w\rangle$ のようなもつれていない状態であれば、$|w\rangle$ を無視して $|v\rangle$ の部分だけに量子ゲートを作用させ続ければ、実質的に $|w\rangle$ を消したことと同じになります。最後に測定するときも、命題 2.4.4 を使えば、全体を測定して $|w\rangle$ の部分のビットを無視することで $|v\rangle$ を単体で測定した結果が得られます。

量子もつれ状態の場合にどのようになるかを考えてみます。$|v_1\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$ と $|v_2\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ を考えます。第 1 量子ビットが何かしらの計算結果で、第 2 量子ビットは補助ビットとしてその計算に用いたという想定です。$|v_1\rangle$ は補助ビットを 0 にリセットしたもの、$|v_2\rangle$ はリセットしていないものになります。$|v_1\rangle, |v_2\rangle$ の測定値の 1 ビット目はどちらも $0, 1$ が等確率で出力され、$|v_1\rangle$ と $|v_2\rangle$ で違いはありません。ここに $H \otimes I$ を作用させると、

$$
\begin{aligned}
(H \otimes I)\,|v_1\rangle &= |00\rangle\,, \\
(H \otimes I)\,|v_2\rangle &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)
\end{aligned}
$$

となります。$(H \otimes I)\,|v_1\rangle$ を測定すると 1 ビット目は必ず 0 が出力されますが、$(H \otimes I)\,|v_2\rangle$ を測定すると 1 ビット目は $0, 1$ が等確率で出力されます。補助ビットという余計な部分ともつれ合っているために、想定とは違う振る舞いをしていることが分かります。

ここまでの回路の例で補助ビットを必ず 0 にリセットしていたのはこのような事情があったからです。計算後に 0 にリセットしておけば、それ以後の計算ではリセットした補助ビットはないものとして扱ってよいということになります。以上で、途中計算で補助ビットを追加し計算後に不要な補助ビットを削除する、という操作が計算後に補助ビットと量子もつれ状態にならないという前提の下正当化されました。以降は、途中計算で補助ビットが必要になる場合でも図 3.10 のように省略して書くことにします。

3.4 古典と量子

Shor のアルゴリズムの手順 3 では x, N を固定したうえで、$f(j) = x^j \bmod N$ という関数を量子回路モデルで計算する必要がありました。3.1.4 節で具体的に構成したように、足し算は基本的な量子ゲートの組み合わせで実現できます。少し複雑になりますが、同様に四則演算 $+ - \times /$ や剰余 mod も計算することができます。人の手で書き下すのは難しいですが、それらを組み合わせれば f を計算する回路も作ることができます。

より一般に、古典的な普通のコンピュータで計算できるような関数 f は量子回路モデルでも近い効率で計算することができます。古典的な計算モデルについてはまったく扱っていなかったので概略になりますが、この結果について述べておきましょう。古典的なコンピュータなどで計算

可能な関数は多くのビットと AND と NOT という演算を組み合わせて実現できることが知られています。足し算が CNOT, Toffoli ゲートという限定的な量子ゲートのみを使って作れたことを考えると、それほど不思議なことではないでしょう。したがって、AND と NOT という演算が量子回路モデルで実現できれば、古典的な意味で計算可能な関数は量子回路モデルで計算可能だと分かります。

$a, b \in \{0, 1\}$ に対して単なる整数としての積 $a \cdot b$ を与える操作が AND です。Toffoli ゲートそのものがこの AND を計算する量子ゲートになっています。実際、Toffoli ゲートの定義から、任意の $a, b \in \{0, 1\}$ に対して

$$CCX_{1,2\to 3} |ab0\rangle = |ab, a \cdot b\rangle$$

となります。次に NOT ですが、これは $a \in \{0, 1\}$ に対して $a \oplus 1 (= 1 - a)$ を与える操作になります。$0, 1$ を反転させる操作なので X ゲートがまさにこの操作そのものです。

古典的な意味で計算可能な関数 f を量子回路に翻訳する方法を述べましょう。f としては $\{0, 1, \ldots, 2^n - 1\}$ から $\{0, 1, \ldots, 2^m - 1\}$ への写像のみを扱います。整数を 2 進法で表せば f は n ビットから m ビットへの写像と思えます。量子回路へ変換したいので、どのような線形写像が欲しいかを定式化しておきましょう。f は n ビットから m ビットへの写像なので、$a \in \{0, 1\}^n$ に対して

$$|a\rangle \otimes |0\rangle^{\otimes m} \mapsto |a\rangle \otimes |f(a)\rangle$$

を対応させるような線形写像が欲しいと考えます。しかし、これでは多くの場合うまくいきません。なぜなら、3.1.4 節での足し算の繰り上がりのように、途中計算で必要な情報を補助ビットに覚えておく必要があるからです。したがって、正確には十分たくさんの補助ビットを付け加えて

$$U_f(|a\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes r}) = |a\rangle \otimes |f(a)\rangle \otimes |0\rangle^{\otimes r}$$

となるような線形写像 U_f を求めることになります。

f を求める計算はいくつかのビットと AND と NOT を組み合わせて実行することができるので、そのまま十分多くの補助ビットと Toffoli ゲートと X ゲートに置き換えれば量子回路が得られます。対応する線形写像を U'_f と表しましょう。U'_f が欲しい U_f になるかというとそういうわけでもありません。何故なら、古典的な計算をそのまま翻訳しただけでは、補助ビットに計算途中に保存した不要な情報が残ってしまっているからです。これは式で、

$$U'_f(|a\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes r}) = |a\rangle \otimes |f(a)\rangle \otimes \underbrace{|g(a)\rangle}_{\text{計算過程の残骸}} \qquad (\forall a \in \{0, 1\}^n)$$

と表せます。$|g(a)\rangle$ を $|0\rangle^{\otimes r}$ に戻したいのですが、無条件に強制的に 0 にリセットすることは量子ゲートではできません[*2]。これを解決する一般的な手法が Bennett's trick という名前で知られ

[*2] 量子ゲートは全単射なので複数の状態を一斉に $|0\rangle$ に送るような線形写像は実現できません。

ています。

Bennett's trick のアイデアは 3.1.3 節ですでに使っています。計算結果を一旦退避させておき、量子回路の逆の操作を行うと元に戻るという性質を使って、退避した結果以外を元に戻す、というものです。補助ビットをさらに m 個増やして

$$\begin{aligned}
|a\rangle \otimes \underbrace{|0\rangle^{\otimes m}}_{\text{計算結果用}} \otimes \underbrace{|0\rangle^{\otimes r}}_{\text{計算過程用}} \otimes \underbrace{|0\rangle^{\otimes m}}_{\text{退避用}} &\xmapsto{U'_f \otimes I^{\otimes m}} |a\rangle \otimes |f(a)\rangle \otimes |g(a)\rangle \otimes |0\rangle^{\otimes m} \\
&\xmapsto{\text{CNOT でコピー}} |a\rangle \otimes |f(a)\rangle \otimes |g(a)\rangle \otimes |f(a)\rangle \\
&\xmapsto{(U'_f)^{-1} \otimes I^{\otimes m}} |a\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes r} \otimes |f(a)\rangle \\
&\xmapsto{\text{ビットの入れ替え}} |a\rangle \otimes |f(a)\rangle \otimes |0\rangle^{\otimes r} \otimes |0\rangle^{\otimes m}
\end{aligned}$$

という手順を行えば、計算結果を保持したまま補助ビットを 0 にリセットできます。計算結果をコピーする部分ですが、例えば前 2 ビットを後ろ 2 ビットにコピーすることは、

$$CX_{2\to 4} CX_{1\to 3}(|xy\rangle \otimes |00\rangle) = |xy\rangle \otimes |xy\rangle \quad (\forall x, y \in \{0,1\})$$

とすればできるので、一般には CNOT を m 個使って計算結果を補助ビットにコピーすることができます。

以下の定理では補助ビットの個数 r に、先ほど追加した退避用の m 個分も含めて記述しています。あとの定理 3.4.3 も同様です。

定理 3.4.1. 古典的なコンピュータで計算可能な関数 $f\colon \{0,1\}^n \to \{0,1\}^m$ に対して、X ゲート、CNOT ゲート、Toffoli ゲートで実現できる線形写像 $U_f\colon (\mathbb{C}^2)^{\otimes n+m+r} \to (\mathbb{C}^2)^{\otimes n+m+r}$ が存在して

$$U_f(|a\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes r}) = |a\rangle \otimes |f(a)\rangle \otimes |0\rangle^{\otimes r} \quad (\forall a \in \{0,1\}^n)$$

が成り立つ。ここで、最後の r 量子ビットは十分多くの補助ビットである。また、U_f を実現する量子回路に必要な量子ゲートの個数は、f を計算するために必要な AND, NOT の個数の定数倍で抑えられる。

系 3.4.2. 定理 3.4.1 の U_f はさらに

$$U_f(|a\rangle \otimes |b\rangle \otimes |0\rangle^{\otimes r}) = |a\rangle \otimes |b \oplus f(a)\rangle \otimes |0\rangle^{\otimes r} \quad (\forall a \in \{0,1\}^n, b \in \{0,1\}^m)$$

を満たすように取れる。ここで、$b \oplus f(a)$ の $\oplus$ は各ビットごとに $\oplus$ を行う演算である。(例えば $110 \oplus 100 = 010$ となる。)

証明. U'_f から U_f を作った手順を以下のように変更すればよい:

$$|a\rangle \otimes |b\rangle \otimes |0\rangle^{\otimes r} \otimes |0\rangle^{\otimes m} \xmapsto{\text{ビットの入れ替え}} |a\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes r} \otimes |b\rangle$$

$$\xmapsto{U'_f \otimes I^{\otimes m}} |a\rangle \otimes |f(a)\rangle \otimes |g(a)\rangle \otimes |b\rangle$$
$$\xmapsto{\text{CNOT で } \oplus \text{ を計算}} |a\rangle \otimes |f(a)\rangle \otimes |g(a)\rangle \otimes |b \oplus f(a)\rangle$$
$$\xmapsto{(U'_f)^{-1} \otimes I^{\otimes m}} |a\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes r} \otimes |b \oplus f(a)\rangle$$
$$\xmapsto{\text{ビットの入れ替え}} |a\rangle \otimes |b \oplus f(a)\rangle \otimes |0\rangle^{\otimes r} \otimes |0\rangle^{\otimes m}. \qquad \square$$

3.1.4 節で足し算の回路を作ったときには、大体 $|a\rangle \otimes |b\rangle \mapsto |a\rangle \otimes |a+b\rangle$ のような線形写像を構成しました。a, b が和を取る対象です。この足し算の回路では、入力 b が保持されていた量子ビットに、計算結果を上書きしたのでした。一方で、この節で作った U_f だと $U_f(|a,b\rangle \otimes |0\rangle) = |a,b\rangle \otimes |a+b\rangle$ のように計算結果を保持するためのビットを追加する必要がありました。この差を埋める方法を紹介しましょう。

定理 3.4.3. $f, g\colon \{0,1\}^{n+m} \to \{0,1\}^m$ を古典的なコンピュータで計算可能な関数とし、

$$g(a, f(a,b)) = b \quad (\forall a \in \{0,1\}^n, b \in \{0,1\}^m)$$

が成り立つと仮定する。このとき、X ゲート、CNOT ゲート、Toffoli ゲートで実現できる線形写像 $\widetilde{U}_f\colon (\mathbb{C}^2)^{\otimes n+m+r} \to (\mathbb{C}^2)^{\otimes n+m+r}$ が存在して

$$\widetilde{U}_f(|a\rangle \otimes |b\rangle \otimes |0\rangle^{\otimes r}) = |a\rangle \otimes |f(a,b)\rangle \otimes |0\rangle^{\otimes r} \quad (\forall a \in \{0,1\}^n, b \in \{0,1\}^m)$$

が成り立つ。ここで、最後の r 量子ビットは十分多くの補助ビットである。また、$\widetilde{U}_f$ を実現する量子回路に必要な量子ゲートの個数は、f, g を計算するために必要な AND, NOT の個数の定数倍で抑えられる。

証明. f, g に対して定理 3.4.1 の U_f, U_g を取る。ただし、r は U_f, U_g の両方が実現できるように大きく取る。このとき、$\widetilde{U}_f$ は以下の合成写像で実現できる:

$$|a\rangle \otimes |b\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes r} \xmapsto{U_f} |a\rangle \otimes |b\rangle \otimes |f(a,b)\rangle \otimes |0\rangle^{\otimes r}$$
$$\xmapsto{\text{ビットの入れ替え}} |a\rangle \otimes |f(a,b)\rangle \otimes |b\rangle \otimes |0\rangle^{\otimes r}$$
$$\xmapsto{(U_g)^{-1}} |a\rangle \otimes |f(a,b)\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes r}. \qquad \square$$

定理 3.4.3 に複雑な条件が課されていますが、これは量子ゲートが可逆であるためです。作りたい写像

$$|a\rangle \otimes |b\rangle \mapsto |a\rangle \otimes |f(a,b)\rangle$$

が量子回路で実現できるためには、この対応が単射である必要があります。したがって、例えば $f(a,b)$ が恒等的に 0 になるような関数はこの形式では実現できません。定理 3.4.3 では、g という f の逆写像のようなものの存在を保証することで、先ほどの写像の単射性も保証しています。

例 3.4.4. N を正の整数、x, y を N と互いに素な整数とし、$xy \bmod N = 1$ と仮定する。m を $2^m \geq N$ となるように取る。このとき、以下の関数 f, g は定理 3.4.3 の条件を満たすように拡張できる。

(1) $f(a,b) = (b+a) \bmod N$, $g(a,b) = (b-a) \bmod N$ $(0 \leq a, b \leq N-1,\ n = m)$
(2) $f(b) = (xb) \bmod N$, $g(b) = (yb) \bmod N$ $(0 \leq b \leq N-1,\ n = 0)$

f, g は $\{0, 1, \ldots, 2^n - 1\} \times \{0, 1, \ldots, 2^m - 1\}$ 上の関数になっていなければいけませんが、定義されていない部分では $f(a,b) = b, g(a,b) = b$ のように拡張すれば条件を満たします。(2) では $n = 0$ となっていますが、f, g に a の部分がなくてもよいという意味です。

実際には、ここで述べた翻訳を機械的に行うと補助ビットが非常に多くなったり非効率的な部分がでてきます。そのため、実用上はさらに何らかの最適化が必要になることが多いでしょう。

3.5 量子ビットはコピーできない

古典的なコンピュータでは計算途中の値やファイルや写真のようなデータをコピーできます。一方、量子回路モデルでは量子ビットをコピーすることはできません。したがって、例えば測定前に量子ビットをコピーして、測定値が悪い値だったらコピーを再度測定するといったことはできません。量子ビットをコピーできないことを証明しておきましょう。

そもそも量子ビットをコピーするといった場合に、何を指すのかを定式化しておきます。非負整数 r と線形写像 U が存在して、任意の n 量子ビットの状態 $|v\rangle$ に対して

$$U(|v\rangle \otimes |0\rangle^{\otimes n} \otimes |0\rangle^{\otimes r}) = |v\rangle \otimes |v\rangle \otimes |0\rangle^{\otimes r}$$

が成り立つとき、n 量子ビットはコピーできる、ということにします[*3]。ここで、$|0\rangle^{\otimes r}$ は補助ビットで、コピーの計算時に補助ビットを使ってもよいという設定になっています。$|v\rangle \otimes |v\rangle$ という状態が得られれば、一つ目の $|v\rangle$ に様々な量子ゲートを作用させたとしても二つ目の $|v\rangle$ は一切影響を受けずそのままなので、確かにコピーと思ってよさそうです。

「任意の $|v\rangle$」という条件は重要で、n 量子ビットの状態 $|v\rangle$ が与えられたときに、$|v\rangle \otimes |v\rangle$ という状態を作ること自体はできることがあります。もし、$|v\rangle$ の作り方が具体的に分かっている場合、つまり $|v\rangle = T|0\rangle^{\otimes n}$ となる線形写像 T が量子回路で具体的に与えられている場合には、

$$(I \otimes T)(|v\rangle \otimes |0\rangle^{\otimes n}) = |v\rangle \otimes |v\rangle$$

とできます。これは $|v\rangle$ を作った計算を繰り返すことで同じ状態を並べており、いわゆるコピーという操作とは違います。コピーするといった場合には、$|v\rangle$ がどんな形であっても同じ方法で行わ

[*3] 厳密には $|v\rangle$ に $\||v\rangle\| = 1$ という条件を課すべきですが、まだノルムを定義していないので省略します。定理 3.5.1 の証明はこの条件があっても正しい議論になっています

れるべきだからです。

定理 3.5.1. n 量子ビットはコピーできない。より詳細に、r をいくら大きくしても、

$$\begin{aligned} U(|0\rangle \otimes |0\rangle \otimes |0\rangle^{\otimes r}) &= |0\rangle \otimes |0\rangle \otimes |0\rangle^{\otimes r}, \\ U(|1\rangle \otimes |0\rangle \otimes |0\rangle^{\otimes r}) &= |1\rangle \otimes |1\rangle \otimes |0\rangle^{\otimes r}, \\ U\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \otimes |0\rangle^{\otimes r}\right) &= \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes |0\rangle^{\otimes r} \end{aligned} \tag{3.2}$$

を満たす線形写像 U は存在しない。

証明. n 量子ビットはコピーができるなら 1 量子ビットのコピーもできるので、定理の最後の主張を示せば十分である。(3.2) を満たす r と U が存在すると仮定して矛盾を導く。U は線形写像なので、

$$\begin{aligned} U((|0\rangle + |1\rangle) \otimes |0\rangle \otimes |0\rangle^{\otimes r}) &= U(|0\rangle \otimes |0\rangle \otimes |0\rangle^{\otimes r}) + U(|1\rangle \otimes |0\rangle \otimes |0\rangle^{\otimes r}) \\ &= |0\rangle \otimes |0\rangle \otimes |0\rangle^{\otimes r} + |1\rangle \otimes |1\rangle \otimes |0\rangle^{\otimes r} \end{aligned}$$

となる。しかし、仮定から

$$\begin{aligned} U\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \otimes |0\rangle^{\otimes r}\right) &= \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes |0\rangle^{\otimes r} \\ &\neq \frac{1}{\sqrt{2}} |0\rangle \otimes |0\rangle \otimes |0\rangle^{\otimes r} + \frac{1}{\sqrt{2}} |1\rangle \otimes |1\rangle \otimes |0\rangle^{\otimes r} \end{aligned}$$

となるので矛盾である。 □

量子ビットをコピーできないことが分かりましたが、古典的なコピーに対応する操作はどうなっているでしょうか。1 量子ビットの場合を考えてみます。古典的なコピーとは、単に $0, 1$ をコピーするということです。よって、

$$\begin{aligned} CX_{1\to 2}(|0\rangle \otimes |0\rangle) &= |0\rangle \otimes |0\rangle, \\ CX_{1\to 2}(|1\rangle \otimes |0\rangle) &= |1\rangle \otimes |1\rangle \end{aligned}$$

のように CNOT ゲートが対応する操作になります。重ね合わせの状態を「コピー」してみると

$$CX_{1\to 2}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

となり量子もつれ状態になります。(3.1) でみたように、第 1 量子ビットに量子ゲートを作用させると間接的に第 2 量子ビットに量子ゲートを作用させたようにも振舞うので、これでは量子ビットのコピーとはいえないというわけです。

第 4 章

初等整数論

この章では、Shor のアルゴリズムの背後にある整数論の結果を扱います。剰余 (余り) の性質は Shor のアルゴリズムのみならず、古典・量子いずれの情報理論でも重要な役割を果たします。余りの計算を代数的に扱いやすくする剰余類から始め、Shor のアルゴリズムで必要な特別な元の数え上げを行います。また、位数を推定するステップで必要となる連分数の性質について解説します。

4.1 剰余と位数

Shor のアルゴリズムでは x(例えば 2) のべき乗を N で割った余りの周期性が中心的な役割を果たしました。このように、注目している整数 N だけでなく、一見無関係なほかの数との関係を見ることで、N に関するより深い情報が得られることがあります。ここでは、Shor のアルゴリズムで用いた N で割った余りの性質をみていきましょう。

整数を 2 で割った余りは 0 または 1 になります。つまり偶数奇数ですが、四則演算に関して以下のような法則があります。

$$\begin{aligned} &\text{偶数} + \text{偶数} = \text{偶数}, \quad \text{偶数} + \text{奇数} = \text{奇数}, \\ &\text{奇数} + \text{奇数} = \text{偶数}, \quad \text{偶数} \times \text{偶数} = \text{偶数}, \quad \cdots \end{aligned}$$

同様の法則は、3 で割った余りや 4 で割った余りなどに対しても書き下すことができます。例えば、3 で割って 2 余る整数を二つ掛けると 3 で割って 1 余る整数になります。

上記のような剰余の間の法則を調べようと思ったとき、いちいち余りを取ると非常に分かりづらくなります。1.2 節で用いた N で割った余りを意味する $\bmod N$ という記号を使うと 奇数 + 奇数 = 偶数 という式は

$$a \bmod N = 1 \text{ かつ } b \bmod N = 1 \Rightarrow (a+b) \bmod N = 0$$

と表せますが、上で書いたような等式の形で素直に書けません。まずは、奇数 + 奇数 = 偶数 のような式を表現できるように新しい記号を導入しましょう。等号 $=$ の代わりになる記号 $\equiv$ を定義する合同式と、虚数単位 i のような新しい記号を用意して都合のよい演算を定義する剰余類という二つの考え方があります。ここでは剰余類を使って議論していきます。

奇数 + 奇数 = 偶数 のような式を表したいという目的でした。そのために、複素数、ベクトル、行列を定義したときのように欲しい機能を持った記号を作ります。目的の式には日本語が混じっているので、まずは「奇数」「偶数」を数学的な対象で表す必要があります。これは単にそれぞれを、奇数全体の集合、偶数全体の集合に置き換えることとし、以下のように記号を定めます。

$$[0]_2 \coloneqq \{n \in \mathbb{Z} \mid n \bmod 2 = 0\}, \quad [1]_2 \coloneqq \{n \in \mathbb{Z} \mid (n-1) \bmod 2 = 0\}.$$

$[0]_2$ が偶数全体の集合で、$[1]_2$ が奇数全体の集合になっています。あとは $[1]_2 + [1]_2 = [0]_2$ となるように和を定義してやれば欲しい機能を持った記号が得られます。以下、N を正の整数として、N で割った余りの場合に正確な定義を述べます。

定義 4.1.1 (剰余類)**.** $a \in \mathbb{Z}$ に対して、

$$[a]_N \coloneqq \{n \in \mathbb{Z} \mid (n-a) \bmod N = 0\}$$

と定め、これを N **を法とする** a **の剰余類**と呼ぶ。$a, b \in \mathbb{Z}$ に対して、剰余類の和と積を

$$[a]_N + [b]_N = [a+b]_N,$$
$$[a]_N \cdot [b]_N = [ab]_N$$

となるように定める。非負整数 n と $a \in \mathbb{Z}$ に対して、$[a]_N$ の n 倍と n 乗を

$$n[a]_N \coloneqq [a]_N + [a]_N + \cdots + [a]_N \ (n \text{ 個}),$$
$$[a]_N^n \coloneqq [a]_N \cdot [a]_N \cdot \cdots \cdot [a]_N \ (n \text{ 個})$$

と定める。ただし、$0 \cdot [a]_N = [0]_N$, $[a]_N^0 = [1]_N$ と約束する。

和と積の定義から $[a]_N$ の n 倍と n 乗は、

$$n[a]_N = [na]_N, \quad [a]_N^n = [a^n]_N$$

と書き直せます。また、n が負の整数の場合にも n 倍を

$$n[a]_N = [na]_N$$

によって拡張しておきましょう。$(-1) \cdot [a]_N$ を単に $-[a]_N$ と表し、$[a]_N + (-[b]_N)$ を $[a]_N - [b]_N$ と表すことにします。これらの約束事により、剰余類を使った式やその変形を、複素数や実数のものと同様に扱えるようになります。

$[a]_N$ は N で割った余りが $a \bmod N$ と等しいような整数の集合です。例えば、$[1]_3$ は 3 で割って 1 余る整数の集合で、$[6]_4$ は 4 で割って 2 余る整数の集合です。$[4]_3$ も 3 で割って 1 余る整数の集合なので $[1]_3 = [4]_3$ となります。分数と既約分数の関係のように、同じものを複数の方法で表せることに注意しましょう。どの程度、違う表現の仕方があるかについては、以下の命題から分かります。

命題 4.1.2. $a, b \in \mathbb{Z}$ とする。$[a]_N = [b]_N$ であることと、$a - b$ が N で割り切れることは同値である。

この命題を踏まえると、以下のような等式が成り立ちます。

$$\begin{aligned}
[1]_2 + [1]_2 &= [1+1]_2 = [2]_2 = [0]_2, \\
[1]_3 &= [4]_3 = [7]_3 = [10]_3 = [-2]_3 = [-5]_3 = \cdots, \\
[2]_{15} &= [17]_{15} = [32]_{15} = [2]^5_{15}, \\
[2]_3 \cdot [2]_3 &= [4]_3 = [1]_3.
\end{aligned}$$

確かに最初の目標だった 奇数 + 奇数 = 偶数 や様々な剰余の関係式を表現できています。

厳密には、定義 4.1.1 にある和と積の定義に矛盾がないことを確かめなければいけません。どういうことかというと、$[1]_3 = [4]_3$ のように一つのものでも複数の表し方があるので、どのような表し方でも同じ計算結果になることを確かめないといけません。実際、具体例でみてみると

$$\begin{aligned}
[1]_3 + [1]_3 &= [2]_3, \\
[1]_3 + [1]_3 &= [4]_3 + [4]_3 = [8]_3 = [2]_3
\end{aligned}$$

のように二通りの計算方法で同じ結果になっていて矛盾していません。一般の場合は次の命題から分かります。

命題 4.1.3. $a, b, c, d \in \mathbb{Z}$ とし $[a]_N = [b]_N, [c]_N = [d]_N$ と仮定する。このとき、$[a+c]_N = [b+d]_N$ となる。

証明. $[a]_N = [b]_N$ から $a - b$ は N で割り切れ、同様に $c - d$ も N で割り切れる。したがって、$(a + c) - (b + d) = (a - b) + (c - d)$ も N で割り切れるが、これは $[a + c]_N = [b + d]_N$ を意味する。 □

このように矛盾なく定義されていることを well-defined であるといいます。命題 4.1.3 から $[a]_N$ の和は well-defined になっています。積が well-defined であることは和の場合と同様に示せるので省略します。

命題 4.1.2 から、N を法とする剰余類は $[0]_N, [1]_N, \ldots, [N-1]_N$ で重複なく尽くされていることが分かります。余りに関する議論を可能な限り剰余類だけで議論できるようにするために、N を法とする剰余類の集合を定義しておきましょう。

定義 4.1.4. $\mathbb{Z}/N\mathbb{Z} \coloneqq \{[0]_N, [1]_N, \ldots, [N-1]_N\}$ と定める。

$\mathbb{Z}/N\mathbb{Z}$ は N で割った余りとしてあり得る整数の集合 $\{0, 1, 2, \ldots, N-1\}$ と似たようなものですが、$\mathbb{Z}$ のように和と積が定義できているという大きな違いがあります。$1+1=2 \notin \{0,1\}$ のように 2 で割った余りの集合 $\{0,1\}$ は和ではみ出てしまいますが、$[1]_2 + [1]_2 = [0]_2 \in \mathbb{Z}/2\mathbb{Z}$ なので $\mathbb{Z}/2\mathbb{Z}$ ならはみ出ず $\mathbb{Z}/2\mathbb{Z}$ の中で計算できます。これにより、N で割った余りに関する定理を、$\mathbb{Z}/N\mathbb{Z}$ という数の体系での定理として扱えるようになります。

$\mathbb{Z}/N\mathbb{Z}$ にも $\mathbb{C}, \mathbb{R}, \mathbb{Z}$ の $0, 1$ のような元が存在することを確認しておきましょう。

命題 4.1.5. $a \in \mathbb{Z}/N\mathbb{Z}$ とすると、

$$\begin{aligned} a + [0]_N &= [0]_N + a = a, \\ a \cdot [0]_N &= [0]_N \cdot a = [0]_N, \\ a \cdot [1]_N &= [1]_N \cdot a = a \end{aligned}$$

となる。

この命題は、$a = [a']_N$ $(a' \in \mathbb{Z})$ と表してから和と積の定義を使えばすぐに分かります。剰余類の和と積は整数の和と積から作られたものなので、ほかにも結合法則・交換法則・分配法則もそのまま成り立ちます。これらの証明も省略しますが、今後は断りなく使います。

このように、$\mathbb{Z}/N\mathbb{Z}$ は $\mathbb{C}, \mathbb{R}, \mathbb{Z}$ などと非常によく似た性質を持ちます。一方で、どんな元も N 倍すると $[0]_N$ になるという、整数や複素数とは全く異なる性質も持ちます。

命題 4.1.6. $a \in \mathbb{Z}/N\mathbb{Z}$ とすると、$Na = [0]_N$ となる。

次に、N と互いに素な整数とそれらの積の構造について、$\mathbb{Z}/N\mathbb{Z}$ の言葉に翻訳しながら考えていきましょう。整数 x と N が互いに素であるとは、$\gcd(x, N) = 1$ が成り立つということでした。あるいは x と N は同じ素因数を持たないといっても同じです。$\mathbb{Z}/N\mathbb{Z}$ で対応するのは、以下の可逆という概念になります。

定義 4.1.7. $g \in \mathbb{Z}/N\mathbb{Z}$ は、ある $h \in \mathbb{Z}/N\mathbb{Z}$ が存在して $gh = [1]_N$ となるとき、**可逆**であるという。このような h は g に対して一通りに定まるので g^{-1} と表す。$(\mathbb{Z}/N\mathbb{Z})^\times$ を $\mathbb{Z}/N\mathbb{Z}$ の可逆な元全体の集合とする。

$\mathbb{C}$ や $\mathbb{R}$ における 0 以外の元、あるいは正則な行列のような性質を持つ可逆元は、「割り算」ができるという点において特別な元になります。つまり、g が可逆なら $gx = a$ という方程式を $x = g^{-1}a$ という形で解くことができます。したがって、特に $g, h \neq [0]_N$ が $gh = [0]_N$ を満たすなら g と h は可逆にはなりえません。なぜなら、g が可逆なら $h = g^{-1}[0]_N = [0]_N$ となりますし、h が可逆でも同様に $g = [0]_N$ となってしまうからです。

例 4.1.8. $N = 6$ の場合、可逆元は $[1]_6, [5]_6$ の二つのみである。実際、

$$[1]_6 \cdot [1]_6 = [1]_6, \quad [5]_6 \cdot [5]_6 = [25]_6 = [1]_6,$$
$$[2]_6 \cdot [3]_6 = [6]_6 = [0]_6, \quad [4]_6 \cdot [3]_6 = [12]_6 = [0]_6$$

となって、$[1]_6, [5]_6$ は可逆で $[2]_6, [3]_6, [4]_6$ は可逆ではない。$N = 7, 15$ の場合は

$$(\mathbb{Z}/7\mathbb{Z})^\times = \{[1]_7, [2]_7, [3]_7, [4]_7, [5]_7, [6]_7\},$$
$$(\mathbb{Z}/15\mathbb{Z})^\times = \{[1]_{15}, [2]_{15}, [4]_{15}, [7]_{15}, [8]_{15}, [11]_{15}, [13]_{15}, [14]_{15}\}$$

となる。

例 4.1.8 から分かるように、N と互いに素な整数と $\mathbb{Z}/N\mathbb{Z}$ の可逆元は対応しています。また、可逆元同士の積も可逆元になることも分かります。これらを示す前に、最大公約数 gcd の特徴づけを示しておきましょう。

定理 4.1.9. $a, b, d \in \mathbb{Z}$ とする。x, y を未知数とする方程式 $ax + by = d$ が整数解をもつことと、d が $\gcd(a, b)$ の倍数であることは同値である。

証明. $ax + by = d$ が整数解をもつなら左辺 $ax + by$ は $\gcd(a, b)$ で割り切れるので、d も $\gcd(a, b)$ で割り切れなければならない。逆を示すためには $ax + by = \gcd(a, b)$ が整数解 x, y を持つことを示せばよい。

a, b を $|a|, |b|$ で置き換えても整数解が存在するかどうかは変わらないので、$a, b \geq 0$ と仮定してよい。また、a, b に関して対称なので $a \geq b$ としてよい。$ax + by = \gcd(a, b)$ に整数解が存在することを b に関する数学的帰納法で示す。$b = 0$ の場合は $\gcd(a, b) = a$ なので、$x = 1, y = 0$ とすればよい。$b \leq k$ $(k \geq 0)$ の場合は成り立つと仮定して、$b = k + 1$ の場合を示す。

a を b で割った商と余りを q と a' と置く。$a' < b$ なので帰納法の仮定から $bx_0 + a'y_0 = \gcd(b, a')$ を満たす整数 x_0, y_0 が存在する。$\gcd(b, a') = \gcd(a, b)$ なので、

$$\begin{aligned} ay_0 + b(x_0 - qy_0) &= (qb + a')y_0 + b(x_0 - qy_0) \\ &= a'y_0 + bx_0 = \gcd(a, b) \end{aligned}$$

となり、$ax + by = \gcd(a, b)$ の整数解 $(x, y) = (y_0, x_0 - qy_0)$ が得られた。 □

定理 4.1.9 の証明は $ax + by = \gcd(a, b)$ の整数解を求めるアルゴリズムも与えています。このアルゴリズムは後で述べる連分数展開 (定理 4.5.10) と本質的に同じもので、計算に必要な割り算の回数も連分数展開のものと等しくなります。

命題 4.1.10. (1) $[1]_N$ は可逆で、$[1]_N^{-1} = [1]_N$ である。

(2) $g, h \in (\mathbb{Z}/N\mathbb{Z})^\times$ とすると、gh も可逆であり $(gh)^{-1} = h^{-1}g^{-1}$ $(= g^{-1}h^{-1})$ となる。

(3) $x \in \mathbb{Z}$ とする。x が N と互いに素であることと $[x]_N$ が可逆であることは同値である。

証明. (1) は $[1]_N \cdot [1]_N = [1]_N$ なのでよい。(2) も $(gh)(h^{-1}g^{-1}) = g[1]_N g^{-1} = gg^{-1} = [1]_N$ から従う。

(3) を示そう。$[x]_N$ が可逆と仮定し x と N が互いに素であることを示す。$[x]_N$ は可逆なので $[x]_N[y]_N = [1]_N$ となる $y \in \mathbb{Z}$ が存在する。このとき、$xy - 1$ は N で割り切れる、つまり $xy + Ns = 1$ となるような $s \in \mathbb{Z}$ が存在する。この式から $\gcd(x, N)$ は 1 を割り切ることが分かり、したがって $\gcd(x, N) = 1$ を得る。よって、x と N は互いに素である。

x と N が互いに素と仮定し逆を示す。定理 4.1.9 より $ax + bN = 1$ となるような整数 a, b が存在する。したがって、$[a]_N[x]_N = [ax + bN]_N = [1]_N$ となり $[x]_N$ が可逆と分かる。 □

命題 4.1.10 (3) から、N と互いに素な 1 以上 N 以下の整数は $(\mathbb{Z}/N\mathbb{Z})^\times$ の元と一対一に対応することが分かります。

Shor のアルゴリズムの手順 2 で、x を固定して $f(j) = x^j \bmod N$ という関数を使いました。この関数の振る舞い、特に周期性を N が素数の場合と合成数の場合で考えていきます。剰余類を導入したので、f の代わりにべき乗 $[x]_N^j$ の振る舞いを調べましょう。

$$\begin{array}{ccccccc} [0]_N & & [1]_N & & [2]_N & \cdots & [N-1]_N \\ & & & & \updownarrow & & \\ 0 & & 1 & & 2 & \cdots & N-1 \end{array}$$

のように対応させれば $[x]_N^j = [x^j]_N$ の性質を f の性質に翻訳できます。例えば $N = 15$, $x = 2$ のとき、

j	1	2	3	4	5	6	7
$[x]_N^j$	$[2]_N$	$[4]_N$	$[8]_N$	$[1]_N$	$[2]_N$	$[4]_N$	$[8]_N$
$f(j)$	2	4	8	1	2	4	8

と対応しています。

以下、$x \in \mathbb{Z}/N\mathbb{Z}$ としましょう。x のべき乗も複素数のべき乗と同様に、以下の指数法則を満たします。

定理 4.1.11 (指数法則)**.** 任意の非負整数 i, j に対して、$x^{i+j} = x^i x^j$ となる。

定理 4.1.11 の特別な場合として、$x^r = [1]_N$ となる r があれば任意の j に対して $x^{j+r} = x^j$ が成り立つことが分かります。f の言葉では、r は f の周期である、つまり $f(j + r) = f(j)$ となると言い換えられます。Shor のアルゴリズムの肝はこのような $r > 0$ を見つけることでした。

$[2]_4^2 = [2]_4^3 = \cdots = [0]_4$ から分かるように、$N = 4, x = 2$ に対しては絶対にこのような r は存在しません。$x^r = [1]_N$ となる r が存在するための条件を与えましょう。

命題 4.1.12. x が可逆であることと、$x^r = [1]_N$ となる正の整数 r が存在することは同値である。また、そのような r が存在するとき、任意の非負整数 j に対して $x^{j+r} = x^j$ となる。

証明. 後半は先ほど述べたとおりである。まず、$x \in (\mathbb{Z}/N\mathbb{Z})^\times$ と仮定し r の存在を示す。x^j の値は $[0]_N, \ldots, [N-1]_N$ のいずれかであり有限通りしかないので、ある正の整数 a, b $(a < b)$ で $x^a = x^b$ となるものが存在する。x は可逆なので $x^a = x^b$ の両辺に $(x^{-1})^a$ を掛けることができ、$x^{b-a} = [1]_N$ を得る。

逆に、$x^r = [1]_N$ となるような正の整数 r が存在すると仮定する。$x \cdot x^{r-1} = x^r = [1]_N$ なので、x は可逆であり、$x^{-1} = x^{r-1}$ となる。 □

以上で、周期 r が存在するための必要十分条件が分かりました。この周期のうち最も小さいものに名前を付けておきましょう。

定義 4.1.13 (位数)**.** $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ に対して、$x^r = [1]_N$ となる最小の正の整数 r を x の**位数**という。また、x' を N と互いに素な整数としたとき、$[x']_N$ の位数を N を法とする x' の**位数**という。N が文脈から明らかな場合は「N を法とする」は省略する。

例 4.1.14. $N = 15$ の場合

$$[2]_N^2 = [4]_N, \quad [2]_N^3 = [8]_N, \quad [2]_N^4 = [1]_N,$$
$$[7]_N^2 = [4]_N, \quad [7]_N^3 = [13]_N, \quad [7]_N^4 = [1]_N$$

となるので、$[2]_N$ と $[7]_N$ の位数はどちらも 4 である。

$x^r = [1]_N$ となる最小の正の整数 r を位数と定義しましたが、最小ではない r はどのような性質を満たすでしょうか。この問に対する答えは以下のようになります。

命題 4.1.15. $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ とし、x の位数を r_0 とする。r を $x^r = [1]_N$ を満たす正の整数とする。このとき、r は r_0 で割り切れる。

証明. r を r_0 で割った余りを s と置き、$r = r_0 \cdot k + s$ $(k \in \mathbb{Z})$ と表す。このとき、

$$\begin{aligned}[1]_N = x^r &= x^{r_0 k + s} \\ &= (x^{r_0})^k x^s \\ &= ([1]_N)^k x^s = x^s\end{aligned}$$

となる。よって、$x^s = [1]_N$ となるが $0 \leq s < r_0$ なので r_0 の最小性から $s = 0$ とならなければいけない。したがって、$r = r_0 \cdot k$ となり r は r_0 で割り切れる。 □

命題 4.1.15 から、位数 r_0 さえ見つけてしまえば $x^r = [1]_N$ となるような r は r_0 の倍数しかないと分かります。逆に、このような r が見つかれば、位数は r の約数であることが分かります。N が素数の場合には実際に $x^r = [1]_N$ となる r を簡単に見つけることができ、フェルマーの小定理と呼ばれています。

定理 4.1.16 (フェルマーの小定理)**.** N を素数とし、$x \in (\mathbb{Z}/N\mathbb{Z})^\times$ とする。このとき、$x^{N-1} = [1]_N$ となる。

例 4.1.17. $f(j) = x^j \bmod N$ の文脈で言い換えると、$f(N-1) = 1$ という式に翻訳される。$N = 7, x = 3$ の場合、$f(j)$ は以下の表のようになり、確かに $f(6) = 1$ である。

j	0	1	2	3	4	5	6	7	$\cdots$
x^j	1	3	9	27	81	243	729	2187	$\cdots$
$x^j \bmod N$	1	3	2	6	4	5	1	3	$\cdots$

$x = 2$ の場合は $f(j)$ は以下の表のようになり、確かに $f(6) = 1$ だが、$f(3) = 1$ で位数は 3 である。定理は x の位数が $N-1$ を割り切るといっているだけで、ちょうど $N-1$ と主張しているわけではない。

j	0	1	2	3	4	5	6	$\cdots$
x^j	1	2	4	8	16	32	64	$\cdots$
$x^j \bmod N$	1	2	4	1	2	4	1	$\cdots$

証明. まず任意の $x \in \mathbb{Z}/N\mathbb{Z}$ に対して $(x + [1]_N)^N = x^N + [1]_N$ となることを示す。実数や複素数の場合と同様に二項定理が成り立ち、

$$(x + [1]_N)^N = \sum_{k=0}^{N} {}_N C_k x^k \tag{4.1}$$

となる。ここで、${}_N C_k$ は二項係数であり、$\frac{N!}{k!(N-k)!}$ で表される整数である。分子 $N!$ は N で割り切れるが、N は素数なので分母 $k!(N-k)!$ は $k \neq 0, N$ なら N で割り切れない。つまり、整数 ${}_N C_k$ は $k \neq 0, N$ の場合 N で割り切れる。$\mathbb{Z}/N\mathbb{Z}$ の元は N 倍すると $[0]_N$ になるので、

$$\begin{aligned}(x + [1]_N)^N &= \sum_{k=0}^{N} {}_N C_k x^k \\ &= [1]_N + [0]_N + \cdots + [0]_N \ (N-2 \text{ 個}) \ + x^N \\ &= [1]_N + x^N\end{aligned}$$

となる。

次に、任意の $x \in \mathbb{Z}/N\mathbb{Z}$ に対して $x^N = x$ となることを示す。$x = [a]_N$ (a は正の整数) と表す。このとき、先ほど示した等式 $(y + [1]_N)^N = y^N + [1]_N$ ($y \in \mathbb{Z}/N\mathbb{Z}$) を繰り返し適用すれば、

$$\begin{aligned}x^N &= ([1]_N + [1]_N + \cdots + [1]_N \ (a \text{ 個}))^N \\ &= ([1]_N + [1]_N + \cdots + [1]_N \ (a-1 \text{ 個}))^N + [1]_N\end{aligned}$$

$$
\begin{aligned}
&= \cdots \\
&= [1]_N + [1]_N + \cdots + [1]_N \ (a \text{ 個}) = [a]_N = x
\end{aligned}
$$

を得る。

$x \in (\mathbb{Z}/N\mathbb{Z})^\times$ とする。上で示したことから $x(x^{N-1} - [1]_N) = x^N - x = [0]_N$ となる。x は可逆なので、$x^{N-1} = [1]_N$ となる。 □

定理 4.1.16 から、$(\mathbb{Z}/N\mathbb{Z})^\times$ の元の位数は $N-1$ 以下であることが分かります。例 4.1.17 の計算から、$[3]_7$ の位数はちょうど $N-1=6$ になっています。$[3]_7$ のべき乗を計算すると

$$[3]_7^0 = [1]_7, \quad [3]_7^1 = [3]_7, \quad [3]_7^2 = [2]_7, \quad [3]_7^3 = [6]_7, \quad [3]_7^4 = [4]_7, \quad [3]_7^5 = [5]_7$$

となり、0 乗から $N-2=5$ 乗までで $(\mathbb{Z}/N\mathbb{Z})^\times$ の元が尽くされていることが分かります。一般に $N-1$ が位数になっている元 $x_0 \in (\mathbb{Z}/N\mathbb{Z})^\times$ を一つ固定すると、

$$(\mathbb{Z}/N\mathbb{Z})^\times = \left\{x_0^0, x_0, x_0^2, \ldots, x_0^{N-2}\right\}$$

のようにすべての元を x_0 のべき乗で表すことができます。この表示は、位数など積構造を調べる際に積を指数の和という単純な操作に翻訳できるため、非常に強力な道具となります。

定義 4.1.18 (原始根)**.** N を素数とする。$x \in (\mathbb{Z}/N\mathbb{Z})^\times$ は位数が $N-1$ のとき $\mathbb{Z}/N\mathbb{Z}$ の**原始根**と呼ばれる。また、N と互いに素な整数 x' は、$[x']_N$ の位数が $N-1$ のとき N を法とする原始根と呼ばれる。

例 4.1.19. 例 4.1.17 の計算から、3 は 7 を法とする原始根だが、2 は原始根ではない。

命題 4.1.20. $x_0 \in (\mathbb{Z}/N\mathbb{Z})^\times$ を原始根とする。このとき、

$$(\mathbb{Z}/N\mathbb{Z})^\times = \left\{x_0^0, x_0, x_0^2, \ldots, x_0^{N-2}\right\}$$

が成り立つ。

証明. 両辺の元の個数が等しいことを示せば等号も示される。$|(\mathbb{Z}/N\mathbb{Z})^\times| = N-1$ なので $x_0^0, x_0, x_0^2, \ldots, x_0^{N-2}$ に同じ元がないことを示そう。$x_0^s = x_0^t$ $(0 \leq s \leq t < N-1)$ とする。両辺に $(x_0^{-1})^s$ を掛ければ $x_0^{t-s} = [1]_N$ となる。x_0 の位数は $N-1$ であり $0 \leq t-s < N-1$ なので、位数の最小性から $t-s=0$、つまり $t=s$ を得る。 □

では、実際に原始根が存在するかという疑問が出ますが、以下の定理で存在することが保証されます。証明にはいくつかの準備が必要で長くなるので、本テキストでは省略することにします。

定理 4.1.21. N を素数とすると、$\mathbb{Z}/N\mathbb{Z}$ の原始根が必ず存在する。

4.2　中国剰余定理

N が合成数の場合にはフェルマーの小定理 (定理 4.1.16) は成り立つとは限りません。例えば $N = 4, x = 3$ の場合、$[x]_4^3 = [3]_4 \neq [1]_4$ となり $N-1$ 乗しても $[1]_4$ にはなりません。

N が合成数の場合には、位数の振る舞いは素数の場合に比べて複雑になります。一方で、N の素因数分解が分かっている状況であれば、N が素数 (または素数のべき乗) の場合に帰着できます。ここでは、$N = ab$ (a, b は互いに素な正の整数) と表し、mod N の話が mod a, mod b の話に帰着できるという中国剰余定理を示します。

中国剰余定理とは、一言でいえば整数 n を N で割った余りは、n を a で割った余りと n を b で割った余りから一通りに計算できる、というものです。$N = 15$ の場合の剰余の表は以下のようになりますが、確かに mod 15 は mod 3 と mod 5 から一通りに決まります。それだけでなく、mod 3 と mod 5 のペアとしてあり得るものがすべて現れています。

$n \bmod 15$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$n \bmod 3$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
$n \bmod 5$	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4

mod 15 が mod 3 と mod 5 から決まる理由は、mod 3 の行は周期 3 の繰り返しになっており、mod 5 の行は周期 5 の繰り返しになっているため、同じ組み合わせが現れるのは最小公倍数の 15 ごとだからである、と直感的に説明できます。

これを剰余類を使って証明するために、N で割った余りだけから、a(または b) で割った余りも分かることを示しておきましょう。

補題 4.2.1. $j, j' \in \mathbb{Z}$ とする。$[j]_N = [j']_N$ ならば $[j]_a = [j']_a$ となる。

証明. $[j]_N = [j']_N$ なので、$j - j'$ は N で割り切れる。特に $j - j'$ は a で割り切れるので、$[j]_a = [j']_a$ となる。 □

N で割った余りから a と b で割った余りのペアを対応させる写像を作って、それらの間の関係を調べます。例えば $N = 15 = 3 \cdot 5$ の場合、

$$\begin{array}{ccc} \{0,1,2,\ldots,14\} & \to & \{0,1,2\} \times \{0,1,2,3,4\} \\ \cup & & \cup \\ n & \mapsto & (n \bmod 3, n \bmod 5) \end{array}$$

という写像を考え、二つの集合の間の関係を調べようという方針です。剰余類の言葉に翻訳して定義すると、

$$\Phi \colon \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z} \tag{4.2}$$

$$\Phi([j]_N) = ([j]_a, [j]_b) \quad ([j]_N \in \mathbb{Z}/N\mathbb{Z})$$

となります。これまでどおり、$[j]_N$ が j を N で割った余り、$([j]_a, [j]_b)$ が a, b で割った余りのペアに対応しています。ここでも、剰余類の和のときに述べた well-defined かどうかが問題になることに注意しましょう。$[j]_N = [j']_N$ のときに、$([j]_a, [j]_b) = ([j']_a, [j']_b)$ とならなければ Φ の定義に矛盾が起きてしまいます。補題 4.2.1 のおかげでこのようなことは起こらず、Φ は well-defined になります。

定理 4.2.2 (**中国剰余定理**)**.** Φ は全単射である。特に、整数を N で割った余りは、a と b で割った余りのペアから一通りに復元できる。[*1]

証明. $\mathbb{Z}/N\mathbb{Z}$ の元の個数は N で、$\mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ の元の個数は $a \times b = N$ なので、Φ が単射であることを示せば全射性も自動的に従う。Φ の単射性を示そう。j, j' を $\Phi([j]_N) = \Phi([j']_N)$ を満たす整数とする。Φ の定義と $\Phi([j]_N) = \Phi([j']_N)$ から、$[j]_a = [j']_a$ かつ $[j]_b = [j']_b$ となる。よって、$j - j'$ は a と b で割り切れるが、a と b は互いに素なので、$j - j'$ は $ab = N$ で割り切れる。したがって、$[j]_N = [j']_N$ となり、Φ は単射である。 □

$\mathbb{Z}/N\mathbb{Z}$ と $\mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ の間の全単射 Φ が得られました。ただし、両者の集合の元の個数は同じなので、その間に全単射があるというのは当たり前のことです。特別な式で作られたこの Φ は、以下のようにより強い性質を持ちます。

$\mathbb{Z}/N\mathbb{Z}$ には和と積が備わっていました。$\mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ にも自然に和と積が

$$\begin{aligned} &(j,k) + (j',k') = (j + j', k + k'), \\ &(j,k) \cdot (j',k') = (jj', kk') \qquad\qquad (j, j' \in \mathbb{Z}/a\mathbb{Z}, k, k' \in \mathbb{Z}/b\mathbb{Z}) \end{aligned}$$

によって定まります。つまり、ベクトルの和とスカラー倍のように、成分ごとに和と積を行うとして定義します。Φ はこれらの和と積と相性がよいことが分かります。

定理 4.2.3. $j, k \in \mathbb{Z}/N\mathbb{Z}$ とすると

$$\begin{aligned} \Phi(j + k) &= \Phi(j) + \Phi(k), \\ \Phi(j \cdot k) &= \Phi(j) \cdot \Phi(k) \end{aligned}$$

が成り立つ。

証明. $j = [j']_N$, $k = [k']_N$ $(j', k' \in \mathbb{Z})$ と表しておく。Φ と和の定義から

$$\begin{aligned} \Phi(j + k) &= \Phi([j' + k']_N) \\ &= ([j' + k']_a, [j' + k']_b) \end{aligned}$$

[*1] このテキストでは扱いませんが、実際に a, b で割った余りのペアから N で割った余りを求めるアルゴリズムが知られており、それも中国剰余定理に含まれます。

$$= ([j']_a, [j']_b) + ([k']_a, [k']_b)$$
$$= \Phi([j']_N) + \Phi([k']_N) = \Phi(j) + \Phi(k)$$

となり、和の等式が示された。積の方の等式も同様なので省略する。 □

定理 4.2.3 の等式が成り立つことを、Φ は和と積を保つ、あるいは Φ は環の**準同型**写像である、という言い方をします。Φ はさらに全単射ですが、全単射な準同型写像を**同型**写像と呼びます。$\mathbb{Z}/N\mathbb{Z}$ と $\mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ は集合としては違いますが、和と積だけを考えるなら同じ性質を持つという意味が同型という言葉に込められています。

Φ は同型写像なので、$\mathbb{Z}/N\mathbb{Z}$ での和と積に関する問題はすべて $\mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ に帰着できます。$\mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ の問題も、$\mathbb{Z}/a\mathbb{Z}$ における問題と $\mathbb{Z}/b\mathbb{Z}$ における問題に分割できるので、結局 N が素数や素数のべき乗の場合に詳しく分かっていれば一般の場合もよく分かるということになります。ただし、抽象的な性質の議論をする場合は問題ないのですが、具体的な計算を行う場合には N の分解 $N = ab$ が分かっている必要があります。

例 4.2.4. $N = 15 = 3 \cdot 5$ の場合、

$$[2]_3^2 = [1]_3,$$
$$[2]_5^2 = [4]_5, \quad [2]_5^3 = [3]_5, \quad [2]_5^4 = [1]_5$$

となる。$\Phi([2]_{15})^j = ([2]_3^j, [2]_5^j)$ なので $[2]_{15}^j = [1]_N$ となるのは $([2]_3^j, [2]_5^j) = ([1]_3, [1]_5)$ となる場合に限る。よって、$[2]_{15}$ の位数は 4 と分かる。

このように、mod a と mod b での位数が分かれば、mod N の位数も簡単に計算できます。

定理 4.2.5. $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ とし、$\Phi(x) = (x_a, x_b)$ により $x_a \in \mathbb{Z}/a\mathbb{Z}, x_b \in \mathbb{Z}/b\mathbb{Z}$ を定める。このとき、x_a, x_b も可逆であり、それぞれの位数を j_0, k_0 とすると、x の位数は最小公倍数 $\mathrm{lcm}(j_0, k_0)$ と等しい。

証明. x の位数を r_0 と置く。Φ は積を保つので、

$$\Phi(x^{r_0}) = \Phi([1]_N) = ([1]_a, [1]_b),$$
$$\Phi(x^{r_0}) = \Phi(x)^{r_0} = (x_a^{r_0}, x_b^{r_0})$$

となる。よって、$x_a^{r_0} = [1]_a$ かつ $x_b^{r_0} = [1]_b$ となる。命題 4.1.12 と命題 4.1.15 より、x_a, x_b も可逆であり、r_0 は j_0 と k_0 で割り切れる。特に r_0 は $\mathrm{lcm}(j_0, k_0)$ で割り切れる。

一方、$r = \mathrm{lcm}(j_0, k_0)$ と置くと

$$\Phi(x^r) = \Phi(x)^r = (x_a^r, x_b^r) = ([1]_a, [1]_b) = \Phi([1]_N)$$

となり、Φ の単射性から $x^r = [1]_N$ となる。再び命題 4.1.15 から r は r_0 で割り切れる。以上より $r_0 = r = \mathrm{lcm}(j_0, k_0)$ となる。 □

特に a, b が素数の場合には、位数の最大値は以下のように評価できます。フェルマーの小定理 (定理 4.1.16) と命題 4.1.15 から、mod a での位数は $a-1$ を割り切ることを思い出しておきましょう。定理 4.2.5 を合わせると以下の系が得られます。

系 4.2.6. x を N と互いに素な整数とする。$[x]_N$ の位数は $\mathrm{lcm}(a-1, b-1)$ を割り切る。さらに、x が mod a と mod b の両方で原始根ならば、$[x]_N$ の位数は $\mathrm{lcm}(a-1, b-1)$ となる。

特に、mod N での位数の最大値は $\mathrm{lcm}(a-1, b-1)$ になります。a, b がどちらも奇数の素数なら、$a-1, b-1$ はどちらも偶数なので、$\mathrm{lcm}(a-1, b-1) = (a-1)(b-1)/\gcd(a-1, b-1)$ は $(a-1)(b-1)/2$ 以下になります。したがって、mod N での位数はどれも $N/2$ 未満になります。

例 4.2.7. $N = 35 = 5 \cdot 7$ の場合に mod 5 と mod 7 の両方で原始根になっているような $\mathbb{Z}/N\mathbb{Z}$ の元を求めてみる。mod 5 での位数は

$[n]_5$	$[1]_5$	$[2]_5$	$[3]_5$	$[4]_5$
位数	1	4	4	2

となるので、$\mathbb{Z}/5\mathbb{Z}$ の原始根は $[2]_5, [3]_5$ である。mod 7 での位数は

$[n]_7$	$[1]_7$	$[2]_7$	$[3]_7$	$[4]_7$	$[5]_7$	$[6]_7$
位数	1	3	6	3	6	2

となるので、$\mathbb{Z}/7\mathbb{Z}$ の原始根は $[3]_7, [5]_7$ である。よって、mod 5 で原始根かつ mod 7 で原始根であるような組み合わせをすべて考慮すると表 4.1 のようになり、位数 $\mathrm{lcm}(5-1, 7-1) = 12$ の元が四つ得られた。ただし、位数 12 の元は $[2]_{35}$ などもあるためすべて尽くされているわけではない。

	$[2]_5$	$[3]_5$
$[3]_7$	$[17]_{35}$	$[3]_{35}$
$[5]_7$	$[12]_{35}$	$[33]_{35}$

表 4.1: mod 5 と mod 7 のペアに対応する mod 35 の元

4.3 N と互いに素な整数の割合

Shor のアルゴリズムでは、ある正の整数 N に対して、N と互いに素な 1 以上 N 以下の整数の割合が成功確率に影響してきます。この節では、N と互いに素な整数の割合を中国剰余定理 (定理 4.2.2) を使って計算しやすい式で表し、下からの評価を与えます。

以下、N を正の整数とします。可逆元の性質を N が素数のべき乗の場合に帰着するために、中国剰余定理と可逆元の関係をみておきましょう。

定理 4.3.1. a, b を互いに素な正の整数とし、$N = ab$ と仮定する。$x \in \mathbb{Z}/N\mathbb{Z}$ とし、$\Phi(x) = (x_a, x_b)$ により $x_a \in \mathbb{Z}/a\mathbb{Z}, x_b \in \mathbb{Z}/b\mathbb{Z}$ を定める。ここで、Φ は (4.2) で定めた写像である。このとき、以下の 2 条件は同値である。

(1) x は可逆である。
(2) x_a と x_b は可逆である。

また、この 2 条件が成り立つとき、$\Phi(x^{-1}) = (x_a^{-1}, x_b^{-1})$ が成り立つ。特に、Φ を $(\mathbb{Z}/N\mathbb{Z})^\times$ に制限したものは、$(\mathbb{Z}/N\mathbb{Z})^\times$ と $(\mathbb{Z}/a\mathbb{Z})^\times \times (\mathbb{Z}/b\mathbb{Z})^\times$ の間の全単射を与える。

証明. $y \in \mathbb{Z}/N\mathbb{Z}$ とし $\Phi(y) = (y_a, y_b)$ により y_a, y_b を定めると、定理 4.2.3 より

$$(x_a y_a, x_b y_b) = \Phi(x)\Phi(y) = \Phi(xy)$$

となる。また、$\Phi([1]_N) = ([1]_a, [1]_b)$ である。よって、

$$\begin{aligned} x_a y_a = [1]_a \text{ かつ } x_b y_b = [1]_b &\Leftrightarrow (x_a y_a, x_b y_b) = ([1]_a, [1]_b) \\ &\Leftrightarrow \Phi(xy) = \Phi([1]_N) \\ &\Leftrightarrow xy = [1]_N \end{aligned}$$

となる。 □

定義 4.3.2. 正の整数 n に対して、n と互いに素な 1 以上 n 以下の整数の個数を $\phi(n)$ と表す。ϕ を **Euler のトーシェント関数**と呼ぶ。

命題 4.1.10 (3) から $\phi(n)$ は $(\mathbb{Z}/n\mathbb{Z})^\times$ の元の個数と等しくなります。n が小さい場合の $\phi(n)$ の値を表にしておきましょう。

n	2	3	4	5	6	7	8	9	10	11	12	13	14
$\phi(n)$	1	2	2	4	2	6	4	6	4	10	4	12	6

表 4.2: 小さい n に対する $\phi(n)$

定理 4.3.1 の後半部分から、a, b が互いに素な正の整数なら

$$|(\mathbb{Z}/ab\mathbb{Z})^\times| = |(\mathbb{Z}/a\mathbb{Z})^\times| \cdot |(\mathbb{Z}/b\mathbb{Z})^\times|$$

となります。実際、表 4.2 を見ると、$\phi(12) = \phi(3)\phi(4)$, $\phi(14) = \phi(2)\phi(7)$ などが成り立っていることが分かります。

系 4.3.3. a, b を互いに素な正の整数とすると、$\phi(ab) = \phi(a)\phi(b)$ が成り立つ。

定理 4.3.4. $N = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ と素因数分解されているとする。このとき、

$$\phi(N) = N\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_r}\right)$$

が成り立つ。

証明. 系 4.3.3 を繰り返し適用すれば、

$$\phi(N) = \phi(p_1^{e_1})\phi(p_2^{e_2})\cdots\phi(p_r^{e_r})$$

が成り立つので、$N = p_1^{e_1}$ の場合を示せば一般の場合も従う。$N = p_1^{e_1}$ の場合、$\phi(N)$ は p_1 で割り切れない 1 以上 N 以下の整数の個数である。p_1 で割り切れる 1 以上 N 以下の整数の個数は N/p_1 なので、

$$\phi(N) = N - \frac{N}{p_1} = N\left(1 - \frac{1}{p_1}\right)$$

となり確かに成り立っている。 □

さて、N と互いに素な整数の割合 $\phi(N)/N$ の具体的な式が得られました。これで、ランダムに $1 \leq x \leq N$ を選んだときに x と N が互いに素になる確率が分かるわけですが、どんなときにどれほど確率が小さくなるかを考察しておきましょう。定理 4.3.4 より、$\phi(N)/N$ をなるべく小さくしようと思った場合、素数を小さい順に掛けていったもの $2 \cdot 3 \cdot 5 \cdot \cdots \cdot p_r$ を N にすれば、効率よく小さくできます。したがって、

$$\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right)\cdots\left(1 - \frac{1}{p_r}\right)$$

という形の積の大きさを評価すれば、どれほど確率が小さくなるかが分かります。この評価には Mertens の定理と呼ばれる素数の分布に関する評価を使いますが、多くの準備が必要なので証明は省略します。

定理 4.3.5. 3 以上の正の整数 N に対して

$$\frac{\phi(N)}{N} \geq \frac{1}{e^{\gamma}\log\log N + \frac{3}{\log\log N}}$$

が成り立つ。ここで、γ はオイラー定数と呼ばれる定数で $\gamma = 0.57721566\cdots$ である。

$\log\log N$ は N が巨大な数だったとしてもかなり小さくなるので、$\phi(N)/N$ は実用上はほぼ定数で下から抑えられるといっても不都合はないでしょう。定理 4.3.5 は証明しませんが、参考にかなり荒い評価の証明を与えておきます。$N = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ と素因数分解されているとき、

$$\frac{\phi(N)}{N} = \left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_r}\right)$$

$$\geq \left(1-\frac{1}{2}\right)\left(1-\frac{1}{3}\right)\cdots\left(1-\frac{1}{r+1}\right) = \frac{1}{r+1}$$

となります。$N = p_1^{e_1}p_2^{e_2}\cdots p_r^{e_r} \geq 2^r$ より $r \leq \log_2 N$ なので、

$$\frac{\phi(N)}{N} \geq \frac{1}{\log_2 N + 1} \tag{4.3}$$

となります。定理 4.3.5 よりかなり弱い評価ですが、Shor のアルゴリズムが古典的なアルゴリズムよりある意味で高速であることを示すには十分な評価になっています。

4.4　良い性質を持つ元の個数

Shor のアルゴリズムでは、N, x に対して N を法とする x の位数 r を求め、それを利用して N の非自明な約数を見つけます。具体的には r が偶数の場合に、$\gcd(N, x^{r/2} \pm 1)$ が N の約数の候補になりました。この方法で約数が求まるためには $x^{r/2}+1$ が N で割り切れないという条件が必要になりますが、一方で条件を満たさない x も多く存在します。

x に必要な条件は

(1) $1 < x < N$ は N と互いに素な整数
(2) x の位数 r は偶数
(3) $x^{r/2}+1$ が N で割り切れない

の三つになります。この条件があれば約数が求まることは補題 5.1.1 で示すとして、ここでは、そのような条件を満たす $1 < x < N$ が十分多く存在することを示します。

位数がいつ偶数になるか、$[x]^{r/2}$ としてどんな元があり得るかを知っておく必要があります。中国剰余定理 (定理 4.2.2) を使って N が素数の場合に帰着させたいので、まずは N が奇素数の場合の元の位数について考えます。

N を奇素数とし、$\mathbb{Z}/N\mathbb{Z}$ の原始根 x_0 を一つ取り固定します。原始根の定義 (定義 4.1.18) から x_0 の位数は $N-1$ となります。命題 4.1.20 より、$(\mathbb{Z}/N\mathbb{Z})^\times$ の元は

$$x_0^0 = [1], x_0, x_0^2, \ldots, x_0^{N-2}$$

で重複なくすべて尽くされていることに注意しましょう。

ここからしばらくは、$(\mathbb{Z}/N\mathbb{Z})^\times$ の元を位数が何回 2 で割り切れるかで類別し、個数を数えることを目指します。

補題 4.4.1. *d を $N-1$ の約数とする。$x^d = [1]$ となる $x \in \mathbb{Z}/N\mathbb{Z}$ の個数はちょうど d 個である。*

証明. 任意の整数 $0 \leq n < N-1$ に対して、

$$(x_0^n)^d = [1] \Leftrightarrow x_0^{nd} = [1]$$

$$\Leftrightarrow nd \text{ は } N-1 \text{ で割り切れる} \qquad \text{(命題 4.1.15)}$$
$$\Leftrightarrow n \text{ は } (N-1)/d \text{ で割り切れる}$$

が成り立つ。$(\mathbb{Z}/N\mathbb{Z})^\times = \left\{x_0^0, x_0, x_0^2, \ldots, x_0^{N-2}\right\}$ なので、$x^d = [1]$ となる $x \in \mathbb{Z}/N\mathbb{Z}$ の個数は 0 以上 $N-2$ 以下の $(N-1)/d$ の倍数の個数と等しく、$(N-1)/((N-1)/d) = d$ 個となる。 □

補題 4.4.2. $x \in \mathbb{Z}/N\mathbb{Z}$ とし、x の位数を r とする。$N-1 = 2^k d$, $r = 2^{k'} d'$ (k, k' は非負整数、d, d' は奇数) と表す。このとき、任意の整数 $s \geq 0$ に対して、以下の 2 条件は同値である。

(1) $k' \leq s$
(2) $x^{2^s d} = [1]$

証明. 命題 4.1.15 とフェルマーの小定理 (定理 4.1.16) から r は $N-1$ を割り切るので、d' は d を割り切る。再び命題 4.1.15 より、任意の整数 $s \geq 0$ に対して

$$k' \leq s \Leftrightarrow r = 2^{k'} d' \text{ は } 2^s d \text{ を割り切る}$$
$$\Leftrightarrow x^{2^s d} = [1]$$

となる。 □

非負整数 j に対して

$$(\mathbb{Z}/N\mathbb{Z})_{2^j} \coloneqq \left\{x \in (\mathbb{Z}/N\mathbb{Z})^\times \;\middle|\; x \text{ の位数は } 2^j \text{ で割り切れ、} 2^{j+1} \text{ で割り切れない}\right\}$$

と定めます。(これはここだけの記号です。) この集合の元の個数 $|(\mathbb{Z}/N\mathbb{Z})_{2^j}|$ を求めましょう。

例 4.4.3. $N = 13$ の場合を考える。$(\mathbb{Z}/13\mathbb{Z})^\times$ の各元の位数は表 4.3 のようになっている。表の最後の行は、位数が 2 で何回割り切れるかを表している。

剰余類	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
位数	1	12	3	6	4	12	12	4	3	6	12	2
j	0	2	0	1	2	2	2	2	0	1	2	1

表 4.3: $(\mathbb{Z}/13\mathbb{Z})^\times$ の元の位数と位数が 2 で割り切れる回数

したがって、

$$(\mathbb{Z}/N\mathbb{Z})_{2^0} = \{[1], [3], [9]\},$$
$$(\mathbb{Z}/N\mathbb{Z})_{2^1} = \{[4], [10], [12]\},$$
$$(\mathbb{Z}/N\mathbb{Z})_{2^2} = \{[2], [5], [6], [7], [8], [11]\}$$

となり、それぞれの元の個数は $3, 3, 6$ となる。

補題 4.4.4. 補題 4.4.2 と同様に $N-1=2^k d$ と表し、$0\leq j\leq k$ とする。このとき、

$$|(\mathbb{Z}/N\mathbb{Z})_{2^j}|=\begin{cases}\frac{N-1}{2^k} & (j=0)\\ \frac{N-1}{2^{k-j+1}} & (j\neq 0)\end{cases}$$

となる。

証明. まず $j>0$ の場合を考える。補題 4.4.2 より位数が 2 で何回割り切れるかは $x^{2^j d}=[1]$ という形の方程式の解かどうかで判別できるので、

$$(\mathbb{Z}/N\mathbb{Z})_{2^j}=\left\{x\in\mathbb{Z}/N\mathbb{Z}\;\middle|\; x^{2^j d}=[1]\text{ かつ }x^{2^{j-1}d}\neq[1]\right\}$$

となる。$x\in(\mathbb{Z}/N\mathbb{Z})^\times$ が $x^{2^{j-1}d}=[1]$ を満たしているなら $x^{2^j d}=[1]$ も満たすということに注意する。補題 4.4.1 から $x^{2^j d}=[1]$ と $x^{2^{j-1}d}=[1]$ の解の個数はそれぞれ $2^j d$ と $2^{j-1}d$ と分かるので、

$$|(\mathbb{Z}/N\mathbb{Z})_{2^j}|=2^j d-2^{j-1}d=2^{j-1}d=\frac{N-1}{2^{k-j+1}}$$

となる。

$j=0$ の場合も同様に

$$|(\mathbb{Z}/N\mathbb{Z})_{2^j}|=|\left\{x\in\mathbb{Z}/N\mathbb{Z}\;\middle|\; x^d=[1]\right\}|=d=(N-1)/2^k$$

となる。 □

目的は、約数を求めるために必要な、$x^{r/2}\neq[-1]$ となる $x\in(\mathbb{Z}/N\mathbb{Z})^\times$ の個数を、N が合成数の場合に数えることでした。r が x の位数であれば $(x^{r/2})^2=x^r=[1]$ が成り立っています。N が素数の場合には 2 乗して $[1]$ になる元が $[\pm1]$ しかないことを示しておきましょう。

補題 4.4.5. $x\in\mathbb{Z}/N\mathbb{Z}$ とする。$x^2=[1]$ ならば $x=[1]$ または $[-1]$ である。

証明. $x^2=[1]$ と仮定すると、$(x-[1])(x+[1])=[0]$ となる。よって、$x-[1]$ か $x+[1]$ のいずれか一方は $\mathbb{Z}/N\mathbb{Z}$ の可逆ではない元になるが、N は素数なので $[0]$ しかありえない。したがって、$x=[1]$ または $[-1]$ となる。 □

次に $N=ab$ (a,b は相異なる奇素数) の場合を考えましょう。$(\mathbb{Z}/N\mathbb{Z})^\times$ に対する中国剰余定理 (定理 4.3.1) の全単射

$$\Phi\colon(\mathbb{Z}/N\mathbb{Z})^\times\to(\mathbb{Z}/a\mathbb{Z})^\times\times(\mathbb{Z}/b\mathbb{Z})^\times$$

を用いて、Shor のアルゴリズムがうまくいくような $(\mathbb{Z}/N\mathbb{Z})^\times$ の元の個数を評価します。具体的には、位数 r が偶数かつ $x^{r/2}\neq[-1]_N$ となる x の個数を調べるのですが、これらの条件を Φ を通して $(\mathbb{Z}/a\mathbb{Z})^\times\times(\mathbb{Z}/b\mathbb{Z})^\times$ 側の言葉に翻訳します。実数や複素数や N が素数の場合と異なり、$x^2=[1]_N$ を満たす $x\in\mathbb{Z}/N\mathbb{Z}$ が四つ存在することに注意してください。

例 4.4.6. $N = 21 = 3 \cdot 7$ の場合を考える。まずは 2 乗して $[1]_N$ を満たす元が何かをみておこう。補題 4.4.5 より、$\mathbb{Z}/3\mathbb{Z}$ の元で 2 乗して $[1]_3$ になるのは、$[1]_3, [-1]_3$ の二つであり、$\mathbb{Z}/7\mathbb{Z}$ の元で 2 乗して $[1]_7$ になるのは、$[1]_7, [-1]_7$ の二つである。よって、中国剰余定理より $\mathbb{Z}/N\mathbb{Z}$ の元で 2 乗して $[1]_N$ になるのは四つあると分かり、具体的に計算すると表 4.4 のようになる。

$\mathbb{Z}/N\mathbb{Z}$	$[1]_N$	$[8]_N$	$[13]_N$	$[-1]_N$
$\mathbb{Z}/3\mathbb{Z}$	$[1]_3$	$[-1]_3$	$[1]_3$	$[-1]_3$
$\mathbb{Z}/7\mathbb{Z}$	$[1]_7$	$[1]_7$	$[-1]_7$	$[-1]_7$

表 4.4: 2 乗して $[1]_N$ になる元

さて、知りたいのは位数 r の元 $[x]_N \in (\mathbb{Z}/N\mathbb{Z})^\times$ に対して $[x]_N^{r/2}$ がどうなるかだった。このとき、$[x]_N, [x]_3, [x]_7$ の位数と $[x]_N^{r/2}$ をまとめると表 4.5 となる。ただし、$[x]_N^{r/2}$ の行の空白は r が奇数であることを意味する。

$[x]_N$	$[1]$	$[4]$	$[16]$	$[8]$	$[2]$	$[11]$	$[13]$	$[10]$	$[19]$	$[20]$	$[5]$	$[17]$
$[x]_N$ の位数 r	1	3	3	2	6	6	2	6	6	2	6	6
$[x]_3$ の位数	1	1	1	2	2	2	1	1	1	2	2	2
$[x]_7$ の位数	1	3	3	1	3	3	2	6	6	2	6	6
$[x]_N^{r/2}$				$[8]$	$[8]$	$[8]$	$[13]$	$[13]$	$[13]$	$[-1]$	$[-1]$	$[-1]$

表 4.5: $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ の位数と $x^{r/2}$ の対応

表は $[x]_N^{r/2}$ の値が同じものが並ぶように、また $[x]_3, [x]_7$ の位数が同じものが並ぶように配置している。ここから、$[x]_3, [x]_7$ の位数の 2 で割り切れる回数によって $[x]_N^{r/2}$ が決まることが読み取れる。特に、r が偶数で $[x]_N^{r/2} \neq [-1]$ となるのは、$[x]_3, [x]_7$ の位数の 2 で割り切れる回数が異なるときと推測できる。

補題 4.4.7. $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ とし、$\Phi(x) = (x_a, x_b)$ によって $x_a \in (\mathbb{Z}/a\mathbb{Z})^\times, x_b \in (\mathbb{Z}/b\mathbb{Z})^\times$ を定める。x, x_a, x_b の位数をそれぞれ r, s, t と置く。また、$x_a \in (\mathbb{Z}/a\mathbb{Z})_{2^i}$, $x_b \in (\mathbb{Z}/b\mathbb{Z})_{2^j}$ となる整数 i, j を取る。このとき、以下の 2 条件は同値である。

(1) $i \neq j$ である。
(2) r は偶数かつ $x^{r/2} \neq [-1]_N$ である。

証明. 定理 4.2.5 から $r = \mathrm{lcm}(s, t)$ なので、$i \neq j$ ならば r は偶数である。よって、定理を示すためには r は偶数と仮定してよい。

r は x の位数なので、$x_a^r = [1]_a$ と $x_b^r = [1]_b$ が成り立つ。また、$x^{r/2} \neq [1]_N$ なので $x_a^{r/2} = [1]_a$

と $x_b^{r/2} = [1]_b$ が同時に成り立つことはない。したがって、補題 4.4.5 から $x_a^{r/2}$ と $x_b^{r/2}$ の組み合わせとしてあり得るのは

(1) $x_a^{r/2} = [-1]_a$ かつ $x_b^{r/2} = [1]_b$
(2) $x_a^{r/2} = [1]_a$ かつ $x_b^{r/2} = [-1]_b$
(3) $x_a^{r/2} = [-1]_a$ かつ $x_b^{r/2} = [-1]_b$ ($\Leftrightarrow x^{r/2} = [-1]_N$)

の 3 通りである。(3) の括弧内の同値性は定理 4.2.2 から従う。

$x^{r/2} = [-1]_N$ と $i = j$ が同値であることを示そう。命題 4.1.15 から $x_a^{r/2} = [1]_a$ と $r/2$ が s で割り切れることは同値である。また、x_b, t に対しても同様のことが成り立つ。したがって、先ほどの分類から

$$\begin{aligned} x_a^{r/2} = [-1]_a \text{ かつ } x_b^{r/2} = [-1]_b &\Leftrightarrow r/2 = \mathrm{lcm}(s,t)/2 \text{ が } s,t \text{ の両方で割り切れない} \\ &\Leftrightarrow \max(i,j) - 1 < i \text{ かつ } \max(i,j) - 1 < j \\ &\Leftrightarrow i = j \end{aligned}$$

となる。$\mathrm{lcm}(s,t)$ の 2 で割り切れる回数が $\max(i,j)$ と等しいことに注意。以上で、$x^{r/2} = [-1]_N$ と $i = j$ が同値であることが示された。 □

補題 4.4.7 の条件を満たす $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ が十分たくさんあることを計算で確かめましょう。定理 4.3.4 から $|(\mathbb{Z}/N\mathbb{Z})^\times| = \phi(N) = (a-1)(b-1)$ ですが、その半数以上が補題 4.4.7 の条件を満たすことが分かります。

定理 4.4.8. 補題 4.4.7 の 2 条件を満たす $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ の個数は $(a-1)(b-1)/2$ 以上である。

証明. 補題 4.4.7 の記号で $i = j$ となるような x の個数を求める。$a - 1 = 2^k a'$, $b - 1 = 2^l b'$ (k, l は整数、a', b' は奇数) と表す。a, b は奇素数なので $k, l \neq 0$ である。このとき、$i = j$ となるような $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ の個数は

$$\begin{aligned} &\underbrace{|(\mathbb{Z}/a\mathbb{Z})_{2^0}||(\mathbb{Z}/b\mathbb{Z})_{2^0}|}_{i=j=0 \text{ の場合}} + \sum_{n=1}^{\min(k,l)} \underbrace{|(\mathbb{Z}/a\mathbb{Z})_{2^n}||(\mathbb{Z}/b\mathbb{Z})_{2^n}|}_{i=j=n \text{ の場合}} \\ &= \frac{a-1}{2^k}\frac{b-1}{2^l} + \sum_{n=1}^{\min(k,l)} \frac{a-1}{2^{k-n+1}}\frac{b-1}{2^{l-n+1}} \qquad \text{(補題 4.4.4)} \\ &= (a-1)(b-1)\left(\frac{1}{2^{k+l}} + \sum_{n=1}^{\min(k,l)} \frac{1}{2^{k+l-2n+2}}\right) \\ &= \frac{(a-1)(b-1)}{2^{k+l}}\left(1 + \sum_{n=1}^{\min(k,l)} 2^{2n-2}\right) \end{aligned}$$

$$\leq \frac{(a-1)(b-1)}{2^{k+l}}\left(1+\sum_{m=0}^{k+l-2}2^m\right) \qquad (2\min(k,l)\leq k+l)$$
$$= \frac{(a-1)(b-1)}{2^{k+l}}\left(1+2^{k+l-1}-1\right) = \frac{(a-1)(b-1)}{2}$$

となる。これが補題 4.4.7 の 2 条件を満たさない x の個数の上からの評価なので、条件を満たすものは $(a-1)(b-1)/2$ 個以上となる。 □

Shor のアルゴリズムで x の位数 r を求める部分がありますが、その部分は量子アルゴリズムを用いており、ごく低確率で $x^r=[1]$ を満たすが位数でない r が得られることがあります。そのような r を位数の代わりに使ったとしても、$x^{r/2}$ の値は変わらず、約数を計算するうえでは問題ないことを示しておきましょう。

命題 4.4.9. $x\in(\mathbb{Z}/N\mathbb{Z})^\times$ とし、r_0 を x の位数とする。正の整数 r を r_0 の奇数倍とし、r_0 は偶数と仮定する。このとき、$x^{r/2}=x^{r_0/2}$ が成り立つ。

証明. $\Phi(x)=(x_a,x_b)$ により x_a,x_b を定める。$x_a^{r_0}=[1]_a$ なので、補題 4.4.5 より $x_a^{r_0/2}=[1]_a$ または $[-1]_a$ である。r/r_0 は奇数なので、どちらの場合でも $x_a^{r/2}=(x_a^{r_0/2})^{r/r_0}=x_a^{r_0/2}$ となる。b に対しても同様なので、

$$\Phi(x^{r/2})=(x_a^{r/2},x_b^{r/2})=(x_a^{r_0/2},x_b^{r_0/2})=\Phi(x^{r_0/2})$$

となり、中国剰余定理 (定理 4.2.2) より $x^{r/2}=x^{r_0/2}$ を得る。 □

4.5 連分数と Legendre の定理

量子や古典コンピュータでは基本的に数は 2 進数で表されます。したがって、例えば $0.33333\cdots$ のような小数は有限桁の 2 進小数では表せず、どうしても近似する必要が出てきます[*2]。ここでは、近似を行った後の数から、近似する前の数を復元できるための条件を考えます。既約分数とは分数 p/q であって p,q が互いに素かつ $q>0$ となるものと約束しましょう。

例として次のような問題を考えてみます。N を正の整数とし p/q を $0<p<q\leq N$ となる既約分数とします。N は既知、p/q は未知であるとし、p/q を 10 進数で小数点以下 $n+1$ 桁目を四捨五入したものを $f_n(p/q)$ と表すことにします。このとき、n をどれくらい大きく取れば $f_n(p/q)$ から確実に p/q が復元できるでしょうか。

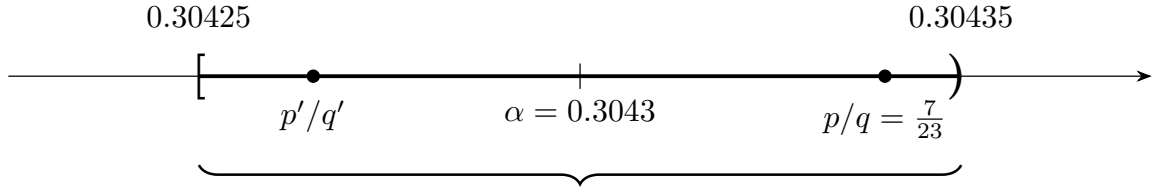
この問題は f_n が単射になるのはいつかと言い換えることができます。p/q の候補は $N(N-1)/2$ 通りあり、小数点以下が n 桁の 0 以上 1 以下の小数は 10^n 通り程度あるので、$N(N-1)/2\leq 10^n$

[*2] もちろん分数にしてから分母分子の 2 進数のペアで表現すれば近似せずに済みますが、量子回路モデルなどで扱いづらくなり、アルゴリズムが複雑化したり効率が悪くなる可能性があります。

でなければ f_n は単射にはなりえません。直感的には f_n の値が極端に偏ったりはしないと思われるので、10^n が $N(N-1)/2$ の数倍程度あれば、f_n での行き先が被ることはなさそうです。実はこの条件はほぼ正解に近く、$N^2 < 10^n$ であれば、$f_n(p/q)$ から p/q を一通りに復元することができます。実際、異なる既約分数 $p/q, p'/q'$ が $\alpha := f_n(p/q) = f_n(p'/q')$ を満たすと仮定すると、図 4.1 のような位置関係になり、

$$\left|\alpha - \frac{p}{q}\right| \leq \frac{1}{2\cdot 10^n}, \quad \left|\alpha - \frac{p'}{q'}\right| \leq \frac{1}{2\cdot 10^n}, \quad \left|\frac{p}{q} - \frac{p'}{q'}\right| < \frac{1}{10^n}$$

といった不等式評価が成り立ちます。よって、三つ目の不等式から $10^n \leq qq' \leq N^2$ が成り立たなければいけません。分母の大きさに制約がある状況では、二つの既約分数は一定の距離より近づくことができないというのがポイントになります。



小数第 5 位を四捨五入して α になる範囲

図 4.1: $p/q = 7/23, \alpha = f_4(p/q)$ の場合

命題 4.5.1. N を正の整数、$\alpha \in \mathbb{R}$ とする。このとき、

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{2N^2} \text{ かつ } q \leq N$$

を満たす既約分数 $p/q \in \mathbb{Q}$ は存在すればただ一つである。

証明. $p/q, p'/q' \in \mathbb{Q}$ のどちらも条件を満たす既約分数とすると、

$$\left|\frac{p}{q} - \frac{p'}{q'}\right| \leq \left|\frac{p}{q} - \alpha\right| + \left|\alpha - \frac{p'}{q'}\right| < 2\cdot\frac{1}{2N^2} = \frac{1}{N^2}$$

となる。また、$q, q' \leq N$ より

$$\left|\frac{p}{q} - \frac{p'}{q'}\right| = \frac{|pq' - qp'|}{qq'} \geq \frac{|pq' - qp'|}{N^2}$$

となる。二つの不等式を合わせると $|pq' - qp'| < 1$ となるが、これは $pq' - qp' = 0$、つまり $p/q = p'/q'$ を意味する。 □

先ほどは 10 進数で考えていましたが、2 進数で考えても同様です。つまり、$N^2 < 2^n$ であれば、p/q の 2 進数の小数点以下 $n+1$ 桁目を 0 捨 1 入したものから p/q を復元することができま

す。これが、1 章の Shor のアルゴリズムで $q = 2^n > N^2$ となるように q を取った理由の一つになります。

次に、命題 4.5.1 のような条件を満たしているときに、α から実際に p/q を復元する方法を与えましょう。これは、$q \leq N$ を満たす既約分数 p/q で α を近似するもののうち、最もよく近似するものを求める問題といえます。実数 α の良い近似になる既約分数を求める方法として、連分数展開という手法を紹介します。

定義 4.5.2 (連分数)**.** 実数列 $(a_0, a_1, \ldots, a_s)$ $(a_1, \ldots, a_s > 0)$ に対して、

$$[a_0; a_1, a_2, \ldots, a_s] \coloneqq a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_{s-1} + \cfrac{1}{a_s}}}}}$$

と定める。ただし、$s = 0$ の場合は $[a_0;] \coloneqq a_0$ とする。右辺の形の分数を**連分数**と呼ぶ。

例 4.5.3.

$$\begin{aligned}
[1; 2] &= 1 + \frac{1}{2} = \frac{3}{2} = 1.5, \\
[1; 2, 2] &= 1 + \cfrac{1}{2 + \cfrac{1}{1}} = \frac{7}{5} = 1.4, \\
[1; 2, 2, 2] &= 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2}}} = \frac{17}{12} = 1.416666\cdots, \\
[1; 2, 2, 2, 2] &= \frac{41}{29} = 1.41379310\cdots
\end{aligned}$$

これらは 2 の個数を増やしていくと $\sqrt{2}$ に収束する連分数として知られている。得られた既約分数は $\sqrt{2}$ をよく近似しており、例えば

$$\left|\sqrt{2} - \frac{41}{29}\right| = 0.0004204\cdots < \frac{1}{2 \cdot 29^2}$$

のように 41/29 の分母の大きさに比べて誤差が非常に小さくなっている。

以下、$(a_0, a_1, \ldots, a_s)$ を固定して、途中で打ち切った連分数 $[a_0; a_1, a_2, \ldots, a_n]$ $(n \leq s)$ たちの関係を考えます。数列 $(p_n)_{-1 \leq n \leq s}, (q_n)_{-1 \leq n \leq s}$ を

$$\begin{matrix} p_{-1} = 1 \\ q_{-1} = 0 \end{matrix}, \quad \begin{matrix} p_0 = a_0 \\ q_0 = 1 \end{matrix}, \quad \begin{matrix} p_n = a_n p_{n-1} + p_{n-2} \\ q_n = a_n q_{n-1} + q_{n-2} \end{matrix} \quad (1 \leq n \leq s)$$

で定めます。また、$\alpha \coloneqq [a_0; a_1, \ldots, a_s]$ と置いておきます。線形な漸化式で定義しているので、行列を使って

$$\begin{aligned}\begin{pmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{pmatrix} &= \begin{pmatrix} p_{n-1} & p_{n-2} \\ q_{n-1} & q_{n-2} \end{pmatrix} \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} \end{aligned} \tag{4.4}$$

と表せます。p_n/q_n は連分数 $[a_0; a_1, a_2, \ldots, a_n]$ を表す既約分数になるのですが、これを示す前に $q_n > 0$ であることを確かめておきましょう。

補題 4.5.4. 任意の $0 \leq n \leq s$ に対して $q_n \geq q_{n-1}, q_n > 0$ が成り立つ。また、$q_n = q_{n-1}$ となるのは $n = 1$ かつ $a_1 = 1$ の場合に限る。

証明. n に関する数学的帰納法で示す。$n = 0$ の場合は定義 $(q_0 = 1, q_{-1} = 0)$ から明らかである。$n \leq k$ $(0 \leq k < s)$ の場合に成り立つと仮定して $n = k+1$ の場合を示す。$q_k, a_{k+1} > 0, q_{k-1} \geq 0$ より、$q_{k+1} = a_{k+1}q_k + q_{k-1} \geq q_k > 0$ となり確かに成り立っている。ここで、$a_{k+1}q_k + q_{k-1} \geq q_k$ の等号が成立するのは、$q_{k-1} = 0$ かつ $a_{k+1} = 1$ の場合、つまり $k = 0$ かつ $a_1 = 1$ の場合に限る。 □

補題 4.5.5. $0 \leq n \leq s$ とすると、$\frac{p_n}{q_n} = [a_0; a_1, a_2, \ldots, a_n]$ が成り立つ。

証明. n に関する数学的帰納法で示す。$n = 0, 1$ の場合は定義から明らかである。$n \leq k$ $(1 \leq k < s)$ の場合に成り立つと仮定して $n = k + 1$ の場合を示す。連分数の定義より

$$[a_0; a_1, a_2, \ldots, a_k, a_{k+1}] = [a_0; a_1, a_2, \ldots, a_{k-1}, a_k + 1/a_{k+1}]$$

なので、数学的帰納法の仮定から

$$\begin{aligned}[a_0; a_1, a_2, \ldots, a_{k+1}] &= [a_0; a_1, a_2, \ldots, a_{k-1}, a_k + 1/a_{k+1}] \\ &= \frac{(a_k + 1/a_{k+1})p_{k-1} + p_{k-2}}{(a_k + 1/a_{k+1})q_{k-1} + q_{k-2}} \\ &= \frac{(a_k a_{k+1} + 1)p_{k-1} + a_{k+1}p_{k-2}}{(a_k a_{k+1} + 1)q_{k-1} + a_{k+1}q_{k-2}} \\ &= \frac{a_{k+1}p_k + p_{k-1}}{a_{k+1}q_k + q_{k-1}} = \frac{p_{k+1}}{q_{k+1}}\end{aligned}$$

となる。 □

ここからは $a_0, a_1, \ldots, a_s$ はすべて整数と仮定します。p_n/q_n が $\alpha = p_s/q_s$ の良い近似であることを示すために、数列 $(p_n/q_n)_n$ が α に近づいていくことをみましょう。

補題 4.5.6. (1) $p_n q_{n-1} - q_n p_{n-1} = (-1)^{n+1}$ が成り立つ。特に p_n, q_n は互いに素であり、p_n/q_n は $[a_0; a_1, a_2, \ldots, a_n]$ を表す既約分数である。

(2) 数列 p_n/q_n は偶数項が単調増加、奇数項が単調減少である。より詳細に、

$$\frac{p_0}{q_0} < \frac{p_2}{q_2} < \frac{p_4}{q_4} < \cdots < \frac{p_s}{q_s} < \cdots < \frac{p_5}{q_5} < \frac{p_3}{q_3} < \frac{p_1}{q_1}$$

が成り立つ。

証明. 2×2 行列 $X = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ の行列式を $\det(X) = ad - bc$ で定義すると、任意の 2×2 行列 Y, Z に対して $\det(YZ) = \det(Y)\det(Z)$ が成り立つ。(4.4) の両辺の行列式を取り、この公式を繰り返し用いれば、

$$\begin{aligned} p_n q_{n-1} - q_n p_{n-1} &= \det \begin{pmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{pmatrix} \\ &= \det \left(\begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} \right) \\ &= \det \begin{pmatrix} a_0 & 1 \\ 1 & 0 \end{pmatrix} \det \begin{pmatrix} a_1 & 1 \\ 1 & 0 \end{pmatrix} \cdots \det \begin{pmatrix} a_n & 1 \\ 1 & 0 \end{pmatrix} \\ &= (-1)^{n+1} \end{aligned}$$

となり、(1) が分かる。

$2 \leq n \leq s$ とすると、定義と (1) から

$$\begin{aligned} \frac{p_n}{q_n} - \frac{p_{n-2}}{q_{n-2}} &= \frac{p_n q_{n-2} - q_n p_{n-2}}{q_n q_{n-2}} \\ &= \frac{(a_n p_{n-1} + p_{n-2}) q_{n-2} - (a_n q_{n-1} + q_{n-2}) p_{n-2}}{q_n q_{n-2}} \\ &= \frac{a_n (p_{n-1} q_{n-2} - q_{n-1} p_{n-2})}{q_n q_{n-2}} \\ &= \frac{(-1)^n a_n}{q_n q_{n-2}} \end{aligned}$$

となる。よって、n が偶数なら $p_n/q_n > p_{n-2}/q_{n-2}$ が、奇数なら $p_n/q_n < p_{n-2}/q_{n-2}$ が成り立つ。また、(1) から

$$\frac{p_s}{q_s} - \frac{p_{s-1}}{q_{s-1}} = \frac{p_s q_{s-1} - q_s p_{s-1}}{q_s q_{s-1}} = \frac{(-1)^{s+1}}{q_s q_{s-1}}$$

となるので、s が偶数なら $p_s/q_s < p_{s-1}/q_{s-1}$ が、奇数なら $p_s/q_s > p_{s-1}/q_{s-1}$ が成り立つ。以上をまとめると (2) の不等式が得られる。 □

補題 4.5.6 から $0 \leq n < s$ なら

$$\left| \frac{p_n}{q_n} - \alpha \right| \leq \left| \frac{p_n}{q_n} - \frac{p_{n+1}}{q_{n+1}} \right| = \frac{|p_n q_{n+1} - q_n p_{n+1}|}{q_n q_{n+1}} = \frac{1}{q_n q_{n+1}} \leq \frac{1}{q_n^2}$$

という評価が成り立つことが分かります。p_n/q_n はこのようなタイプの不等式を満たす既約分数のうち、いくつかの意味で最適なものになっています。最適性の一つであり、本テキストで必要となる以下の補題を示しましょう。

補題 4.5.7. N を正の整数とし、$q_n \leq N$ を満たす最大の整数 n を取る。$q \leq N$ を満たす既約分数 $p/q \in \mathbb{Q}$ を動かしたときの $|q\alpha - p|$ の最小値は $|q_n\alpha - p_n|$ である。

証明. $n = s$ の場合は $|q\alpha - p|$ の最小値は $|q_s\alpha - p_s| = 0$ である。$n < s$ の場合を考えよう。

$p/q \in \mathbb{Q}$ を $q \leq N$ を満たす既約分数とする。補題 4.5.6 の (1) から $\det \begin{pmatrix} p_n & p_{n+1} \\ q_n & q_{n+1} \end{pmatrix} = (-1)^{n+1}$ なので、$\begin{pmatrix} p_n & p_{n+1} \\ q_n & q_{n+1} \end{pmatrix}$ の逆行列は

$$\begin{pmatrix} p_n & p_{n+1} \\ q_n & q_{n+1} \end{pmatrix}^{-1} = (-1)^{n+1} \begin{pmatrix} q_{n+1} & -p_{n+1} \\ -q_n & p_n \end{pmatrix}$$

となり成分はすべて整数である。よって、

$$\begin{pmatrix} p_n & p_{n+1} \\ q_n & q_{n+1} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} p \\ q \end{pmatrix}$$

を満たす整数 a, b が存在する。

$b = 0$ なら $p/q = p_n/q_n$ となるので、$b \neq 0$ の場合を考える。b は整数なので $b(1-b) \leq 0$ となり、$q \leq N < q_{n+1}$ と合わせると、

$$\begin{aligned} abq_n &= b(q - bq_{n+1}) \\ &= bq - b^2q_{n+1} \\ &< bq - b^2q = qb(1-b) \leq 0 \end{aligned}$$

を得る。よって、$abq_n < 0$ かつ $q_n > 0$ なので、a と b は異なる符号を持つ。

補題 4.5.6 の (2) から $\alpha - p_n/q_n$ と $\alpha - p_{n+1}/q_{n+1}$ は異なる符号を持つか後者が 0 なので、$a(q_n\alpha - p_n)$ と $b(q_{n+1}\alpha - p_{n+1})$ は同じ符号を持つ、または後者は 0 である。よって、

$$\begin{aligned} |q\alpha - p| &= |(aq_n + bq_{n+1})\alpha - (ap_n + bp_{n+1})| \\ &= |a(q_n\alpha - p_n) + b(q_{n+1}\alpha - p_{n+1})| \\ &= |a(q_n\alpha - p_n)| + |b(q_{n+1}\alpha - p_{n+1})| \geq |q_n\alpha - p_n| \end{aligned}$$

となる。以上より、$|q\alpha - p|$ の最小値は $|q_n\alpha - p_n|$ である。 □

補題 4.5.7 で $|q\alpha - p|$ が最小となる p/q は多くの場合 p_n/q_n ただ一通りですが、$n = s-1$ の場合に $(p_{n+1} - p_n)/(q_{n+1} - q_n)$ でも最小値を取ることがあります。証明中の記号で $(a, b) = (-1, 1)$

の場合に対応し、これ以外に例外はありません。$a_s > 1$ のときに $\alpha = [a_0; a_1, \ldots, a_s] = [a_0; a_1, \ldots, a_s - 1, 1]$ と 2 通りに表せるため、このような現象が発生します。

次の定理により、α のある意味での良い近似は、必ず連分数を途中で打ち切ったものによって得られることが保証されます。Shor のアルゴリズムの連分数を使う部分はこの定理を根拠としています。

定理 4.5.8 (Legendre の定理). 既約分数 $p/q \in \mathbb{Q}$ が $\left|\alpha - \frac{p}{q}\right| < \frac{1}{2q^2}$ を満たすとする。このとき、ある整数 $0 \leq n \leq s$ が存在して、$p/q = p_n/q_n$ となる。

証明. $q_n \leq q$ となる最大の整数 n を取る。補題 4.5.7 より、

$$|q_n\alpha - p_n| \leq |q\alpha - p| < \frac{1}{2q}$$

となる。よって、

$$\left|\frac{p_n}{q_n} - \frac{p}{q}\right| \leq \left|\frac{p_n}{q_n} - \alpha\right| + \left|\alpha - \frac{p}{q}\right| < \frac{1}{2qq_n} + \frac{1}{2q^2} \leq \frac{1}{qq_n},$$
$$\left|\frac{p_n}{q_n} - \frac{p}{q}\right| = \frac{|p_nq - q_np|}{q_nq}$$

となる。したがって、$|p_nq - q_np| < 1$ となり $p_n/q_n = p/q$ を得る。 □

命題 4.5.1 とまとめて Shor のアルゴリズムで使いやすい形にしておきましょう。

系 4.5.9. N を正の整数とする。既約分数 $p/q \in \mathbb{Q}$ が $\left|\alpha - \frac{p}{q}\right| < \frac{1}{2N^2}$ と $q \leq N$ を満たすとする。$q_n \leq N$ を満たす最大の n を取れば、$p/q = p_n/q_n$ となる。

証明. 仮定から

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{2N^2} \leq \frac{1}{2q^2}$$

となる。よって、Legendre の定理 (定理 4.5.8) から、$p_m/q_m = p/q$ となる m が存在する。

n の最大性から $m \leq n, q_m \leq q_n$ となる。よって、補題 4.5.7 から

$$\left|\alpha - \frac{p_n}{q_n}\right| \leq \frac{q_m}{q_n}\left|\alpha - \frac{p_m}{q_m}\right| < \frac{1}{2N^2}$$

となる。命題 4.5.1 より、$|\alpha - p'/q'| < 1/2N^2$ かつ $q' \leq N$ を満たす既約分数 p'/q' はただ一つなので、$p_n/q_n = p_m/q_m = p/q$ となる。 □

最後に、$\alpha \in \mathbb{Q}$ が与えられたときに、$\alpha = [a_0; a_1, \ldots, a_s]$ となる連分数を求めるアルゴリズムについて述べておきましょう。

定理 4.5.10 (有理数の**連分数展開**)**.** $p/q \in \mathbb{Q}$ を既約分数とする。$a_0, \ldots, a_s$ が整数であるような連分数 $[a_0; a_1, \ldots, a_s]$ で p/q と等しくなるものが存在する。また、この連分数を求めるために必要な割り算の回数は $2\log_2(q)$ 以下である。

証明. q に関する数学的帰納法で示す。$q = 1$ の場合は $a_0 = p, s = 0$ とすればよい。$q \leq k$ $(k \geq 1)$ に対して示されたと仮定して、$q = k+1$ の場合を示す。p を q で割った商と余りを a_0 と p' $(0 \leq p' < q)$ と置く。p, q は互いに素なので、p', q も互いに素である。よって、q/p' は既約分数である。q/p' に数学的帰納法の仮定を用いて連分数 $q/p' = [a_1; a_2, \ldots, a_s]$ を求める。そうすると

$$\begin{aligned}\frac{p}{q} &= a_0 + \frac{1}{\frac{q}{p'}} \\ &= a_0 + \frac{1}{[a_1; a_2, \ldots, a_s]} = [a_0; a_1, a_2, \ldots, a_s]\end{aligned}$$

となり、p/q を表す連分数が得られる。

以上より、このアルゴリズムで割り算を行う回数は s と等しい。q_n の漸化式から数学的帰納法で $q_n \geq \sqrt{2}^n$ $(n \geq 2)$ が示され、$q = q_s$ から $s \leq 2\log_2(q)$ を得る。 □

証明中では $q_n \geq \sqrt{2}^n$ という少し粗い評価から割り算の回数を見積もりましたが、フィボナッチ数列を使うことでより精密な評価ができます。q_s が最小になるのは $a_1 = a_2 = \cdots = a_s = 1$ のときで、そのとき q_s はフィボナッチ数列の第 $s+1$ 項目になります。したがって、q がフィボナッチ数列の $n+1$ 項目未満なら、p/q の連分数展開に必要な割り算は $n-1$ 回以下と見積もれます。

例 4.5.11. $q \leq 50$ であるような既約分数 p/q の小数点以下 5 桁目を四捨五入したものが 0.3043 だったとする。p/q を復元しよう。

$$0.3043 = \frac{3043}{10000}$$

この分数に定理 4.5.10 を適用する。各ステップでの分母分子は表 4.6 のようになり、

$$\frac{3043}{10000} = \cfrac{1}{3 + \cfrac{1}{3 + \cfrac{1}{2 + \cfrac{1}{39 + \cfrac{1}{11}}}}}$$

を得る。

分母	分子	商	余り	
10000	3043	3	871	$\frac{3043}{10000} = \frac{1}{3+\frac{871}{3043}}$
3043	871	3	430	$\frac{871}{3043} = \frac{1}{3+\frac{430}{871}}$
871	430	2	11	$\frac{430}{871} = \frac{1}{2+\frac{11}{430}}$
430	11	39	1	$\frac{11}{430} = \frac{1}{39+\frac{1}{11}}$

表 4.6: 連分数展開の途中計算

この連分数を途中で打ち切って得られる分数列 p_n/q_n は

$$\frac{p_0}{q_0} = \frac{0}{1}, \quad \frac{p_1}{q_1} = \frac{1}{3}, \quad \frac{p_2}{q_2} = \frac{3}{10}, \quad \frac{p_3}{q_3} = \frac{7}{23}, \quad \frac{p_4}{q_4} = \frac{276}{907}, \quad \frac{p_5}{q_5} = \frac{3043}{10000}$$

となる。分母が 50 以下で最大になっているのは p_3/q_3 なので、$p/q = p_3/q_3 = 7/23$ と分かる。$7/23 = 0.3043478\cdots$ であり確かに小数点以下 5 桁目を四捨五入したものが 0.3043 になっている。

第 5 章

Shor のアルゴリズム

この章では 1 章で紹介した Shor のアルゴリズムの詳細を解説します。必要な整数論の大部分はすでに 4 章で解説したので、量子アルゴリズムで位数を推定する部分を中心に詳しく扱います。位数を推定するために必要な量子フーリエ変換と呼ばれる操作がこの章の山場になります。

5.1 アルゴリズムの再考

量子回路モデルを定義し Shor のアルゴリズムの細部を述べる準備が整いました。N を合成数としますが、議論を簡単にするため $N = ab$ (a, b は異なる奇素数) とします。RSA 暗号ではこのような奇素数二つの積を用いるので、RSA 暗号を解読するという目的であればこれで十分です。一般の場合でも、定理 4.4.8 を証明し直す必要がある程度で議論はほとんど変わりませんが、N が偶数の場合にはあらかじめ 2 で割っておくなど簡単な前処理が必要になります。

Shor のアルゴリズムは N の非自明な約数を求めるというアルゴリズムでした。今の設定の場合、a または b が求めるべきものになります。Shor のアルゴリズムで量子アルゴリズムを使う部分は x の位数 r を求める部分だけで、残りは整数の基本的な演算で処理できます。位数 r を求める部分は後に回して、まずは残りのステップを確認していきましょう。

(1) $1 < x < N$ を取る。
(2) x と N が互いに素でなかったら、$\gcd(x, N)$ が N の非自明な約数を与えるので終了。
(3) (位数推定) 量子アルゴリズムで $[x]^r = [1]$ となる r を見つける。
(4) r が偶数で $[x]^{r/2} = [1]$ なら r を $r/2$ で置き換える、という操作を可能な限り繰り返す。
(5) r が奇数、または $[x]^{r/2} = [-1]$ ならうまくいかない x なので (1) に戻る。
(6) そうでなければ、$\gcd(x^{r/2} + 1, N)$ と $\gcd(x^{r/2} - 1, N)$ が N の素因数 a, b を与える。

(1), (2) 準備

まず位数を求める対象の整数 $1 < x < N$ を一つ選びます。x と N が互いに素にならない場合、つまり $\gcd(x, N) \neq 1$ の場合、$\gcd(x, N)$ が N の非自明な約数になります。これは、x が a か b の倍数になる場合で、おおよそ $(a + b)/N$ の確率で起こります。ごく低確率で運がよいケースですが、例外扱いで処理しておかないとこの後の計算で支障が出てきます。以下、x と N が互いに素、つまり x が a, b で割り切れない場合だけを考えます。

(4), (5), (6) 約数の計算

x の mod N での位数とは $[x]^r = [1]$ となる最小の正の整数のことでした。(3) の位数の推定では $[x]^r = [1]$ を満たす整数 r が得られますが、位数よりも大きい可能性が僅かながらあります。そのため (4) では、r が偶数かつ $[x]^{r/2} = [1]$ なら r を $r/2$ で置き換える、という操作を繰り返して余分な 2 の素因数を除去しています。この操作を行うと、最終的に r が奇数または $[x]^{r/2} \neq [1]$ が成り立ちます。置き換えた後の r も位数ではないかもしれませんが、位数の奇数倍にはなっています。命題 4.4.9 より、r が位数そのものでも奇数倍になっていたとしても $[x]^{r/2}$ は同じになるので、この後の処理には一切影響はありません。

r は以下のパターンに分けられます。

(a) r が奇数
(b) r が偶数かつ $[x]^{r/2} = [-1]$
(c) r が偶数かつ $[x]^{r/2} \neq [-1]$

定理 4.4.8 で示したように、(c) を満たす x は半数以上あります。したがって、ランダムに x を選んだ場合、1/2 以上の確率で (c) になります。(c) の条件から N の素因数が求まることを確かめておきましょう。

補題 5.1.1. *r が偶数かつ $[x]^{r/2} \neq [-1]$ なら、$\gcd(x^{r/2} + 1, N)$ と $\gcd(x^{r/2} - 1, N)$ が N の素因数 a, b を与える。*

証明. ここまでの設定と r の仮定から、$x^r - 1$ は N で割り切れ、$x^{r/2} - 1$ は N で割り切れない。また、仮定より $x^{r/2} + 1$ も N で割り切れない。

$$x^r - 1 = (x^{r/2} - 1)(x^{r/2} + 1)$$

で a は素数なので、a は $x^{r/2} - 1$ と $x^{r/2} + 1$ の少なくとも一方を割り切る。b に関しても同様である。$x^{r/2} - 1$ と $x^{r/2} + 1$ のどちらも $N = ab$ で割り切れないことと合わせて、

- $\gcd(x^{r/2} + 1, N) = a$ かつ $\gcd(x^{r/2} - 1, N) = b$

- $\gcd(x^{r/2}+1, N) = b$ かつ $\gcd(x^{r/2}-1, N) = a$

のどちらか一方が成り立つことが分かる。 □

(3) 位数の推定

残った位数 r を推定するステップを詳しくみていきましょう。m を $2^m \geq N$ となるように取り、$q = 2^n$ を N^2 より大きくなるように取っておきます。m は単に 0 から $N-1$ を 2 進数として表すために必要なビット数というだけで本質的ではありません。また、q は N^2 より大きければ位数の推定には支障はありませんが、量子ゲートの個数を見積もるときには、$N^2 < q < 2N^2$ となる q を取っていると仮定して計算を行うこととします。写像 f を $f(j) = x^j \bmod N$ (j は非負整数) で定めます。

(a) $n+m$ 量子ビットの初期状態 $|0\rangle^{\otimes n+m}$ から $\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle \otimes |0\rangle^{\otimes m}$ を作る。

(b) $U_f(|j\rangle \otimes |0\rangle^{\otimes m}) = |j\rangle \otimes |f(j)\rangle$ $(0 \leq \forall j < q)$ となる線形写像 U_f を実現する量子回路を作り、状態

$$U_f\left(\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle \otimes |0\rangle^{\otimes m}\right) = \frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle \otimes |f(j)\rangle$$

を作る。

(c) $\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle \otimes |f(j)\rangle$ の前半 n 量子ビットに (逆) 量子フーリエ変換を施し測定する。測定値を k とすると、k/q はある既約分数 k'/r $(0 \leq k' < r)$ の近似値となっている。

(d) k/q に連分数展開を使って k'/r を復元する。

(e) $[x]^r = [1]$ になっていれば r を位数[*1]と推定して終了。そうでなければ (a) に戻る。

(a) は次のステップで $f(j)$ をすべての j にわたって計算するための準備で、そのために 0 から $q-1$ が一様に重ね合わせになっている状態 $\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle$ を作っています。Hadamard ゲートを使えば実現でき、後ろの m 量子ビットは無関係なので省略して書くと、

$$\begin{aligned} H^{\otimes n}|0\rangle^{\otimes n} &= (H|0\rangle) \otimes (H|0\rangle) \otimes \cdots \otimes (H|0\rangle) \\ &= \frac{1}{\sqrt{2}^n}(|0\rangle + |1\rangle)^{\otimes n} = \frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle \end{aligned}$$

となります。例えば、$n = 2$ の場合は

$$H^{\otimes 2}|0\rangle^{\otimes 2} = (H|0\rangle) \otimes (H|0\rangle)$$

[*1] 実際にはこの r が位数でないこともあります。測定値が悪く、推定した r が実際の位数より大きくなることがあるからです。

$$= \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)$$
$$= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

のようになっています。

(e) では、求めた r が位数かチェックしています。r が実際の位数の約数になる場合、位数の倍数になる場合、まったく無関係な値になる場合などが発生します。r が位数の倍数だと $[x]^r = [1]$ が成り立ちチェックを素通りしますが、そのままでも Shor のアルゴリズムでは支障はないので位数と推定して終了します。

(b), (c), (d) について残りの節で解説していきます。

5.2 $f(j)$ の計算

$f(j) = x^j \bmod N$ を計算する量子回路を考えましょう。このステップは 1.2 節の手順 3 で述べた行列 A を作る部分に対応します。

定理 3.4.1 で述べたように、古典的なコンピュータで計算できる関数は量子回路でも計算可能です。したがって、$U_f(|j\rangle \otimes |0\rangle) = |j\rangle \otimes |f(j)\rangle$ となる線形写像 U_f を量子回路で実現することは難しくありません。ただし、うまい計算方法を使わないと必要な量子ゲートの数が膨大になってしまいます。

$x^j \bmod N$ を計算する最も単純な方法として

$$x^k \bmod N = ((x^{k-1} \bmod N) \cdot x) \bmod N$$

という漸化式を使う方法があります。この方法では、$x \bmod N, x^2 \bmod N, x^3 \bmod N, \ldots$ と順番に計算していくため、掛け算と余りの計算を q 回程度行うことになります。例えば N が 2^{2024} のような巨大な数だと、膨大な回数の掛け算と割り算が必要になってしまいます。

これに対して、以下のような劇的に高速化する方法が知られています。j を $j = j_1 j_2 \cdots j_n$ のように 2 進数で表します。つまり、

$$j = 2^{n-1} j_1 + 2^{n-2} j_2 + \cdots + 2 j_{n-1} + j_n$$

となっています。このとき、$\bmod N$ を使うと式が煩雑になるので剰余類 $[x]$ を使って表すと

$$[x]^j = ([x]^{2^{n-1}})^{j_1} ([x]^{2^{n-2}})^{j_2} \cdots ([x])^{j_n}$$

と式変形できます。$[x]^{2^k}$ たちは j とは無関係なうえ

$$[x]^{2^k} = ([x]^{2^{k-1}})^2$$

という漸化式を満たすので、n 回の掛け算と余りの計算だけで $x^{2^k} \bmod N$ たちを求めることができます。k は高々 $n = \log_2(q) \fallingdotseq 2\log_2(N)$ なので、この程度であれば古典的なコンピュータでも

高速に計算できます。計算しておいた $x^{2^k} \bmod N$ を使えば、$x^j \bmod N$ は以下のように n 回程度の掛け算と余りの計算で求まります。

(1) $k = 0, 1, \ldots, n-1$ に対して、$x_k \coloneqq x^{2^k} \bmod N$ をあらかじめ計算しておく。
(2) $y = 1$ に初期化する。
(3) $j_1, j_2, \ldots, j_n$ を先頭からみていき、各 j_k に対して、j_k が 0 なら何もせず、1 なら y を $(x_{n-k}y) \bmod N$ で置き換える。
(4) 最終的な y が $x^j \bmod N$ となる。

仮に N が 2^{2024} 程度でも n は $2 \cdot 2024$ 程度なので、最初の単純な方法より大幅に高速化されています。

あとは定理 3.4.1 や定理 3.4.3 を使って量子回路に翻訳するだけですが、どのような量子回路になるかを述べておきましょう。定理 3.4.3 と例 3.4.4 を使って、U_{x_k} という線形写像を

$$U_{x_k} |y\rangle = |(x_k y) \bmod N\rangle \quad (0 \leq y \leq N-1)$$

を満たすように作ります。この線形写像を使って、目的の線形写像 $U_f(|j\rangle \otimes |0\rangle) = |j\rangle \otimes |f(j)\rangle$ を作ります。先ほどの $f(j)$ を計算するアルゴリズムの (3) の処理「j_k が 0 なら何もせず 1 なら y を $(x_{n-k}y) \bmod N$ で置き換える」は

$$|j\rangle \otimes |y\rangle \mapsto \begin{cases} |j\rangle \otimes |y\rangle & (j_k = 0) \\ |j\rangle \otimes U_{x_{n-k}} |y\rangle & (j_k = 1) \end{cases}$$

と表せます。これは制御ビット付きの $U_{x_{n-k}}$ にほかなりません。まとめると、$f(j)$ を計算する量子回路は図 5.1 のようになります。ただし、一番下の線は m 量子ビットを 1 本にまとめて表しています。さらに、$|0\rangle^{\otimes m}$ の最後の量子ビットに X ゲートを作用させて $|00\cdots01\rangle$ にする処理を最初に追加すれば U_f が実現できます。

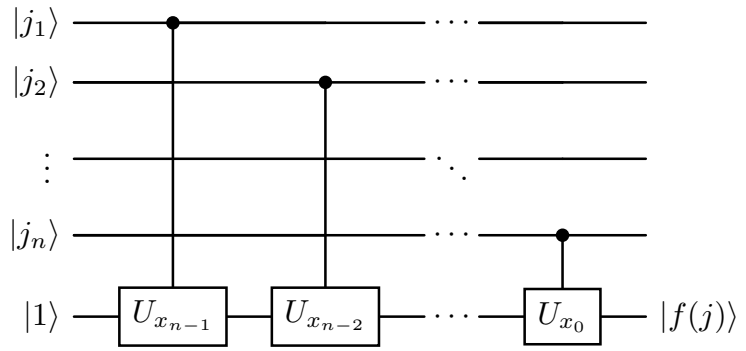


図 5.1: $x^j \bmod N$ を計算する回路

U_f にどれくらいの量子ゲートが必要かを見積もっておきましょう。$U_{x_0}, U_{x_1}, \ldots, U_{x_{n-1}}$ たち

は掛ける数が異なるだけで、どれも

$$U_{x_k}|y\rangle = |(x_k y) \bmod N\rangle$$

という x_k を掛けて mod N するという写像でした。2 進数の筆算を使う場合、N の桁数の 2 乗程度の回数の演算 (1 桁の和と積) で掛け算も割り算も実現できます。3.1.4 節で述べた整数の和の回路と同様に量子ゲートに翻訳すれば、やはり $\log_2(N)^2$ 個程度の X, CNOT, Toffoli ゲートで実現できます。実際には $\log_2(N)$ の 2 次関数で上から抑えられるといったほうが正確ですが、一番影響のある部分だけを抜き出しています。

U_f には U_{x_k} たちが $\log_2(q) = n$ 個使われているので、まとめると以下の定理が得られます。ただし、q は N^2 程度の大きさに取っているものと仮定しています。

定理 5.2.1. 線形写像 U_f は $\log_2(N)^3$ 個程度の X, CNOT, Toffoli ゲートで実現できる。

これで U_f を $\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle\otimes|0\rangle$ に作用させることで、$\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle\otimes|f(j)\rangle$ という状態を作ることができました。$[x]_N$ の位数を r とすると f の周期は r になるので、$f(j)$ は $x^0 \bmod N, x^1 \bmod N, \ldots, x^{r-1} \bmod N$ の r 通りの値しかとりません。$f(j)$ が同じ値になる j をまとめると、

$$\sum_{j=0}^{q-1}|j\rangle\otimes|f(j)\rangle = \sum_{d=0}^{r-1}\sum_{j'=0}^{M_d-1}|rj'+d\rangle\otimes\left|x^d \bmod N\right\rangle$$

と変形できます。ここで、M_d は $r(M_d-1)+d < q \le rM_d+d$ を満たす整数です。d を固定した部分の前半の n 量子ビット

$$\sum_{j'=0}^{M_d-1}|rj'+d\rangle = |d\rangle + |r+d\rangle + |2r+d\rangle + \cdots + |(M_d-1)r+d\rangle$$

という状態から周期 r を取り出すのが次の量子フーリエ変換での目的になります。

5.3 フーリエ変換

Shor のアルゴリズムの肝は x の位数を推定する部分 (1.2 節の手順 4) です。その中で量子フーリエ変換と呼ばれる操作が中心的な役割を果たします。1.2 節で述べたとおり、(逆) 量子フーリエ変換とは

$$B = (b_{k,j})_{0\le k<q, 0\le j<q}, \quad b_{k,j} = \exp\left(-\frac{2\pi\sqrt{-1}}{q}jk\right) \tag{5.1}$$

という一見複雑な行列を掛けるという操作でした。この節ではいったん量子アルゴリズムから離れて、行列 B を掛けることでなぜ位数が求まるのかについて考えていきます。

5.3.1 指数関数とフーリエ変換

定義 2.5.4 で定義した $\exp(\cdot)$ という関数を思い出しましょう。$\exp(\sqrt{-1}x) = \cos(x) + \sqrt{-1}\sin(x)$ によって定義されていました。cos と sin は周期 2π の周期関数なので、$e^{\sqrt{-1}x}$ も周期 2π の周期関数になります。つまり、$e^{\sqrt{-1}(x+2\pi)} = e^{\sqrt{-1}x}$ が任意の $x \in \mathbb{R}$ に対して成り立ちます。さらに $e^{2\pi\sqrt{-1}x}$ と定数倍ずらせば、周期 1 の周期関数になります。

命題 5.3.1. q, j を正の整数とする。実数 x に関する関数 $\exp\left(\frac{2\pi\sqrt{-1}}{q}jx\right)$ は周期 q/j の周期関数である。

このように、(5.1) に現れている $\exp\left(\frac{2\pi\sqrt{-1}}{q}jk\right)$ は周期 q/j の周期関数に $x = k$ を代入したものと思えます。

さて、フーリエ変換の一種であるフーリエ級数展開について述べておきましょう。大雑把に述べると、関数 $f(x)$ が $e^{\sqrt{-1}nx}$ $(n \in \mathbb{Z})$ という関数たちの線形結合で表されているときに、各 $e^{\sqrt{-1}nx}$ の係数を取り出す操作がフーリエ変換です。式で表すと、$f(x) = \sum_{n\in\mathbb{Z}} c_n e^{\sqrt{-1}nx}$ という関数から数列 $\{c_n\}_{n\in\mathbb{Z}}$ を取り出す操作です。

$$f(x) = e^{\sqrt{-1}x} + 2e^{2\sqrt{-1}x} \xmapsto{\text{フーリエ変換}} \begin{array}{c|ccccccccc} n & \cdots & -1 & 0 & 1 & 2 & 3 & 4 & \cdots \\ \hline c_n & \cdots & 0 & 0 & 1 & 2 & 0 & 0 & \cdots \end{array}$$

この例の場合、f が初めから線形結合で書けているのであまり意味があるように見えないかもしれませんが、実際には f が分かりやすい形でなくても、積分によって c_n を求めることができます:

$$f(x) = (\text{複雑な形}) \xmapsto{\text{フーリエ変換}} c_n = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-\sqrt{-1}nx}dx. \tag{5.2}$$

係数 c_n は、f の中に $e^{\sqrt{-1}nx}$ がどの程度の割合で含まれているかを表す量と思えます。

Shor のアルゴリズムにおいて重要なのは、この c_n たちの振る舞いによって f の周期が読み取れるという事実です。例えば、f の周期が $2\pi/m$ であることと、c_{nm} $(n \in \mathbb{Z})$ 以外の係数がすべて 0 になることが同値になります。

5.3.2 離散フーリエ変換

フーリエ変換を施すことで周期に関する情報を取り出すことができるという事実を述べました。ここまでの話は f が $\mathbb{R}$ 上の周期関数の場合だったので、そのままでは量子ビットや数列には適用できません。Shor のアルゴリズムに使うため、ここでは数列に対するフーリエ変換を考えます。

Shor のアルゴリズムでは、例えば

$$\underbrace{0,0,1,0}_{1\text{ 周期分}},0,0,1,0,0,0,1,0,0,0,1,0,\ldots,1,0 \tag{5.3}$$

のような周期的な数列にフーリエ変換を施します。$e^{2\pi\sqrt{-1}\frac{k}{q}x}$ が数列にどれだけ多く含まれているかという情報を取り出すことで、元の数列がどのような周期をもっているかを検出しようという戦略です。

q を正の整数とし、整数 k に対して $\mathbb{Z}$ 上の関数 χ_k を

$$\chi_k(j) = \exp\left(\frac{2\pi\sqrt{-1}}{q}kj\right) \quad (j \in \mathbb{Z})$$

で定めます。exp の周期性 (命題 5.3.1) から χ_k は周期 q の周期関数になります。k に関する周期性から $\chi_{k+q} = \chi_k$ なので、

$$\chi_0, \quad \chi_1, \quad \cdots, \quad \chi_{q-1}$$

の q 個の関数が得られました。χ_k の k は $e^{\sqrt{-1}nx}$ の n のような、周期をコントロールするパラメータだと思っています。

例 5.3.2. $q = 2$ の場合、

$$\begin{aligned}
&\chi_0(0) = 1, \quad \chi_0(1) = 1,\\
&\chi_1(0) = 1, \quad \chi_1(1) = -1,\\
&\chi_2(0) = 1, \quad \chi_2(1) = 1,\\
&\qquad\vdots
\end{aligned}$$

となる。また、$q = 4$ の場合は

$$\begin{aligned}
&\chi_0(0) = 1, \quad \chi_0(1) = 1, \quad \chi_0(2) = 1, \quad \chi_0(3) = 1,\\
&\chi_1(0) = 1, \quad \chi_1(1) = \sqrt{-1}, \quad \chi_1(2) = -1, \quad \chi_1(3) = -\sqrt{-1},\\
&\chi_2(0) = 1, \quad \chi_2(1) = -1, \quad \chi_2(2) = 1, \quad \chi_2(3) = -1,\\
&\chi_3(0) = 1, \quad \chi_3(1) = -\sqrt{-1}, \quad \chi_3(2) = -1, \quad \chi_3(3) = \sqrt{-1}
\end{aligned}$$

となる。

exp の性質を χ_k で書き直しておきましょう。

命題 5.3.3. $i, j, k, l \in \mathbb{Z}$ とすると、以下の性質が成り立つ。

(1) $\chi_k(0) = \chi_k(q) = 1$
(2) $\chi_k(i + j) = \chi_k(i)\chi_k(j)$
(3) $\chi_k(j)\chi_l(j) = \chi_{k+l}(j)$
(4) χ_k は周期 q の周期関数
(5) $\overline{\chi_k(j)} = \chi_k(-j) = \chi_{-k}(j)$

(6) $\chi_k(j) = \chi_k(1)^j$

命題 5.3.3 の (1),(2) を満たすような $\mathbb{Z}$ 上の関数 χ は、必ず χ_k の形で表されることが示せます。以下で行う議論では (1),(2) の性質が本質的で、χ_k の具体的な形はそれほど重要ではありません。

$a = (a_0, a_1, \ldots, a_{q-1})$ を長さ q の複素数列とします。この数列は $\{0, 1, \ldots, q-1\}$ という集合から $\mathbb{C}$ への写像 (関数) と思えます。つまり、a は $j \in \{0, 1, \ldots, q-1\}$ に対して複素数 a_j を対応させる関数です。このような視点からフーリエ級数展開の類似を考えると、a は複素数 $c_0, \ldots, c_{q-1}$ を使って

$$a_j = \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} c_k \chi_k(j) \quad (j \in \{0, 1, \ldots, q-1\}) \tag{5.4}$$

と表すことができます。この事実は定理 5.3.6 で示すことにして、先に数列 a から c_k を求める式を与えましょう。

補題 5.3.4. $0 \leq k, l < q$ とすると、

$$\sum_{j=0}^{q-1} \chi_k(j)\overline{\chi_l(j)} = \begin{cases} q & (k = l) \\ 0 & (k \neq l) \end{cases}$$

が成り立つ。

証明. 求める和を S と置くと、

$$S = \sum_{j=0}^{q-1} \chi_k(j)\overline{\chi_l(j)} = \sum_{j=0}^{q-1} \chi_k(j)\chi_{-l}(j) = \sum_{j=0}^{q-1} \chi_{k-l}(j)$$

となる。$k = l$ なら $\chi_{k-l}(j) = 1$ $(\forall j)$ なので、$S = q$ となる。$k \neq l$ とし、$z := \chi_{k-l}(1) = \exp(2\pi\sqrt{-1}\frac{k-l}{q})$ と置く。$0 < |(k-l)/q| < 1$ なので、$e^{\sqrt{-1}x}$ の定義から $z \neq 1$ である。また、$z^q = \chi_{k-l}(q) = 1$ となる。よって、等比数列の和の公式から

$$S = \sum_{j=0}^{q-1} z^j = \frac{1 - z^q}{1 - z} = 0$$

を得る。 □

定理 5.3.5. (5.4) が成り立っていると仮定する。$0 \leq k < q$ とすると、

$$c_k = \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} \overline{\chi_k(j)} a_j$$

が成り立つ。

証明. 右辺に $a_j = \frac{1}{\sqrt{q}} \sum_{l=0}^{q-1} c_l \chi_l(j)$ を代入すると、

$$\begin{aligned}\frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} \overline{\chi_k(j)} a_j &= \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} \overline{\chi_k(j)} \frac{1}{\sqrt{q}} \sum_{l=0}^{q-1} c_l \chi_l(j) \\ &= \frac{1}{q} \sum_{l=0}^{q-1} c_l \sum_{j=0}^{q-1} \chi_l(j) \overline{\chi_k(j)}\end{aligned}$$

となる。補題 5.3.4 より、二つ目の和 $\sum_{j=0}^{q-1} \chi_l(j)\overline{\chi_k(j)}$ は $l = k$ のときのみ q でそれ以外は 0 となる。よって、

$$\frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} \overline{\chi_k(j)} a_j = c_k$$

を得る。 □

定理 5.3.5 の式はちょうど、(5.2) の積分を和に変えたようなものになっています。数列 a から数列 c を得る操作は、まさにフーリエ変換の離散的な類似と思うことができるのです。

a, c を数ベクトルで表すと、定理 5.3.5 の式は行列で表すことができます。その行列が Shor のアルゴリズムで現れる B にほかなりません。まずは、行列や行列の操作について復習しておきましょう。

(複素数成分)$n \times m$ 行列とは複素数を縦 n 行、横 m 列の長方形に並べたもののことでした。m 次元数ベクトルは $m \times 1$ 行列とみなせます。

$$\begin{pmatrix} X_{00} & X_{01} & \cdots & X_{0,m-1} \\ X_{10} & X_{11} & \cdots & X_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n-1,0} & X_{n-1,1} & \cdots & X_{n-1,m-1} \end{pmatrix}, \quad \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{m-1} \end{pmatrix}$$

ここではいくつかの式に合わせて左上を $(0,0)$ 成分としています。行列 X に対して X_{ij} で (i,j) 成分を表すと、$n \times m$ 行列 X と $m \times l$ 行列 Y の積 XY は

$$(XY)_{ij} = \sum_{k=0}^{m-1} X_{ik} Y_{kj}$$

によって定義される $n \times l$ 行列になります。$n \times n$ 行列 X, Y が $XY = YX = I$ (単位行列) を満たすとき、Y を X^{-1} と表し X の**逆行列**と呼びました。また、逆行列を持つ行列は**正則**であるといわれます。$XY = I$ か $YX = I$ の一方を満たすなら自動的にもう一方も成り立ちます。

$n \times m$ 行列 X の転置 ${}^tX$ と複素共役 $\overline{X}$ と随伴 X^* を

$$({}^tX)_{ij} = X_{ji}, \quad (\overline{X})_{ij} = \overline{X_{ij}}, \quad X^* = {}^t\overline{X}$$

によって定めます。例えば 2×2 の場合では、

$$X=\begin{pmatrix}a & b\\ c & d\end{pmatrix},\quad {}^tX=\begin{pmatrix}a & c\\ b & d\end{pmatrix},\quad \overline{X}=\begin{pmatrix}\overline{a} & \overline{b}\\ \overline{c} & \overline{d}\end{pmatrix},\quad X^*=\begin{pmatrix}\overline{a} & \overline{c}\\ \overline{b} & \overline{d}\end{pmatrix}$$

となります。

さて、フーリエ変換と行列 B の話に戻りましょう。$B=\left(\exp\left(\frac{-2\pi\sqrt{-1}}{q}jk\right)\right)_{0\le k,j<q}$ と定義されていました。記号の付け方の都合上、k を行、j を列にしています。χ を使って表すと

$$B=\left(\overline{\chi_k(j)}\right)_{0\le k,j<q}$$

となり、B を使うと定理 5.3.5 は

$$\begin{pmatrix}c_0\\ c_1\\ \vdots\\ c_{q-1}\end{pmatrix}=\frac{1}{\sqrt{q}}B\begin{pmatrix}a_0\\ a_1\\ \vdots\\ a_{q-1}\end{pmatrix}\tag{5.5}$$

と表せます。つまり、Shor のアルゴリズムの B を掛けるステップは、数列 a から関数 χ_k が a にどのくらい寄与しているかという情報 $(=c)$ を取り出していると解釈できます。

χ_k たちの満たす関係式 (補題 5.3.4) を使うと、$\frac{1}{\sqrt{q}}B$ の逆行列を具体的に作ることができます。これを利用して、認めて進めていた (5.4) を満たす c がいつでも取れることを証明しましょう。

定理 5.3.6. $\frac{1}{\sqrt{q}}B$ の逆行列は $\frac{1}{\sqrt{q}}B^*$ である。特に B は正則であり、任意の a に対して、(5.4) を満たす $c_0,\dots,c_{q-1}$ が存在する。

証明. BB^* を計算しよう。$(B^*)_{jl}=\overline{B_{lj}}=\chi_l(j)$ なので、

$$\begin{aligned}(BB^*)_{kl}&=\sum_{j=0}^{q-1}B_{kj}(B^*)_{jl}\\ &=\sum_{j=0}^{q-1}\overline{\chi_k(j)}\chi_l(j)=\begin{cases}q & (k=l)\\ 0 & (k\neq l)\end{cases}\qquad\text{(補題 5.3.4)}\end{aligned}$$

となる。つまり、$\frac{1}{q}BB^*$ の (k,l) 成分は $k=l$ のときだけ 1 でそれ以外は 0 なので、$\frac{1}{q}BB^*=I$ となる。したがって、$\frac{1}{\sqrt{q}}B$ の逆行列は $\frac{1}{\sqrt{q}}B^*$ であり、特に B は正則である。

$a=(a_0,a_1,\dots,a_{q-1})$ を複素数列として、(5.4) の形に表せることを示そう。数列 $c_0,\dots,c_{q-1}$ を

$$\begin{pmatrix}c_0\\ c_1\\ \vdots\\ c_{q-1}\end{pmatrix}=\frac{1}{\sqrt{q}}B\begin{pmatrix}a_0\\ a_1\\ \vdots\\ a_{q-1}\end{pmatrix}$$

によって定義する。$\frac{1}{\sqrt{q}}B$ の逆行列が $\frac{1}{\sqrt{q}}B^*$ なので、

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{q-1} \end{pmatrix} = \frac{1}{\sqrt{q}} B^* \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{q-1} \end{pmatrix}$$

となる。右辺を $\frac{1}{\sqrt{q}}B^* = \frac{1}{\sqrt{q}}\left(\chi_k(j)\right)_{0\le j,k<q}$ を使って計算すると、任意の $0 \le j < q$ に対して

$$a_j = \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} c_k \chi_k(j)$$

となる。 □

系 5.3.7. $\sum_{j=0}^{q-1} |a_j|^2 = \sum_{k=0}^{q-1} |c_k|^2$ が成り立つ。

証明. 定理 5.3.5 から

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{q-1} \end{pmatrix} = \frac{1}{\sqrt{q}} B \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{q-1} \end{pmatrix},$$

$$\begin{pmatrix} \overline{c_0} & \overline{c_1} & \cdots & \overline{c_{q-1}} \end{pmatrix} = \begin{pmatrix} \overline{a_0} & \overline{a_1} & \cdots & \overline{a_{q-1}} \end{pmatrix} \frac{1}{\sqrt{q}} B^*$$

である。ここで、$\frac{1}{\sqrt{q}}B$ の逆行列は $\frac{1}{\sqrt{q}}B^*$ なので

$$\begin{aligned} \sum_{k=0}^{q-1} |c_k|^2 &= \begin{pmatrix} \overline{c_0} & \overline{c_1} & \cdots & \overline{c_{q-1}} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{q-1} \end{pmatrix} \\ &= \begin{pmatrix} \overline{a_0} & \overline{a_1} & \cdots & \overline{a_{q-1}} \end{pmatrix} \frac{1}{\sqrt{q}} B^* \cdot \frac{1}{\sqrt{q}} B \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{q-1} \end{pmatrix} \\ &= \begin{pmatrix} \overline{a_0} & \overline{a_1} & \cdots & \overline{a_{q-1}} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{q-1} \end{pmatrix} = \sum_{j=0}^{q-1} |a_j|^2 \end{aligned}$$

を得る。 □

定義 5.3.8 (離散フーリエ変換)**.** $\frac{1}{\sqrt{q}}B$ を**離散フーリエ変換**という。ベクトル (または数列)c を a の離散フーリエ変換、各成分 c_k を a の**フーリエ係数**と呼ぶ。

最後に、(5.3) のような周期的な数列 a のフーリエ係数 c_k がどんな値になるかを計算してみましょう。ただし、ここでは周期 r が q を割り切るような簡単な場合を扱い、一般の場合は次の節で計算します。

命題 5.3.9. r を q の約数、$0 \leq d < r$ とし $M := q/r$ と置く。長さ q の数列 a を

$$a_j = \begin{cases} 0 & (j \bmod r \neq d) \\ 1 & (j \bmod r = d) \end{cases}$$

で定める。このとき、

$$c_k = \begin{cases} \sqrt{\frac{M}{r}}\overline{\chi_k(d)} & (k \bmod M = 0) \\ 0 & (k \bmod M \neq 0) \end{cases}$$
$$a_j = \frac{1}{r}\sum_{k'=0}^{r-1} \overline{\chi_{k'M}(d)} \exp\left(\frac{2\pi\sqrt{-1}}{r}k'j\right)$$

となる。

証明. $0 \leq k < q$ とすると、

$$\begin{aligned} c_k &= \frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}\overline{\chi_k(j)}a_j \\ &= \frac{1}{\sqrt{q}}\sum_{j'=0}^{M-1}\overline{\chi_k(rj'+d)} \\ &= \frac{1}{\sqrt{q}}\overline{\chi_k(d)}\sum_{j'=0}^{M-1}\overline{\chi_k(rj')} \end{aligned}$$

となる。$\chi_k(rj') = \exp(2\pi\sqrt{-1}\frac{kj'}{M})$ なので、補題 5.3.4 より、最後の和は k が M で割り切れるなら M になり、そうでないなら 0 になる。したがって、

$$c_k = \begin{cases} \sqrt{\frac{M}{r}}\overline{\chi_k(d)} & (k \bmod M = 0) \\ 0 & (k \bmod M \neq 0) \end{cases}$$

となる。a に関する式は、(5.4) に c の式を代入すれば得られる。 □

量子回路モデルの文脈で命題 5.3.9 を言い換えておきましょう。周期的な数列 a は状態

$$\sum_{j=0}^{q-1} a_j |j\rangle = \sum_{j'=0}^{M-1} |d+rj'\rangle = |d\rangle + |d+r\rangle + |d+2r\rangle + \cdots + |d+(M-1)r\rangle$$

に対応します。これに $\frac{1}{\sqrt{q}}B$ を作用させると

$$\sum_{k=0}^{q-1} c_k \left|k\right\rangle = \sqrt{\frac{M}{r}} \sum_{k'=0}^{r-1} \overline{\chi_{k'M}(d)} \left|k'M\right\rangle$$
$$= \sqrt{\frac{M}{r}} \left(\left|0\right\rangle + \overline{\chi_M(d)} \left|\frac{q}{r}\right\rangle + \overline{\chi_{2M}(d)} \left|\frac{2q}{r}\right\rangle + \cdots + \overline{\chi_{(r-1)M}(d)} \left|\frac{(r-1)q}{r}\right\rangle \right)$$

になります。$\overline{\chi_{k'M}(d)}$ たちの絶対値は 1 なので、この状態を測定すると、$0, \frac{q}{r}, \frac{2q}{r}, \ldots, \frac{(r-1)q}{r}$ のどれかが等確率で出力されます。例えば、$r = 8, d = 3, q = 128$ なら確率は図 5.2 のような分布になります。

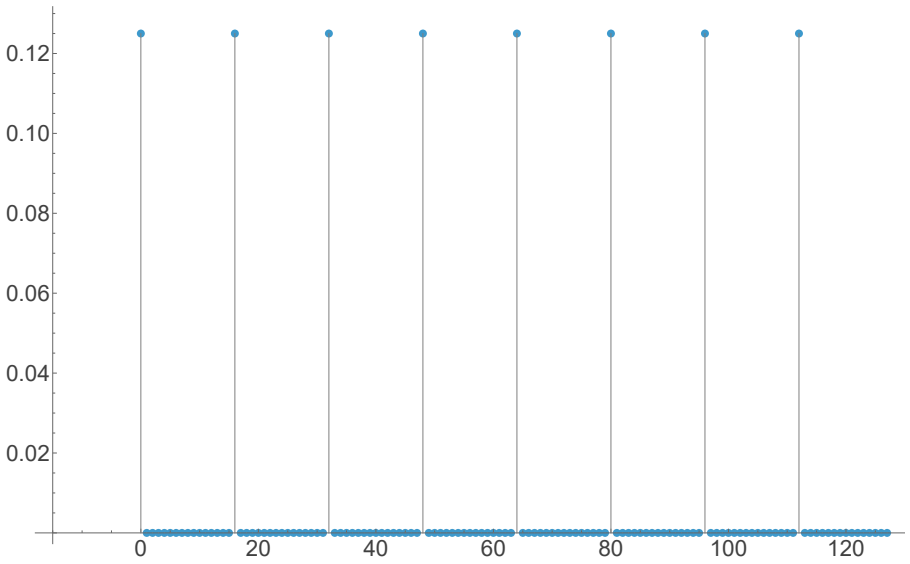


図 5.2: $r = 8, d = 3, q = 128$ の場合の確率分布

測定値を q で割ると、$0, 1/r, 2/r, \ldots, (r-1)/r$ となるので分母をみれば a の周期 r が分かります。注意として、分子が r と互いに素でない場合では、約分ができてしまい分母が r より小さくなってしまうことがあります。しかし、定理 4.3.5 からそのような約分できるケースは低確率でしか起こらないので、$\log \log r$ 回程度繰り返せば十分な確率で r を復元できます。

r が q の約数でないと c_k が複雑な形になり確率の評価が難しくなります。その場合については次の節 (5.3.3 節) で扱います。また、フーリエ変換 $\frac{1}{\sqrt{q}}B$ を基本的な量子ゲートを使って実現する方法については 5.4 節で扱います。

5.3.3　確率の評価

前節の最後で r が q の約数の場合に、$\left|d\right\rangle + \left|d+r\right\rangle + \left|d+2r\right\rangle + \cdots + \left|d+(M-1)r\right\rangle$ にフーリエ変換を施すことで周期を取り出しやすい形 $\left|0\right\rangle + \left|q/r\right\rangle + \left|2q/r\right\rangle + \cdots + \left|(r-1)q/r\right\rangle$(係数は無視) に変換できることが分かりました。

Shor のアルゴリズムに使おうとすると、2 点、大きな問題が発生します。第 1 に r が未知であること、第 2 に $q = 2^n$ という形でなければならないことです。一つ目に関してはそもそも r が求めたいものだったので、未知なのは当然です。二つ目の $q = 2^n$ は量子回路モデルで扱いやすいようにするための制約で、この条件のおかげでフーリエ変換を綺麗な回路で実現できるようになります。いずれにせよ、r が q の約数になるように q を取ることは一般には不可能です。ここでは、r が q の約数でない場合に、命題 5.3.9 からどの程度ずれるかを評価します。

q, r を正の整数、$0 \leq d < r$ として

$$a_j = \begin{cases} 1 & (j \bmod r = d) \\ 0 & (j \bmod r \neq d) \end{cases} \quad (j = 0, 1, 2, \ldots)$$

という数列を考え、前節と同じく a の離散フーリエ変換を c と表します。$r = 7, d = 3, q = 128$ の場合、c に対応する状態に測定を行うと図 5.3 のような確率分布で測定値が得られ、$k = 0, 18, 37, 55, 73, 91, 110$ で確率が極大になることが分かります。これらはそれぞれ、

$$\frac{0}{7} = \frac{0}{128}, \quad \frac{1}{7} \fallingdotseq \frac{18}{128}, \quad \frac{2}{7} \fallingdotseq \frac{37}{128}, \quad \frac{3}{7} \fallingdotseq \frac{55}{128}, \quad \frac{4}{7} \fallingdotseq \frac{73}{128}, \quad \frac{5}{7} \fallingdotseq \frac{91}{128}, \quad \frac{6}{7} \fallingdotseq \frac{110}{128}$$

という近似に対応しています。また、図 5.2 と異なり、ピークの周辺にも低確率で測定される値があることが分かります。

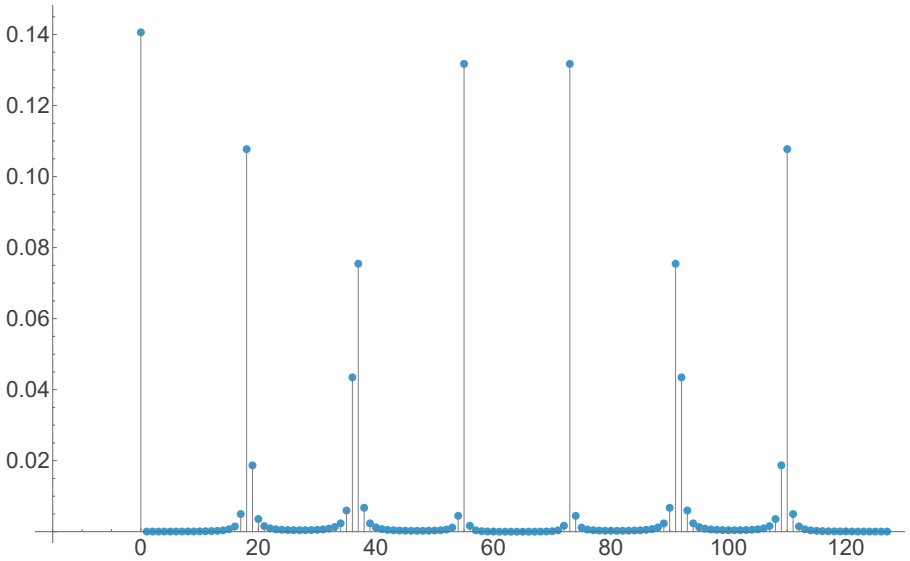


図 5.3: $r = 7, d = 3, q = 128$ の場合の確率分布

まずは図 5.3 のようになる理由を考えてみましょう。$q = r$ の場合の命題 5.3.9 を a に適用すると、

$$a_j = \frac{1}{r} \sum_{k'=0}^{r-1} e_{k'} \exp\left(\frac{2\pi\sqrt{-1}}{r} k' j\right)$$

と表せます。ここで、$e_{k'}$ は具体的な絶対値 1 の複素数です。正の整数 q を r に比べて十分大きく取り、qk'/r の小数点以下を四捨五入したものを k と置くと、

$$\frac{k'}{r} \fallingdotseq \frac{k}{q}, \quad \exp\left(\frac{2\pi\sqrt{-1}}{r}k'j\right) \fallingdotseq \exp\left(\frac{2\pi\sqrt{-1}}{q}kj\right)$$

と近似できます。したがって、大体

$$a_j = \frac{1}{r}\sum_{k'=0}^{r-1} e_{k'} \exp\left(\frac{2\pi\sqrt{-1}}{q}kj\right)$$

のようになると予想できます。図 5.3 の極大になっている点はこの近似した点 k/q で、例えば

$$\frac{2}{7} \fallingdotseq \frac{36}{128}, \quad \frac{2}{7} \fallingdotseq \frac{37}{128}$$

のような点では近似の精度が悪く、k'/r に近い k/q が複数あるため、フーリエ係数が 1 点に集中せず周辺に分散してしまっています。

$|c_k|$ を具体的に計算して、位数 r の推定の成功確率を下から評価していきましょう。a_j は $j \bmod r = d$ の場合だけ 1 になる数列だったので、1 になる項の添え字は

$$d, \quad r+d, \quad 2r+d, \quad \ldots, \quad j'r+d, \quad \cdots, \quad (M-1)r+d$$

となります。ここで、M は $(M-1)r+d < q \leq Mr+d$ を満たす唯一の整数です。定理 5.3.5 から

$$\begin{aligned} c_k &= \frac{1}{\sqrt{q}}\sum_{j=0}^{q-1} \overline{\chi_k(j)} a_j \\ &= \frac{1}{\sqrt{q}}\overline{\chi_k(d)} \sum_{j'=0}^{M-1} \exp\left(-2\pi\sqrt{-1}\frac{kr}{q}j'\right) \end{aligned}$$

という具体的な形は分かっているので、まずは等比数列の和の公式を使って整理します。

補題 5.3.10. $x \in \mathbb{R}$、n を正の整数とする。このとき、

$$\left|\sum_{j=0}^{n-1} e^{2\pi\sqrt{-1}jx}\right| = \begin{cases} n & (x \in \mathbb{Z}) \\ \left|\frac{\sin(\pi n x)}{\sin(\pi x)}\right| & (x \notin \mathbb{Z}) \end{cases}$$

となる。

証明. $x \in \mathbb{Z}$ の場合は明らかなので、$x \notin \mathbb{Z}$ の場合を示す。$e^{2\pi\sqrt{-1}jx} = (e^{2\pi\sqrt{-1}x})^j$ なので、等比数列の和の公式から

$$\sum_{j=0}^{n-1} e^{2\pi\sqrt{-1}jx} = \frac{1 - e^{2\pi\sqrt{-1}nx}}{1 - e^{2\pi\sqrt{-1}x}}$$

となる。ここで、任意の $y \in \mathbb{R}$ に対して

$$\begin{aligned} 1 - e^{2\pi\sqrt{-1}y} &= e^{\pi\sqrt{-1}y}(e^{-\pi\sqrt{-1}y} - e^{\pi\sqrt{-1}y}) \\ &= e^{\pi\sqrt{-1}y}(-2\sqrt{-1}\sin(\pi y)) \end{aligned}$$

となるので、合わせて

$$\left|\sum_{j=0}^{n-1} e^{2\pi\sqrt{-1}jx}\right| = \left|\frac{-2\sqrt{-1}e^{\pi\sqrt{-1}nx}\sin(\pi nx)}{-2\sqrt{-1}e^{\pi\sqrt{-1}x}\sin(\pi x)}\right| = \left|\frac{\sin(\pi nx)}{\sin(\pi x)}\right|$$

を得る。 □

以下、$\sin(\pi nx)/\sin(\pi x)$ という形の関数は x が整数の点にも連続的に拡張しているものとします。$\sin(10\pi x)/\sin(\pi x)$ のグラフは図 5.4 のようになります。絶対値が最大になるのは x が整数の場合であり、その周辺では急激に絶対値が小さくなることが分かります。

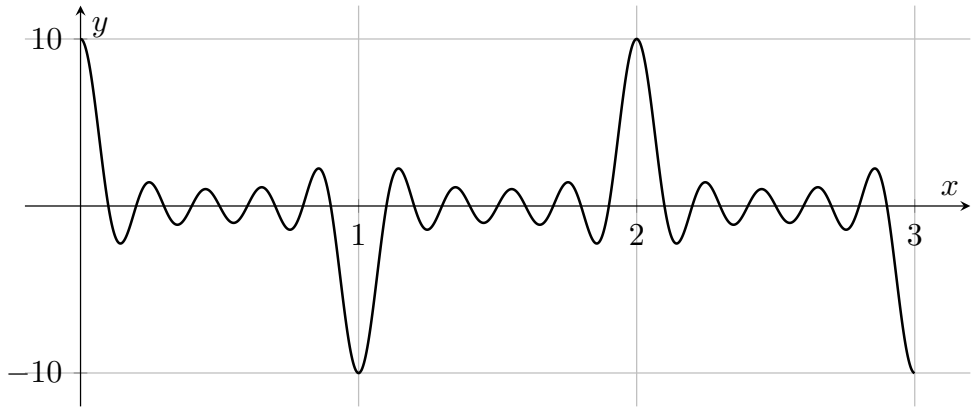


図 5.4: $\sin(10\pi x)/\sin(\pi x)$ のグラフ

補題 5.3.11. n を 1 より大きい整数とすると、$0 \leq x \leq 1/n$ において関数 $f(x) = \frac{\sin(\pi nx)}{\sin(\pi x)}$ は単調減少である。

証明. f の導関数が $0 < x < 1/n$ で常に負であることを示せばよい。

$$\frac{df}{dx}(x) = \frac{\pi n\cos(\pi nx)\sin(\pi x) - \pi\sin(\pi nx)\cos(\pi x)}{\sin^2(\pi x)}$$

である。さらに、$g(x) \coloneqq n\cos(\pi nx)\sin(\pi x) - \sin(\pi nx)\cos(\pi x)$ を微分すると

$$\begin{aligned} \frac{dg}{dx}(x) &= -\pi n^2\sin(\pi nx)\sin(\pi x) + \pi n\cos(\pi nx)\cos(\pi x) \\ &\quad + \pi\sin(\pi nx)\sin(\pi x) - \pi n\cos(\pi x)\cos(\pi nx) \\ &= -\pi(n^2-1)\sin(\pi nx)\sin(\pi x) \end{aligned}$$

となり、$\frac{dg}{dx}$ は $0 < x < 1/n$ において常に負である。$g(0) = 0$ なので $g(x)$ は $0 < x < 1/n$ において常に負であり、したがって、$f(x)$ は $0 \leq x \leq 1/n$ において単調減少である。 □

$|c_k|^2$ に補題 5.3.10 を用いると

$$|c_k|^2 = \frac{1}{q}\left|\sum_{j'=0}^{M-1} \exp\left(-2\pi\sqrt{-1}\frac{kr}{q}j'\right)\right|^2 = \frac{1}{q}\left|\frac{\sin\left(\pi M r\frac{k}{q}\right)}{\sin\left(\pi r\frac{k}{q}\right)}\right|^2$$

となります。図 5.4 から、k を動かしたときに $|c_k|^2$ が大きくなるのは $r\frac{k}{q}$ が整数に近いとき、すなわちある整数 k' に対して $k/q \fallingdotseq k'/r$ となっている場合と予想できます。そして、補題 5.3.11 から、k/q と k'/r の誤差が大きいほど $|c_k|^2$ は小さくなることも分かります。これは図 5.3 やその後の考察とも整合しています。

以下では、Shor のアルゴリズムの設定に合わせて N を 15 以上の整数とし、$r < N/2, N^2 < q$ と仮定します。a の離散フーリエ変換

$$\frac{1}{\sqrt{q}}\sum_{k=0}^{q-1} c_k\,|k\rangle$$

を測定すると、$|c_k|^2/\sum_{l=0}^{q-1}|c_l|^2$ の確率で k が出力されるのでした。そして、k/q を連分数展開し r を復元するという流れですが、どんな k でもうまくいくわけではありません。K を以下の条件を満たす整数 $0 \leq k < q$ の集合と定義すると、K は r を復元できる k の集合となります。

- r と互いに素な整数 k' が存在して、$|\frac{k'}{r} - \frac{k}{q}| < \frac{1}{2N^2}$ を満たす。

$k \in K$ から r が復元できることは系 4.5.9 で示したとおりですが、今の設定で述べておきましょう。

補題 5.3.12. $k \in K$ とし、4.5 節で扱った k/q を連分数展開し得られる分数列 $(p_n/q_n)_n$ を考える。$q_n < N$ となる最大の n を取れば、$q_n = r$ となる。

K を使うと、求めたい確率は大体

$$\frac{\sum_{k\in K}|c_k|^2}{\sum_{k=0}^{q-1}|c_k|^2} = \frac{\sum_{k\in K}|c_k|^2}{\sum_{j=0}^{q-1}|a_j|^2} = \frac{\sum_{k\in K}|c_k|^2}{M} \tag{5.6}$$

となります。一つ目の等号は系 5.3.7 から得られます。

(5.6) の分子を下から評価したいのですが、K に関する和は少し複雑なので、$|c_k|$ がピークになる点だけを考えます。1 以上 r 以下で r と互いに素な整数全体の集合を K' と置き、

$$\psi\colon K' \to \mathbb{Z}, \quad \psi(k') = \left\lfloor \frac{k'q}{r} + \frac{1}{2} \right\rfloor$$

という写像を考えます。ψ の定義の $\lfloor y \rfloor$ は実数 y 以下の最大の整数を表し、したがって、$\psi(k')$ は $\frac{k'q}{r}$ の小数第 1 位を四捨五入した整数になっています。つまり、$k = \psi(k')$ は

$$\frac{qk'}{r} - \frac{1}{2} < k \leq \frac{qk'}{r} + \frac{1}{2}$$

となるただ一つの整数で、

$$\left| \frac{k'}{r} - \frac{k}{q} \right| \leq \frac{1}{2q} < \frac{1}{2N^2}$$

を満たします。よって、$\psi(k') \in K$ となります。

補題 5.3.13. (1) K' の元の個数 $|K'|$ は $\phi(r)$ と等しい。
(2) ψ は単射である。

証明. (1) Euler のトーシェント関数 $\phi(r)$ は r と互いに素な 1 以上 r 以下の整数の個数として定義されていた (定義 4.3.2) ので、$|K'| = \phi(r)$ は定義そのものである。
(2) 任意の $k', k'' \in K'$ $(k' > k'')$ に対して

$$\begin{aligned} \psi(k') - \psi(k'') &> \left(\frac{k'q}{r} - \frac{1}{2} \right) - \left(\frac{k''q}{r} + \frac{1}{2} \right) \\ &= \frac{(k' - k'')q}{r} - 1 > k' - k'' - 1 \geq 0 \end{aligned}$$

となるので、ψ は単射である。 □

$k' \in K', k = \psi(k')$ とすると、k'/r は k/q の非常に良い近似になっているので、$|c_k|^2$ が比較的大きくなると予想できます。実際、以下のような下からの評価が成り立ちます。

補題 5.3.14. $k' \in K'$ とし $k \coloneqq \psi(k')$ と置くと以下の不等式が成り立つ:

$$|c_k|^2 \geq \frac{4}{\pi^2} \cdot \frac{M}{r} \left(1 - \frac{1}{N} \right).$$

証明. $\delta \coloneqq \frac{kr}{q} - k', \varepsilon \coloneqq \frac{r}{q}$ と置くと、

$$|\delta| \leq \frac{\varepsilon}{2}, \quad M\varepsilon = \frac{Mr}{q} < \frac{r+q}{q} = 1 + \varepsilon < 2$$

となる。補題 5.3.10 を使い $|c_k|^2$ を sin 関数で表し補題 5.3.11 の単調性を用いると、

$$|c_k|^2 = \frac{1}{q} \left| \frac{\sin\left(\pi M r \frac{k}{q}\right)}{\sin\left(\pi r \frac{k}{q}\right)} \right|^2 = \frac{1}{q} \frac{\sin\left(\pi M |\delta|\right)^2}{\sin\left(\pi |\delta|\right)^2} \geq \frac{1}{q} \frac{\sin\left(\pi M \cdot \frac{\varepsilon}{2}\right)^2}{\sin\left(\pi \cdot \frac{\varepsilon}{2}\right)^2}$$

となる。分母に対して $|\sin(x)| \leq |x|$ $(\forall x \in \mathbb{R})$、分子に対して $\sin(\pi/2 + x) = \cos(x) \geq 1 - \frac{x^2}{2}$ $(\forall x \in \mathbb{R})$ というよく知られた不等式を用いると、

$$\begin{aligned}|c_k|^2 &\geq \frac{1}{q} \cdot \frac{1}{(\pi \cdot 2^{-1}\varepsilon)^2} \cdot \left(1 - \frac{\pi^2}{2}\left(2^{-1}\varepsilon\right)^2\right)^2 \\ &= \frac{4}{\pi^2} \cdot \frac{M}{r} \frac{q}{Mr} \left(1 - \frac{\pi^2}{8}\varepsilon^2\right)^2 \\ &\geq \frac{4}{\pi^2} \cdot \frac{M}{r} \frac{1}{1+\varepsilon} \left(1 - \frac{\pi^2}{4}\varepsilon^2\right)\end{aligned}$$

となる。$2Nr < q$ と $N \geq 15$ から $2\varepsilon < 1/N \leq 1/15$ となり、特に $\frac{\pi^2}{4}\varepsilon \leq 1$ となる。よって、

$$\frac{1}{1+\varepsilon}\left(1 - \frac{\pi^2}{4}\varepsilon^2\right) \geq \frac{1}{1+\varepsilon}(1-\varepsilon) \geq (1-\varepsilon)^2 \geq 1 - 2\varepsilon > 1 - \frac{1}{N}$$

となり、

$$|c_k|^2 \geq \frac{4}{\pi^2} \cdot \frac{M}{r}\left(1 - \frac{1}{N}\right)$$

を得る。 □

定理 5.3.15. 次の不等式が成り立つ:

$$\sum_{k \in K} |c_k|^2 \geq \frac{\phi(r)}{r} \cdot \frac{4}{\pi^2} \cdot M \left(1 - \frac{1}{N}\right).$$

証明. 補題 5.3.13 と補題 5.3.14 から、

$$\begin{aligned}\sum_{k \in K} |c_k|^2 &\geq \sum_{k' \in K'} |c_{\psi(k')}|^2 \\ &\geq |K'| \frac{4}{\pi^2} \cdot \frac{M}{r}\left(1 - \frac{1}{N}\right) = \frac{\phi(r)}{r} \cdot \frac{4}{\pi^2} \cdot M\left(1 - \frac{1}{N}\right)\end{aligned}$$

となる。 □

最後に、Shor のアルゴリズムの位数を推定する部分において、測定 1 回で正しい位数が求まる確率を評価しましょう。最初に

$$\frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} |j\rangle \otimes \left|x^j \bmod N\right\rangle = \frac{1}{\sqrt{q}} \sum_{d=0}^{r-1} \sum_{j'=0}^{M_d - 1} |d + rj'\rangle \otimes \left|x^d \bmod N\right\rangle$$

という状態を用意し、前半の量子ビットにフーリエ変換 $\frac{1}{\sqrt{q}}B$ を作用させます。ここで、M_d は $\lceil \frac{q-d}{r} \rceil$ でこれまで M と表していた整数です。同様に、c_k と表していたものを $c_{d,k}$ と表すことに

すると、

$$\left(\frac{1}{\sqrt{q}}B\otimes I\right)\left(\frac{1}{\sqrt{q}}\sum_{j=0}^{q-1}|j\rangle\otimes\left|x^j \bmod N\right\rangle\right)=\frac{1}{\sqrt{q}}\sum_{d=0}^{r-1}\sum_{k=0}^{q-1}c_{d,k}|k\rangle\otimes\left|x^d \bmod N\right\rangle \tag{5.7}$$

となります。系 5.3.7 より

$$\sum_{d=0}^{r-1}\sum_{k=0}^{q-1}|c_{d,k}|^2=\sum_{d=0}^{r-1}\sum_{j'=0}^{M_d-1}1^2=\sum_{j=0}^{q-1}1^2=q$$

となります。また、この式から $\sum_{d=0}^{r-1}M_d=q$ となることも分かります。(5.7) の状態を測定し測定値を (k,l) とすると、$k\in K$ となる確率は定理 5.3.15 より

$$\begin{aligned}\frac{\sum_{d=0}^{r-1}\sum_{k\in K}|c_{d,k}|^2}{q}&\geq\frac{1}{q}\sum_{d=0}^{r-1}\frac{\phi(r)}{r}\cdot\frac{4}{\pi^2}\cdot M_d\left(1-\frac{1}{N}\right)\\&=\frac{\phi(r)}{r}\cdot\frac{4}{\pi^2}\cdot\left(1-\frac{1}{N}\right)\cdot\frac{1}{q}\sum_{d=0}^{r-1}M_d\\&=\frac{\phi(r)}{r}\cdot\frac{4}{\pi^2}\cdot\left(1-\frac{1}{N}\right)\end{aligned}$$

と評価できます。定理 4.3.5 と合わせて以下の結果が得られました。

定理 5.3.16. Shor のアルゴリズムの位数を推定する部分において、1 回の測定で正しい位数が求まる確率は

$$\frac{\phi(r)}{r}\cdot\frac{4}{\pi^2}\cdot\left(1-\frac{1}{N}\right)\left(\geq\frac{1}{e^\gamma\log\log N+\frac{3}{\log\log N}}\cdot\frac{4}{\pi^2}\cdot\left(1-\frac{1}{N}\right)\right)$$

以上である。

定理 5.3.16 の右辺は N が 2^{2024} 程度でも 0.03 程度になります。つまり、150 回程度繰り返せば、99% 以上の確率で正しい位数が求まります。これでも十分高い確率といえますが、ここで行った評価はかなり粗いものになっており、実際の成功確率はもっと高くなります。例えば、

- 図 5.3 のピークになっているような k しか考慮していない
- sin 関数と $\phi(r)/r$ の評価で実用上あまり起きない最悪の場合を考慮している

といった部分で粗い評価を行っています。

さらに成功確率を上げる方法はいくつか知られており、例えば

- q を大きく取る (ただし、量子ビットが増えてしまう)
- 測定値 k だけでなく、その周辺 $k\pm1,k\pm2,\ldots$ もチェックする

- $[x]^r \neq [1]$ となり失敗した r を位数の約数だと仮定して、そのような複数の r の最小公倍数を取ることで位数を推定する
- 推定した r だけでなく、r に小さな整数を掛けたものもチェックする

などがあります。

5.4 量子フーリエ変換

ここまでで、Shor のアルゴリズムの大部分の説明が終わりました。あとは、離散フーリエ変換 $\frac{1}{\sqrt{q}}B$ を量子回路で実現する部分だけが残っています。この節では、実際に離散フーリエ変換を実現する量子回路を構成します。ここでも 5.3.2 節で定義した $\chi_k(j)$ をそのまま使います。

5.3.2 節で考察した数列 a の離散フーリエ変換を量子ビットの記号で表してみましょう:

$$\begin{aligned}\sum_{j=0}^{q-1} a_j |j\rangle \mapsto \sum_{k=0}^{q-1} c_k |k\rangle \\ &= \sum_{k=0}^{q-1} \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} \overline{\chi_k(j)} a_j |k\rangle \\ &= \frac{1}{\sqrt{q}} \sum_{j=0}^{q-1} a_j \left(\sum_{k=0}^{q-1} \overline{\chi_k(j)} |k\rangle \right).\end{aligned}$$

特に基底の行き先は

$$|j\rangle \mapsto \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \overline{\chi_k(j)} |k\rangle \quad (0 \leq j < q)$$

となります。

定義 5.4.1. 正の整数 q に対して、**量子フーリエ変換** QFT_q を以下が成り立つように定義する:

$$\begin{aligned}\mathrm{QFT}_q |j\rangle &= \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \chi_k(j) |k\rangle \\ &= \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \exp\left(2\pi\sqrt{-1}\frac{kj}{q}\right) |k\rangle \qquad (0 \leq \forall j < q).\end{aligned}$$

逆量子フーリエ変換 QFT_q^{-1} は QFT_q の逆写像であり、以下の式を満たす:

$$\begin{aligned}\mathrm{QFT}_q^{-1} |j\rangle &= \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \overline{\chi_k(j)} |k\rangle \\ &= \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \exp\left(-2\pi\sqrt{-1}\frac{kj}{q}\right) |k\rangle \qquad (0 \leq \forall j < q).\end{aligned}$$

定義の中の QFT_q^{-1} の式は定理 5.3.6 から得られます。数学で離散フーリエ変換と呼ばれるものは、量子情報理論ではしばしば逆量子フーリエ変換と呼ばれていることに注意してください。

QFT_q が実現できれば、回路の並びを逆にし位相ゲート $R(\theta)$ を $R(-\theta)$ で置き換えれば、QFT_q^{-1} の回路も簡単に作ることができます。あるいは、QFT_q^{-1} は QFT_q と exp 内の符号が異なるだけでほぼ同じものなので、以下で構成する回路内の $R(\theta)$ を $R(-\theta)$ で置き換えれば実現できます。$q = 2^n$ とし、以下では量子フーリエ変換 QFT_q を実現する回路について考えましょう。

例 5.4.2 ($q = 2$ の場合)**.** $q = 2$ の場合、つまり 1 量子ビットの場合、

$$\chi_k(j) = \exp\left(\frac{2\pi\sqrt{-1}}{q} jk\right) = (-1)^{jk}$$

なので、量子フーリエ変換は

$$\mathrm{QFT}_2 |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$\mathrm{QFT}_2 |1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

となる。これは Hadamard ゲート H と一致し、$j \in \{0, 1\}$ を使って表すと

$$\begin{aligned} H|j\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^j |1\rangle\right) \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle + \chi_{2^0}(j)|1\rangle\right) \end{aligned}$$

と表せる。

QFT_q を実現するための基本的なアイデアは $x^j \bmod N$ を計算するアルゴリズム (5.2 節) と同様で、$\chi_k(j)$ のパラメータの j や k を 2 進数で表して、2 のべき乗の積み重ねに分解するというものです。$\chi_k(j)$ は $\exp\left(2\pi\sqrt{-1}\frac{kj}{q}\right)$ で定義されていたので、

$$\chi_k(j) = \chi_k(1)^j, \quad \chi_k(j) = \chi_1(j)^k$$

という式が成り立ちます。

補題 5.4.3. $0 \le j < q$ とすると、

$$\mathrm{QFT}_q |j\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \chi_{2^{n-1}}(j)|1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + \chi_{2^{n-2}}(j)|1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + \chi_{2^0}(j)|1\rangle\right)$$

が成り立つ。

証明. 定義から、

$$\mathrm{QFT}_q |j\rangle = \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \chi_k(j) |k\rangle$$

である。ここで、$k = k_1 k_2 \cdots k_n$ と 2 進数で表すと、

$$\begin{aligned} k &= k_1 2^{n-1} + k_2 2^{n-2} + \cdots + k_n 2^0, \\ \chi_k(j) &= \chi_{k_1 2^{n-1}}(j) \chi_{k_2 2^{n-2}}(j) \cdots \chi_{k_n 2^0}(j) \end{aligned}$$

となる。したがって、

$$\begin{aligned} \mathrm{QFT}_q |j\rangle &= \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} \chi_k(j) |k\rangle \\ &= \frac{1}{\sqrt{q}} \sum_{k_1, k_2, \ldots, k_n \in \{0,1\}} \chi_{k_1 2^{n-1}}(j) |k_1\rangle \otimes \chi_{k_2 2^{n-2}}(j) |k_2\rangle \otimes \cdots \otimes \chi_{k_n 2^0}(j) |k_n\rangle \\ &= \frac{1}{\sqrt{2}} \left(|0\rangle + \chi_{2^{n-1}}(j) |1\rangle\right) \otimes \frac{1}{\sqrt{2}} \left(|0\rangle + \chi_{2^{n-2}}(j) |1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}} \left(|0\rangle + \chi_{2^0}(j) |1\rangle\right) \end{aligned}$$

となる。$\chi_0(j) = 1$ に注意。 □

例 5.4.4 ($q = 4$ の場合)**.** $q = 4$ の場合、つまり 2 量子ビットの場合、

$$\chi_k(j) = \exp\left(\frac{2\pi\sqrt{-1}}{q} jk\right) = \sqrt{-1}^{jk}$$

となる。量子フーリエ変換は

$$\begin{aligned} \mathrm{QFT}_4 |00\rangle &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\ &= \frac{1}{2} \left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle + |1\rangle\right) \\ \mathrm{QFT}_4 |01\rangle &= \frac{1}{2}(|00\rangle + \sqrt{-1} |01\rangle - |10\rangle - \sqrt{-1} |11\rangle) \\ &= \frac{1}{2} \left(|0\rangle - |1\rangle\right) \otimes \left(|0\rangle + \sqrt{-1} |1\rangle\right) \\ \mathrm{QFT}_4 |10\rangle &= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \\ &= \frac{1}{2} \left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle - |1\rangle\right) \\ \mathrm{QFT}_4 |11\rangle &= \frac{1}{2}(|00\rangle - \sqrt{-1} |01\rangle - |10\rangle + \sqrt{-1} |11\rangle) \\ &= \frac{1}{2} \left(|0\rangle - |1\rangle\right) \otimes \left(|0\rangle - \sqrt{-1} |1\rangle\right) \end{aligned}$$

となる。基底の行き先はどれも量子もつれ状態ではなく、しかも計算結果を上から順に見ていくと、第 1 量子ビットと第 2 量子ビットはどちらも規則的に並んでいる様子がわかる。一般的に書いてみると

$$\mathrm{QFT}_4 |j_1 j_2\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + (-1)^{j_2} |1\rangle\right) \otimes \frac{1}{\sqrt{2}} \left(|0\rangle + \sqrt{-1}^{2j_1 + j_2} |1\rangle\right) \quad (j_1, j_2 \in \{0, 1\})$$

あるいは

$$\mathrm{QFT}_4 |j\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \chi_{2^1}(j)|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + \chi_{2^0}(j)|1\rangle) \quad (0 \le j < 4)$$

となる。

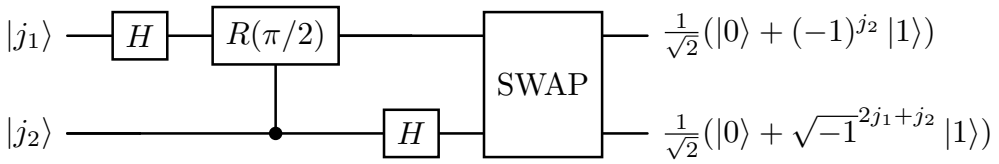


図 5.5: QFT_4 の量子回路

QFT_4 は図 5.5 の量子回路で実現できる。最後の SWAP は定理 3.1.1 で作った、量子ビットの順番を入れ替える回路である。$R(\pi/2)$ は 2.5.6 節で導入した位相ゲートで

$$R(\pi/2)(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle + e^{\sqrt{-1}\pi/2}\beta|1\rangle = \alpha|0\rangle + \sqrt{-1}\beta|1\rangle \quad (\alpha, \beta \in \mathbb{C})$$

というものだった。図 5.5 では、$R(\pi/2)$ を制御ゲートにした $CR(\pi/2)_{2\to1}$ (3.1.2 節で導入) を使っているので、$j_2 = 1$ の場合だけ $R(\pi/2)$ が第 1 量子ビットに作用する。これは $|1j_2\rangle$ に $\sqrt{-1}^{j_2}$ を掛けるという操作と等しい。したがって、

$$\begin{aligned} CR(\pi/2)_{2\to1}(H \otimes I)|j_1 j_2\rangle &= \frac{1}{\sqrt{2}} CR(\pi/2)_{2\to1}(|0\rangle + (-1)^{j_1}|1\rangle) \otimes |j_2\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{j_1}\sqrt{-1}^{j_2}|1\rangle) \otimes |j_2\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + \sqrt{-1}^{2j_1+j_2}|1\rangle) \otimes |j_2\rangle \end{aligned}$$

となり、$\mathrm{QFT}_4 |j\rangle$ の第 2 量子ビットを作ることができる。ただし、SWAP を行う前なので第 1 量子ビットに保持されていることに注意しよう。

補題 5.4.3 で示したように $\mathrm{QFT}_q |j\rangle$ の各量子ビットはもつれておらず独立した状態になっているので、出力の量子ビットを一つ一つ順々に作っていけばよさそうです。図 5.5 の最後でなぜ SWAP が必要になるかについては後に回すとして、まずは $\mathrm{QFT}_q |j\rangle$ の $n-l$ 番目の量子ビット $|0\rangle + \chi_{2^l}(j)|1\rangle$ を作る方法を述べましょう。厳密には $1/\sqrt{2}$ が必要ですが、以下では省略して説明します。

この部分に関しては $x^j \bmod N$ を計算するアルゴリズムとほぼ同じです。$j = j_1 j_2 \cdots j_n$ のように 2 進数で表すと、

$$\begin{aligned} \chi_{2^l}(j) &= \chi_{2^l}(j_1 2^{n-1})\chi_{2^l}(j_2 2^{n-2}) \cdots \chi_{2^l}(j_n 2^0) \\ &= \chi_{2^l}(2^{n-1})^{j_1}\chi_{2^l}(2^{n-2})^{j_2} \cdots \chi_{2^l}(2^0)^{j_n} \end{aligned}$$

となります。χ_{2^l} は周期が 2^{n-l} なので、$\chi_{2^l}(2^{n-l}) = \chi_{2^l}(2^{n-l+1}) = \cdots = 1$ になります。したがって、

$$\begin{aligned}\chi_{2^l}(j) &= \chi_{2^l}(2^{n-l-1})^{j_{l+1}}\chi_{2^l}(2^{n-l-2})^{j_{l+2}}\cdots\chi_{2^l}(2^0)^{j_n} \\ &= (-1)^{j_{l+1}}\exp\left(\frac{\pi}{2}\sqrt{-1}\right)^{j_{l+2}}\cdots\exp\left(\frac{\pi}{2^{n-l-1}}\sqrt{-1}\right)^{j_n}\end{aligned}$$

と表せます。この式から、$|0\rangle + \chi_{2^l}(j)\,|1\rangle$ を作るには、まず Hadamard ゲートで $|0\rangle + (-1)^{j_{l+1}}\,|1\rangle$ を作り、次に位相ゲート $R(\frac{\pi}{2})$、$R(\frac{\pi}{4})$、…、$R(\frac{\pi}{2^{n-l-1}})$ を j の各ビットの値に応じて掛けていけばよいことが分かります。$l+i$ 番目 $(i \geq 2)$ のビット j_{l+i} が 1 のときだけ位相ゲート $R(\frac{\pi}{2^{i-1}})$ を作用させる処理なので、第 $l+i$ 量子ビットを制御ビットとする制御位相ゲートを使えば実現できます。以上より、図 5.6 の量子回路で $|0\rangle + \chi_{2^l}(j)\,|1\rangle$ を作ることができます。

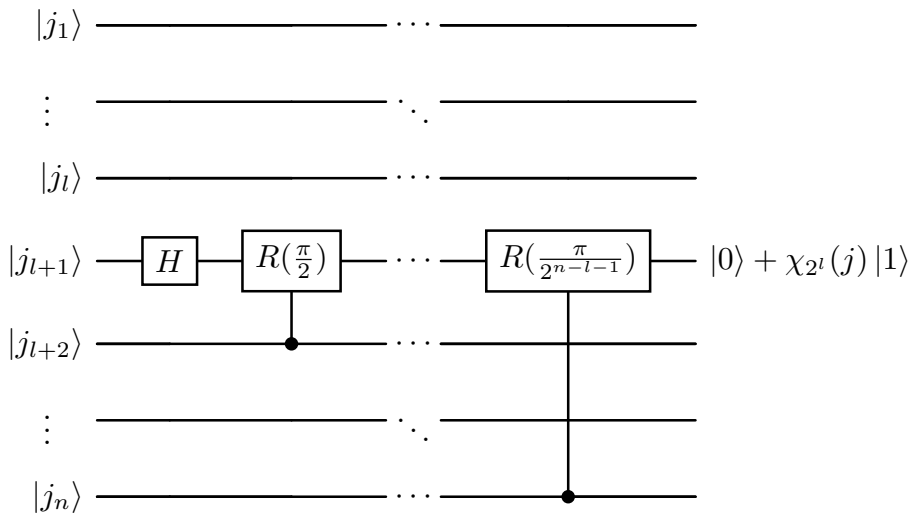


図 5.6: $\mathrm{QFT}_q\,|j\rangle$ の $n-l$ 番目の量子ビットの状態を作る回路

あとは、図 5.6 の回路を $l = 0, 1, \ldots, n-1$ に対して順番に並べていけば、QFT_q を作ることができます。ただし、並べる順番は何でもよいわけではなく、$l = 0$ を最初に、次に $l = 1, 2, \ldots$ の順で並べていき、そして $l = n-1$ を最後に置く必要があります。これは、図 5.6 の回路を実行すると入力の $l+1$ 番目の量子ビット $|j_{l+1}\rangle$ が変化し、その後の計算で j_{l+1} が使えなくなってしまうからです。図 5.7 は並べる順番を間違えた QFT_4 の回路ですが、最初の Hadamard ゲートで $|j_2\rangle$ を壊しているため、その後の計算がうまくいかなくなっています。

$\mathrm{QFT}_q\,|j\rangle$ の各ビットを計算するために必要な入力 j のビットは表 5.1 のようになっています。j_1 が必要なのは $\mathrm{QFT}_q\,|j\rangle$ の第 n 量子ビットだけなので、最初に $\mathrm{QFT}_q\,|j\rangle$ の第 n 量子ビットを計算して、その結果を j_1 の場所に保存する、という順番がよいことが分かります。

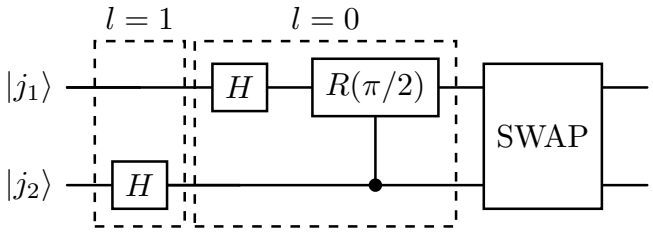


図 5.7: QFT_4 の間違った量子回路

出力 $\mathrm{QFT}_q\,\|j\rangle$ のビット	計算に必要な入力 j のビット
1 番目	j_n
2 番目	j_{n-1}, j_n
⋮	⋮
$n-1$ 番目	$j_2, j_3, \ldots, j_n$
n 番目	$j_1, j_2, j_3, \ldots, j_n$

表 5.1: 各出力ビットと計算に必要な入力ビットの対応

以上をまとめれば、QFT_q を実現する量子回路が完成します。例えば、$q=8$ の場合は図 5.8 の量子回路になります。先ほど述べた計算順序の都合上、最後に量子ビットの順番を反転する必要があるので、SWAP を施しています。$q=8$ では第 2 量子ビットは移動させる必要はありませんが、一般の場合も考慮してすべての量子ビットを含むように SWAP を書いています。SWAP 部分を除くと $1+2+\cdots+n=n(n+1)/2$ 個の量子ゲートが必要で、SWAP には $3\lfloor n/2 \rfloor$ 個の量子ゲートが必要なので、全体ではおおよそ $n^2/2+2n$ 程度のゲート数になっています。

定理 5.4.5. 量子フーリエ変換 QFT_q は、量子ゲート数がおおよそ $n^2/2+2n$ 個の量子回路で実現できる。

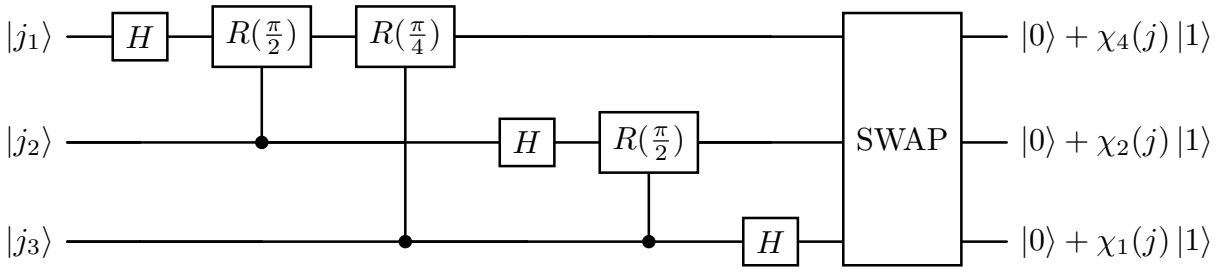


図 5.8: QFT_8 の量子回路 (出力側の $1/\sqrt{2}$ 倍は省略している)

以上で離散フーリエ変換の量子回路版である量子フーリエ変換を実現できました。量子ビット

の状態をベクトルとみなすと、離散フーリエ変換と逆量子フーリエ変換はまったく同じ線形写像で表されます。離散フーリエ変換は $q \times q$ の行列を掛ける操作なので、行列とベクトルの積の定義に基づいて計算すると、q^2 回の掛け算が必要になります。古典的なコンピュータで離散フーリエ変換を高速に計算する手法として高速フーリエ変換が知られていますが、それでもおおよそ $q\log_2(q)$ 回の掛け算が必要です。q 次元の数ベクトルの成分をすべて変化させる以上、古典的なコンピュータでは最低でも q 回の何らかの操作が必要になります。

一方、量子フーリエ変換では $n^2 = (\log_2 q)^2$ 程度の量子ゲート数で実現できています。これは古典的なコンピュータでの高速フーリエ変換よりもはるかに効率的です。ベクトルの成分の個数 q よりも小さい量子ゲート数で計算できていると考えると不思議に思えます。この理由を考えてみましょう。

例えば、量子フーリエ変換で用いた第 1 量子ビットに H を作用させる量子ゲート $H \otimes I^{\otimes n-1}$ は、一般的な量子ビットの状態に作用させると

$$(H \otimes I^{\otimes n-1}) \sum_{j=0}^{q-1} a_j |j\rangle = \sum_{j=0}^{q-1} a_j \left(\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{j_1} |1\rangle) \right) \otimes |j_2 j_3 \cdots j_n\rangle$$

となり、q 個の成分すべてに影響を与えています。1 つの量子ビットを操作する量子ゲートであっても、ベクトルとしては大量の成分を同時に変化させており、これが量子回路モデルで量子フーリエ変換が効率的に実現できる理由の一つとなっています。

5.5 位数推定まとめ

Shor のアルゴリズムの位数推定全体で必要な量子ゲート数と成功確率などをまとめておきましょう。ここでの定量的な評価の多くは、N に対してどの程度の大きさかを見やすくするために、定数倍や小さい次数の項を無視しています。

位数推定全体は以下のような処理でした。

(1) $N, x, q = 2^n$ から図 5.9 の量子回路を作り、測定値の前半 n ビットを k とする。
(2) k/q を連分数展開 (定理 4.5.10) して得られる分数列 p_i/q_i のうち、q_i が N 未満かつ最大のものを k'/r とする。
(3) $[x]_N^r = [1]_N$ かチェックし、違っていれば (1) に戻る。
(4) そうでなければ r を位数と推定し終了。

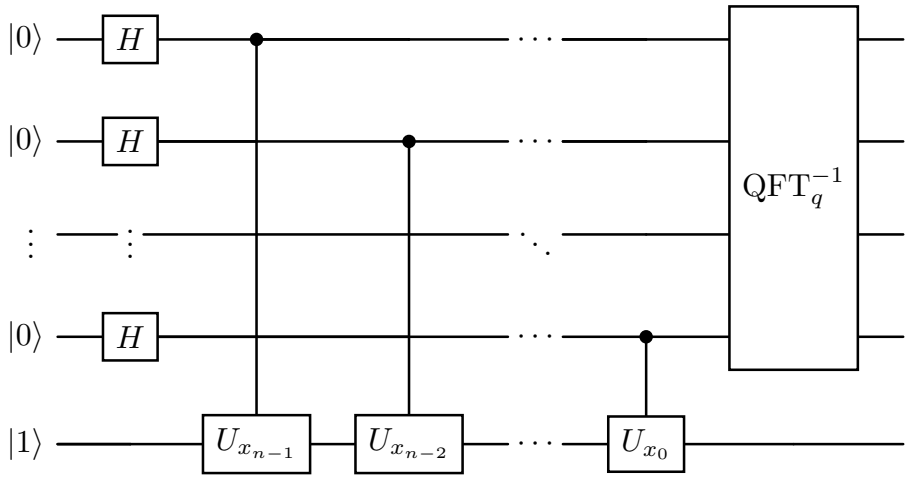


図 5.9: 位数推定を行う回路

まず回路に必要な量子ゲートの個数ですが、定理 5.2.1 と定理 5.4.5 を合わせると、$\log_2(N)^3$ 程度の量子ゲート数になります。前半の $f(j)$ を計算する部分が最も重い処理になっています。

(2), (3) は古典的なコンピュータで計算します。計算は 2 進数で行い、1 桁の和や積に 1 単位時間かかると仮定して計算時間を見積もることにしましょう。例えば、2 進数で m 桁の整数の掛け算は、筆算を使うと m^2 程度の時間がかかります。

(2) の連分数展開に必要な割り算の回数は定理 4.5.10 で評価しており、$\log_2(N)$ で抑えられます。割り算は、2 進数の筆算で行うなら、1 回あたり $\log_2(N)^2$ 程度の計算時間になります。よって、連分数展開全体では $\log_2(N)^3$ 程度の計算時間がかかります。

(3) の $[x]_N^r = [1]_N$ かチェックする部分は、5.2 節で紹介した方法を使えば、$\log_2(N)^3$ 程度の計算時間になります。この部分は、量子か古典かの違いだけで U_f と同じ計算になっています。

1 回の (1) から (4) の計算で位数が求まる確率は定理 5.3.16 で下から評価しており、

$$\frac{1}{e^\gamma \log\log N + \frac{3}{\log\log N}} \cdot \frac{4}{\pi^2} \cdot \left(1 - \frac{1}{N}\right)$$

以上の成功確率でした。N が十分大きければ、$3/\log\log N$ や $1/N$ を 0 とみなしてもこの関数の値はほとんど変わらないので、

$$\frac{1}{e^\gamma \log\log N} \cdot \frac{4}{\pi^2}$$

程度になります。したがって、(1) から (4) を $\log\log N$ の数倍の回数繰り返せば、ほぼ確率 1 で位数が求まるといえます。

以上をまとめると、

- 量子回路に必要な量子ゲート数は $\log_2(N)^3$。
- 連分数展開や位数のチェックにかかる時間は $\log_2(N)^3$。

- 全体の計算を $\log\log N$ の数倍程度繰り返せば、ほぼ確率 1 で位数が求まる。

となります。2 から $\sqrt{N}$ までの整数で N を割って約数を見つける場合、$\sqrt{N} = 2^{\log_2(N)/2}$ という $\log_2(N)$ に関して指数関数的な時間がかかります。Shor のアルゴリズムは、多項式それも 3 次という低い次数の計算時間しかかからず、はるかに高速だと分かります。ちなみに、古典コンピュータで高速な素因数分解アルゴリズムとして知られている一般数体篩法では、おおよそ $\exp\left((64/9)^{1/3}\log(N)^{1/3}(\log\log(N))^{2/3}\right)$ という計算時間になり、$\log_2(N)$ の指数関数より高速ですが Shor のアルゴリズムのような多項式ではありません。

本テキストでは、整数の積や剰余の計算アルゴリズムとして 2 進数の筆算を使っています。積や剰余を計算する高速なアルゴリズムとして、高速フーリエ変換や Karatsuba 法と呼ばれるものが知られており、それらで置き換えることもできます。ただし、高速フーリエ変換のような複雑なアルゴリズムの場合、量子ゲートに翻訳したときに回路が複雑になる、量子ビットが大量に必要になる、といった弊害もあり単純に置き換えればよいというわけでもありません。

参考文献

量子情報理論・量子アルゴリズム全般に関する文献

[1] 上坂吉則, 『量子コンピュータの基礎数理』, コロナ社 (2000).

[2] 中原幹夫, “量子計算入門 (講義ノート)”, 物性研究 **83**(6), 699–786 (2005).
http://hdl.handle.net/2433/110153

[3] 中山茂, 『量子アルゴリズム』, 技報堂出版 (2014).

[4] 西野哲朗, 『量子コンピュータ入門』, 東京電機大学出版局 (1997).

[5] 縫田光司, “量子計算、量子アルゴリズムと有限群の表現論”, 2009 年度表現論シンポジウム講演集, 74–85 (2009).
https://doi.org/10.34508/repsympo.2009.0_74

[6] 細谷暁夫, 『量子コンピュータの基礎 第 2 版』 (別冊数理科学 SGC ライブラリ 69), サイエンス社 (2009).

[7] 宮野健次郎, 古澤明, 『量子コンピュータ入門 第 2 版』, 日本評論社 (2016).

[8] J. A. Buchmann, *Introduction to Quantum Algorithms* (Pure and Applied Undergraduate Texts 64), American Mathematical Society (2024).

[9] A. Glassner, *Quantum Computing: From Concepts to Code*, No Starch Press (2025).

[10] N. D. Mermin (著), 木村元 (訳), 『量子コンピュータ科学の基礎』, 丸善出版 (2009).

[11] M. A. Nielsen, I. L. Chuang (著), 木村達也 (訳), 『量子コンピュータと量子通信 II －量子コンピュータとアルゴリズム－』, オーム社 (2005).

Shor のアルゴリズムに関する文献

[12] 國廣昇, “Shor のアルゴリズムに基づく素因数分解実験の調査”, 第 39 回量子情報技術研究会 (QIT39) 予稿集, 2018 年 11 月 26 日.

図に関して

[13] Wolfram Research, Inc., Mathematica, Version 14.3, Champaign, IL, 2025.
図 1.1、図 5.2、図 5.3 は本ソフトウェアを利用して作成された。

索引

A

ancilla 41

B

Bell 状態 49

C

CNOT ゲート 27

E

EPR ペア 49
Euler のトーシェント関数 70

H

Hadamard ゲート 27

L

Legendre の定理 83

S

Shor のアルゴリズム 4

T

Toffoli ゲート 28

X

XOR 28

あ

アルゴリズム 1

い

位数 63
位相ゲート 29

え

エンタングル状態 46

か

可逆 60
重ね合わせ 17

き

基底 16
逆行列 96
逆量子フーリエ変換 108

け

ゲート操作 2
計算モデル 2
原始根 65

さ

作用 24

し

次元 16
準同型 68
N を法とする a の剰余類 58

す

数ベクトル 13
数ベクトル空間 14

せ

制御ビット 28
正則行列 96
線形結合 15
線形写像 25

そ

測定 2, 22

た

第 1 量子ビット 21
多重線形性 18

ち

中国剰余定理 67

て

テンソル積 18, 32

と

同型 68

は

排他的論理和 28

ひ

ビット 17
標準基底 16

標的ビット 28

ふ

フーリエ係数 98

へ

ベクトル 14
ベクトル空間 14

ほ

補助ビット 41

り

離散フーリエ変換 98
量子回路図 34
量子回路モデル 2, 34
量子ゲート 2
量子ビット 17
量子フーリエ変換 108
量子もつれ状態 46, 49

れ

レジスタ 22
連分数 79
連分数展開 84

内閣府総合科学技術・イノベーション会議の研究開発と Society 5.0 との橋渡しプログラム (BRIDGE)
令和 7 年度採択施策「量子人材教育エコシステムの開発と試行」

量子数理教育
～数学的アプローチによる量子人材育成～
入門編

著者：　落合啓之 (九州大学マス・フォア・インダストリ研究所/教授)
北川宜稔 (九州大学マス・フォア・インダストリ研究所/学術研究員)

連絡先：　ochiai@imi.kyushu-u.ac.jp

印刷：　城島印刷株式会社

2026 年 2 月 11 日　発行