Google Verifiable Quantum Circuit Compilation

KickMix の 3 課題を理解する

invsqrt・dqi・dfthcsos の数学的背景と回路最適化

A Self-Contained Guide to the KickMix Benchmarks and Circuit Optimization

2026 年 11 月 2 日改訂
公開回路の基準日：2026 年 10 月 30 日

はじめに

本書は Kaggle「Google Verifiable Quantum Circuit Compilation」の参加者に向けて、invsqrt・dqi・dfthcsos の数学的背景、公開回路の構造、費用と検証の読み方を説明する技術ガイドである [2, 1]。可逆論理と基本的な線形代数を出発点とし、符号理論と電子構造計算の必要な定義は本文で導入する。

3 課題は何を計算するのか

表 1　科学的目標とコンペが要求する部分計算。

課題	アルゴリズム全体の目的	提出する部分計算
invsqrt	Coulomb 相互作用などのシミュレーション	$1/\sqrt{x}$ に比例する位相を付ける
dqi	OPI 最適化問題を量子干渉で扱う	RS 復号を用い測定記録に対応する位相を補正する
dfthcsos	電子ハミルトニアンのエネルギー推定	係数の因子分解と回転角の lookup を最適化する

どの課題も量子アルゴリズム全体の一部である。dqi は物理量子ビットの誤り訂正ではなく、dfthcsos は FeMoCo の基底状態エネルギーを実際に計算する課題ではない。

構成と数値の読み方

第 1 章で KickMix と Verifier、第 2 章で Carleman 線形化と Walsh 展開・Horner 法・ANF、第 3 章で共通の最適化手法を学ぶ。第 4〜6 章は各課題を背景から費用分析へ順に説明する。第 7 章は再現と検証、第 8 章は比較と今後の課題を扱う。付録には有限体、多項式 Euclid 法、記号と採点条件、解析対象の識別子と再現条件をまとめる。

本書の回路比較は **2026 年 10 月 30 日時点の取得版**を基準とする。以前の版を解析する場合はその取得日を添える。現在の順位や最良記録を主張するものではない。公開回路は mrmeows、SomeJoint、busyaprime、Ozer らの成果物であり、作者の説明と回路を出典として扱う [3, 6, 4]。費用は小さいほどよく、参照比は「参照費用／提出費用」で大きいほどよい。3 課題の参照比の幾何平均が総合評価になる。

数値は既存の公開物とその検証記録に基づく歴史的比較であり、本書の改訂時に全回路を再実行した結果ではない。公式 Verifier の判定、全入力の厳密計算、著者報告、解析モデルの推定を区別する。macOS で公開ソースからビルドした KickMix の検証記録を含み、競技配布 Linux バイナリとの同一性を保証するものではない。

表 2　参照費用と公開回路の比較例（2026 年 10 月 30 日取得版）。公開物の数値を丸めて掲載。invsqrt は全有効入力各 64 ショット、dqi は既定 102,400 ショット、dfthcsos は既定検査に対応する。再現条件は第 7 章。

課題	公開回路の出典・構成	参照費用	公開費用
invsqrt	mrmeows 系・busyaprime 改良、63 量子ビット	2.07677×10^{6}	102,349.24
dqi	SomeJoint 配布、889 量子ビット	2.44395×10^{13}	4.1421×10^{11}
dfthcsos	SomeJoint、$(R, B, C) = (8, 143, 60)$、$J = 5$	3.35560×10^{9}	340,232,090

目次

はじめに i

第 1 章 KickMix と Verifier 1
1.1 KickMix とは . . . 1
1.2 使える命令 . . . 2
1.3 なぜ古典的にシミュレートできるのか . . . 4
1.4 何を検証でき、何を仮定しているか . . . 6

第 2 章 回路最適化の数学 8
2.1 出発点：位相オラクルは「線形結合」で書ける . . . 9
2.2 Carleman 線形化 . . . 9
2.3 Walsh 展開：ビット全体での厳密な持ち上げ . . . 13
2.4 Horner 計算：持ち上げない、因数分解する . . . 20
2.5 ANF：GF(2) 上の多項式としての持ち上げ . . . 26
2.6 比較と適用条件 . . . 32

第 3 章 KickMix 回路の最適化手法 36
3.1 課題ごとの費用モデル . . . 36
3.2 費用モデルが報いるもの . . . 37
3.3 Toffoli を減らす . . . 38
3.4 CX は無料だが、無害ではない . . . 43
3.5 位相回転の最適化 . . . 44
3.6 量子ビット数を減らす技法 . . . 47
3.7 ルックアップ：パリティ積（ANF）型 . . . 49
3.8 signed-digit Horner . . . 51
3.9 精度と回路資源の取引 . . . 51
3.10 公開回路における位置づけのまとめ . . . 51
3.11 本章のまとめ . . . 52

第 4 章 invsqrt 53
4.1 物理的背景 . . . 53
4.2 課題：位相 $1/\sqrt{x}$ を作る . . . 53
4.3 260 量子ビットの参照回路 . . . 54

4.4 公開回路の構造 . . . 56
4.5 区分多項式と Horner 法 . . . 57
4.6 lookup・算術・位相回転 . . . 57
4.7 費用分析 . . . 57
4.8 今後の最適化方向 . . . 57

第 5 章 dqi 58
5.1 エラー訂正と Reed–Solomon 符号 . . . 58
5.2 シンドローム . . . 59
5.3 DQI と課題の範囲 . . . 60
5.4 Euclid 法から復号へ . . . 60
5.5 鍵方程式 . . . 61
5.6 拡張 Euclid 法 . . . 61
5.7 Chien 探索 . . . 62
5.8 Forney の式 . . . 62
5.9 逆計算して S を復元する理由 . . . 63
5.10 参照回路と費用の考え方 . . . 64

第 6 章 dfthcsos 66
6.1 課題：FeMoCo のブロックエンコーディング . . . 66
6.2 baseline と公開回路の再現 . . . 76
6.3 ANGLE 回路の構成 . . . 78
6.4 費用分析と今後の最適化方向 . . . 82

第 7 章 公開回路の評価と検証方法 84
7.1 公開回路の再現方法 . . . 84
7.2 全入力検証とランダム検証 . . . 84
7.3 正しさと近似誤差 . . . 85
7.4 解析的費用モデル . . . 85
7.5 結果の信頼性と再現可能性 . . . 85

第 8 章 3 課題の比較と今後の最適化課題 86
8.1 3 課題の比較 . . . 86
8.2 最適化で重要な共通原理 . . . 86
8.3 残る研究課題 . . . 86
8.4 参加者への技術的示唆 . . . 87

付録 A 有限体 GF(2^m) の基礎 88

付録 B 多項式 Euclid 法の計算例 89

付録 C 用語・記号一覧 90

付録 D　Verifier とスコアの定義 91

付録 E　解析対象と再現条件 92
E.1　回路と提出物の識別子 92
E.2　Verifier の版と実行環境 93
E.3　Walsh 打ち切りと係数探索の条件 93

参考文献 94

第 1 章

KickMix と Verifier

この章で学ぶこと

- KickMix 回路のクラスと、使える命令を知る。
- 重ね合わせを作らない回路が古典的に高速シミュレートできる理由を理解する。
- 公式 Verifier が何を検査し、何を仮定しているかを区別する。

1.1 KickMix とは

KickMix は、Google Quantum AI が耐故障量子計算のフレームワーク Qualtran の中で公開した、回路の記述形式と、その高速シミュレータ（C++20 ／ Python）である [1]。

記述形式は「低レベル」である。すなわち、「$1/\sqrt{x}$ を計算する」「足し算をする」といった高水準の記述ではなく、個々のゲートを実行順に 1 行ずつ並べた命令列で回路を表す。プログラムでいえばアセンブリに近い。同じ回路を、人が読めるテキスト形式（拡張子 `.kmx`）と、機械が速く読めるバイト列形式（`.kmb`）の 2 通りで保存でき、両者は 1 対 1 に対応する。Verifier に提出するのは `.kmb` である。図 1.1 に、公開 63-Q 回路の `.kmx` の先頭部分を示す（全体で 46,633 行）。

```
R q22
X q22
CX q21 q22
APPEND_TO_REGISTER q0 r0      # 入力 x のビット0 を宣言
...
REGISTER r0 "x"
CX q20 q23
CCX q23 q19 q24               # q23 と q19 が両方1なら q24 を反転
CCX q24 q18 q25
...
HMR q47 b317                  # q47 をX基底で測って結果を b317 に
CZ q41 q1 if b317             # b317 が1のときだけ位相補正
...
Z_POW q59 0.0000076293958...  # 一般角の位相回転（半回転単位）
```

図 1.1　`.kmx` 形式の例（公開 63-Q 回路の抜粋。`#` 以降は本書で付けた注釈で、実際のファイルにはない）。

耐故障量子アルゴリズムの費用の大半を占めるサブルーチン、すなわち可逆な古典算術、QROM によるデータ参照、位相オラクルなどを対象とし、AI エージェントが回路を自動探索・最適化できる環境を提供することを目的としている。

KickMix が扱うのは、次の要素だけでできた回路である。

- 古典可逆な論理ゲート（X、CX、CCX、SWAP、CSWAP）
- 対角な位相ゲート（Z、CZ、CCZ、任意角の Z_POW、大域位相反転 NEG）
- X 基底での測定とリセット（HMR）、リセット（R）
- 古典ビットの操作と、古典ビットによる条件付き実行

どの計算基底入力 $|x\rangle$ から始めても、状態は常に「一つの計算基底状態 × スカラー位相」にとどまる。名称の *kick* は、HMR や位相ゲートによる位相キックバック（phase kickback）に由来すると説明されている。*mix* の由来は資料中に明記されていない。

1.2　使える命令

表 1.1　KickMix の主な命令（公開 KickMix の命令仕様）。どの命令にも `ifbN`（古典ビットが 1 のときだけ実行）を付けられる。

分類	命令	作用
可逆論理	X, CX, CCX	ビット反転、制御反転、二重制御反転（Toffoli）
	SWAP, CSWAP	交換、制御付き交換（Fredkin）
対角位相	Z, CZ, CCZ	$\|1\rangle$、$\|11\rangle$、$\|111\rangle$ に -1
	Z_POW q t	$\|1\rangle \mapsto \mathrm{e}^{\mathrm{i}\pi t}\|1\rangle$（$t$ は半回転単位。$t = 1, \frac{1}{2}, \frac{1}{4}$ が Z, S, T）
	NEG	大域位相に -1
測定・リセット	HMR q b	X 基底で測定し結果を b に書き、q を 0 にリセット
	R q	q を 0 にリセット（$q = 1$ だった場合は位相をランダムに反転）
古典ビット	BIT_STORE0/1, BIT_INVERT	古典ビットへの 0・1 の書き込み、反転
制御	PUSH_CONDITION, POP_CONDITION	ブロック単位の条件（条件スタック）
宣言	REGISTER, APPEND_TO_REGISTER	入出力レジスタの宣言（シミュレーションには影響しない）

1.2.1　HMR と測定ベースの逆計算

HMR は Hadamard → 測定 → リセット（”hammer”）の略である。補助量子ビットを「計算をやり直さずに」0 へ戻すために使う。具体例で順に追う。

■状況　以前に CCX(a, c, f) で、0 だった補助量子ビット f に $f = a \wedge c$（a と c がともに 1 のときだけ 1）を書き込んだとする。入力は重ね合わせなので、状態は

$$\sum_x \alpha_x \left|x\right\rangle \left|a(x)\right\rangle \left|c(x)\right\rangle \left|f(x)\right\rangle, \qquad f(x) = a(x)\, c(x)$$

である。f はもう不要なので 0 に戻したい。素直には CCX(a, c, f) をもう一度掛ければよいが、Toffoli が 1 個かかる。

■手順 1：X 基底で測る　f に Hadamard を掛けると

$$\left|f\right\rangle \;\longmapsto\; \frac{1}{\sqrt{2}}\big(\left|0\right\rangle + (-1)^f \left|1\right\rangle\big)$$

となる。これを計算基底で測り、結果を古典ビット b に書く。どの x についても $\left|0\right\rangle$ と $\left|1\right\rangle$ の振幅の大きさは $1/\sqrt{2}$ で等しいので、b は x によらず確率 $1/2$ ずつで 0 か 1 になる。**b は x について何の情報も持たないので、入力の重ね合わせは壊れない。**

■手順 2：測定後の状態　結果が b だったときの状態は、$\left|b\right\rangle$ 成分の係数 $(-1)^{b \cdot f}$ を各 x に残したものになる。f をリセットすると

$$\sum_x \alpha_x \, (-1)^{b \cdot f(x)} \left|x\right\rangle \left|a\right\rangle \left|c\right\rangle \left|0\right\rangle$$

である。

- $b = 0$ なら位相は付かない。そのまま終わりである。
- $b = 1$ なら、$f(x) = 1$、つまり $a = c = 1$ の成分にだけ -1 が付く。放置すると位相の誤差になる。

■手順 3：補正　CZ(a, c) は「a と c がともに 1 の成分に -1 を掛ける」ゲートで、ちょうど $(-1)^{a \cdot c} = (-1)^f$ を掛ける。そこで $b = 1$ のときだけ CZ(a, c) を掛けると（kmx では `CZqaqcifb`）、

$$(-1)^{f(x)} \cdot (-1)^{a(x)c(x)} = (-1)^{2f(x)} = 1$$

となり、余分な位相が消える。

表 1.2　$b = 1$ のときの各成分の位相。測定で付いた -1 を、CZ がちょうど打ち消す。

a	c	$f = ac$	測定後（$b = 1$）	CZ(a, c)	合計
0	0	0	$+1$	$+1$	$+1$
0	1	0	$+1$	$+1$	$+1$
1	0	0	$+1$	$+1$	$+1$
1	1	1	-1	-1	$+1$

■費用と条件　CCX で消せば Toffoli が 1 個かかるが、HMR と CZ で消せば Toffoli は 0 個である（CZ は Clifford なので費用 0）。これが測定ベースの逆計算（measurement-based uncomputation）である（第 3 章）。ただし、測る時点で a と c が f を作ったときと同じ値を保っている必要がある。

一般に、補正には位相 $(-1)^f$ を掛けられればよい。f が生きている量子ビットの XOR なら Z の組で、積なら CZ で済む。f が複雑な関数のときは、補正のために f を計算し直す必要があり、測定で消す利点がなくなる。算術の結果を消すのに逆計算が要るのはこのためである（第 4 章）。

1.2.2　R とファズテスト

R は、計算を終えて 0 に戻った補助量子ビットを捨てるための命令である。シミュレータは、R を掛けた時点で量子ビットが 1 だった場合に位相をランダムに反転する（公開シミュレータの R の定義）。片付け忘れた量子ビットを R で黙って捨てると、ショットごとに位相がばらつき、Verifier で検出される。

1.3　なぜ古典的にシミュレートできるのか

1.3.1　一般の量子回路シミュレーションが重い理由

n 量子ビットの一般の状態は、2^n 個の複素振幅を並べた状態ベクトル

$$|\psi\rangle = \sum_{y\in\{0,1\}^n} \alpha_y \,|y\rangle$$

で表される。Hadamard のように重ね合わせを作るゲートを 1 個掛けるたびに、0 でない振幅の数は最大で 2 倍になる。$n = 40$ ではすでに約 10^{12} 個の振幅（16 TB 程度）が必要で、260 量子ビットでは原理的に保持できない（図 1.2 左）。

1.3.2　KickMix の命令は「基底を基底へ」写す

KickMix の命令は、どれも次のどちらかである。

- **置換**（X、CX、CCX、SWAP、CSWAP）：基底状態 $|y\rangle$ を別の基底状態 $|\pi(y)\rangle$ に移す。行列で書くと、各列に 1 が 1 個だけある置換行列である。
- **対角**（Z、CZ、CCZ、Z_POW、NEG）：基底状態はそのままで、位相 $\mathrm{e}^{\mathrm{i}\phi(y)}$ を掛ける。行列は対角行列である。

置換行列と対角行列の積は、各列に 0 でない成分が 1 個だけある行列（単項行列、monomial matrix）になり、単項行列どうしの積もまた単項行列である（図 1.3）。したがって回路全体も単項行列で、計算基底入力 $|x\rangle$ に対して

$$\boxed{V\,|x\rangle = \mathrm{e}^{2\pi\mathrm{i}\,\varphi(x)}\,|\pi(x)\rangle}$$

と、**一つの基底状態と一つの位相**だけを返す。1 ショットの模擬に必要なのは、各量子ビットのビット値（n ビット）と位相（1 個の数）だけになる（図 1.2 右）。

1.3.3　重ね合わせ入力でも、基底入力の結果で決まる

回路は線形なので、重ね合わせ入力に対しては

$$V\sum_x \alpha_x\,|x\rangle = \sum_x \alpha_x\,\mathrm{e}^{2\pi\mathrm{i}\varphi(x)}\,|\pi(x)\rangle$$

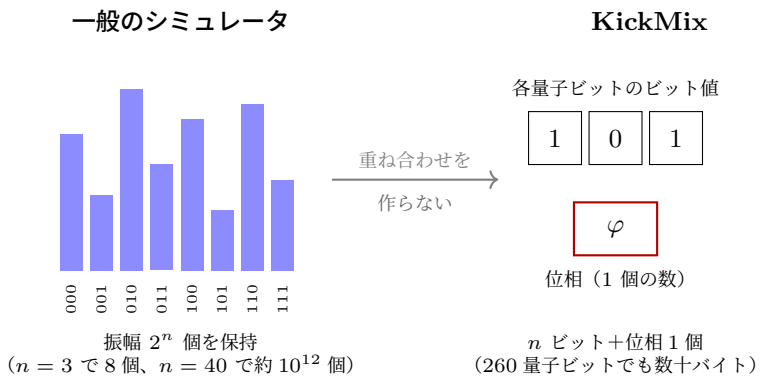


図 1.2　状態の持ち方の比較。KickMix 回路は基底状態を基底状態へ写すので、1 ショットはビット列と位相 1 個で表せる。

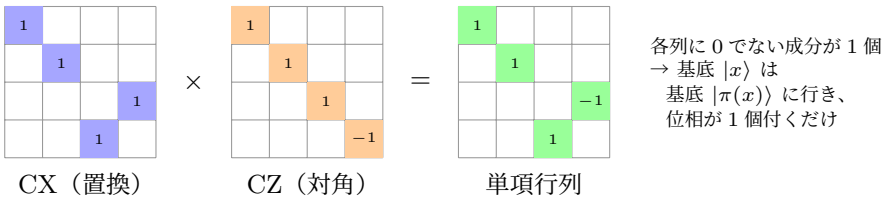


図 1.3　2 量子ビットでの例（行と列は基底 $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ の順）。置換と対角の積は単項行列になり、何個掛けても単項行列のままである。

となる。右辺は各 x の結果（行き先 $\pi(x)$ と位相 $\varphi(x)$）だけで決まる。したがって、すべての基底入力について「入力が戻り（$\pi(x)=x$、補助は 0）、位相が正しい」ことを確かめれば、回路が任意の入力に対して望みの対角ユニタリとして働くことになる。HMR の測定結果がランダムでも、その結果 b は x によらず一様なので、この議論は測定の各結果ごとに成り立つ（論文は、測定ベースの逆計算が基底を保つという仮定の下でこれを示している [1]）。

1.3.4　1 ショットのトレース

表 1.3 は、AND を作って位相を掛け、測定で消す小さな回路を 1 ショットずつ追ったものである。保持しているのはビット値と位相だけで、HMR の結果 b は乱数で決める。

表 1.3　入力 $a=c=1$ での 1 ショットのトレース（f は補助量子ビット、位相は turn 単位）。$b=0$ と $b=1$ のどちらでも、最後の位相は $t/2$ になる。

命令	a	c	f	位相 φ
（初期状態）	1	1	0	0
CCX a c f	1	1	1	0
Z_POW f t	1	1	1	$t/2$
HMR f b（$b=0$ が出た場合）	1	1	0	$t/2$
CZ a c if b（実行されない）	1	1	0	$t/2$
HMR f b（$b=1$ が出た場合）	1	1	0	$t/2+\frac{1}{2}$（$f=1$ なので -1）
CZ a c if b（実行される）	1	1	0	$t/2+1\equiv t/2$

1.3.5　ビット並列で多数のショットを同時に進める

ここでいう **word** は、CPU が 1 命令でまとめて扱う固定長のビット列のことで、典型的には 64 ビット整数（`uint64`）1 個である。SIMD 命令を使えば、128〜512 ビットのレジスタ 1 本を 1 word として扱える。

1 ショットの状態がビット列なので、多数のショットを「量子ビットごとに word を 1 個」用意して詰め込める。第 k ショットでの量子ビット q の値を、量子ビット q 用の word W_q の第 k ビットに置く。つまり、量子ビットの並びが word の並び（$W_0, W_1, \ldots, W_{Q-1}$）に、ショットの並びが word の中のビット位置に対応する。すると、

$$\mathrm{CX}(a,t):\ W_t \leftarrow W_t \oplus W_a, \qquad \mathrm{CCX}(a,c,t):\ W_t \leftarrow W_t \oplus (W_a \wedge W_c)$$

のように、1 回の論理演算で全ショットを同時に更新できる（図 1.4）。KickMix は word の幅に応じて 64〜512 ショットを 1 word に詰め（64 ビット整数なら 64 ショット、512 ビットの SIMD レジスタなら 512 ショット）、論文によれば 1 コアで毎秒数百億命令の速度に達する。ビット並列処理により、多数の基底入力を状態ベクトル全体を保存せずに検査できる。実行時間は命令数と実装・環境に依存する。

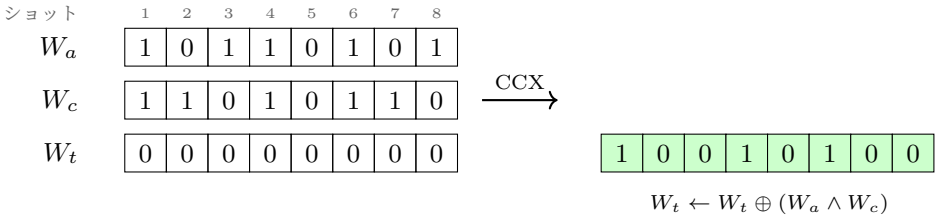


図 1.4　ビット並列の模式図（8 ショットの例）。各 word のビット k が第 k ショットの値を表し、CCX は word どうしの AND と XOR の 1 回ずつで全ショットに作用する。

Note

本書の解析（第 7 章）も同じ性質を使っている。位相を無視してビット値だけを追えば、全入力を一度に（各入力 1 ショットで）厳密に模擬できる。測定結果で分岐する条件付き命令は、位相補正に限られる部分なら値に影響しない。

1.4　何を検証でき、何を仮定しているか

1.4.1　公式 Verifier が検査すること

公式 Verifier（`verify_submission_invsqrt.py`）は、すべての $x \in [1, 3{,}133{,}452]$ について各 64 ショットを実行し、次を検査する。

- 入力レジスタが x に戻っていること、他の量子ビットがすべて 0 に戻っていること。
- 各ショットの位相 θ（turn、mod 1）と目標 $1/\sqrt{x}$ の円周距離が、許容値 $2^{-24}/\sqrt{x}$ 以下であること。1 ショットでも超えれば失格である。

- 最悪誤差比と、最小の余裕 min_margin = min(許容値 − 誤差)。

費用は次の式で計算する。

$$\boxed{\begin{aligned} C &= \overline{N_{\text{Toffoli}}} + \frac{1}{4}\Big(T_{\text{base}} + n_{\text{rot}}\big(0.56\log_2(1/\varepsilon) + 5.3\big)\Big), \\ \varepsilon &= \frac{\pi \cdot \text{min_margin}}{n_{\text{rot}}}, \qquad \text{score} = Q \times C \end{aligned}}$$

ここで $\overline{N_{\text{Toffoli}}}$ はショット平均の Toffoli 様ゲート数（CCX、CCZ、CSWAP）、T_{base} は T 角の回転数、n_{rot} は Clifford/T 以外の回転数である。Clifford 角（$t \in \frac{1}{2}\mathbb{Z}$）は費用 0、T 角（$t \in \frac{1}{4}\mathbb{Z}$）は 1 T $= \frac{1}{4}$ Toffoli である。

1.4.2　仮定していること

- **計算基底での検査から一般の入力への拡張。**論文は、測定ベースの逆計算が基底を保つという仮定の下で、ランダムな基底入力での誤り率から重ね合わせ入力での誤差を抑える議論を与えている [1]。Verifier 自体が検査するのは計算基底入力である。
- **測定のランダム性。**HMR の結果は擬似乱数（回路ハッシュ＋ salt）で決まり、物理的な測定の一様性を仮定している。
- **回転合成の誤差モデル。**一般角の回転は Clifford+T に展開せず、合成費用の漸近式で事後的に課金する。誤差は全回転が最悪の向きにそろうと仮定して配分している。

1.4.3　検証可能な構造と候補探索

一般の量子回路の等価性検査は計算量的に難しいが、KickMix 回路は数秒〜数分で正しさと費用を確定できる。そのため、AI エージェントが候補を大量に生成し、許容誤差の限界まで攻めた近似的な変換（レジスタ幅の短縮、多項式次数の変更、キャリーの刈り込みなど）を安全に試せる。テストケースは回路のハッシュから決まる（Fiat–Shamir 型）ので、秘密のテストセットも要らない。論文はこの性質を verifiable structure と呼び、invsqrt を含む三つのベンチマークを Kaggle コンペティションとして公開した [1, 2]。

注意

公式 Verifier の PASS は「全入力・各 64 ショットで許容誤差内だった」ことを示す。回路の数学的な正しさの証明や、最適性の証明ではない。各課題で検査する入力集合と統計的保証は異なるため、検証条件を添えて解釈する。

第 2 章

回路最適化の数学

この章で学ぶこと

- Carleman 線形化を、一変数の例とテンソル積の構成の両方で導き、打ち切りと冗長性（対称基底）まで追う。
- Walsh 展開・Horner 計算・ANF が、それぞれ何を持ち上げ、何を持ち上げないのかを、小さな例の手計算で区別する。
- 位相オラクルの費用が「特徴量の数」と「算術の量」の取引で決まることを、公開 63-Q 版の数値で理解する。

本章では、**非線形な計算をどのような変数・基底で表し、どの順序で評価するか**という観点から、Carleman 線形化、Walsh 展開、Horner 法、ANF を比較する。まず Carleman 線形化で、単項式を新しい変数として導入し、非線形な時間発展を線形系に表す考え方を学ぶ。その上で、有限集合上の関数を基底展開する Walsh 展開、多項式の評価順序を変える Horner 法、ブール関数を GF(2) 上の多項式で表す ANF と、その共通点と違いを整理する。

比較するのは、各手法が何を表現・変換するのか、表現が有限で閉じるか、どこで近似や切り捨てが入るか、そして変数や演算の数がどう増減するかである。これにより、形式的な類似だけで手法を同一視せず、回路設計に役立つ考え方と、それぞれの適用条件を理解する。

今回の 3 課題について本書で扱う参照回路・公開回路では、Carleman 線形化と Walsh 展開は使用していない。この二つは、手法の比較を通じて数学的な見通しを得るために取り上げる。実際の invsqrt の公開回路では、係数表の読み出しに ANF・パリティ積の考え方を使い、区分多項式を Horner 法で評価して、その出力ビットを位相に変える。以下は、比較のために学ぶ手法と、対象回路で採用されている手法の役割を区別して読むとよい。

以下の公開 63-Q 版の詳細数値は、特記がなければ 2026 年 10 月 28 日取得の 41 回転版（費用 103,546.17）を指す。構成例の探索値は指定した係数幅・誤差条件の範囲での比較であり、全構成に対する最適性を意味しない。Walsh の打ち切りや係数探索、lookup の費用推定の表は、当時の解析記録から引き継いだ参考値である。探索した係数表や乱数 seed の全てを掲載しているわけではないため、本文だけで同一の探索結果を再現できるとは限らない。以下では計算方法と数理的な理由を示し、これらの値を最適性の証明として使わない。

2.1　出発点：位相オラクルは「線形結合」で書ける

invsqrt 課題で作る回路は、入力 x の計算基底状態に位相を掛ける対角ユニタリ

$$\boxed{D\,|x\rangle = \mathrm{e}^{2\pi\mathrm{i}\,\theta(x)}\,|x\rangle\,, \qquad \theta(x) = \frac{1}{\sqrt{x}} \pmod 1}$$

である（詳細は第 4 章）。位相は足し算で合成できる。すなわち $\theta = \theta_1 + \theta_2$ なら $\mathrm{e}^{2\pi\mathrm{i}\theta} = \mathrm{e}^{2\pi\mathrm{i}\theta_1}\mathrm{e}^{2\pi\mathrm{i}\theta_2}$ である。したがって $\theta(x)$ を何らかの「特徴量」$b_j(x)$ の線形結合

$$\boxed{\theta(x) = \sum_j \alpha_j\, b_j(x)}$$

に書ければ、各特徴量を量子ビットに用意して角度 α_j の回転を掛けるだけで位相が付く。Walsh 展開と ANF は関数を特徴量の線形結合で表す。一方、Horner 法は多項式の評価順序を変える方法である。Carleman 線形化は時間発展を特徴量の線形系に表す別の操作である。そこでまず Carleman 線形化を一通り導き、変数・基底の選択と評価方法という観点から、Walsh 展開・Horner 法・ANF との共通点と違いを比べる。

2.2　Carleman 線形化

2.2.1　基底の線形結合と、方程式の線形化は別の操作

「関数を基底関数の線形結合で表すこと」と「非線形な方程式を線形方程式に直すこと」は別の操作である。例えば $f(x) = x^2 + 3x + 2$ は $\varphi_0 = 1,\ \varphi_1 = x,\ \varphi_2 = x^2$ を使えば $f = 2\varphi_0 + 3\varphi_1 + \varphi_2$ と書ける。これは係数について線形な結合だが、基底 φ_2 自身は x の 2 次関数である。線形結合というとき、要請されるのは係数 a_j について足し算と定数倍で組み合わせることだけで、基底関数が一次であることではない。Walsh 展開は前者（基底展開）にあたり、Carleman 線形化は後者（時間発展の線形化）にあたる。特徴量を増やす点に形式的な類似はあるが、有限集合の基底展開と時間発展の線形化は異なる操作である。

Carleman 線形化が対象にするのは、2 次の非線形項をもつ常微分方程式

$$\boxed{\dot{\boldsymbol{x}} = F_1\boldsymbol{x} + F_2(\boldsymbol{x}\otimes\boldsymbol{x})}, \qquad \boldsymbol{x}\in\mathbb{R}^d,\ F_1\in\mathbb{R}^{d\times d},\ F_2\in\mathbb{R}^{d\times d^2} \tag{2.1}$$

である。偏微分方程式を Galerkin 法で有限個の係数に射影すると、右辺に係数の積が残って、この形になることが多い。右辺が未知数 $\boldsymbol{x}$ に線形かどうかは、関数の変数に関する形の複雑さとは別に判断しなければならない。

2.2.2　一変数の例：$y_k = s^k$ を新しい変数にする

最小の例として、スカラー $s(t)$ の方程式

$$\frac{\mathrm{d}s}{\mathrm{d}t} = -\gamma s + \beta s^2$$

を考える。$y_k = s^k$ と置き、積の微分を使うと

$$\dot{y}_k = k\, s^{k-1}\dot{s} = k\, s^{k-1}(-\gamma s + \beta s^2) = -k\gamma\, y_k + k\beta\, y_{k+1}$$

となる。$k = 1, 2, 3$ を書き下すと

$$\begin{aligned} \dot{y}_1 &= -\gamma\, y_1 + \beta\, y_2, \\ \dot{y}_2 &= -2\gamma\, y_2 + 2\beta\, y_3, \\ \dot{y}_3 &= -3\gamma\, y_3 + 3\beta\, y_4 \end{aligned}$$

である。各式は y について線形だが、$\dot{y}_k$ の式には必ず y_{k+1} が現れるので階層は閉じない。$\boldsymbol{y} = (y_1, y_2, y_3, \dots)^{\top}$ と並べれば、$\dot{\boldsymbol{y}} = A\boldsymbol{y}$ の A は無限次元の上二重対角行列

$$A = \begin{pmatrix} -\gamma & \beta & 0 & 0 & \cdots \\ 0 & -2\gamma & 2\beta & 0 & \cdots \\ 0 & 0 & -3\gamma & 3\beta & \cdots \\ 0 & 0 & 0 & -4\gamma & \ddots \\ \vdots & & & & \ddots \end{pmatrix} \tag{2.2}$$

になる。K 次で $y_{K+1} = 0$ と打ち切れば、$K \times K$ の有限行列が得られる。$K = 3$ なら

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} -\gamma & \beta & 0 \\ 0 & -2\gamma & 2\beta \\ 0 & 0 & -3\gamma \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

で、最後の行は $\dot{y}_3 = -3\gamma y_3$ と閉じている。$K = 1$ は 2 次項を捨てた線形系 $\dot{s} = -\gamma s$ そのものである。

表 2.1 は、$\gamma = 1,\ \beta = 0.5,\ s(0) = 1$ について $t = 1$ の y_1 を、厳密解と打ち切り次数ごとの解で比べたものである。この例の打ち切り解は、$s_0 = s(0)$ として次の有限等比和でも再計算できる：

$$y_1^{(K)}(t) = s_0 \mathrm{e}^{-\gamma t} \sum_{j=0}^{K-1} \left[\frac{\beta s_0}{\gamma}(1 - \mathrm{e}^{-\gamma t})\right]^j, \qquad \gamma \neq 0.$$

厳密解は $s(t) = \left[\beta/\gamma + (1/s_0 - \beta/\gamma)\mathrm{e}^{\gamma t}\right]^{-1} \approx 0.53788$ である。

この数値例では次数を 1 つ上げるごとに誤差がおよそ $1/3$ に減る。一般の収束は初期値、係数、対象時刻に依存し、この例の比だけから一律には判断できない。

表 2.1　一変数 Carleman 線形化の打ち切り誤差（$\gamma = 1, \beta = 0.5, s_0 = 1, t = 1$、厳密解 0.53788）。

打ち切り次数 K	$y_1(1)$	絶対誤差
1	0.36788	1.7×10^{-1}
2	0.48415	5.4×10^{-2}
3	0.52090	1.7×10^{-2}
4	0.53252	5.4×10^{-3}
6	0.53735	5.4×10^{-4}

2.2.3　テンソル積による一般の構成

d 変数の場合は、$\boldsymbol{y}_k = \boldsymbol{x}^{\otimes k}$（$d^k$ 次元のベクトル）を新しい変数に取る。積の微分（Leibniz 則）により、$\boldsymbol{x}^{\otimes k}$ の時間微分は「どれか 1 つの因子だけを微分する」項の和になる：

$$\dot{\boldsymbol{y}}_k = \sum_{r=0}^{k-1} \boldsymbol{x}^{\otimes r} \otimes \dot{\boldsymbol{x}} \otimes \boldsymbol{x}^{\otimes (k-1-r)}.$$

式 (2.1) を代入すると、$F_1\boldsymbol{x}$ の部分は $\boldsymbol{y}_k$ に戻り、$F_2(\boldsymbol{x}\otimes\boldsymbol{x})$ の部分は因子が 1 個増えて $\boldsymbol{y}_{k+1}$ に出る。結果は

$$\begin{aligned} \dot{\boldsymbol{y}}_k &= A_{k,k}\boldsymbol{y}_k + A_{k,k+1}\boldsymbol{y}_{k+1}, \\ A_{k,k} &= \sum_{r=0}^{k-1} I^{\otimes r} \otimes F_1 \otimes I^{\otimes (k-1-r)}, \qquad A_{k,k+1} = \sum_{r=0}^{k-1} I^{\otimes r} \otimes F_2 \otimes I^{\otimes (k-1-r)} \end{aligned} \tag{2.3}$$

である。$A_{k,k}$ は F_1 の Kronecker 和で $d^k \times d^k$、$A_{k,k+1}$ は $d^k \times d^{k+1}$ である。$d = 1$ なら $A_{k,k} = -k\gamma$、$A_{k,k+1} = k\beta$ となり、式 (2.2) に一致する。K 次で打ち切ると、ブロック上三角行列

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_K \end{pmatrix} = \begin{pmatrix} A_{1,1} & A_{1,2} & & \\ & A_{2,2} & A_{2,3} & \\ & & \ddots & \ddots \\ & & & A_{K,K} \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_K \end{pmatrix} \tag{2.4}$$

が得られる。初期値は $\boldsymbol{y}_k(0) = \boldsymbol{x}_0^{\otimes k}$ で、解は行列指数関数 e^{tA} で書け、第 1 ブロックが元の変数の近似である。捨てた $A_{K,K+1}\boldsymbol{y}_{K+1}$ が誤差の源で、隣り合うブロックの結合を通じて下位にも伝わる。

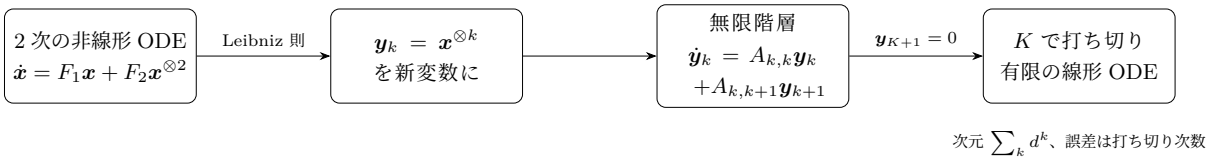


図 2.1　Carleman 線形化の流れ。非線形性は消えるのではなく、階層の次元に押し込まれる。

2.2.4　非対称テンソル基底の冗長性と対称基底

$\boldsymbol{x}\otimes\boldsymbol{x}$ は d^2 個の成分 x_ix_j を持つが、$x_ix_j = x_jx_i$ なので、異なる成分は $d(d+1)/2$ 個しかない。一般に $\boldsymbol{x}^{\otimes k}$ の独立な成分は、次数 k の単項式 $x^\alpha = x_1^{\alpha_1}\cdots x_d^{\alpha_d}$（多重指数 $|\alpha| = k$）の個数 $\binom{d+k-1}{k}$ である。したがって

$$\boxed{D_{\mathrm{full}} = \sum_{k=1}^{K} d^k, \qquad D_{\mathrm{sym}} = \sum_{k=1}^{K} \binom{d+k-1}{k} = \binom{d+K}{K} - 1} \tag{2.5}$$

である（後者の和は $\sum_{k=0}^{K}\binom{d+k-1}{k} = \binom{d+K}{K}$ という恒等式から $k=0$ の 1 を引いたもの）。表 2.2 に、いくつかの (d, K) での値を示す。d が大きいほど冗長性の比が急に大きくなる。

表 2.2　テンソル基底 D_{full} と対称単項式基底 D_{sym} の次元。

d	K	D_{full}	D_{sym}	比
2	2	6	5	1.2
2	3	14	9	1.6
3	3	39	19	2.1
4	4	340	69	4.9
6	5	9,330	461	20.2
8	5	37,448	1,286	29.1

2.2.5　小さな例：$d=2,\ K=2$ の 5×5 行列

対称基底で、$d=2, K=2$ を具体的に書く。変数を $\boldsymbol{y}=(x_1,\ x_2,\ x_1^2,\ x_1x_2,\ x_2^2)^{\top}$ とし、方程式を

$$\begin{aligned}\dot{x}_1 &= a\,x_1 + b\,x_2 + p\,x_1^2 + q\,x_1x_2 + r\,x_2^2,\\ \dot{x}_2 &= c\,x_1 + e\,x_2 + p'x_1^2 + q'x_1x_2 + r'x_2^2\end{aligned}$$

とする（$F_1=\begin{pmatrix}a & b\\ c & e\end{pmatrix}$ で、F_2 の成分は x_1x_2 と x_2x_1 を 1 つにまとめてある）。2 次の変数の微分は連鎖律で

$$\begin{aligned}\tfrac{\mathrm{d}}{\mathrm{d}t}x_1^2 &= 2x_1\dot{x}_1 = 2a\,x_1^2 + 2b\,x_1x_2 + (3\text{ 次以上}),\\ \tfrac{\mathrm{d}}{\mathrm{d}t}(x_1x_2) &= x_2\dot{x}_1 + x_1\dot{x}_2 = c\,x_1^2 + (a+e)\,x_1x_2 + b\,x_2^2 + (3\text{ 次以上}),\\ \tfrac{\mathrm{d}}{\mathrm{d}t}x_2^2 &= 2x_2\dot{x}_2 = 2c\,x_1x_2 + 2e\,x_2^2 + (3\text{ 次以上})\end{aligned}$$

となる。3 次以上は $y_3=0$ として捨てると、5×5 の行列

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}x_1\\ x_2\\ x_1^2\\ x_1x_2\\ x_2^2\end{pmatrix} = \left(\begin{array}{cc|ccc} a & b & p & q & r\\ c & e & p' & q' & r'\\ \hline 0 & 0 & 2a & 2b & 0\\ 0 & 0 & c & a+e & b\\ 0 & 0 & 0 & 2c & 2e\end{array}\right)\begin{pmatrix}x_1\\ x_2\\ x_1^2\\ x_1x_2\\ x_2^2\end{pmatrix} \tag{2.6}$$

が得られる。左上が $A_{1,1}=F_1$、右上が $A_{1,2}$（F_2 の同類項をまとめたもの）、右下が $A_{2,2}$ である。区画の次元は $2+3=5=D_{\mathrm{sym}}$ で、非対称テンソル基底なら $2+4=6=D_{\mathrm{full}}$ になる。この行列の各行は、ランダムな係数について単項式の微分を直接計算して確かめてある。

Note

Carleman 線形化の要点は三つある。(i) 非線形性を単項式の新変数に押し込む。(ii) その先は線形演算だけで扱える。(iii) 一般には階層が閉じず、打ち切り誤差と次元の取引になる。以下では、この変数と次元の選択を、有限基底を使う Walsh・ANF と、評価順序を変える Horner 法に対比する。

演習：Carleman 線形化

(1) $\dot{u} = u - u^2$ に対し $y_k = u^k$ の方程式を $k = 1, 2, 3$ について書き、$K = 3$ の行列を求めよ。(2) $d = 3,\ K = 4$ の D_{full} と D_{sym} を式 (2.5) で計算せよ（答え：120 と 34）。

2.2.6　Carleman 線形化と量子アルゴリズム

K 次で打ち切った行列を A_K とすると、その解は $\boldsymbol{y}_K(t) = \mathrm{e}^{A_K t}\boldsymbol{y}_K(0)$ である。A_K は一般に Hermitian でも反 Hermitian でもないため、$\mathrm{e}^{A_K t}$ は一般にユニタリではなく、そのまま量子ゲートとして適用できない。線形常微分方程式を解く量子アルゴリズムや、行列を大きなユニタリの一部として表すブロックエンコーディングを利用する方法が研究されている [17, 18]。

ただし打ち切りが有限次元の線形問題を与えることだけでは、量子優位は得られない。初期ベクトルに比例する量子状態の準備、行列へのアクセスと非ユニタリ演算の実装、解の増減や条件数に関わる安定性、Carleman 階層の打ち切り誤差を評価する必要がある。さらに量子状態として符号化された解から必要な物理量を測定する費用も含め、同じ精度と出力を要求する古典計算と比較する。Carleman 線形化は本書の 3 課題の公開回路で採用されていると主張するものではなく、特徴量による線形化の背景として紹介している。

2.3　Walsh 展開：ビット全体での厳密な持ち上げ

2.3.1　概観：Walsh 展開は何をしているのか

Walsh 展開は、ビット列を入力とする関数を、値が $+1$ と -1 だけの基底関数の線形結合で表す方法である。通常の Fourier 展開は sin・cos（複素指数関数）を基底に使い、関数を「周波数」の重ね合わせとして見る。Walsh 展開は基底に「どのビットの XOR（パリティ）を取るか」で決まる矩形波を使い、関数を「パリティ」の重ね合わせとして見る。短く言えば、古典的には ***XOR* を基底にした *Fourier* 展開**であり、量子的には **計算基底で対角な演算子を *Pauli-Z* の積に分解すること**である（§2.3.5）。ビット単位の条件分岐、パリティ、論理関数のように、2 進表現に沿った構造を持つ関数は Walsh 展開と相性がよい。ただし、少ない項で表せるかどうかは関数次第で、一般の関数には 2^n 項すべてが必要になる（§2.3.9）。以下ではまず定義と係数の計算法を述べ、次に位相回路への対応、項数と打ち切り誤差の順に見る。

表 2.3　Fourier 展開と Walsh 展開の対比。

	Fourier 展開	Walsh 展開
基底	$\sin, \cos$、複素指数関数	値が ± 1 の矩形波 χ_S
基底の番号の意味	周波数	どのビットのパリティを取るか（部分集合 S）
基底を作る演算	加算・乗算	XOR と符号反転
高速変換	FFT，$O(N \log N)$	高速 Walsh–Hadamard 変換（FWHT），$O(N \log N)$
量子演算との対応	位相・シフトなど	Pauli-Z の積 Z_S

2.3.2　ブール関数の Fourier 解析

n ビット入力 $x \in \{0,1\}^n$ と部分集合 $S \subseteq \{1,\dots,n\}$ に対し、パリティと指標（Walsh 関数）を

$$p_S(x) = \bigoplus_{i \in S} x_i, \qquad \chi_S(x) = (-1)^{p_S(x)} = (-1)^{S \cdot x}$$

と定める。指標は積について閉じる：$\chi_S \chi_T = \chi_{S \triangle T}$（指数の和が mod 2 で対称差に対応する）。さらに $U \neq \emptyset$ なら

$$\sum_{x} \chi_U(x) = \prod_{i=1}^{n} \sum_{x_i \in \{0,1\}} (-1)^{u_i x_i} = \prod_{i=1}^{n} \bigl(1 + (-1)^{u_i}\bigr) = 0$$

（$u_i = 1$ となる因子が 1 つでもあれば 0）、$U = \emptyset$ なら 2^n である。この二つから直交性

$$\sum_{x} \chi_S(x) \chi_T(x) = 2^n \, \delta_{S,T} \tag{2.7}$$

が従う。2^n 個の指標は 2^n 次元空間の直交基底をなすので、任意の実関数 θ は

$$\boxed{\theta(x) = \sum_{S} c_S \, \chi_S(x), \qquad c_S = \frac{1}{2^n} \sum_{x} \theta(x) \chi_S(x)} \tag{2.8}$$

と一意に展開できる。反転公式は、$\theta = \sum_T c_T \chi_T$ の両辺に χ_S を掛けて x で和を取り、式 (2.7) で $T = S$ の項だけが $2^n c_S$ として残ることから出る。行列で書けば、$H_{xS} = \chi_S(x)$ とすると $\boldsymbol{\theta} = H \boldsymbol{c}$、$H^{\top} H = 2^n I$、$\boldsymbol{c} = H \boldsymbol{\theta} / 2^n$ である。

2.3.3　高速 Walsh–Hadamard 変換と Kronecker 構造

H は $H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ の Kronecker 積で

$$H_{2^n} = H_2 \otimes H_{2^{n-1}}, \qquad H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

と再帰的に書ける（行 $x = x_1 + 2x_2$、列 $S = s_1 + 2s_2$ の順）。入力ベクトルを前半 $\boldsymbol{v}_0$ と後半 $\boldsymbol{v}_1$ に分けると

$$(H_2 \otimes H_{2^{n-1}}) \begin{pmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \end{pmatrix} = \begin{pmatrix} H_{2^{n-1}}(\boldsymbol{v}_0 + \boldsymbol{v}_1) \\ H_{2^{n-1}}(\boldsymbol{v}_0 - \boldsymbol{v}_1) \end{pmatrix}$$

なので、演算量は $T(n) = 2T(n-1) + 2^n$、すなわち $T(n) = O(n 2^n)$ で、2^{2n} の素朴な行列積より桁違いに小さい。22 ビットなら $N = 2^{22}$ 個の係数を対象に、$O(N \log N)$ の演算で変換できる。

2.3.4　位相係数と定数項

$\chi_S = 1 - 2 p_S$ を式 (2.8) に代入すると

$$\theta(x) = \sum_{S} c_S - 2 \sum_{S} c_S p_S(x) = \theta(0) + \sum_{S \neq \emptyset} b_S \, p_S(x), \qquad \boxed{b_S = -2 c_S}$$

となる（$\theta(0)=\sum_S c_S$ は $x=0$ で全パリティが 0 になることから）。各項 $b_S p_S(x)$ は、パリティ p_S を CX だけで 1 本の量子ビットに集めて回転を掛け、CX で戻せば実装できる。CX の費用は 0 なので、費用は回転の数だけで決まる。公式 Verifier の Z_POW(t) は $\mathrm{diag}(1,\mathrm{e}^{\pi \mathrm{i} t})$（半回転単位）なので、$|1\rangle$ に位相 b_S turn を付けるには $t=2b_S=-4c_S$ を渡す。

位相は mod 1 で測るので、定数項 $\theta(0)$ の扱いが二通りある。

- **anchored**：$\theta(0)$ を整数に取り、定数項を落とす。$x=0$ での位相はぴったり 0 になる。
- **global_offset**：定数位相を $X;\mathrm{Z_POW};X;\mathrm{Z_POW}$ の 2 回転で付ける（どちらの基底状態にも同じ位相が付く）。定数ずれを自由に調整できる。

全項を残せば両者は同じだが、項を捨てると両者は違う振る舞いをする（後述）。

2.3.5　量子演算との対応：Pauli-Z 積への展開

Walsh 指標は、Pauli-Z の固有値そのものである。$Z_i|x\rangle=(-1)^{x_i}|x\rangle$ なので、部分集合 S に対して $Z_S=\prod_{i\in S} Z_i$（$Z_\emptyset=I$）と置けば

$$Z_S|x\rangle=(-1)^{\sum_{i\in S}x_i}|x\rangle=\chi_S(x)\,|x\rangle$$

となる（指数は mod 2 で効くので χ_S と一致する）。したがって、計算基底で対角な演算子 $D=\sum_x \theta(x)|x\rangle\langle x|$ は、式 (2.8) の $\chi_S(x)$ を Z_S に置き換えるだけで

$$\boxed{D=\sum_S c_S\,Z_S}$$

と Pauli-Z 積の線形結合に書ける。c_S は Walsh 係数そのものである。位相オラクルは

$$U_\theta=\sum_x \mathrm{e}^{2\pi\mathrm{i}\theta(x)}|x\rangle\langle x|=\exp\Bigl(2\pi\mathrm{i}\sum_S c_S Z_S\Bigr)=\prod_S \exp\bigl(2\pi\mathrm{i}\,c_S Z_S\bigr)$$

と分解できる。Z_S はすべて対角で互いに可換なので、最後の等号は Trotter 分解のような近似ではなく厳密に成り立ち、積の順序も自由である。$Z_S=I-2P_S$（P_S は $p_S(x)$ を対角に並べた射影）を代入すると、各因子は大域位相 $\mathrm{e}^{2\pi\mathrm{i}c_S}$ を除いて $\mathrm{e}^{2\pi\mathrm{i}b_S P_S}$（$b_S=-2c_S$）になり、前節の「CX でパリティを集めて回転し、CX で戻す」回路に一致する。Hamiltonian の言葉で言えば、対角 Hamiltonian $H=\sum_x f(x)|x\rangle\langle x|$ を $H=\sum_S \hat f(S)Z_S$ と分解し、$\mathrm{e}^{-\mathrm{i}tH}=\prod_S \mathrm{e}^{-\mathrm{i}t\hat f(S)Z_S}$ と実装することと同じである。

■例：AND 条件と CZ。　$f(x_1,x_2)=x_1x_2$ は $x=3$ でだけ 1 になる。$x_i=(1-\chi_{\{i\}})/2$ を掛け合わせると

$$x_1x_2=\frac{1}{4}\bigl(1-\chi_{\{1\}}-\chi_{\{2\}}+\chi_{\{1,2\}}\bigr),\qquad |11\rangle\langle 11|=\frac{1}{4}\bigl(I-Z_1-Z_2+Z_1Z_2\bigr)$$

で、Walsh 係数は $(c_\emptyset,c_{\{1\}},c_{\{2\}},c_{\{1,2\}})=(\frac14,-\frac14,-\frac14,\frac14)$ である。位相を $\theta=\frac12 x_1x_2$ turn とすると係数は半分になり、$b_S=-2c_S$ は $b_{\{1\}}=b_{\{2\}}=\frac14$、$b_{\{1,2\}}=-\frac14$ である。Z_POW の引数 $2b_S$ で言えば、x_1 と x_2 にそれぞれ Z_POW($\frac12$)（S ゲート）、$p_{\{1,2\}}$ に Z_POW($-\frac12$)（$S^\dagger$）を掛けることになる。$x=3$ では $\frac14+\frac14+0=\frac12$ turn、$x=1,2$ では $\frac14-\frac14=0$ となり、$U_\theta=\mathrm{diag}(1,1,1,-1)=\mathrm{CZ}$

が得られる。条件付き位相を、CX と Z 軸回転の組み合わせとして組み立てられることがわかる。一方で、この 2 ビットの AND は 4 項すべてを必要とし、n ビットの AND $x_1 \cdots x_n$ も 2^n 項すべてを持つ。論理的に単純な関数でも、Walsh 展開では密になりうる。

■CX の並べ方。 各項の「CX で集めて戻す」をそのまま並べると、隣り合う項の CX が重複する。パリティ S を Gray 符号の順に並べると、隣り合う項の S は 1 ビットしか違わないので、戻す CX と次に集める CX の多くが打ち消し合う [14]。CX と位相回転だけからなる回路の CX 数の最適化は、[15] でも扱われている。本課題の Verifier では CX の費用が 0 なので、この工夫はスコアには直接効かず、ゲート数と深さを減らす効果にとどまる。

2.3.6　手計算の例：$n = 2$

$x = x_1 + 2x_2$ とし、位相の表を $\theta = (\theta(0), \theta(1), \theta(2), \theta(3)) = (0, \frac{1}{8}, \frac{1}{4}, \frac{1}{2})$ とする。係数は $\boldsymbol{c} = H_4\boldsymbol{\theta}/4$ である。FWHT の 2 段階で計算すると、x_1 の段（隣同士の和と差）は

$$(0 + \tfrac{1}{8},\ 0 - \tfrac{1}{8},\ \tfrac{1}{4} + \tfrac{1}{2},\ \tfrac{1}{4} - \tfrac{1}{2}) = (\tfrac{1}{8}, -\tfrac{1}{8}, \tfrac{3}{4}, -\tfrac{1}{4}),$$

x_2 の段（半分ずつの和と差）は

$$(\tfrac{1}{8} + \tfrac{3}{4},\ -\tfrac{1}{8} - \tfrac{1}{4},\ \tfrac{1}{8} - \tfrac{3}{4},\ -\tfrac{1}{8} + \tfrac{1}{4}) = (\tfrac{7}{8}, -\tfrac{3}{8}, -\tfrac{5}{8}, \tfrac{1}{8})$$

で、4 で割って

$$c_\emptyset = \tfrac{7}{32}, \quad c_{\{1\}} = -\tfrac{3}{32}, \quad c_{\{2\}} = -\tfrac{5}{32}, \quad c_{\{1,2\}} = \tfrac{1}{32}$$

を得る。和は $\sum_S c_S = 0 = \theta(0)$ になっている。位相係数 $b_S = -2c_S$ と Z_POW の引数 $-4c_S$ は表 2.4 のとおりである。

表 2.4　$n = 2$ の例（$\theta = (0, \frac{1}{8}, \frac{1}{4}, \frac{1}{2})$）の各項と回路。

S	c_S	$b_S = -2c_S$	Z_POW の引数 $-4c_S$	回路
$\{1\}$	$-3/32$	$3/16$	$3/8$	x_1 に直接
$\{2\}$	$-5/32$	$5/16$	$5/8$	x_2 に直接
$\{1,2\}$	$1/32$	$-1/16$	$-1/8$	CX($x_1 \to x_2$)、回転、CX

$x = 3$ で確かめると、$p_{\{1\}} = p_{\{2\}} = 1$、$p_{\{1,2\}} = 0$ なので位相は $\frac{3}{16} + \frac{5}{16} = \frac{1}{2} = \theta(3) - \theta(0)$ になる。4 個の入力すべてで、3 本の回転の積が $\mathrm{e}^{2\pi\mathrm{i}(\theta(x)-\theta(0))}$ に一致することを数値で確かめた。回路は図 2.3 の形で、$S = \{1, 2\}$ の項が「CX で p_S を集める→回転→ CX で戻す」にあたる。

2.3.7　行列イメージ

$n = 3$ の場合に、同じことを行列で見る。行を入力 x、列を部分集合 S で並べた 8×8 の Sylvester–Hadamard 行列 H（成分 $\chi_S(x) = (-1)^{x \cdot S}$）に係数ベクトル $\boldsymbol{c}$ を掛けると、位相の表 $\boldsymbol{\theta}$ が得られる（図 2.2(a)）。x, S は 3 ビットの 2 進数で、最下位ビットが x_1（S では要素 1）に対応する。H は全成分が ± 1 の密な行列である。$\chi_S = 1 - 2p_S$ を使うと、各列を 0/1 のパリティ行列 $P_{xS} = p_S(x)$ に置き換えた形 $\theta(x) - \theta(0) = \sum_{S \neq \emptyset} (-2c_S)\, p_S(x)$ になる（図 2.2(b)）。回路が実際に実装するのはこちらである。

top-k 切り捨ては、この行列から k 本の列だけを残す操作にあたる。図 2.2 では、残した 3 本の列を青、捨てた列を灰色で示した。残した列 1 本が回転 1 個に対応するので、費用は列の本数 k で決まる。

(a) 位相の表 $\theta = Hc$（± 1 行列）

$x\backslash S$	000	001	010	011	100	101	110	111		c		θ
000	+	+	+	+	+	+	+	+		c_{000}		$\theta(000)$
001	+	−	+	−	+	−	+	−		c_{001}		$\theta(001)$
010	+	+	−	−	+	+	−	−		c_{010}		$\theta(010)$
011	+	−	−	+	+	−	−	+		c_{011}		$\theta(011)$
100	+	+	+	+	−	−	−	−	×	c_{100}	=	$\theta(100)$
101	+	−	+	−	−	+	−	+		c_{101}		$\theta(101)$
110	+	+	−	−	−	−	+	+		c_{110}		$\theta(110)$
111	+	−	−	+	−	+	+	−		c_{111}		$\theta(111)$

(b) 0/1 パリティ行列による形（$S = \emptyset$ の列は定数項 $\theta(0)$ に吸収される）

$x\backslash S$	001	010	011	100	101	110	111			
000	0	0	0	0	0	0	0		$-2c_{001}$	$\theta(000) - \theta(0)$
001	1	0	1	0	1	0	1		$-2c_{010}$	$\theta(001) - \theta(0)$
010	0	1	1	0	0	1	1		$-2c_{011}$	$\theta(010) - \theta(0)$
011	1	1	0	0	1	1	0		$-2c_{100}$	$\theta(011) - \theta(0)$
100	0	0	0	1	1	1	1	×	$-2c_{101}$	= $\theta(100) - \theta(0)$
101	1	0	1	1	0	1	0		$-2c_{110}$	$\theta(101) - \theta(0)$
110	0	1	1	1	1	0	0		$-2c_{111}$	$\theta(110) - \theta(0)$
111	1	1	0	1	0	0	1			$\theta(111) - \theta(0)$

図 2.2　$n = 3$ の Walsh 展開の行列表示。青の列（$S = 001, 100, 111$）が top-k（$k = 3$）で残す列で、灰色は捨てる列である。残した列 1 本が回転 1 個になる。選び方は例示であり、実際の $|c_S|$ の順位ではない。

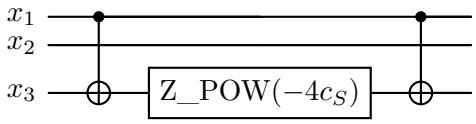


図 2.3　$S = \{1, 3\}$ の 1 項分の回路。最初の CX で x_3 の線が $p_S = x_1 \oplus x_3$ になり、回転で位相 $\mathrm{e}^{2\pi \mathrm{i}(-2c_S)p_S}$ が付き、最後の CX で x_3 を元に戻す。ここで回転は位相 $-2c_S$ turn を付ける操作を指し、Verifier の Z_POW の引数は半回転単位でその 2 倍の $-4c_S$ になる。CX は費用 0 なので、回転 1 個分だけが費用になる。

2.3.8　Carleman との違い

- **展開は有限で厳密である。** $x_i^2 = x_i$ なので、特徴量は高々 2^n 個で閉じる。Carleman の閉包問題は起きない。
- **誤差の出どころが違う。** 打ち切り次数ではなく、どの係数を捨てるか（台の選択）で誤差が決

まる。

- **費用は特徴量と回転の種類に依存する。**一般角回転 1 個は約 7.2 Toffoli 相当（公開 63-Q 版では回転 295 Toffoli を 41 回転で割った値、下の閾値の逆算では約 7.27）なので、公開 63-Q 版と競うには 22 量子ビットで 647 個未満、44 量子ビットでも 329 個未満に収める必要がある。

この閾値は、公開 63-Q 版のスコア $Q \times C = 103{,}546$ を超えないための上限である。$Q = 22$ なら $C < 103{,}546/22 \approx 4{,}706.6$、$Q = 44$ なら $C < 2{,}353.3$ で、Toffoli 0、近似誤差 0 という理想条件で $4{,}706.6/647 \approx 7.27$ Toffoli である。実際には近似誤差があれば 1 回転あたりの費用が上がるので、この個数は上限である。

2.3.9　項数の疎性と打ち切り誤差

Walsh 展開そのものに高速化の効果があるわけではない。n ビットの一般の関数は 2^n 個の係数を持ち、回路の回転数はそのまま非零係数の数になる。少ない項で済むのは、関数が構造を持つときに限られる。

実数係数の多重線形多項式で次数が k の関数を考える。単項式 $\prod_{i\in T} x_i$ は $x_i = (1 - \chi_{\{i\}})/2$ を代入すると T の部分集合 U の指標 χ_U だけで書けるので、Walsh 係数は $|S| \le k$ の S にしか現れない。項数の上限は表 2.5 のとおりである。例えば 1 次式なら $x_i = (1 - \chi_{\{i\}})/2$ か

表 2.5　関数の形と Walsh 項数の上限（実数上の多項式次数による分類）。

関数	Walsh 項数の上限
定数	1
1 次式 $\sum_i a_i x_i + a_0$	$n + 1$
2 次式	$1 + n + \binom{n}{2} = O(n^2)$
k 次式（k 固定）	$\sum_{j=0}^{k} \binom{n}{j} = O(n^k)$
一般の関数	2^n

ら $\theta = a_0 + \frac{1}{2}\sum_i a_i - \frac{1}{2}\sum_i a_i \chi_{\{i\}}$ で、2^n 個の入力上の関数でも回転は n 個で済む（演習の $\theta = x_1/4 + x_2/4$ がこの場合で、CX が要らない）。ここでの次数は実数上の多項式としての次数であり、ANF（GF(2) 上の次数、§2.5）とは別物である。パリティ $x_1 \oplus \cdots \oplus x_n$ は Walsh では $\chi_{[n]}$ を含む 2 項で済むが、実数上の次数は n になる。逆に AND $x_1 \cdots x_n$ は ANF では 1 項だが、Walsh では 2^n 項になる。

小さい係数を捨てて近似することもできる。残す集合を K とし、定数項も含めて $\theta_K = \sum_{S\in K} c_S \chi_S$ と再構成すれば、$|\chi_S(x)| = 1$ から各入力で

$$|\theta(x) - \theta_K(x)| \le \sum_{S \notin K} |c_S|$$

が成り立つ。位相オラクルの作用素ノルムで言えば $\|U_\theta - U_{\theta_K}\| \le 2\pi \sum_{S\notin K} |c_S|$ である。右辺は捨てた係数の絶対値の和なので、係数が少数の項に集中していれば項数を大きく減らせる。逆に、係数が多くの項に薄く広がっていれば、この上界は小さくならない。次の 22 ビットの例はその場合にあたる。

2.3.10　計算例：22 ビットでの top-k 打ち切り

$|c_S|$ の大きい順に k 項を残す実験を、22 ビット（有効入力 $x = 1, \ldots, 3{,}133{,}452$）で考える。表 2.6 は、誤差比 $\max_x e(x)\sqrt{x}\,2^{24}$（許容誤差 $2^{-24}/\sqrt{x}$ turn に対する最大誤差の比。1 以下で合格）である。不正入力は逆平方根の延長で埋め、$\theta(0) = 0$ に取った設定（anchor 0、continued）を示す。

表 2.6　22 ビット Walsh 展開の top-k の誤差比（anchor 0、continued。設定についての計算例）。

残した項数 k	anchored	global_offset
0	1.68×10^{7}	1.67×10^{7}
256	4.63×10^{8}	1.65×10^{7}
4,096	1.81×10^{9}	1.59×10^{7}
65,536	6.72×10^{9}	1.32×10^{7}
262,144	1.19×10^{10}	9.25×10^{6}
1,048,576	1.08×10^{10}	1.17×10^{7}
4,194,303（全項）	1.39×10^{-6}	1.19×10^{-6}

$k = 10^6$ でも誤差比は 10^7 以上に残り、全項（$k = 2^{22} - 1$）でようやく 10^{-6} 程度で通る。anchored では、多くの打ち切り点で項数を増やしても改善せず、誤差比が非常に大きいまま残る。表では $k = 262{,}144$ から 1,048,576 に増やすとわずかに改善しており、項数に対して単調に悪化するわけではない。その機構は次のとおりである。残す集合を K（$\emptyset \in K$）とし、捨てた部分を $r = \theta - \theta_K = \sum_{S \notin K} c_S \chi_S$ とする。$\theta(0) = 0$ とすれば $r(0) = -\theta_K(0)$ で、anchored の再構成（$x = 0$ で位相 0）の誤差は

$$e(x) = r(0) - r(x) = 2 \sum_{S \notin K} c_S \, p_S(x) \tag{2.9}$$

である。どの $S \neq \emptyset$ でも $p_S(x)$ は入力の半分で 1 になるので、e の平均は $\sum_{S \notin K} c_S = -\theta_K(0)$ に近い。上位 $|c_S|$ は $x = 0, 1$ 付近の段差に支配され、$k \leq 262{,}144$ では選んだ係数がすべて正だった。すると $\theta_K(0)$ は k とともに単調に増え（$k = 256$ で約 0.016 turn、$k = 262{,}144$ で約 0.40 turn）、$x \geq 1$ の全体がほぼ一様にずれる。誤差比は円周誤差が最大の 0.5 turn で頭打ちになる。$x = 3{,}133{,}452$ では $0.5 \times \sqrt{3{,}133{,}452} \times 2^{24} \approx 1.48 \times 10^{10}$ で、表の 1.19×10^{10} はその約 80% である。つまり anchored の比は大きい x のずれで決まり、候補同士の比較にはほとんど使えない。一様なずれを除く global_offset でも、捨てた残差の二乗平均平方根は 4×10^{-4}〜1.4×10^{-3} turn で、許容誤差（$x = 3{,}133{,}452$ で約 3.4×10^{-11} turn）には遠く、比は全 k で 10^7 付近に残る。$1/\sqrt{x}$ には $x = 0$ 付近の急峻な変化があり、top-k の台は小さい x の段差の表現に項を使い切っていると考えられる（これは未検証の仮説である）。これは、Carleman 線形化を局所化せずに行うと次元が爆発するのと同じ構図である。

注意

この失敗は「固定した拡張と整数リフトでの上位 k 項」に限った結果である。不正入力 1,060,851 個（2^{22} の約 25%）を don't-care として使う自由、入力ごとの整数リフト $n(x)$（位相は mod 1 なので $\theta(x) + n(x)$ を自由に選べる）、上位 $|c_S|$ 以外の台の選び方は検証してい

ない。「不可能が示された」ではなく「検証した top-k では届かない」と読むべきである。

演習：Walsh 展開

$n=2$ で $\theta=(0,\frac{1}{4},\frac{1}{4},\frac{1}{2})$ のとき、c_S を求め、必要な回転の数を答えよ（答え：$c_\emptyset=\frac{1}{4}$, $c_{\{1\}}=c_{\{2\}}=-\frac{1}{8}$, $c_{\{1,2\}}=0$ で、回転は 2 個。$\theta=x_1/4+x_2/4$ は加法的なので CX が要らない）。

2.4　Horner 計算：持ち上げない、因数分解する

2.4.1　概観：Horner 法は何をしているのか

Horner 法は、多項式の**表し方**ではなく**評価の順序**を変える方法である。Walsh 展開や ANF が「どの基底で関数を書くか」を選ぶのに対し、Horner 法は基底 $1, x, x^2, \dots$ をそのまま使い、括弧の付け方だけを変えて冪 x^k を明示的に作らずに済ませる。

n 次多項式 $f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ は

$$\boxed{f(x)=(\cdots((a_nx+a_{n-1})x+a_{n-2})\cdots)x+a_0}$$

と括り直せる。計算は漸化式

$$b_n=a_n,\qquad b_k=b_{k+1}\,x+a_k\quad(k=n-1,\dots,0),\qquad f(x)=b_0$$

で、各段は「掛けて足す」1 回である。

例えば $f(x)=2x^3-3x^2+4x-5$ は $f(x)=((2x-3)x+4)x-5$ となり、$x=2$ では

$$b_3=2,\quad b_2=2\cdot2-3=1,\quad b_1=1\cdot2+4=6,\quad b_0=6\cdot2-5=7$$

と、掛け算 3 回、加減算 3 回で $f(2)=7$ を得る（$16-12+8-5=7$ と一致する）。古典計算では中間値 b_{k+1} を b_k で上書きできるので、作業領域は $O(1)$ で済む。量子回路では上書きが可逆でないため、中間値を保持するか後で逆計算する必要があり、この利点はそのままでは得られない（§2.4.7）。

2.4.2　多項式評価の掛け算の数

m 次の多項式 $y=C_0+C_1d+\cdots+C_md^m$ を、冪を順に作って評価すると、$d^2,\dots,d^m$ に $m-1$ 回、各 C_jd^j に m 回で、掛け算は $2m-1$ 回になる。これを $y=C_0+d(C_1+d(C_2+\cdots+dC_m))$ と括り直すのが Horner 法で、掛け算は m 回で済む。一般の係数ではこれより少なくならないことが知られている。

公開 63-Q 版は区間ごとの 2 次近似 $y\approx C_0+C_1d+C_2d^2$ を

$$\boxed{y=C_0+d\,(C_1+d\,C_2)}$$

と括り直し、$t_1=C_2d$、$t_2=(t_1+C_1)d$、$y=t_2+C_0$ の順に計算する。Horner 法は単項式を新変数に**しない**。むしろ逆向きの操作で、d^2 という単項式を作らずに済むよう式を因数分解し、中間値 t_1 を再利用する。

表 2.7　m 次多項式の掛け算の回数。

次数 m	冪を作る方法	Horner 法
1	1	1
2	3	2
3	5	3
4	7	4

2.4.3　固定小数点の厳密なパイプライン

量子回路では実数ではなく整数の演算を行うので、各段で下位ビットを切り捨てる。計算例での整数 Horner は、整数 C_0, C_1, C_2, d と切り捨て幅 k_1, k_2 に対し

$$\boxed{t_1 = \left\lfloor \frac{C_2 d}{2^{k_1}} \right\rfloor, \quad t_2 = \left\lfloor \frac{(t_1 + C_1)\, d}{2^{k_2}} \right\rfloor, \quad v = t_2 + C_0, \qquad y = \frac{v}{2^F}} \tag{2.10}$$

で定義される（右シフトは切り捨て除算 $\lfloor \cdot \rfloor$ に等しい）。切り捨てを無視すれば $v \approx C_0 + C_1 d/2^{k_2} + C_2 d^2/2^{k_1+k_2}$ なので、実数の係数 a_1, a_2 に対する整数係数の格子は、C_1 が $2^{-(k_2+F)}$、C_2 が $2^{-(k_1+k_2+F)}$ である。公開 63-Q 版（A=8、256 区間）を模したモデルのパラメータは、小数部 $F = 25$、$k_1 = 12$、$k_2 = 13$、$C_0/C_1/C_2$ の幅が 26/18/10 ビット、d が 14 ビット、C_2 が 11 ビット、$t_1 + C_1$ が 19 ビットである。ここでの F, k_1, k_2 は本書のモデル値であり、公開ノートブックの値とは別に確認していない。区間のアドレス a（$A = 8$ ビット、先頭の 1 の直後の $A - 1$ ビットとオクターブ 1 ビット）が C_0, C_1, C_2 を選び、d は区間内の位置を表す。

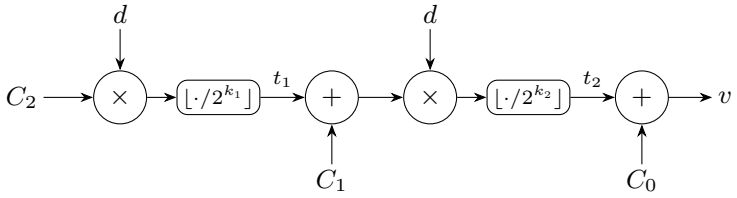


図 2.4　整数 Horner のパイプライン。各掛け算の直後に下位ビットを切り捨てる。幅は $C_2 d$ が $11 + 14$ ビット、$(t_1 + C_1)d$ が $19 + 14$ ビットの積である（A=8 のモデル値）。

2.4.4　誤差予算の導出

全体の誤差は、近似誤差、係数の丸め、切り捨ての三つの和である。

■切り捨て。 式 (2.10) の各段は $0 \le x/2^k - \lfloor x/2^k \rfloor < 1$ を満たす。t_1 の切り捨て誤差を $\epsilon_1 \in [0, 1)$ とすると、第 2 段では $(t_1 + C_1)d = (u_1 - \epsilon_1 + C_1)d$（$u_1 = C_2 d/2^{k_1}$）となるので、$\epsilon_1$ は $\epsilon_1 d/2^{k_2}$ に拡大される。$d < 2^{14}$ かつ $k_2 = 13$ なので、この拡大は最大でも $2^{14-13} = 2$ である。第 2 段の切り捨ては高々 1 なので、v の切り捨て誤差は片側で 2^{-F}（1 ulp）の 3 倍未満である。

■係数の丸め。 C_0 は丸めで最大 0.5 ulp、C_1 は $0.5\, d/2^{k_2} < 1$ ulp、C_2 は t_1 に $0.5\, d/2^{k_1} < 2$ を与え、第 2 段で再び $d/2^{k_2} < 2$ 倍されて < 4 ulp である。したがって最悪ケースの丸めは < 5.5

ulp、切り捨てと合わせて < 8.5 ulp、つまり $8.5 \times 2^{-25} \approx 2.5 \times 10^{-7}$ である。ただし、この単純な上界は許容誤差を超えている。許容誤差は $2^{-24}y$（$y \in (1,2]$）で、$F = 25$ では 2〜4 ulp しかない。したがって、各項の誤差を独立に最悪へそろえる評価では合格を示せない。実際のモデルでは、d を区間の中心からの符号付きの値（$|d| \le 2^{13}$）にとって拡大を抑え、区間ごとに C_0 の偏りを中心へ寄せ、C_1, C_2 を局所探索（± 2）で選ぶことで、誤差どうしを打ち消し合わせている。その結果の最大誤差比（$\rho_{\max} = 0.873$、$A = 8$）は、上界ではなく全入力での評価で確かめた値である。誤差が上界の議論では抑えられず、全入力の評価が必要になるのはこのためである。

■近似誤差。 2 次式で区間幅 h の関数を近似する誤差は、3 階微分に比例して h^3 で減る。アドレスを 1 ビット増やすと区間幅が半分になり、誤差は 1/8 になる。公開の分析では、丸めなしの実数フィットの最大誤差比 $\rho_{\max}$ が $A = 7, 8, 9$ で $0.61,\ 0.078,\ 0.0097$ と、ほぼ 1/8 ずつ減る。アドレスを上位 A ビットで区切る方式では下のオクターブの区間が相対的に 2 倍の幅になり、$A = 8$ で 0.61 と約 8 倍悪くなる。公開 63-Q 版と同じ「オクターブ 1 ビット＋先頭の 1 の直後 $A-1$ ビット」の方式では、$A = 8$ で 0.078 である。

公開ノートブックの分析では、誤差予算の 60〜97% が切り捨てと係数の丸めに使われ、近似そのものの誤差は 1〜4% にすぎない。$A = 8$ のモデル値 $\rho = 0.078$（約 8%）は、この「1〜4%」と同じ桁である。誤差が丸めで決まるので、係数の幅（列数）を減らせるかどうかが費用を決める。$\rho_{\max} \le 0.9$ を満たす、ここで比較した構成を表 2.8 に示す。

表 2.8　$\rho_{\max} \le 0.9$ を満たす比較構成（数値例）。列は符号と絶対値を数えた係数ビット数。

A	区間数	$C_0/C_1/C_2$ 幅	列	$d/C_2/(t_1+C_1)$ 幅	$\rho_{\max}$
7	128	27/20/12	59	15/13/21	0.897
8	256	26/18/10	54	14/11/19	0.873
9	512	26/17/8	51	13/9/18	0.897
10	1024	26/17/5	48	12/6/18	0.883

A を増やすと係数の幅と列が減る一方で、lookup の表は 2^A エントリとなり、A を 1 増やすと 2 倍になる。この取引の下で公開 63-Q 版は $A = 8$ を採用する（第 4 章）。

■参考：浮動小数点での誤差評価。 浮動小数点演算では、単位丸め誤差を u、$\gamma_m = mu/(1-mu)$（$2nu < 1$ を仮定）とすると、Horner 法の計算値 $\tilde{p}(x)$ について

$$|\tilde{p}(x) - p(x)| \le \gamma_{2n} \sum_{k=0}^{n} |a_k|\,|x|^k$$

という標準的な上界が知られている。右辺は $|a_k||x|^k$ の和なので、正負の項が大きく打ち消し合って $|p(x)|$ が小さい場合には、相対誤差が大きくなりうる。本書の回路は浮動小数点ではなく固定小数点で、各段の誤差は上で導いたとおり切り捨てと係数の丸めで決まる。この上界はそのままでは使えないが、「各段の誤差が後の掛け算で拡大される」という構造は同じである。

2.4.5　行列イメージ：部分積の格子

Horner 法は、行ベクトルと列ベクトルの内積 $y = [1\ d\ d^2]\,[C_0\ C_1\ C_2]^{\top}$ を、d の冪を作らない入れ子の形に括り直したものと見なせる。左の形では d^2 を作ってから C_2 に掛けるので掛け算が 3 回

要る。右の形は、図 2.5 のように「掛ける、足す、掛ける、足す」の 2 回の掛け算で済む。

(a) Vandermonde 型の内積

$$y = \begin{pmatrix} 1 & d & d^2 \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix}$$

(b) 入れ子の評価 $y = C_0 + d(C_1 + dC_2)$ のデータフロー

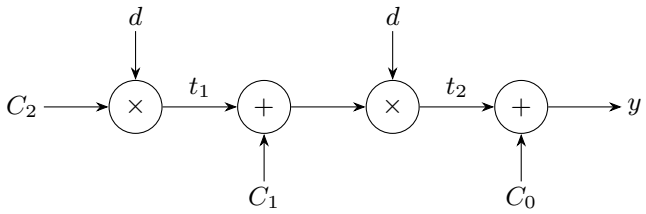
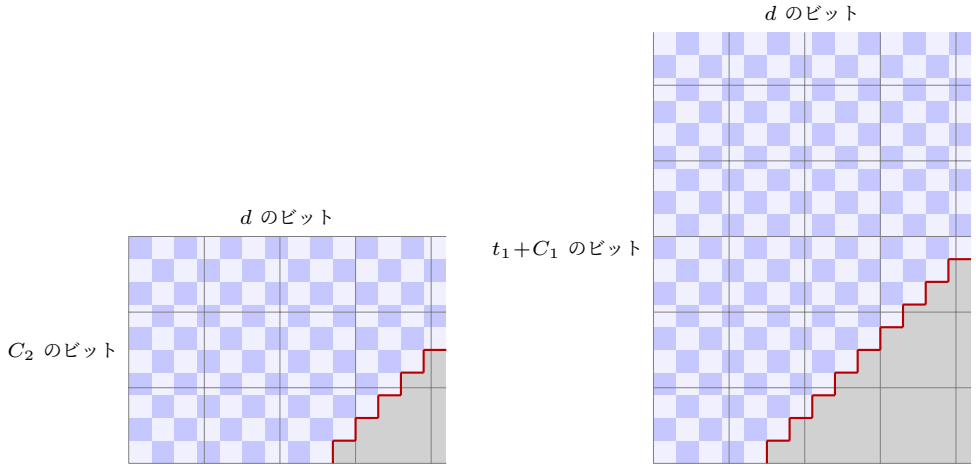


図 2.5　Horner 法のデータフロー。d の冪を作らず、各段の中間値 $t_1 = C_2 d$、$t_2 = (t_1 + C_1)d$ を次の段に渡す。

算術の費用は掛け算の部分積ビットの数でほぼ決まる。可逆回路では、掛け算を「被乗数のビットが 1 なら乗数を加算する」行を重ねた制御付き加算の列として実装する。部分積ビット 1 個は 2 ビットの AND、すなわち Toffoli 1 個であり、加算器（Gidney 型で n ビットあたり約 n 個の Toffoli）の費用が加わる。図 2.6 は、掛け算を「被乗数のビット × 乗数のビット」の格子として描いたもので、同じ重み 2^{i+j} をもつ部分積は反斜め線上に並び、列ごとにまとめて足される。公開 63-Q 版の A=8 のモデルでは C_2 が約 10〜11 ビット、d が約 14 ビット、$t_1 + C_1$ が約 19 ビットで、1 回目の格子は $10 \times 14 = 140$ ビット、2 回目は $19 \times 14 \approx 270$ ビットである（図の 10 ビットは例示）。モデルの部分積ビット数は $(C_2\text{幅} + (t_1{+}C_1)\text{幅}) \times d\text{幅} = (11 + 19) \times 14 = 420$ で、計算と逆計算を合わせた Horner の 749 Toffoli を $749/420 \approx 1.78$ Toffoli ／ビットで較正している。



(a) $t_1 = C_2 d$ の格子（10×14）　　(b) $t_2 = (t_1 + C_1)d$ の格子（19×14）

図 2.6　Horner の部分積の格子（模式図）。各マスが部分積ビット 1 個、濃淡の縞が同じ重みの反斜め線である。灰色のマスが切り捨てで落ちる部分、赤線が切り捨ての境界で、(a) では $i+j < 5$、(b) では $i + j < 9$ を例示した（値は例示）。

赤線より右下の灰色は、結果の下位 k_1 ビット（2 回目では k_2 ビット）を切り捨てることで不要

になる低重みの部分積を表す。切り捨てを $t_1 = C_2 d$ の直後に入れるので、2 回目の格子の被乗数は $C_2 d$ の全幅ではなく上位だけになる。$C_2 d^2$ を単項式 $C_{2,k} d_i d_j$ にすべて展開すると、重みの下限を誤差予算に合わせても部分積が約 500 個残る。Horner 法ではこれが約 140 個の格子に収まる。この差が、展開が算術より高くつく理由である。

■3 ビット ×3 ビットの例。 $C = 7$（111_2）と $d = 7$（111_2）の積 49 を、重みで切り捨てて 2^3 で割る（$\lfloor 49/8 \rfloor = 6$）。部分積は 9 個（$i, j \in \{0, 1, 2\}$、重み 2^{i+j}）である。重み 2^{i+j} が 2^g 未満（$i + j < g$）の部分積を捨てた結果を表 2.9 に示す。

表 2.9　7×7 の部分積の切り捨て（厳密値 $\lfloor 49/8 \rfloor = 6$）。

捨てる境界 g	残す部分積の数	捨てた重みの合計	$\lfloor$残りの和$/8\rfloor$	誤差
0	9	0	6	0
2	6	5	5	-1
3	3	17	4	-2

捨てた重みの合計は一般に $\sum_{w<g}(w+1)2^w = (g-1)2^g + 1$ なので、2^k で割った結果への誤差は最大で約 $g - 1$ 単位に達する。Toffoli を減らしたいなら境界を上げるほどよいが、誤差が増える。実際の回路は余分な下位ビット（ガードビット）を残して境界を決めるので、結果の誤差予算と部分積の数の取引になる。

2.4.6　なぜ単項式に展開しないのか

2 次式を部分積 $C_{1,k} d_j$ と $C_{2,k} d_i d_j$ にすべて展開して位相に直す案がある。重みごとに Wallace 木と Hamming weight phasing を使う。A=8（C_1 18 ビット、C_2 10 ビット、d 14 ビット）で、重みが 2^{-L} 未満の部分積を捨てた場合の数を表 2.10 に示す。

表 2.10　2 次式を単項式（部分積）に展開した場合の Toffoli 数（A=8、計算例）。

L	$C_1 d$	$C_2 d^2$	AND	キャリー	Toffoli	捨てた重みの合計$/2^{-24}$
29	207	349	611	≈534	≈1,145	2.06
30	216	406	683	≈599	≈1,282	1.03
31	224	465	755	≈665	≈1,420	0.51

誤差予算に収まる $L = 31$ 前後で約 1,400 Toffoli となり、現行の 749 より悪い。$L = 29, 30$ は捨てた重みの合計が 2^{-24} の 2.06 倍、1.03 倍で、誤差予算を超える。Horner は $t_1 = C_2 d$ の直後に k_1 ビット切り捨てるので、$C_2 d^2$ の部分積が約 500 から約 140 に縮む。展開するとこの効果を失う。

2.4.7　位相オラクルへの二つの道：Horner 算術と Walsh 直接位相

位相オラクル $U_\theta |x\rangle = \mathrm{e}^{2\pi \mathrm{i} \theta(x)} |x\rangle$ を作るとき、θ が多項式（またはその近似）なら、大きく二つの道がある。

- **方式 A（Horner 算術）。** 可逆算術で $|x\rangle|0\rangle \to |x\rangle|v\rangle$ と値 v を計算し、v に応じた位相を付け、逆計算で $|x\rangle|0\rangle$ に戻す。

- **方式 B（Walsh 直接位相）。**$\theta = \sum_S c_S \chi_S$ と展開し、$U_\theta = \prod_S \exp(2\pi \mathrm{i} c_S Z_S)$ を各項の回転で直接作る（§2.3.5）。算術用の補助量子ビットは要らない。

方式 A で位相を付ける段は、実は Walsh 展開の最も疎な場合になっている。$y = v/2^F = \sum_j v_j 2^{j-F}$ は出力ビット v_j の 1 次式なので、$\mathrm{e}^{2\pi \mathrm{i} y} = \prod_j \mathrm{e}^{2\pi \mathrm{i}\, 2^{j-F} v_j}$ と出力ビットごとの回転 1 個ずつに分かれる（§2.3.9 の 1 次式の場合）。公開 63-Q 版の 41 回転はこれにあたる。つまり方式 A は、「入力 x の上では密な位相関数を、算術で出力ビットの上の 1 次式に変換してから回転する」方法と見なせる。その代わりに、可逆な掛け算と足し算、中間値を保持する量子ビット、逆計算の費用を払う。

表 2.11　Horner 算術（方式 A）と Walsh 直接位相（方式 B）の比較。

	Horner 算術	Walsh 直接位相
基底	$1, x, x^2, \ldots$（評価順序を変える）	パリティ χ_S
得意な対象	低次の多項式（区分近似を含む）	Walsh 係数が少数に集中する関数
主な演算	可逆な掛け算・足し算	CX と Z 軸回転
補助量子ビット	中間値と出力に必要	不要
費用を決めるもの	掛け算の Toffoli と逆計算	非零係数の数（回転数）

同じ関数でも、二つの表し方で大きさがまったく違うことがある。3 ビット整数 $x = x_1 + 2x_2 + 4x_3 \in \{0, \ldots, 7\}$ 上の $f(x) = x^2 + 2x + 1$ は、Horner 法では $f = (x+2)x + 1$ で掛け算 1 回である。Walsh 展開では、x^2 がビットの実数上の 2 次式なので $\chi_{\{1,2,3\}}$ の係数だけが 0 になり、8 項中 7 項が残る（$c_\emptyset = \frac{51}{2}$、$c_{\{3\}} = -18$ など）。ビット数が増えると、Walsh の項数は $O(n^2)$ で増える一方、Horner の掛け算の回数は次数だけで決まる。ただし可逆な掛け算 1 回の Toffoli 数はビット幅とともに増えるので、どちらが安いかは幅・精度・補助量子ビットの制約を含めて比べる必要がある。本課題の $1/\sqrt{x}$ では、方式 B を入力全体に使うと項数が多すぎることを §2.3.9 の後の 22 ビットの例で見た。

近似多項式を Chebyshev 多項式の基底で $P(x) = \sum_k c_k T_k(x)$ と持つ場合は、単項式に戻してから Horner 法を使うより、三項漸化式 $T_{k+1} = 2xT_k - T_{k-1}$ を使う Clenshaw 法で評価するほうが数値的に安定である。本書の回路は単項式の係数表を Horner で評価しており、Clenshaw 法は使っていない。

2.4.8　Carleman との違い

Horner 法は「持ち上げる」代わりに「因数分解する」操作で、Carleman とは向きが逆である。

- **出力ビットごとに回転する。**最後に必要なのは y の各ビットだけで、公開 63-Q 版は出力ビットごとに回転を 1 個掛ける（41 個）。
- **代わりに算術が要る。**中間値の掛け算・足し算は可逆回路で行い、最後に逆計算で片付ける。公開 63-Q 版では、計算（約 350 Toffoli）と逆計算（約 399）の合計が Toffoli の 55% を占める。逆計算が計算本体と同程度の費用になるのは、v のビットが非線形関数で、測定で消しても位相補正に再計算が要るからである。
- **誤差は丸めで決まる。**切り捨てと係数の丸めが誤差予算の 60〜97% を占め、近似そのものの誤差は 1〜4% にすぎない（公開ノートブックの分析）。

演習：Horner 計算

$k_1 = k_2 = 2$、$C_2 = 5$, $C_1 = 9$, $C_0 = 3$, $d = 6$ のとき、式 (2.10) で t_1, t_2, v を求めよ（答え：$t_1 = \lfloor 30/4 \rfloor = 7$、$t_2 = \lfloor 16 \cdot 6/4 \rfloor = 24$、$v = 27$）。切り捨てなしの実数評価 $3 + 9 \cdot 6/4 + 5 \cdot 36/16 = 27.75$ との差（0.75）を、本文の切り捨て誤差の上界（片側で 3 未満）と比べよ。

2.5　ANF：GF(2) 上の多項式としての持ち上げ

2.5.1　概観：ANF は何をしているのか

ANF（代数標準形）は、ブール関数を XOR と AND だけで書く方法である。GF(2) では XOR が足し算、AND が掛け算にあたるので、ANF はブール関数を「GF(2) 上の多項式」として書くことにほかならない。NOT は $1 \oplus a$、OR は $a_1 \oplus a_2 \oplus a_1 a_2$ と書けるので、NOT や OR を使わなくても任意のブール関数を表せ、その表し方は一意である（表 2.12）。Walsh 展開がブール入力の関

表 2.12　2 ビットの基本演算の ANF。

演算	ANF	演算	ANF
$a_1 \wedge a_2$（AND）	$a_1 a_2$	$\neg a_1$（NOT）	$1 \oplus a_1$
$a_1 \vee a_2$（OR）	$a_1 \oplus a_2 \oplus a_1 a_2$	NAND	$1 \oplus a_1 a_2$
$a_1 \oplus a_2$（XOR）	$a_1 \oplus a_2$	XNOR	$1 \oplus a_1 \oplus a_2$

数を実数係数のパリティの和に分解する（パリティ・スペクトルを取り出す）のに対し、ANF は関数を単項式（ビットの積）の XOR に分解して論理構造を取り出す。Horner 法はそのどちらでもなく、多項式の評価順序を変える。三つの違いを表 2.13 にまとめる。同じ AND $a_1 a_2$ でも、ANF で

表 2.13　Walsh 展開・ANF・Horner 法の対比。

	Walsh 展開	ANF	Horner 法
数学的構造	GF(2) の指標による Fourier 解析	GF(2) 上の多項式	多項式の評価順序
基底	$\chi_S(x) = (-1)^{p_S(x)}$	単項式 $\prod_{i \in T} a_i$	$1, x, x^2, \ldots$
係数	実数	0 または 1	一般の数
主な演算	実数の和	XOR と AND	掛け算と足し算
得意な対象	対角位相	論理関数・可逆回路	数値関数・算術回路
高速変換	FWHT，$O(n2^n)$	高速 Möbius 変換，$O(n2^n)$	（変換ではない）

は 1 項、Walsh 展開では $\frac{1}{4}(1 - \chi_{\{1\}} - \chi_{\{2\}} + \chi_{\{1,2\}})$ の 4 項（§2.3.5）、算術としては掛け算 1 回である。逆にパリティ $a_1 \oplus \cdots \oplus a_n$ は、ANF では n 項、Walsh では 2 項になる。どの表し方が短いかは関数によって決まり、どれかが常に短いわけではない。

2.5.2　代数標準形と Möbius 変換

ブール関数 $f:\{0,1\}^A \to \{0,1\}$ は、GF(2) 上の多項式（代数標準形、ANF）として一意に

$$f(a) = \bigoplus_{T \subseteq \{1,\dots,A\}} \lambda_T \prod_{i \in T} a_i, \qquad \lambda_T \in \{0,1\}$$

と書ける。Walsh 展開が実数係数でパリティの線形結合を取るのに対し、ANF は係数が GF(2) で、単項式（ビットの積）の XOR を取る。入力 x を部分集合と見なすと、$\prod_{i\in T} x_i = 1$ となるのは $T \subseteq x$ のときだけなので

$$f(x) = \bigoplus_{T \subseteq x} \lambda_T. \tag{2.11}$$

逆に、真理値表から係数を取り出す式が

$$\boxed{\lambda_T = \bigoplus_{U \subseteq T} f(U)} \tag{2.12}$$

である。導出は、式 (2.11) を式 (2.12) の右辺に代入して

$$\bigoplus_{U\subseteq T}\ \bigoplus_{V\subseteq U} \lambda_V = \bigoplus_{V\subseteq T} \Big(\#\{U : V \subseteq U \subseteq T\} \bmod 2\Big)\lambda_V = \bigoplus_{V\subseteq T} 2^{|T\setminus V|}\lambda_V$$

と整理すると、$2^{|T\setminus V|}$ が偶数になるのは $V \neq T$ のときだけで、$V = T$ の項 λ_T だけが残ることによる。

行列では、$M_{xT} = 1 \iff T \subseteq x$ と置くと $f = M\lambda \pmod 2$ で、

$$M = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{\otimes A}$$

と Kronecker 積の形に書ける（1 ビットごとに T が x に含まれるかどうかの条件が掛け合わされる）。$M^2 = I \pmod 2$ なので、式 (2.12) の変換にも同じ行列を使う。M は 2 進数として $T \le x$ の領域だけに 1 がある下三角行列で、成分は Sierpinski 三角形（Pascal 三角形の偶奇）になる。Kronecker 構造により、高速 Walsh–Hadamard 変換と同じバタフライで $O(A\,2^A)$ で計算できる：各ビット i について「x の第 i ビットが 1 のものに、第 i ビットを 0 にした x の値を XOR する」を A 回繰り返す。

小さな例を 2 ビットで示す。$A=2$ の $M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$（行 $x = 00, 01, 10, 11$）で、AND（真理値表 $(0,0,0,1)$）の係数は $(0,0,0,1)$ すなわち a_1a_2、OR（$(0,1,1,1)$）の係数は $(0,1,1,1)$ すなわち $a_1 \oplus a_2 \oplus a_1a_2$、XOR（$(0,1,1,0)$）は $(0,1,1,0)$ すなわち $a_1 \oplus a_2$ である。いずれも式 (2.12) で手計算し、プログラムでも確かめた。

2.5.3　次数と疎性

ANF の次数 $\deg f$ は、$\lambda_T = 1$ となる最大の $|T|$ である。$\deg f \le A$ であり、次数が小さいほど現れ得る単項式の種類は制限される。次数が d 以下（$0 \le d \le A$）なら、定数項を含めた候補数は最

大 $\sum_{k=0}^{d}\binom{A}{k}$ 個である。ただし、実際の非零項数は関数に依存する。例えば 8 変数では、1 次関数 $a_1 \oplus \cdots \oplus a_8$ は 8 項を持つ一方、8 次関数 $a_1a_2\cdots a_8$ は 1 項だけである。乱数的な関数では、約半数の T に $\lambda_T = 1$ が立つので、$A = 8$ では単項式が列あたり $2^7 = 128$ 個程度（期待値）で、高い次数の項も含む。各単項式をそのまま積で実装すると、$|T| - 1$ 個の AND が要り、項数が爆発する。そこで、単項式の数ではなく、積を何回使うかだけを数える費用モデルに移る。

2.5.4　乗法複雑度：XOR–AND 回路の最小 AND 数

このコンテストの費用モデルでは、CX と X によるアフィン演算は無料である。アフィン式とは、定数とビットの XOR で書ける式をいう。二つのビットの AND を、0 で初期化した別の量子ビットへ書く操作には 1 Toffoli を使う。

■入力だけのアフィン積で表す場合　まず、L_0, L_r, M_r が元の入力ビット $\boldsymbol{a} = (a_1, \ldots, a_A)$ だけのアフィン式である場合を考える。

$$f(\boldsymbol{a}) = L_0(\boldsymbol{a}) \oplus \bigoplus_{r=1}^{R} \big(L_r(\boldsymbol{a}) M_r(\boldsymbol{a})\big) \tag{2.13}$$

この式は、入力アフィン式の積を R 個使う 2 次以下の表現である。一般の乗法複雑度には、次に示す計算済みの積の再利用も含める。

■一般の定義：計算済みの積も再利用する　一般の**乗法複雑度**は、XOR、AND、NOT を使う回路で関数 f を計算するための**最小 AND 数**である [19]。XOR と NOT の個数は、この尺度では数えない。AND を計算順に並べると、その出力 u_r と最終出力は

$$\begin{aligned} u_r &= L_r(\boldsymbol{a}, u_1, \ldots, u_{r-1})\, M_r(\boldsymbol{a}, u_1, \ldots, u_{r-1}), \quad r = 1, \ldots, R, \\ f(\boldsymbol{a}) &= L_{\mathrm{out}}(\boldsymbol{a}, u_1, \ldots, u_R) \end{aligned}$$

と書ける。ここで各 $L_r, M_r, L_{\mathrm{out}}$ は、入力と**先に計算した *AND* の出力**に関するアフィン式である。この再帰的な使い方まで許した最小の R が、一般の乗法複雑度になる。

例えば 3 次関数 $f = a_1a_2a_3$ は式 (2.13) だけでは表せないが、$u_1 = a_1a_2$、$u_2 = u_1a_3$、$f = u_2$ とすれば 2 AND で計算できる。1 AND の回路の出力は入力アフィン式と一つのアフィン積の XOR であり、次数が高々 2 なので、この 3 次関数には 2 AND が必要かつ十分である。

この最小 AND 数は、可逆回路全体の Toffoli 数と常に等しいわけではない。積の順方向計算を Toffoli で実装しても、補助ビットの確保と消去、入力の復元、測定後の位相補正を別に設計する必要がある。逆計算に追加の Toffoli が必要な場合もあり、測定による消去が無料になるかは、補正を含む回路構成に依存する。

2.5.5　2 次形式とシンプレクティック階数

まず 2 次の部分だけを考える。f の 2 次の係数 λ_{ij}（$i < j$）を、対角を 0 とした対称行列 B（$B_{ij} = B_{ji} = \lambda_{ij}$、GF(2) 上）に並べる。これは交代行列（シンプレクティック形式）で、階数は偶数になる。アフィン形の積 $(u \cdot a)(v \cdot a)$ の 2 次部分は $B = uv^{\top} \oplus vu^{\top}$（$a_i^2 = a_i$ は 1 次に吸収）で、階数は高々 2 である。次の結果が知られている（Dickson の定理、GF(2) の場合）。2 次形式 Q に付

随する交代行列 B の階数が $2r$ なら、Q は適当な可逆線形変換で

$$Q = y_1y_2 \oplus y_3y_4 \oplus \cdots \oplus y_{2r-1}y_{2r} \ \oplus \ (\text{1 次式})$$

の形にでき、1 次式は CX で作れる。積は r 個で足りる。入力アフィン式の積を足す分解では、1 個の積が寄与する階数は高々 2 なので、r 個未満では階数 $2r$ に届かない。さらに 2 次 Boolean 関数については、一般の XOR–AND 回路まで許しても最小 AND 数がこの r と一致することが知られている [20]。したがって、2 次関数の乗法複雑度は階数の半分である。

3 変数の例を示す。$f = a_1a_2 \oplus a_1a_3 \oplus a_2a_3$（過半関数）の $B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ は、3 行を足すと 0 になるので階数 2、$r = 1$ である。$y_1 = a_1 \oplus a_3,\ y_2 = a_2 \oplus a_3$ と置くと $y_1y_2 = a_1a_2 \oplus a_1a_3 \oplus a_2a_3 \oplus a_3$ なので

$$f = (a_1 \oplus a_3)(a_2 \oplus a_3) \oplus a_3$$

となる（$a_3a_3 = a_3$ が $\oplus a_3$ と打ち消す）。単項式で書けば 3 積、アフィン形で書けば 1 積である。4 変数の「すべての 2 変数の積の和」の階数は 4（$r = 2$）、5 変数でも 4、6 変数なら 6 である（GF(2) 上の消去で確かめた）。

2.5.6　行列イメージ

ANF の係数 λ_T から真理値表への変換は、GF(2) 上の行列積である。$A = 3$ のとき、行を入力 x、列を単項式の添字 T とし、$M_{xT} = 1 \iff T \subseteq x$ と定めると $f = M\lambda \pmod 2$ になる（図 2.7）。図の例は $\lambda_{011} = \lambda_{101} = \lambda_{110} = 1$、すなわち $f = a_1a_2 \oplus a_1a_3 \oplus a_2a_3$（$x$ のビット列は $a_3a_2a_1$ の順）で、f は 2 個以上のビットが 1 のとき 1 になる過半関数である。8 行すべてを実際に計算して確かめてある。Walsh の行列がすべて ± 1 の密行列だったのに対し、ここでは約半分が 0 で、しかも $2^A \times 2^A$ の下三角である。

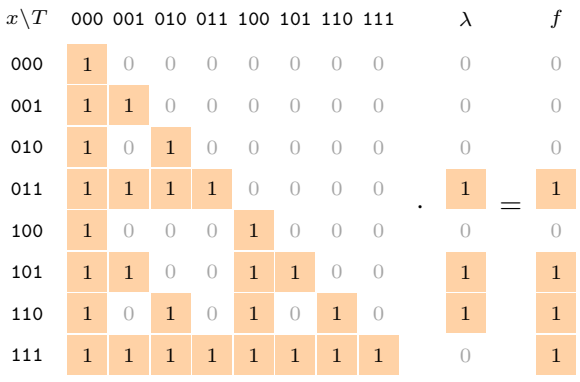


図 2.7　$A = 3$ の Möbius 行列 M（橙が 1）による $f = M\lambda \pmod 2$。係数ベクトル λ は過半関数に対応し、結果 f は x のビット数が 2 以上の行で 1 になる。

2 次の部分は、階数 $2r$ の対称行列 B に対応する。図 2.8 の例では B の階数が 2 なので、積は 1 個で済む。表 2.14 で 8 入力すべてについて、単項式の形と 1 個の積の形が一致することを確かめた。単項式で書けば 3 積、アフィン形で書けば 1 積であり、この差が公開 63-Q 版で「アフィン形同士の積」を使う理由である。

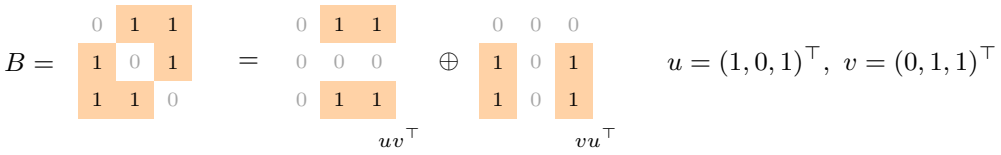


図 2.8　過半関数の 2 次部分の行列 B（GF(2)、対角 0 の対称行列）。階数 2 で、$B = uv^\top \oplus vu^\top$ と 1 組の (u, v) に分解できるので、積は 1 個で足りる。u, v は積 $(a_1 \oplus a_3)(a_2 \oplus a_3)$ のアフィン形の係数である。

表 2.14　過半関数の検証。単項式で書いた f と、1 個の積に直した $g = (a_1 \oplus a_3)(a_2 \oplus a_3) \oplus a_3$ が全入力で一致する。

a_1	a_2	a_3	a_1a_2	a_1a_3	a_2a_3	f	$a_1 \oplus a_3$	$a_2 \oplus a_3$	積	g
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	1	0	0
1	1	0	1	0	0	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	0
1	0	1	0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	1	0	0	1
1	1	1	1	1	1	1	0	0	0	1

2.5.7　高次の部分：seed による引き下げ

3 次以上の項は、先に作った積（seed）を素材にして下げる。例えば $f_1 = a_1a_2a_3$ と $f_2 = a_1a_2a_4 \oplus a_3$ では、seed $s = a_1a_2$ を 1 個作ると、$f_1 = s \cdot a_3$、$f_2 = s \cdot a_4 \oplus a_3$ がそれぞれ 1 積で済み、積は計 3 個である。別々に作れば $2 + 2 = 4$ 個かかる。列が多いほど、共有できる seed の効果が大きくなる。

2.5.8　公開 63-Q 版の係数表 lookup

公開 63-Q 版の係数表 lookup は、8 ビットのアドレス a から 35 列の係数ビット $f_j(a)$ を出力する（256 行 × 35 列、真理値表は 8,960 ビット）。各列をそのまま単項式に展開すると項数が爆発するので、式 (2.13) のアフィン積を部品とし、先に計算した seed や係数列も後続のアフィン式に使う XOR–AND 回路を構成する。これにより高次の関数も扱える。アフィン演算は CX と X で作れるため、主な非線形費用は新たに作る積の個数になる。共有した積の費用を列ごとに重複計上せず、消去と補正も含む回路全体で Toffoli 数を数える。2 次の部分はシンプレクティック形式の階数で積の数が決まり、高次の部分は先に作った積（seed）を素材にして下げる。公開 63-Q 版は seed 35 個と、列ごとの 4〜5 個の積（計 151 個）で、$35 + 151 = 186$ Toffoli に収めている（単項反復 QROM なら $2^8 - 1 = 255$、比は $186/255 \approx 0.73$）。この表を可能にしているのは、CX を無料とする費用モデルである。積の素材として使えるアフィン形は量子ビットの任意の GF(2) 結合なので、「各列を、できるだけ少ない積で書ける形にする」問題が、GF(2) 上の線形代数（消去と階数）の問題になる。

積の数を概算する一つの方法は、パリティ積で列を分解する上界を計算する。ランダムなアフィン形同士の積を P 個の seed として作り、各列を「アフィン関数＋積 r 個」で表せる最小の r を

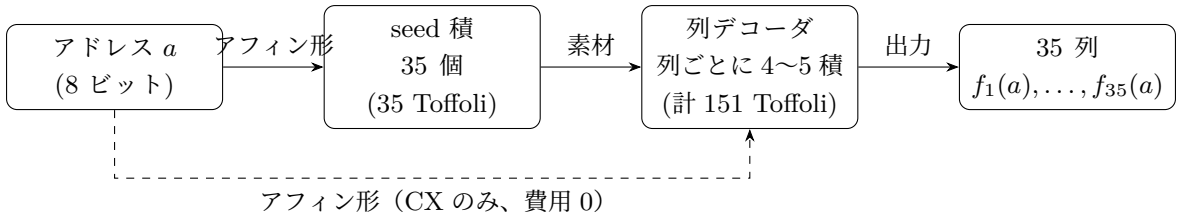


図 2.9　公開 63-Q 版の係数表 lookup の構造（模式図）。35 個の seed 積は全列で共有され、各列のデコーダは seed とアドレスのアフィン形から 4〜5 個の積で列を作る。計 186 Toffoli（内訳の 151 は $186 - 35$ による）。

GF(2) 消去で求め、作った列を後の列の素材として再利用する。候補は全アドレスの真理値表で照合する必要がある。結果を表 2.15 に示す。

表 2.15　lookup の積の数の推定（数値例）。公開 63-Q 版の実測は A=8 で 186（35 列、符号化後）である。

	$A = 8$	$A = 9$
列（生）	54	51
積の数（プール上限 40 本）	205	395
積の数（$P = 60$）	184	319
積の数（最良、$P = 100/130$）	174	262
単項反復 QROM（$2^A - 1$）	255	511
比（上限 40 本）	0.80	0.77

$A = 8$ の推定 205 は実測 186 より約 10% 弱い（列数が 54 と 35 で表が違うので想定の範囲内）。$A = 9$ は損益分岐に届かない公算が大きいと見積もられている（推定）。

2.5.9　量子回路との対応：ビットオラクルと位相オラクル

ANF の演算は、量子回路の基本ゲートにそのまま対応する。定数 1 の XOR（ビット反転）は X、$a_i \oplus a_j$ を作るのは CX、積 $a_i a_j$ を 0 の補助ビットに書くのは Toffoli、3 個以上の積は多重制御 X である。

■ビットオラクル。 $U_f|\boldsymbol{a}\rangle|y\rangle = |\boldsymbol{a}\rangle|y \oplus f(\boldsymbol{a})\rangle$ は、ANF の各項を y へ XOR すれば作れる。例えば OR $f = a_1 \oplus a_2 \oplus a_1 a_2$ なら、CX$(a_1 \to y)$、CX$(a_2 \to y)$、Toffoli$(a_1, a_2 \to y)$ の 3 ゲートで済む。ANF は一意だが、そこから作る回路は一意ではない。共通する積の再利用やゲートの並べ替えで回路は変わる。例えば $f = a_1 a_2 a_3 \oplus a_1 a_2 a_4$ は $f = a_1 a_2 (a_3 \oplus a_4)$ とくくり出せ、積 $a_1 a_2$ を 1 回作って $(a_3 \oplus a_4)$ との積を取れば 2 積で済む（単項式ごとに作ると 4 積）。共通因子をくくり出すこの操作は Horner 法の括り直しに近く、上の seed による引き下げはこれを多数の列にわたって行うものである。

■位相オラクル（符号）。 $(-1)^{f(\boldsymbol{a})}$ を付けるオラクルは、XOR が符号の積になる（$(-1)^{u \oplus v} = (-1)^u (-1)^v$）ので、ANF の項ごとに分解できる。

$$(-1)^{f(\boldsymbol{a})} = \prod_{T : \lambda_T = 1} (-1)^{\prod_{i \in T} a_i}$$

1 次の項は Z、2 次の項は CZ、3 次の項は CCZ にあたり、例えば $f = a_1 \oplus a_2 a_3$ なら $U_f = Z_1\,\mathrm{CZ}_{2,3}$ である。

■一般の角度の位相。 この分解がそのまま成り立つのは、位相が ± 1（半回転）のときだけである。一般の角度 $\mathrm{e}^{2\pi \mathrm{i}\phi f(\boldsymbol{a})}$ では、f を 0/1 の**実数**として扱う必要があり、XOR は $u \oplus v = u + v - 2uv$ と実数の式に直さなければならない。項ごとの積には分解できない。道は二つある。一つは f を補助ビットに計算し、そのビットに Z_POW(2ϕ) を 1 個掛けてから逆計算する方法で、費用は f の乗法複雑度（§2.5 の乗法複雑度の小節）で決まる。もう一つは f を Walsh 展開して直接位相にする方法（§2.3.5）で、項数は ANF の短さとは関係なく決まる。公開 63-Q 版の invsqrt は前者の考え方を使っており、ANF とアフィン積で係数ビットを計算し、Horner 算術を経て、出力ビットごとに回転を掛ける（§2.4.7）。

■注意。 ANF の項数や次数は回路の大きさの目安になるが、項数が最少の ANF が Toffoli 数や T 数の最小の回路を与えるとは限らない。本節で見たように、費用を決めるのは単項式の数ではなく、積の共有を含めた乗法複雑度と、消去や位相補正を含む回路全体の構成である。

2.5.10　Carleman との違い

- **持ち上げる体が違う。**実数ではなく GF(2) 上の持ち上げで、目的は位相ではなくビット値を作ることである。
- **展開は有限で厳密だが、そのまま使うと高価である。**単項式の数（A=8 の表で列あたり百個以上）を、積の共有で「乗法的複雑度」程度まで圧縮するのが本質である。
- **再利用の構造がある。**作った積を後の列の素材にする点は、Carleman 階層で低次の変数が高次の式に現れる構造に似ている。

演習：ANF

$f = a_1 a_2 \oplus a_2 a_3 \oplus a_3 a_4$ の交代行列 B を書き、階数が 4 であることを確かめよ。また、$f = a_2(a_1 \oplus a_3) \oplus a_3 a_4$ と 2 個の積で書けることを確かめ、1 個の積では書けない理由を述べよ（答え：1 個の積の階数は高々 2）。

2.6　比較と適用条件

2.6.1　四手法の役割と行列表現

三つの展開を、行列の形で並べると次のようになる。表 2.17 の「疎か密か」が、費用が何で決まるかを表している。

2.6.2　表現と実行費用の取引

Walsh 展開や部分積の直接位相合成では、算術の結果を保持せず特徴量へ位相を付けられるが、特徴量を増やすと回転数が増える。Horner 法では出力ビットに位相を付けるための算術と逆計算が要る。公開 63-Q 版の direct phase taps（22 回転）は、一部の項を算術から外して直接位相にした中

表 2.16　Carleman 線形化と三つの展開の比較。

	持ち上げる先	有限性・誤差	回路での費用
Carleman 線形化	単項式 u^k（実数）	無限階層。打ち切り次数で誤差	線形系の次元
Walsh 展開	パリティ $p_S(x)$（入力全体）	有限 2^n 項で厳密。誤差は捨てる項で決まる	残す項の数＝回転数（1 個約 7.2 Toffoli）
Horner 計算	持ち上げない（因数分解）	近似と丸めで誤差	算術（掛け算・足し算）とその逆計算
ANF	GF(2) の単項式（アドレス）	有限で厳密	積の数（Toffoli）。共有で圧縮

表 2.17　行列で見た三つの展開の大きさと疎密。

	行列の形	疎か密か	費用を決めるもの
Walsh 展開	$2^n \times 2^n$、成分 ± 1（$\theta = Hc$）	密。列を k 本だけ残す（列の疎化）	残す列の数 k（回転数）
Horner 計算	係数は 3 本 $[C_0\ C_1\ C_2]$ だが、掛け算は部分積の格子（例：10×14 と 19×14）	格子は密で、切り捨てで下位の三角形を落とす	格子のビット数（算術とその逆計算）
ANF	$2^A \times 2^A$ の下三角、成分 0/1（$f = M\lambda$、GF(2)）	約半分が 0（Sierpinski 状）。2 次部分は階数 $2r$ の交代行列	積の数（階数と seed の共有）

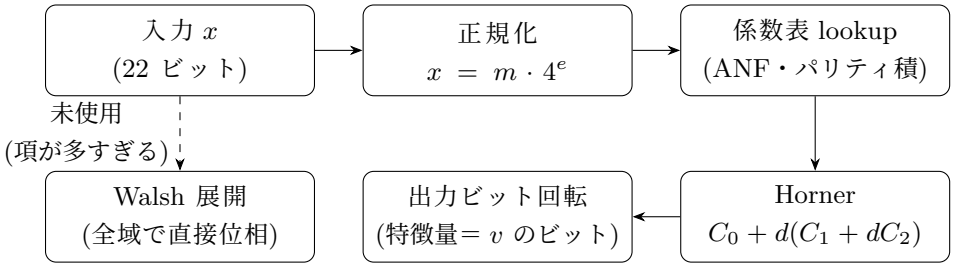


図 2.10　公開 63-Q 版での三つの展開の位置づけ。ANF は係数表、Horner は算術、最後に v のビットを特徴量として回転する。Walsh 展開を入力全体にそのまま使う道は、項数が多すぎて採られていない。

間的な例である。ANF では積を共有して非線形演算を減らせる一方、保持用の線が増える。このように、何を特徴量にし、何を算術で評価し、何を共有するかを、精度と $Q \times C$ で比較する。

Carleman 線形化で見た次元と打ち切りの取引とも比較できるが、この公開回路は ODE の線形化を実行していない。第 4 章では、区間ごとの 2 次多項式を算術で評価する公開構成を参照回路と比べる。

2.6.3　Walsh 展開・Horner 評価・ANF の適用条件

ここでは上の比較を、数値評価の保証と有限ビット写像の更新へ具体化する。

Horner 法が変えるもの：評価順序

第 2.4 節の多項式の恒等性は厳密な実数演算についてのものであり、浮動小数点では評価順序の変更で丸め誤差が変わる。Horner 法だけで近似次数や近似誤差が改善するわけではない。固定小数点の回路では、さらに各段の切り捨てやオーバーフローがある。同じ実数多項式から出発しても、途中で切る位置が違えば整数出力は一致しないことがある。

したがって invsqrt では、係数・スケール・各段のビット幅・切り捨て位置を固定した整数演算として検証する。多項式としての恒等式を確かめただけでは、実装された位相や補助レジスタの消去まで保証したことにはならない。乗算回数の削減も、そのまま Toffoli 数の同率削減を意味せず、オペランドの幅と可逆な作業領域を含めて比較する。

有限ビット写像を特徴量で表す場合

Walsh 展開と ANF は、位相関数や表の各ビットを表すだけでなく、有限ビット写像の合成を特徴量の更新として書くことにも使える。これは比較のための補足であり、本課題の回路をこの形へ置き換えて実行しているという意味ではない。

完全に指定された写像 $F : \{0,1\}^b \to \{0,1\}^b$ を考える。全状態に対する値が決まっていることが前提であり、有効入力だけが指定されている課題では、範囲外をどう定めるかも必要になる。$N = 2^b$ として全 Walsh 指標を並べれば

$$K_{st} = \frac{1}{N} \sum_{x \in \{0,1\}^b} \chi_s(F(x)) \chi_t(x), \qquad \boldsymbol{\chi}(F(x)) = K \boldsymbol{\chi}(x)$$

と書ける。これは、入力に非線形な特徴量を計算した後では、その更新を行列で表せるという意味での**持ち上げ**である。観測量 $g(x) = \boldsymbol{c}^{\mathsf{T}} \boldsymbol{\chi}(x)$ の合成 $g \circ F$ の係数は $K^{\mathsf{T}} \boldsymbol{c}$ となる。K は状態 x 自体ではなく、選んだ特徴量を更新する。

係数の計算には、非正規化 Walsh 行列 $H_{xs} = \chi_s(x)$ を用いられる。$H^{\mathsf{T}} H = N I$ より、$\boldsymbol{v} = H \boldsymbol{c}$ を全入力上の関数値とすると、合成後の係数は

$$\boldsymbol{c}' = \frac{1}{N} H^{\mathsf{T}} \boldsymbol{v}_F, \qquad (\boldsymbol{v}_F)_x = \boldsymbol{v}_{F(x)}$$

である。高速 Walsh–Hadamard 変換（FWHT）を使えば、密な K を保存せずに係数を更新できる。F の真理値表が与えられ、各参照を定数時間で行える場合でも、$O(N \log N)$ 時間・$O(N)$ 作業領域を要し、$N = 2^b$ の指数的な大きさは残る。

F が非可逆でも特徴量の式は成立するが、K は一般にユニタリではない。例えばすべての入力を同じ値へ写す F では情報が失われる。F が全状態の置換なら完全 Walsh 基底での K は直交行列になるが、そのことだけから安価な量子回路が得られるわけではない。

ANF の線形性は GF(2) 上のもの

Boolean 単項式 $m_s(x) = \prod_{i \in s} x_i$ を全て並べると、各 $m_s(F(x))$ も ANF で表せるため

$$\boldsymbol{m}(F(x)) = M \boldsymbol{m}(x) \pmod 2$$

と書ける。各行の係数は、その特徴量の真理値表に XOR による部分集合 Möbius 変換を施して求める。$x_i^2 = x_i$ なので同じ変数の高いべきは要らないが、完全基底は定数を含めて 2^b 個あり、常に

小さくなるわけではない。

演算体の違いは XOR の例ではっきりする。$a, b \in \{0, 1\}$ に対し

$$\begin{aligned} &\text{GF(2) 上：} && a \oplus b = a + b \pmod 2, \\ &\mathbb{R} \text{ 上：} && a \oplus b = a + b - 2ab, \\ &\text{Walsh 指標：} && (-1)^{a \oplus b} = (-1)^a (-1)^b. \end{aligned}$$

例えば $a = b = 1$ なら XOR は 0 だが、実数の和は 2 である。実数上では $\chi_s(x) = \prod_{i \in s} (1 - 2x_i)$ と Boolean 単項式を結び付けられる一方、ANF の mod 2 の演算規則まで実数へ移せるわけではない。

■同じ写像を二つの演算体で書く例　$F(x_0, x_1) = (x_1, x_0 \oplus x_1)$ とする。ANF では

$$\begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \pmod 2.$$

入力 $(1, 1)$ は $(1, 0)$ に移るが、この行列を実数演算で使うと $(1, 2)$ になり、違う写像になってしまう。一方、$\chi_0 = (-1)^{x_0}$、$\chi_1 = (-1)^{x_1}$、$\chi_{01} = (-1)^{x_0 + x_1}$ を使えば、実数上で

$$\begin{pmatrix} \chi_0(F(x)) \\ \chi_1(F(x)) \\ \chi_{01}(F(x)) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \chi_0(x) \\ \chi_1(x) \\ \chi_{01}(x) \end{pmatrix}$$

と書ける。同じビット出力を再現していても、ANF は 2 個のビット特徴を GF(2) 上で、ここで選んだ Walsh 表現は 3 個の指標を実数上で更新している。特徴数だけを比べて、量子振幅の線形代数でも同じ圧縮ができると判断してはいけない。

必要な特徴だけで閉じる条件と検証

すべての特徴を持つ代わりに一部の集合 S を使うには、更新後の各特徴が同じ集合の線形結合で表せなければならない。Walsh なら

$$\chi_s \circ F \in \operatorname{span}_{\mathbb{R}} \{\chi_t : t \in S\} \qquad (s \in S)$$

という**閉包**条件である。ANF でも GF(2) 上の同じ条件を課す。上の例では、Walsh の一つの指標から始めても更新を繰り返すと残りの二つが必要になる。

必要な行の非ゼロ支持を順に加え、増えなくなったところで閉包が得られる。Walsh 和を整数として、ANF の係数をビットとして計算すれば、ゼロかどうかを厳密に判断できる。小さい浮動小数点係数を閾値で捨てた場合や、特徴数の上限で打ち切った場合は、別途誤差を評価する近似となる。

小さな写像では全入力に対し、元の F と特徴量の更新から読み戻した出力を比較できる。基底表現が厳密でも、真理値表による直接実行より速いとは限らない。また、古典的な特徴量更新の一致だけでは、KickMix 回路の位相、測定に伴う補正、補助量子ビットのゼロ復元は保証しない。回路として採用する際は、これらを含む第 7 章の検証へ戻る必要がある。

第 3 章

KickMix 回路の最適化手法

この章で学ぶこと

- 課題ごとの費用式が何を報い、Toffoli・量子ビット幅・精度がどう関係するかを理解する。
- 2026 年 10 月 28 日に取得した 63 量子ビット公開回路の最適化を、Toffoli 削減・位相回転・幅圧縮・lookup に分けて整理する。
- 最適化手法の原理と、2026 年 10 月 28 日に取得した公開回路の一例で確認された使い方を区別して説明できるようになる。

本章は、第 1 章で述べたベンチマークの費用式を出発点にして、回路側でどのような工夫がその費用式に効くかを整理する。課題ごとの費用モデルを説明した後、具体例として、2026 年 10 月 28 日に取得した invsqrt の公開 41 回転回路を扱う（以下「10/28 公開回路」）[3]。回路に実装された工夫、公開作者の説明、回路命令列からの推定は区別して示す。この取得版は歴史的な教材例であり、現時点の最良提出物や一般的な最適性を主張するものではない。

注意

以下の数値はすべて 2026 年 10 月 28 日に取得した公開 41 回転版と、その時点で参照できた公開資料に関する例である。公式費用式から直接得られる値、公開作者の報告、命令列からの逆解析、粗い費用推定を区別し、推定には「推定」と明記する。生成器の一部が公開資料だけでは分からない場合、その実装詳細を確定事項として扱わない。

3.1　課題ごとの費用モデル

本章では T を Toffoli 様演算数、Q を回路が使う量子ビット数、C を invsqrt の重み付きゲート費用とする。三課題のスコアは同じ形ではない。

$$\mathrm{cost}_{\mathrm{invsqrt}} = QC, \qquad \mathrm{cost}_{\mathrm{dqi}} = R(\varepsilon_+)Q^2T, \qquad \mathrm{cost}_{\mathrm{dfthcsos}} = \lambda_{\mathrm{eff}}TQ.$$

dqi の T はショット平均、dfthcsos の T は ANGLE lookup の回路評価と他部分の解析的費用モデルを合わせた量である。費用の小ささだけでは不十分で、各課題の正しさ、精度ゲート、レジスタ制約を満たさなければならない。数値表現の word 幅も、必要な精度と回路資源をつなぐ設計変数であ

る。word 幅を狭めると算術や lookup が小さくなる場合があるが、近似誤差が増えるため、各課題の許容条件と同時に評価する。

3.2　費用モデルが報いるもの

3.2.1　公式の費用式

invsqrt の公式 Verifier [2] は、シミュレーションした全ショットの平均として費用を数える。

$$\boxed{\text{score} = Q \times C, \qquad C = \overline{N_{\text{Tof}}} + \frac{1}{4}\Big(N_T + n_{\text{rot}}\left(0.56 \log_2 \tfrac{1}{\varepsilon} + 5.3\right)\Big)} \tag{3.1}$$

ここで Q は `circuit.num_qubits`、$\overline{N_{\text{Tof}}}$ は Toffoli 様ゲート（CCX、CCZ、CSWAP）の 1 ショットあたり実行回数の平均、N_T は T 角の回転の個数、n_{rot} は一般角回転の個数である。合成誤差 ε は、回転ごとの誤差を合計した最悪値が、シミュレーションで観測された余裕に収まるように決まる。

$$\boxed{\varepsilon = \frac{\pi \cdot \text{min_margin}}{n_{\text{rot}}}} \tag{3.2}$$

ここで $\text{min_margin} = \min_{x,\text{shot}}\big(\text{tol}(x) - |\text{error}(x)|\big)$ は turn 単位の最小余裕である（Verifier 内部では 2^{-64} turn 単位で保持し、式では 2^{64} で割っている）。1 回転あたり最大 2ε ラジアンの位相誤差を許し、それを n_{rot} 個で均等に割ると、$n_{\text{rot}} \cdot 2\varepsilon \le 2\pi\,\text{min_margin}$ から式 (3.2) が得られる。

3.2.2　ゲートの分類と単価

Z_POW の分類は Verifier の `z_pow_class` が決める。引数は半回転単位の t であり、$\text{Z_POW}(t) = \text{diag}(1, e^{i\pi t})$ である。t が 1/2 の整数倍なら Clifford、1/4 の整数倍なら T、それ以外が一般角回転である。

表 3.1　公式費用式における単価

ゲート	扱い	単価（Toffoli 換算）
CX、X、SWAP	費用に入らない	0
Clifford 角の Z_POW、CZ、Z	無料	0
T（t が 1/4 の奇数倍）	T 1 個	1/4
一般角の Z_POW	$(0.56 \log_2(1/\varepsilon) + 5.3)$ 個の T	約 7.2
CCX、CCZ、CSWAP	Toffoli 様ゲート。実行されたときだけ数える	1
HMR、R（測定・リセット）	費用に入らない（乱数は回路ハッシュと salt で決まる）	0

一般角回転の単価は、10/28 公開回路の値を例にすると見積もれる。回転数 $n_{\text{rot}} = 41$、合成誤差 $\varepsilon = 2.434 \times 10^{-13}$ を式 (3.1) に入れると、1 個あたり $0.56 \log_2(1/\varepsilon) + 5.3 \approx 28.8$ 個の T、すなわち約 7.2 Toffoli である。41 個では約 295 Toffoli 換算になり、10/28 公開回路で報告された

$C = 1{,}643.59$ の約 18% に当たる。ε の対数にしか依存しないので、最小余裕が 2 倍になっても節約は 1 回転あたり $0.56/4 = 0.14$ Toffoli にとどまる。

3.2.3　なぜ Q と Toffoli だけがてこか

式 (3.1) から、操作できるのは次の 3 つだけである。

1. Q：回路が使う量子ビット数。
2. $\overline{N_{\mathrm{Tof}}}$：実行される Toffoli 様ゲートの平均個数。
3. 回転の個数と最小余裕（n_{rot}、min_margin）：Toffoli 換算の回転費用を決める。

CX、X、SWAP、Clifford 角は式に現れないので、これらをいくら増やしても score は変わらない。一方、交換比率は次のようになる（10/28 公開回路の費用値による例）。

- Toffoli を 1 個減らすと、score は $Q = 63$ だけ下がる。
- 量子ビットを 1 本減らすと、$Q \times C$ を保つなら C を約 26.5 Toffoli 増やしてよい（$103{,}546.17/62 - 1{,}643.59 \approx 26.5$）。

つまり、この回路幅・費用の近傍では、量子ビットを減らす代わりに払える費用は 1 本あたり約 26.5 Toffoli である。この比率は例のスコアを比べる物差しであり、他の回路や課題の普遍的な上限ではない。

Note

Verifier が報告する `max_toffoli_count` は回路命令から得る静的上限であり、適応実行の最大値を別に測ったものではない。score に使われるのはショット平均である。2026 年 10 月 28 日取得版では、静的な Toffoli 様ゲートが 1,416 個、平均実行数が約 1,348.50 個で、その差は条件付きブロックによる（3.3.6 節）。この数値は当該版の例である。

3.3　Toffoli を減らす

3.3.1　論理 AND の測定ベース消去

2 つの制御の論理積を新しい量子ビットに書く操作は、Toffoli 1 個で行える。通常の可逆計算ではこの AND を消すのにもう 1 個の Toffoli が要る。しかし標的が $|0\rangle$ から始まって $a \wedge b$ になっているだけなら、標的を X 基底で測定し、結果 m に応じて位相を直せば Toffoli なしで消せる。測定ベース消去（HMR + Clifford CZ の補正）である。

標的が $|a \wedge b\rangle$ のとき、X 基底測定の結果 m により状態に位相 $(-1)^{m\,(a \wedge b)}$ が残る。これは $m = 1$ のとき $\mathrm{CZ}(a, b)$ を掛ければ打ち消せる。結果として、AND の計算は Toffoli 1 個、消去は 0 個で済む。

この手法は参照回路にすでに入っている。参照回路の `_add_row` は、制御ビットとオペランドビットの AND を作業レジスタに書く `init_and`、加算、`del_and` という流れで、`del_and` が測定ベース消去である。10/28 公開回路では HMR が 959 個あり、算術のキャリーや係数列の消去に広く使

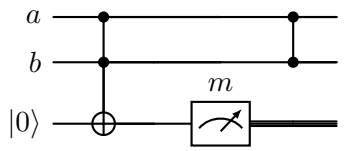


図 3.1　AND の計算（Toffoli 1 個）と測定ベース消去の概念図。標的を X 基底で測定する HMR の結果を m とし、$m = 1$ のときだけ a と b の間に CZ を掛ける（古典制御は図では省略）。CZ は Clifford なので費用は 0 である。

われている。

■公開 41 回転版についての検査例　対象の回路命令列について、CCX の直前に標的が制御の積と全入力で一致するもの（AND の逆計算）が残っていないかを古典シミュレーションで調べる方法がある。該当を全入力で確認した範囲では **0 件**だった。これはこの取得版に対する検査結果であり、別回路や別の状態準備では追加候補があり得る。

3.3.2　ダーティ借用線による 5 CCX から 4 CCX への書き換え

公開作者の README が報告した、この取得版の直前の改良（スコア比 20.032 から 20.056）がこの書き換えの例である。これは特定の版に関する報告で、一般的な最適化限界を示すものではない。制御 a, b、制御兼標的 c、標的 t に加えて、任意の値を持つ借用線 d（ダーティ補助）を使う、次の 5 CCX のブロックを考える。

```
（旧）  CCX a b d        （新）  CX  d c
        CCX d c t                CCX a b d
        CCX a b d                CCX d c t
        CCX d c t                CX  d c
        CCX a b c                CCX a b d
                                 CCX d c t
                                 CX  d t
```

旧ブロックは CCX を 5 個、新ブロックは CCX を 4 個と CX を 3 個使う。5 本の線はすべて異なり、d と標的の初期値は任意でよい。両者は次の同じ置換を実装する。

$$\boxed{c' = c \oplus ab, \qquad t' = t \oplus abc \ (c \text{ は書き換え前の値}), \qquad a' = a,\ b' = b,\ d' = d} \tag{3.3}$$

■恒等式の確認　新ブロックを順に追うと、次のようになる。

1. $\mathrm{CX}(d \to c)$：$c_1 = c \oplus d$。
2. $\mathrm{CCX}(a, b \to d)$：$d_1 = d \oplus ab$。
3. $\mathrm{CCX}(d, c \to t)$：$t_1 = t \oplus d_1 c_1 = t \oplus (d \oplus ab)(c \oplus d)$。
4. $\mathrm{CX}(d \to c)$：ここで d の位置には d_1 があり、$c_2 = c_1 \oplus d_1 = c \oplus ab$。
5. $\mathrm{CCX}(a, b \to d)$：d が元に戻る。
6. $\mathrm{CCX}(d, c \to t)$：$t_2 = t_1 \oplus d(c \oplus ab)$。
7. $\mathrm{CX}(d \to t)$：$t' = t_2 \oplus d$。

t' を展開すると、$(d \oplus ab)(c \oplus d) = dc \oplus d \oplus abc \oplus abd$ と $d(c \oplus ab) = dc \oplus abd$ の和は $d \oplus abc$ に

なり、最後の CX で d が消えて $t' = t \oplus abc$ を得る。これは式 (3.3) に一致する。

公開作者の説明では、32 通りの基底割当て（借用線と標的の任意の初期値を含む）すべてで一致を全数検査している。このブロックは位相を持たない基底置換なので、基底での一致は任意の重ね合わせ、および他の線とのエンタングルメントに対しても成り立つ。10/28 公開回路では、ソース行（0 始まり）27939 と 29213 の 2 箇所で $(a, b, c, d, t) = (46, 17, 18, 0, 35)$ に適用され、無条件の Toffoli がちょうど 2 個減った。$Q = 63$ なので score は約 126 下がる（103,672.22 から 103,546.17）。

3.3.3　履歴キャッシュ

履歴キャッシュ（history cache）は、あとで再び必要になる Toffoli の積を捨てずに CX でコピーして保持し、再計算を CNOT に置き換える手法である。公開作者のノートブックは、2026 年 10 月 28 日に取得した 41 回転版でこれにより 23 個の Toffoli が CNOT になったと報告している（著者報告）。平均 Toffoli の推移（著者報告）では、「round13」の 1,392.09 から「history caches」の 1,368.0 へ下がっており、これが最大の単独段差である。

図 3.2 に、同じ積 $a \wedge b$ を 2 度使う場面での両方式を並べる。(a) の再計算では、最初の CCX で新しい線 w に積を書き、使った後に HMR と CZ で消す。後で同じ積が必要になると、2 個目の CCX で作り直す。(b) のキャッシュでは、最初の使用の後も w を消さずに残し、2 回目は CX で w から値を取り出す。この間 a, b が書き換えられないことが条件で、そのとき両者は同じ置換を与える（基底状態の全数列挙で確認）。取引は「Toffoli 1 個の削減」対「区間 $[t_3, t_5]$ のあいだ生きている量子ビット 1 本の増加」である。

この公開スナップショットでは、この置換で 23 個の Toffoli が CNOT になったと報告されている。

保持には量子ビットが要るので、キャッシュは幅との交換になる。10/28 公開回路の命令列を調べた一つの解析例では、再計算を伴う同じ積の候補を 289 組抽出した。対象回路の占有履歴に照らすと、これらは保持区間のどこかで占有 63 に達し、同じ幅のまま追加の線を保持する余地がなかった。この結果は当該命令列とスケジュールに限られる。

図 3.3 は、当該回路で観察された幅の問題を占有曲線で模式的に示す。保持区間が占有の台地（63）にかかると、w を足した占有は 64 になり、Q を増やさずには置けない。

そこで Q を増やしてキャッシュする取引を区間スケジューリング（早く終わるものから貪欲に選ぶ）で見積もった（推定）。

表 3.2　再計算キャッシュを Q の増加と交換した場合の見積もり（推定）

Q	キャッシュできる組	推定 score
64	3	約 104,998
65	5	約 106,508
66	7	約 108,015
68	10	約 111,084
70	12	約 114,211

このスケジュール推定の各点は例の公開版の score 103,546 を下回らなかった。$Q = 64$ の場合で

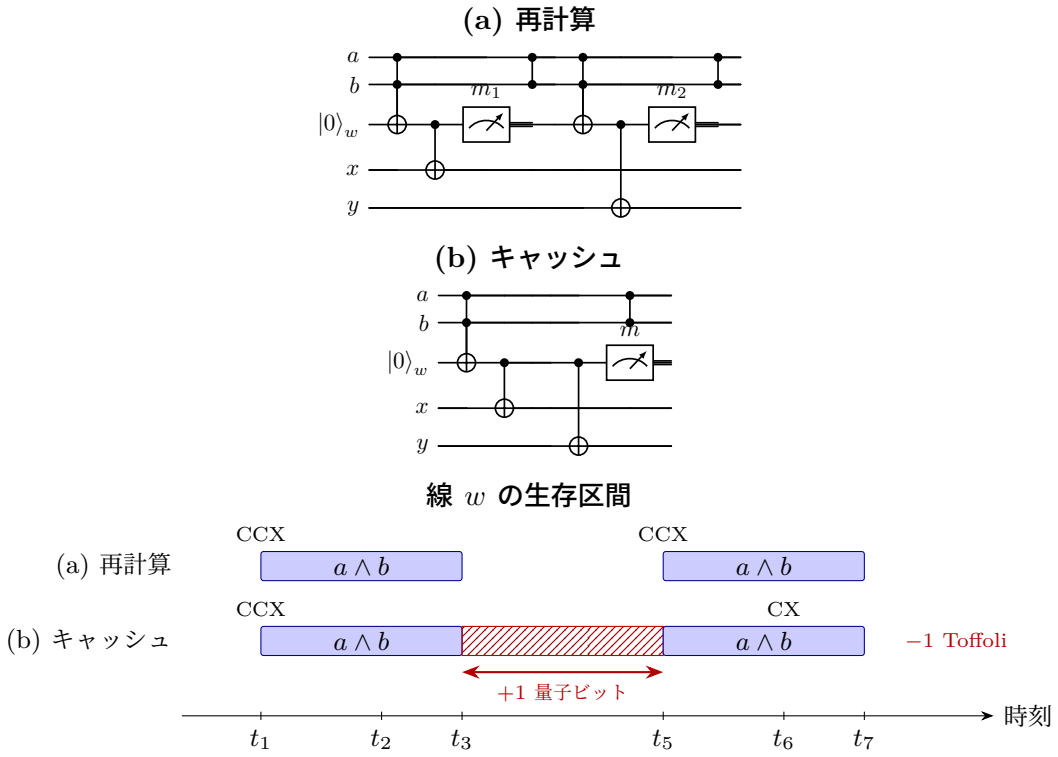


図 3.2　履歴キャッシュ。(a) は $a \wedge b$ を CCX で作り、使い、HMR と CZ で消し、後で 2 個目の CCX で作り直す。(b) は最初の使用後も w を残し、2 回目は CX で使う。図の x, y は積を使う側の標的の代役、HMR の結果 m が 1 のときだけ a, b に CZ を掛ける（古典制御は省略）。下段は w の生存区間で、斜線の部分が余分に生きる区間である。

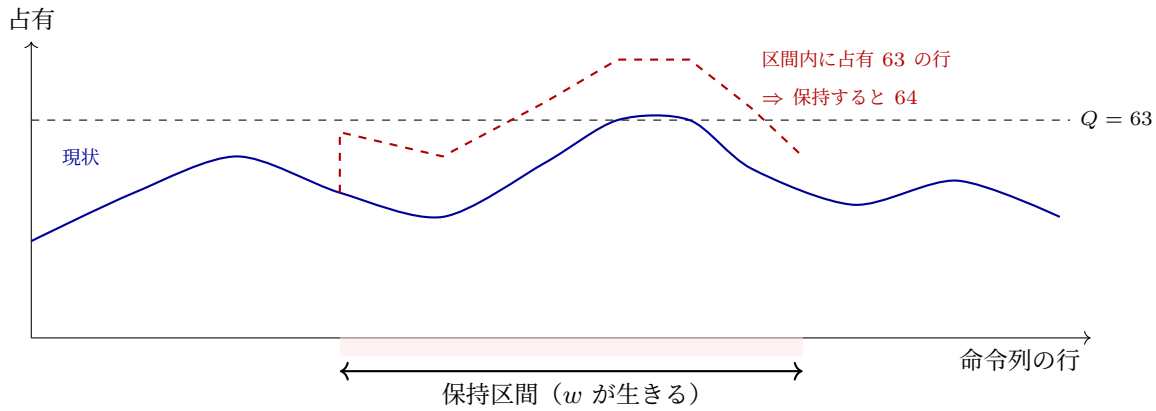


図 3.3　再計算の 2 回目の CCX を CX にするために w を保持する区間（下の矢印）と、占有曲線（模式図、縮尺は実測ではない）。実線が現状、破線が w を保持した場合の占有。289 組すべてで保持区間内のどこかに占有 63 の行があり、破線は $Q = 63$ を超える。

も約 25.7 Toffoli の追加削減が必要と見積もられたが、保持区間の重なりを含む当該命令列での推定であり、別方式の不可能性を示すものではない。

3.3.4　アフィンで置換できる Toffoli

Toffoli の制御の積が、生きている量子ビットと定数の GF(2) 上の線形結合（アフィン関数）として書けるなら、その Toffoli は CX に置き換えられる（常に 0 なら削除できる）。たとえば、制御 2 本が常に等しい値を持つ場合、積は片方のビットそのものになる。この置換は厳密で、CX は無料なので Toffoli が丸ごと消える。

図 3.4 に、置換が成り立つ 3 つの典型を示す。いずれも到達可能な状態の集合の上でだけ成り立つ恒等式であり、全入力で成り立つ置換ではない。状態は基底状態の置換にしか作用しないので、到達可能な基底状態が張る部分空間の上では、重ね合わせに対しても同じである。

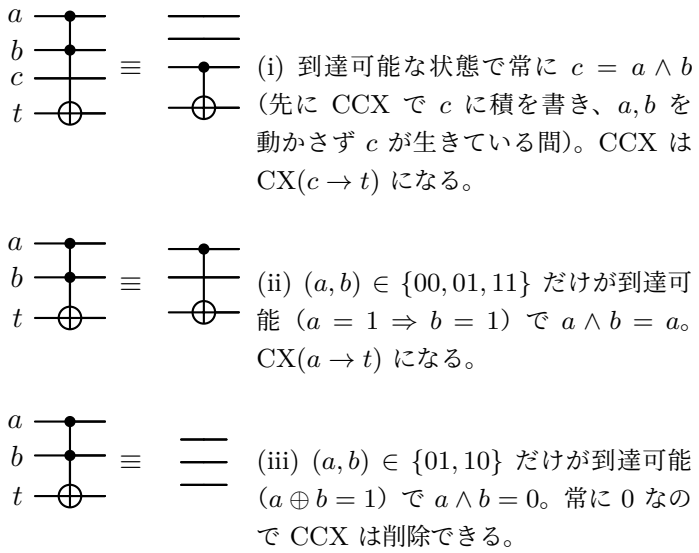


図 3.4　アフィンで置換できる Toffoli。到達可能な状態の集合の上でだけ成り立つ恒等式である。(i) は履歴キャッシュ（図 3.2）の裏返しで、保持した積を制御に使う。基底状態の列挙で、各到達可能集合の上の全状態について両辺が一致することを確認した。到達不能な状態（たとえば (i) で $c \neq a \wedge b$）では一致しない。

一般には、$a \wedge b$ が生きている量子ビットの XOR と定数の和に等しい（アフィンである）ことが、全到達可能状態で成り立てばよい。その成立を確かめるには、サンプルでなく全入力で調べる必要がある。

10/28 公開回路の末尾の後始末（命令列の 46,246 行以降）を除く Toffoli 様演算 1,140 個を、対象ベンチマークの入力全域で検査した例では、

- 入力 4,101 点のサンプルでは 31 件の候補が現れたが、全 3,133,452 入力ではすべて非線形で、まれな入力でだけ発火することが分かった。
- この取得版の対象範囲で追加候補は **0 件**だった。サンプルだけに基づく置換は誤りを招くため、回路置換を主張するにはその対象範囲での全入力検査が必要である。

CCX 1,140 個のうち、0 の標的に AND を計算するものが 779 個、0 でない値に足し込むもの（キャ

リーや累算）が 361 個である。

3.3.5　再計算とキャッシュの取引

再計算かキャッシュかは、次の 3 つを比べる問題である。

1. 再計算の Toffoli 数（キャッシュで節約できる量）。
2. 保持のために増える量子ビット数。1 本あたり約 26.5 Toffoli が上限。
3. 保持区間が占有ピークと重なるか。重なれば Q が直接増える。

10/28 公開回路の占有プロファイルでは 63 の台地が 3 つあった（第 4 章）。このスケジュールではキャッシュを追加しても幅を増やさずに保持できるのはピークの外に限られた。逆向きの取引（量子ビットを減らして再計算する）も同様で、係数表の列を早く消して必要なときに作り直すと、1 列あたり約 +4.3 Toffoli がかかる（推定）。係数 lookup を 2 分割する案は $\Delta C \approx +90\text{〜}200$ になり、$Q = 62$ の許容 +26.5 を超える（推定）。

3.3.6　条件付きブロックと、平均 Toffoli が静的な数より小さい理由

10/28 公開回路には PUSH_CONDITION が 128 対ある。ブロックの内側の演算は、条件が成り立つ（測定結果が 1 など）ときだけ実行される。内側には CCX 62、CCZ 48、CX 899、X 364、HMR 70 がある（実測、命令列の集計）。

Verifier は実行された Toffoli 様ゲートだけを平均に入れるので、条件が成立しないショットではブロック内の Toffoli が 0 になる。10/28 公開回路では、静的な Toffoli 様ゲートは 1,416 個、平均実行数は 1,348.50 個で、差は約 67.5 個である。条件の成立確率を 0.5 として重み付けした合計は 1,360.5 で、実測平均と近い。

46,246 行より前の 128 ブロックは、全入力の古典シミュレーションで値中立（実行の有無でビット値が変わらない）と確認した。値中立なブロックは位相の補正だけを担うので、実行しなくてもビットの正しさに影響しない。末尾の後始末（46,246 行以降）は測定結果で分岐する相補ブロックの対であり、値中立ではない。上記の厳密な書き換え検査はこの区間を除いたものである。

注意

HMR の乱数は回路の SHA-256 と salt で決まるので、同じ回路でも salt により平均 Toffoli が変わる。10/28 公開回路を同一条件で実行した報告値では、異なる salt での平均は約 1,348.49972、公開作者の値は 1,348.49915 だった。これはショット乱数を含む当該回路の再実行例である。

3.4　CX は無料だが、無害ではない

CX は式 (3.1) に入らないので、回路設計では自由に使える。パリティ（XOR）を作るには CX のネットワークを使えばよく、たとえば複数の量子ビットの XOR を 1 本に集めてから Toffoli の制御にすれば、アフィン形の積を直接作れる。10/28 公開回路の D 領域では、CCX 48 個のうち 36 個

が、片方の制御に約 15〜24 列の XOR を CX で集めてから使っている。

ただし CX が無料なのは費用だけである。

- 10/28 公開回路の CX は 29,815 個で、全演算の過半を占める。
- `num_ops` とシミュレーション時間は増え得る。10/28 公開回路を 4 worker で 200,540,928 ショット実行した記録では 135 秒だったが、実行環境の詳細が揃っていないため、これは性能比較ではなく一つの実行例である。
- Gray code 順の走査などでパリティ網の CX を共有すれば、演算数は減らせる。8 bit の係数表を項ごとに計算・復元する単純な合成例では 1,793 演算となるが、これは特定実装の演算数であり、Gray code 共有を含む一般的な最小値ではない。score は CX 数を直接数えない。

CX だけで量子ビットを減らせるかどうかは別問題で、生きている量子ビットの値ベクトルの GF(2) 階数で決まる。10/28 公開回路の占有 63 の行の大半（B と D の領域）では線形冗長性が 0 であり、CX だけで 63 から 62 にする道はない（実測、サンプル 4,101 点の階数。サンプルで独立なら全入力で独立）。

3.5　位相回転の最適化

invsqrt の出力は位相なので、算術で作った値をどう位相へ移すかが、回転の個数と費用を決める。

3.5.1　出力ビットごとの回転

参照回路は結果 y_1（40 ビット）をレジスタに作り、各ビット j に Z_POW(2^{j-f+1}) を 1 個ずつ掛け、その後で計算全体を逆に走らせる。最上位 3 ビットは $t = 1,\ 1/2,\ 1/4$ なので Z、S、T であり、残りの 37 個が一般角回転になる。この 37 個は、参照回路の一般角回転数 37（Toffoli 7,735、$C = 7{,}987.59$）と一致する。

公開ノートブックの説明（著者報告）では、44 回転版も「v を保持する代わりに、各ビットに 1 回ずつ回転する」。v のビットは算術で一度だけ計算され、回転の前後で消される。多項式の各項を直接位相に変換しているわけではない。図 3.5 がこの構成である。回転の個数は保持するビット数に等しい。

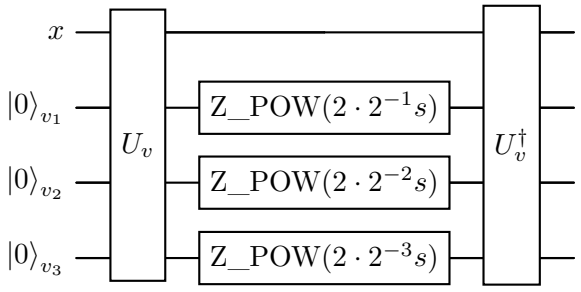


図 3.5　「v を保持して、出力ビットごとに回転する」構成（3 ビットの例）。U_v が $v = \sum_j v_j 2^{-j}$ を算術で一度だけ計算し、各ビット v_j に Z_POW($2 \cdot 2^{-j}s$) を 1 個ずつ掛け、$U_v^\dagger$ で消す。因子 2 は、Z_POW の引数が半回転単位であるという規約による（$t = 2$ が 1 周）。全体の位相は $\prod_j e^{i\pi \cdot 2 \cdot 2^{-j} s\, v_j} = e^{2\pi i\, s\, v}$ になる（全ビット列で確認）。

41 回転版では、さらに位相の直接付与（3.5.4 節）を加え、回転が 44 から 41 になった。

3.5.2　同じ線上の Z_POW の統合

Z_POW は同じ量子ビットに連続して掛ければ角度が加算される。$\mathrm{Z_POW}(t_1)\,\mathrm{Z_POW}(t_2) = \mathrm{Z_POW}(t_1 + t_2)$ である。合計の t が Clifford や T 角になれば費用が下がり、一般角のままでも n_{rot} が減って、式 (3.2) の ε が大きくなる。

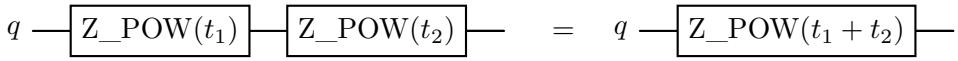


図 3.6　同じ線上の連続する Z_POW の統合。$\mathrm{diag}(1, e^{i\pi t_1})\,\mathrm{diag}(1, e^{i\pi t_2}) = \mathrm{diag}(1, e^{i\pi(t_1+t_2)})$ なので、これは厳密な恒等式である。

図 3.6 のとおり、角度は単に加算される。10/28 公開回路の 48 個の Z_POW（Clifford/T 7、一般角 41）は、17,600 行付近の 22 個（q59 と q40、Clifford/T を含む。direct phase taps と推定）と、28,000 行付近の 26 個（q40、出力ビット回転）として、別々の時点に分かれている。連続する Z_POW を統合する手法は一般に使えるが、10/28 公開回路に適用されているかは公開情報から確認できない。

3.5.3　角度の LP 較正

回転の個数 n_{rot} とは別に、回転角の微調整で最小余裕を最大にできる。入力 x ごとの符号付き誤差は、小さな角度変更に対して線形に変わるので、最小余裕を最大化する線形計画（LP）として解ける。公開ノートブックは 44 回転版で、この LP が出す最良の余裕が 47,884,703.7（2^{-64} turn 単位）、出荷済み角度が 47,884,702 で、差が 2 単位未満であると報告している。公開作者の報告では、44 回転版の出荷済み角度はこの LP の目的に対してほぼ最適だった（当該版・当該較正条件での報告）。

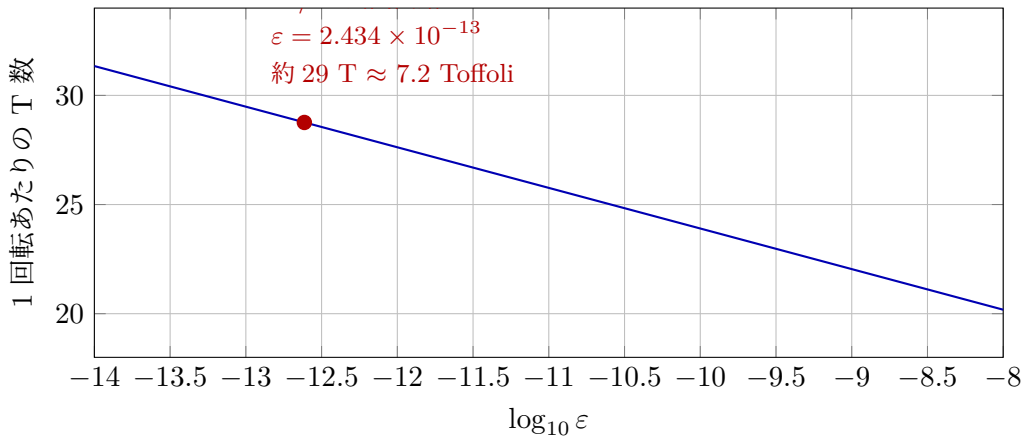


図 3.7　一般角回転 1 個あたりの T 数 $0.56\log_2(1/\varepsilon) + 5.3$。$\varepsilon$ の桁が 1 つ下がるごとに約 1.86 T（約 0.47 Toffoli）増えるだけで、対数的にしか効かない。

図 3.7 は式 (3.1) の回転項を $\log_{10}\varepsilon$ に対して描いたものである。最小余裕を改善しても 1 回転あたりの費用はゆっくりとしか下がらない。

図 3.8 は、式 (3.2) で使う最小絶対余裕を改善する目的を示す。回転角の局所補正を $\delta\theta_j$ とし、位

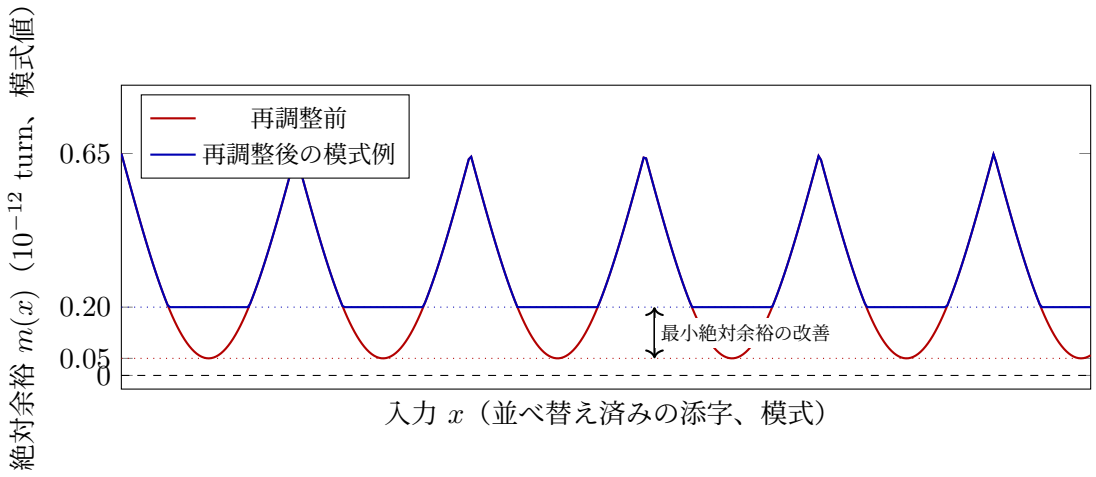


図 3.8　LP 較正の目的を示す模式図。縦軸は $m(x) = \mathrm{tol}(x) - |e(x)|$ という絶対余裕で、0 未満なら不合格となる。最小絶対余裕を大きくする目的を示す曲線であり、実測値や実際に実現した較正結果ではない。相対誤差比 $|e(x)|/\mathrm{tol}(x)$ の最大値を小さくする目的とは一般に同値ではない。

相の整数リフトと各回転の適用条件を固定した範囲で、符号付き誤差を

$$e_x(\delta\boldsymbol{\theta}) = e_x(0) + \sum_{j} B_{xj}\delta\theta_j$$

と書く。このとき、全入力について

$$-\mathrm{tol}(x) + m \le e_x(\delta\boldsymbol{\theta}) \le \mathrm{tol}(x) - m$$

を課し、m を最大化するのが絶対余裕の LP である。一方、$\rho_{\max} = \max_x |e_x|/\mathrm{tol}(x)$ の最小化は別の目的であり、許容値が入力によって変わる場合に $m_{\min} = \min_x(\mathrm{tol}(x) - |e_x|)$ の最大化を保証しない。Verifier の回転合成費用に直接入るのは後者である。最小余裕が 30 倍になっても、節約は invsqrt 費用の約 1.75% にとどまる（静的モデル、ノートブック報告）。

3.5.4　direct phase taps とガードビット補正

41 回転版は、多項式コアに次を加えたハイブリッドと報告されている。

- direct phase taps：算術の途中の量子ビットに、直接 Z_POW を掛けて位相を取り出す。解析では、17,601 行付近の 22 個の回転（q59 / q40）がこれに当たると推定している。
- ガードビット位相補正：上流 README によれば、直接計算した位相制御補正を 1 つ追加し、同じ 41 個の角度を再較正した。得られる最小余裕が大きくなり、回転合成費用が下がって、この Toffoli の追加を上回る。公開ノートブックは、20x の段差で位相余裕を使うのではなく上げた、と述べている（著者報告）。

後者は、費用式の構造を使う典型例である。補正のために Toffoli が増えても、回転 41 個分の単価が下がれば合計は減る。

図 3.9 の構成では、算術の途中にある積に直接位相を載せられる。補正の CZ は Clifford なので無料である。

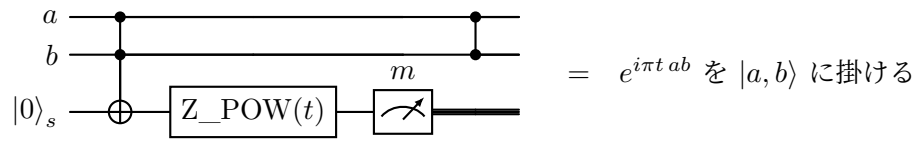


図 3.9　direct phase tap。積 $a \wedge b$ を CCX でスクラッチ線 s に書き、Z_POW を掛け、HMR（X 基底測定）で消す。結果 $m = 1$ のときだけ a, b に CZ を掛けて符号 $(-1)^{m\,ab}$ を打ち消す。正味の作用は位相 $e^{i\pi t\,ab}$ だけで、Toffoli 1 個と回転 1 個を使う。$m = 0, 1$ の両方で位相の一致を確認した。

3.5.5　Clifford と T の角度クラス

回転角を Clifford（1/2 の整数倍）または T（1/4 の整数倍）に収められれば、一般角費用（約 7.2 Toffoli）が 0 または 1/4 になる。参照回路でも最上位 3 ビットの 3 個がこれに当たる。10/28 公開回路の Z_POW 48 個のうち、Clifford/T は 7 個、一般角は 41 個である。角度を丸めると位相誤差が増えて最小余裕が減るので、丸めは余裕との交換になる。

3.6　量子ビット数を減らす技法

公開ノートブックの報告（著者報告）では、10/28 公開回路は 78、76、68、66、63 と量子ビット数を段階的に減らした。多項式と回転角は変えず、保存と消去だけを変えている。用いた手法は次のとおりである。

早期キャリー消去　加算のキャリーを完成した直後に測定ベース消去で消す。E 領域（Horner 算術）で HMR が 285 個あり、列をその場でオペランドや累算器として消費しながらキャリーを消す。

バッファ再利用　同じ量子ビットを時間をずらして別の役割に使う。q59 はタップ用スクラッチ、q40 は出力ビット回転用スクラッチである。Horner の累算器とキャリーは、係数表の列の量子ビットをその場で再利用している。

関数関係にある列の再計算　係数表の列が他の列の関数なら、保持せず必要なときに再計算する。G 領域では 9〜11 列を再生成して測定消去する。上流 README は、10 本の導関数 seed ビットを最後の位相補正まで保持し、7 個の共有非線形ヘルパーで退役させる構成も述べている。

ピーベリング（pebbling）　可逆計算で、中間値をいつ保持し、いつ消して再計算するかを、時間と空間の交換として決める一般的な枠組みである。公開資料にこの語は出てこない。公開回路での対応物は、作業プールの上限が 40 本に達したとき seed を HMR で消し、CZ で補正する運用である（解析での記述）。

図 3.10 は、量子ビットの生存区間（ガント形式）で、2 つの手法の効果を分けて示す。早期消去は同時に生きている値の数を減らし、再利用は異なる時期の値に同じ線を割り当てて線の本数 Q を減らす。消去を早めるだけでは、線を共有しない限り Q は減らない。

測定ベース消去（図 3.1）は、早期キャリー消去の部品である。

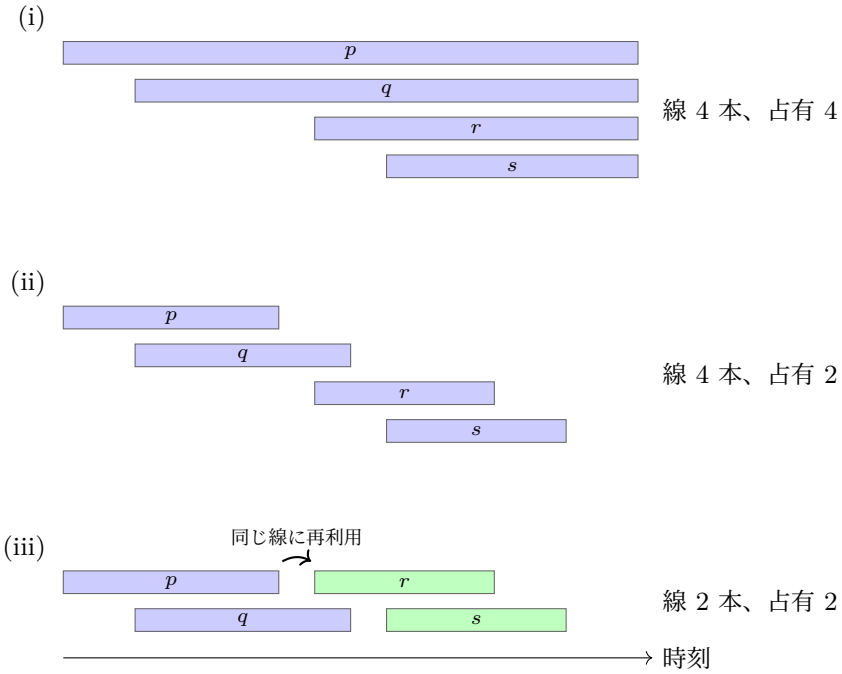


図 3.10　量子ビットの生存区間。(i) 値を使い終わっても消さずに最後まで残す。(ii) 使い終えた直後に測定ベース消去（図 3.1）で消すと、同時に生きる値は最大 2 個に減る。(iii) さらに p, r と q, s を同じ線に割り当てると、線の本数が 4 から 2 になる。

■ピーベリングの小さな例　入力 v_0 から $v_1 \to v_2 \to v_3 \to v_4$ と順に作る 4 ステップの鎖を考える。各値は、直前の値が置かれているときに限り、計算（ピーブルを置く）も消去（取り除く）もできる。最後に v_4 を出力へコピーし、中間値をすべて消す。図 3.11 の上段が Bennett の全保存で、すべてを保持して逆順に消す。下段は v_2 をチェックポイントにして v_1 を早く消し、後で v_1 を作り直す方法である。

表 3.3　4 ステップ鎖のピーベリング（図 3.11）

方式	最大ピーブル数	ゲート適用数	再計算
Bennett 全保存	4	8	0
チェックポイント v_2	3	10	+2（v_1 の再計算と再消去）

ピーブルは量子ビットを、再計算は Toffoli を意味する。この例では量子ビットが 1 本減る代わりに、ゲートが 2 個増える。式 (3.1) の交換比率では、再計算 1 回が Toffoli を何個使うかが 26.5 以下であれば得になる。

■10/28 公開回路での幅の観察　この命令列では実占有のピークが 63（宣言幅と一致）で、占有 63 の行は 15,619 行（全体の 33%）だった。占有 63 の区間は B、D、G の 3 つに現れた。この回路表現の占有プロファイルについての観察であり、別表現や全ての回路に対する幅の下界ではない。

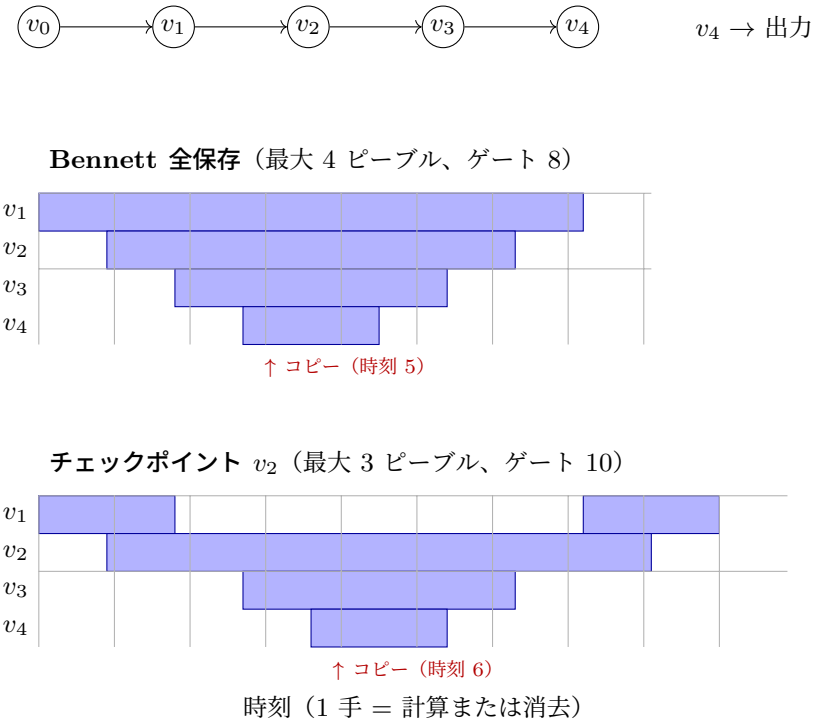


図 3.11　4 ステップ鎖のピーブリング。各行が値 v_r で、塗られた区間のあいだ量子ビットが生きている。上は全保存、下は v_2 をチェックポイントにした場合で、v_1 を 2 回計算する。規則（計算と消去は直前の値が置かれているときだけ）を満たすことを列挙で確認した。

表 3.4　幅を減らす技法の整理

技法	節約するもの	費用	10/28 公開回路
早期キャリー消去	同時に生きる値の数（占有）	ほぼなし（測定ベース消去は無料）	あり（E 領域 HMR 285 個）
バッファ再利用	線の本数 Q	値の生存区間が重ならないことが条件。Toffoli は増えない	あり（q59、q40 ほか）
関数関係にある列の再計算	保持する列の線	再計算の Toffoli（1 列あたり約 +4.3、推定）	あり（G 領域 9〜11 列）
ピーベリング（チェックポイント）	保持する中間値の線	再計算のゲート。長い鎖ほど増える	運用のみ（プール上限 40 本）
履歴キャッシュの逆（再計算で保持を避ける）	線	Toffoli	該当区間は既に採用

3.7　ルックアップ：パリティ積（ANF）型

区分多項式の係数は、入力の上位ビットで選ぶ表から引く。アドレスが A ビットなら表は 2^A エントリで、単項反復（unary iteration）による QROM は $2^A - 1$ 個の Toffoli を使う。10/28 公開回路は $A = 8$（256 エントリ）、出力 35 列で、これを 186 Toffoli で実装している（上流の証明書の値）。単項反復なら 255 個かかるので、約 73% である。

3.7.1　考え方

各出力列は、アドレスビットの Boolean 関数であり、代数的正規形（ANF）で書ける。ここでの要点は、次の 4 点である。

1. アドレスビットのアフィン関数同士の積（パリティ積）を、seed として新しい量子ビットに書く。1 個あたり Toffoli 1 個。
2. 各列は、アドレスビット、seed、先に作った列の XOR（CX で無料）に、少数の追加の積を足した形に分解する。
3. 作った列は後の列の材料として再利用できる。
4. 作業プールが上限に達したら、seed を HMR で消し、CZ で補正する。

10/28 公開回路の構成は、35 個の seed（各 1 CCX）と 35 本のデコーダ（各 4〜5 CCX、計 151）で、合計 186 である。列数が多くても、列を XOR で共有できれば Toffoli 数は列数に比例しない。単項反復の費用は列数に依存しないが、ANF 型は表の符号化に強く依存する。

3.7.2　ANF lookup の費用を見積もる

係数表をパリティ積に分解するときの Toffoli 数の上界は、候補となるアフィン積を作り、各出力列をそれらの積と生存中の線形量の XOR で表す GF(2) 線形代数問題として見積もれる。手順は、ランダムなアフィン形同士の積を P 個作り、各列を「生きている量子ビットのアフィン関数と積 r 個」で表して、GF(2) 消去で表せる最小の r を求めるものである。全アドレスで真理値表との一致を確認した（$A = 8$ で 256/256、$A = 9$ で 512/512）。結果（推定）を表 3.5 に示す。

表 3.5　ANF lookup の積数の推定例（上界。10/28 公開版の実値は 186）

	$A = 8$	$A = 9$
列（生、符号＋絶対値）	54	51
積の数（プール上限 40 本）	205	395
積の数（$P = 60$）	184	319
積の数（最良、$P = 100/130$）	174	262
単項反復 QROM（$2^A - 1$）	255	511
比（上限 40 本）	0.80	0.77

$A = 8$ での推定器は 205 で、公開版の 186 より約 10% 弱い。表の符号化が違う（生の 54 列に対して公開版は 35 列）ので、差は想定の範囲内である。$A = 9$ でこの公開版の費用水準に並ぶには、lookup 本体を約 335 以下にする必要があるが、この候補集合とプール制約に基づく推定は 395 だった。この探索結果は候補集合に依存する費用の目安であり、不可能性や一般的な下界を意味しない。プールを広げれば積は減るが、量子ビットが増えて $Q = 61$ の前提が崩れる。

3.8　signed-digit Horner

区分 2 次多項式は Horner 法 $v = C_0 + d\,(C_1 + C_2\,d)$ で評価する。signed-digit 表現では、乗数の桁を $\{-1, 0, +1\}$ にして、加算する行の数を減らす。公開ノートブックの説明（著者報告）によれば、10/28 公開回路もこれを用いる。10/28 公開回路の命令列では、E 領域（23,326〜28,000 行）が Horner 算術に当たり、Toffoli 様演算 350 個、HMR 285 個を含むと分類できる。ただし桁の符号化自体は公開資料から個別には確認できない。

参照回路では、乗数のビットを制御にした `_add_row`（AND、加算、AND の測定消去）を、δ のビット数だけ繰り返す。行数は δ のビット数に比例するので、桁表現と切り捨て位置が費用に直結する。このスナップショットを対象にした静的費用推定では、Horner の費用を部分積ビット数の定数倍（較正係数 1.78）と置いた（推定）。Horner の逆計算は、計算本体とほぼ同じ費用になる。v のビットは非線形関数なので、測定で消しても位相補正に再計算が要るためである。

Note

2 次式を部分積に展開して「書き込み一回型」にし、逆計算をなくす案も検討した。展開すると、誤差予算に収まる範囲で約 1,400 Toffoli となり、この取得版の Horner 推定値（749）より大きかった。Horner が $t_1 = C_2 d$ の直後に下位ビットを切り捨てるため、$C_2 d^2$ の部分積が約 500 から約 140 に縮むが、展開するとこの効果を失うためである。

3.9　精度と回路資源の取引

多項式次数・区間数・word 幅を減らすと算術や表は小さくなる一方、近似誤差は増え得る。invsqrt では近似誤差そのものに加えて、最小余裕が小さくなると回転合成費用が増える。dqi では復号失敗率の上限が $R(\varepsilon_+)$ に影響する。dfthcsos では係数再構成誤差のゲートを満たした上で λ_{eff} と回路資源を比べる。したがって、回路局所のゲート削減だけでなく、数値表現やアルゴリズムの選択も一緒に最適化する。

3.10　公開回路における位置づけのまとめ

表 3.6 に、手法ごとに、2026 年 10 月 28 日取得回路での確認状況と、その根拠および当該回路の検査例をまとめる。

表 3.6: 2026 年 10 月 28 日取得回路で確認した最適化手法の例

手法	10/28 公開回路	根拠と当該回路の検査例
AND の測定消去	あり	HMR 959 個（命令列集計）。参照回路の `del_and` も同じ原理。対象範囲で追加機会 0 件
ダーティ 5 → 4 CCX	あり（2 箇所）	公開作者の README 報告（行 27939、29213）。恒等式を確認。これは当該版の 2 箇所の例

次ページへ続く

表 3.6：最適化手法の例（続き）

手法	10/28 公開回路	根拠と当該回路の検査例
履歴キャッシュ	あり	公開ノートブックの報告（23 Toffoli を CNOT に）。抽出した候補 289 組はこのスケジュールで占有 63 に到達
アフィンで置換可能な Toffoli	対象範囲では追加なし	命令列に対する全入力検査。31 候補を全入力で調べた例ではすべて非線形
条件付きブロック	あり	PUSH_CONDITION 128 対。平均 1,348.50 対 静的 1,416。値中立を古典シミュレーションで確認
出力ビットごとの回転	あり	ノートブックの記述と q40 への 26 回転。—
LP 角度較正	あり（44 回転版で確認）	ノートブックの LP 再現（差 2 単位未満）。41 回転版は未再検査
direct phase taps	あり（推定）	q59/q40 の 22 回転（命令列から推定）。—
ガードビット補正	あり	上流 README。—
Z_POW の統合	不明	確認できる資料なし。—
早期キャリー消去・バッファ再利用	あり	HMR 285 個（E 領域）、q59/q40 の再利用。線形冗長 0（B。D は 88 行中 87 行）
列の再計算	あり	G 領域の 9〜11 列再生成。係数 lookup の 2 分割は $\Delta C \approx +90$〜200（推定）
ピーベリング	運用のみ	プール上限 40 本の seed 消去。—
パリティ積 lookup	あり	公開証明書：35 seed + 151 decoder = 186。$A = 9$ のモデル費用 395 は目安で、一般的な下界や不可能性の証明ではない（推定）
signed-digit Horner	あり（著者報告）	ノートブックの記述。展開案は約 1,400 Toffoli で悪化（推定）

3.11　本章のまとめ

- 公式費用 $Q \times C$ では、CX と Clifford 角は無料であり、てこは Q、実行 Toffoli、回転の個数と最小余裕である。1 量子ビット $\approx$ 26.5 Toffoli の交換比率が、取引の物差しになる。
- 10/28 公開回路は、測定ベース消去、ダーティ借用線による 5 → 4 CCX、履歴キャッシュ、条件付きブロック、LP 較正した出力ビット回転、パリティ積 lookup、signed-digit Horner を組み合わせている。
- 2026 年 10 月 28 日版の所定範囲に対する全入力検査例では、局所的な書き換え（CX 置換・削除、AND の測定消去、再計算キャッシュ）による追加の Toffoli 削減は見つからなかった。この結果は対象回路の対象命令範囲に限られ、他の回路に対する下界ではない。
- 最適化の判断には、対象のスコア式、精度条件、幅制約を合わせて見る必要がある。ここで扱った回路例は 2026 年 10 月 28 日取得版の歴史的スナップショットである。

演習：交換比率

$Q = 63$、$C = 1{,}643.59$ の回路が、量子ビットを 2 本減らして $Q = 61$ にするとき、$Q \times C$ を保つために許される C の増加量を求めよ。$Q = 62$ の場合の約 26.5 との関係を確認せよ。

第 4 章

invsqrt

この章で学ぶこと

- 逆平方根の位相が物理シミュレーションに現れる理由を知る。
- 参照回路と公開回路の正規化・近似・算術を比較する。
- 費用削減と精度条件の関係を説明する。

4.1 物理的背景

Coulomb ポテンシャルは粒子間距離 r の逆数に比例する。位置を格子点 ν, ν' で表すと距離の二乗は $x = |\nu - \nu'|^2$ なので、その値は $1/\sqrt{x}$ に比例する。ポテンシャルによる時間発展は位置基底で対角であり、各配置にエネルギー依存の位相を掛ければよい。符号・時間刻みなどを係数に含めたこの部分演算が背景である [1]。課題では次の正の位相規約を固定する。距離の計算やシミュレーション全体は提出対象に含まれない。

4.2 課題：位相 $1/\sqrt{x}$ を作る

公式 Verifier[2] の仕様は次のとおりである。

$$\boxed{|x\rangle\,|0\cdots0\rangle \;\longmapsto\; e^{2\pi i\,\beta/\sqrt{x}}\,|x\rangle\,|0\cdots0\rangle\,, \qquad \beta = 1,\ x \in [1,\ 3{,}133{,}452]} \tag{4.1}$$

- 入力 $x = |\nu - \nu'|^2$ は、各運動量成分が $n_p = 10$ ビットのときの値で、$\max x = 3(2^{10} - 2)^2 = 3{,}133{,}452$。レジスタ幅は $2n_p + 2 = 22$ ビット。
- x レジスタは相異なる量子ビットで構成し、最後に元の値に戻す。それ以外の量子ビットはすべて作業用で、$|0\rangle$ に戻さなければならない。
- 追跡した位相 θ（turn 単位）は、許容誤差 $\mathrm{tol}(x) = 2^{-24}\beta/\sqrt{x}$ 以内で $\beta/\sqrt{x}$ に一致する（mod 1）。すなわち**相対許容** 2^{-24} が全域で一定である。
- 位相は 2^{-64} turn 単位の固定小数点で比較する。許容誤差の最小値は $\mathrm{tol}(\max x) \approx 3.37 \times 10^{-11}$ turn である。

■ショットの取り方　各 x について 64 ショット実行し、合計は 3,133,452 × 64 = 200,540,928 ショットである。HMR と R の測定結果は、回路の `.kmb` 表現の SHA-256（`--salt` があれば併用）から決まる乱数で与えられる。同じ x を繰り返し、測定ベース消去のランダムな分岐を検査する。64 ショットは全分岐の列挙ではない。スモーク（$x = 1..64$）の費用と full の費用は一致しないので、score の比較には必ず full の結果を使う。

■位相の単位　`Z_POW(t)` は $\mathrm{diag}(1, e^{i\pi t})$ で、t は半回転単位である。$|1\rangle$ には $t/2$ turn が付く。$y/2^f$ turn の位相は、ビット j に 2^{j-f+1} 半回転の `Z_POW` を掛けて作る。

4.3　260 量子ビットの参照回路

参照回路の生成器は 公式配布物の `nr_reference_circuit.py`[2] である。Newton–Raphson（NR）法の 3 段階を、KickMix の算術で組み立てている。基準の実行結果は 260 量子ビット、Toffoli 7,735 個、$C = 7{,}987.59$、score 2,076,774.23 である（公式参照実装の既定設定）。

4.3.1　アルゴリズムの 3 段階

1. **正規化。**先頭のゼロのペアの個数 s を、単項（unary）で `flags`（$n_p = 10$ 本）に数える。x をその場で $x_{\mathrm{norm}} = x \cdot 2^{2s}$ にシフトし、$M = x_{\mathrm{norm}}/2^{2n_p} \in [1, 4)$ とする。$1/\sqrt{x} = 2^{s-n_p}/\sqrt{M}$。
2. **初期値（3 次の seed）。**上位 3 ビットで 4 つの区間 $[1, 1.5)$、$[1.5, 2)$、$[2, 3)$、$[3, 4)$ を選ぶ。係数は、2 つの AND ビットで制御した Clifford ネットワークで 4 本のレジスタ $c_0, \ldots, c_3$ に読み込む。$\delta = M - \{1, 1.5, 2, 3\}$ として、3 次多項式を Horner 法で評価する。

$$r_1 = c_2 - T(c_3\delta), \quad r_2 = c_1 - T(r_1\delta), \quad y_0 = c_0 - T(r_2\delta) \tag{4.2}$$

   ここで $T(\cdot)$ は出力レジスタの最下位より下の部分積ビットを捨てる列切り捨てである。
3. **Newton–Raphson 1 反復。**

$$p = T\Big(y_0\big(3 - T(M\,T(y_0^2))\big)\Big) \approx \frac{2}{\sqrt{M}}, \qquad y_1 = p \cdot 2^s \tag{4.3}$$

   最後に単項制御のシフトで逆正規化し、$y_1/2^F \approx 1/\sqrt{x}$（$F = \mathrm{result_bits} + n_p + 1$ ビットの小数）を得る。

4.3.2　回路全体の構造：計算、位相、逆計算

y_1 に `Z_POW` をビットごとに掛け、その後、計算全体を逆順に走らせて補助量子ビットを $|0\rangle$ に戻す。

$$\mathrm{compute}(y_1) \;\to\; \prod_j \mathrm{Z_POW}(y_{1,j}) \;\to\; \mathrm{compute}^{-1} \tag{4.4}$$

`_compute_y1` は `inverse` フラグで逆計算も同じ関数から出す。加算は減算に、`init_and` は `del_and`（測定ベース消去、無料）に変わり、順序のある段階は逆順に並ぶ。計算と逆計算で Toffoli がほぼ等しくなるため、7,735 は片方向 約 3.9k の 2 倍である（生成器の docstring が「方向あたり 約 3.9k Toffoli」と記している）。

4.3.3　レジスタ（既定の精度）

既定の精度は `Precision(delta=14,coef=17,square=28,product=28,result=29)` である。生成器の docstring では、モデルの最悪相対誤差は 2.84×10^{-8}（$2^{-24} = 5.96 \times 10^{-8}$）である。全域検証での最大誤差／許容は 0.4769 で、おおよそ一致する。

表 4.1　参照回路の明示的な量子ビット確保（`build_nr_circuit`）

レジスタ	役割	幅
`x`	入力（正規化後は x_{norm} を保持）	22
`y1`	結果 y_1（result_bits $+n_p+1$）	40
`flags`	先頭ゼロペア数の単項表現	10
`selectors`	区間選択（t_a、t_b、δ の 1/2 ビット）	3
`c0..c3`	係数、Horner の中間値（17 ビット ×4）	68
`square`	$T(y_0^2)$	28
`z`	My_0^2 と $3 - My_0^2$（product_bits +2）	30
`scratch`	部分積の AND を置く作業領域	30
	明示的な確保の合計	231

明示的な確保は 231 本である。公式の宣言は 260 本なので、残る約 29 本は組込みの加算器が内部で確保する補助と推定される（生成器のソースからは確認していない）。

4.3.4　主な部品

`_add_row`　制御ビット × オペランドのビットの AND を `scratch` に書き、加算し、AND を測定で消す。部分積 1 行ぶん。Horner、平方、$M\cdot$ 平方、$y_0 \cdot z$ のすべてがこれで組まれている。

`_normalize`　各 flag k について、上位のペアが 00 かを AND で検出し、偶数ビット列と奇数ビット列を制御付きで 1 段回転する。逆計算は逆順。

`_square`　対角項は CX で無料、交差項は 2 倍にして `_add_row` で足す。

`_correct`　$z \leftarrow 3 - z$ を 2 の補数で行う。

`_phase_by_bits`　y_1 の 40 ビットに、2^{j-f+1} 半回転の `Z_POW` を掛ける。上位 3 ビットが Z、S、T で、残る 37 個が一般角回転である。

4.3.5　回路の構成図

図 4.1 は、各レジスタが段階ごとに何をしているかを示す。参照回路の特徴は、結果 y_1（40 本）・係数（68 本）・平方（28 本）・z（30 本）を**同時に保持したまま**位相を掛け、その後で全体を鏡像に逆計算する点にある。保持する中間値がすべて量子ビットを占めるので、幅は 260 本に達する。

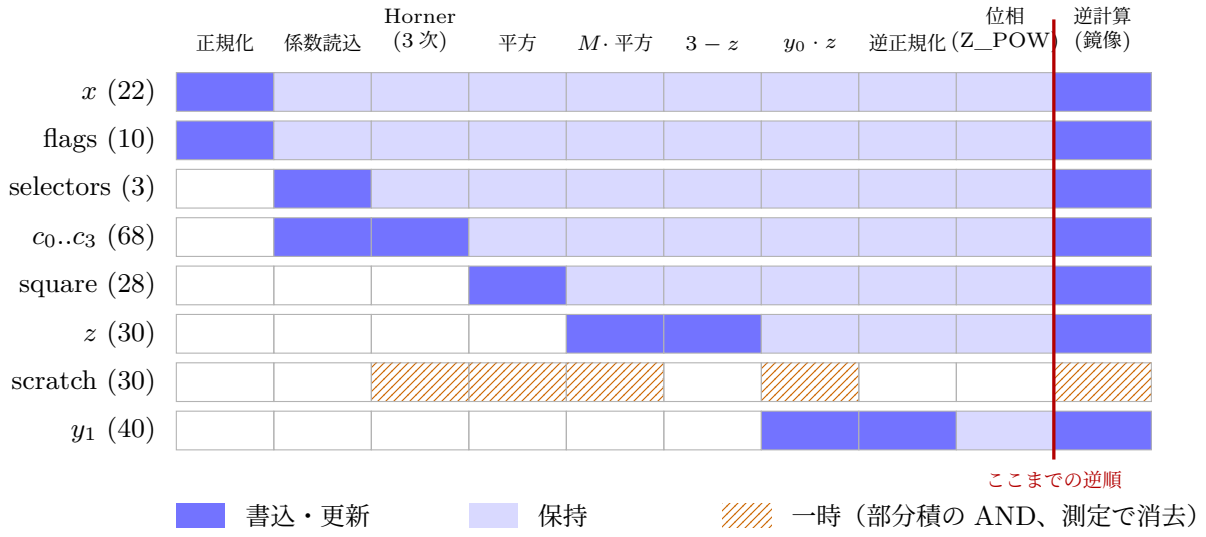


図 4.1　260 量子ビット参照回路のレジスタ使用の模式図（`nr_reference_circuit.py` の手順に基づく。括弧内は量子ビット数で、明示的な確保の合計は 231 本）。位相を掛ける時点で、すべての中間値を同時に保持している。

4.4　公開回路の構造

mrmeows が公開した 63 量子ビット系の回路を、busyaprime の改良と SomeJoint の配布物とともに比較する [3, 6]。以下の構成説明は 2026 年 10 月 28 日取得の 41 一般角回転版に対応する。図は各段階の役割を示し、特定のゲート分解が唯一であるとは主張しない。

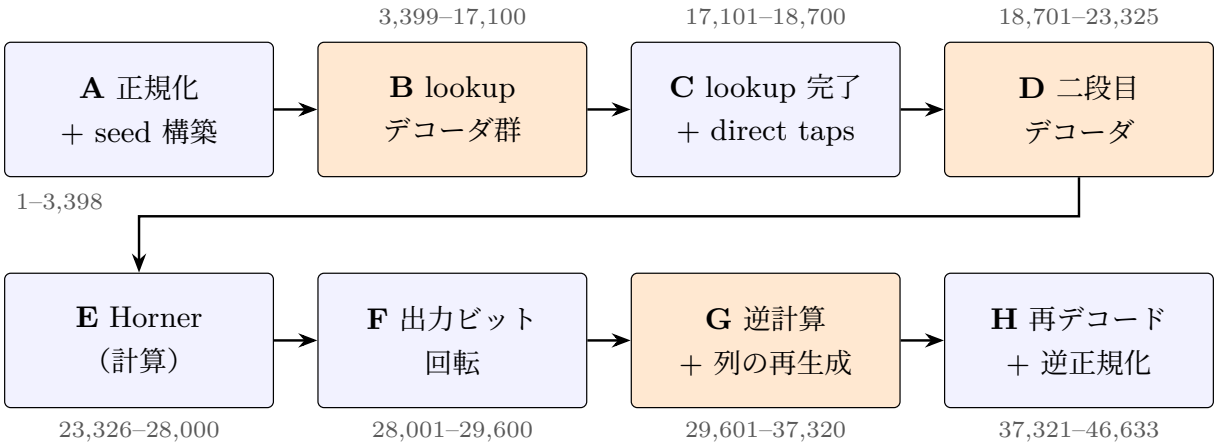


図 4.2　公開 63-Q 回路の段階 A–H（行番号は `invsqrt.kmx`）。橙色は係数と作業領域の使用が集中する B、D、G。入力レジスタは途中で正規化され、最後に元の値に戻る。

作業領域の再利用が参照回路との大きな違いである。参照回路と違い、係数・中間値・結果を別々のレジスタに持たず、同じプール（q22–q62）を段階ごとに使い回す。結果 v も保持せず、出力ビットごとに回転を掛けて消す。

4.5　区分多項式と Horner 法

$x = m4^{e}$、$1 \le m < 4$ とすれば $x^{-1/2} = 2^{-e} m^{-1/2}$ である。指数の偶奇と仮数の上位ビットから 8 ビットのアドレスを作り、256 区間の係数表を参照する。局所変数 d に対して $c_0 + d(c_1 + dc_2)$ を固定小数点で評価するため、Newton 反復を省ける。区間を増やすと近似次数を下げられるが、係数表は大きくなる。算術の各段で桁を切る場合、実数多項式の誤差だけでなく、実際の整数パイプラインを全有効入力で評価する必要がある。

4.6　lookup・算術・位相回転

係数ビットの表をパリティ積の共有で合成し、線形な組合せは CX で扱う。作業プールを係数、派生ビット、Horner 累算器の間で再利用し、確定した出力ビットから回転を掛けて消す。数値 $v/2^{f}$ を位相に変えるには、ビット v_j に Z_POW(2^{j-f+1}) を掛ける。算術キャリーなどの AND は測定と補正で消せるが、位相補正を残したまま入力と補助を復元する必要がある。

4.7　費用分析

表 4.2　invsqrt の歴史的比較。参照は既定設定。公開回路は全 3,133,452 入力各 64 ショット、$\beta = 1$、相対許容 2^{-24} の公開検証値を丸めたもの。平均は乱数条件で微小に変わる。

回路・取得版	Q	平均 Toffoli	C	QC
参照（Newton）	260	7,735	7,987.59	2,076,774.23
mrmeows 系（10 月 28 日）	63	1,348.50	1,643.59	103,546.17
busyaprime 改良（10 月 30 日）	63	1,329.50	1,624.59	102,349.24

参照との差は、幅で $260/63 \simeq 4.13$、換算費用で $7987.59/1624.59 \simeq 4.92$、積で約 20.29 倍となる。公開 2 版は一般角回転 41 個と T 回転 1 個を含む。10 月 28 日版の $\varepsilon \simeq 2.434 \times 10^{-13}$ なら、回転寄与は $[1 + 41(0.56 \log_2(1/\varepsilon) + 5.3)]/4 \simeq 295.09$ で、表の C を再計算できる。Newton の追加乗算を省くことと、中間値の保持を減らすことが異なる資源に効いている。

4.8　今後の最適化方向

区間数と次数、係数の符号化、signed-digit の乗算、精度配分、計算と再計算のスケジュールを同時に選ぶ余地がある。局所書き換えが見つからないことは、その表現に関する観察であって最適性の証明ではない。実際、同じ 63 量子ビットでも lookup と派生係数デコーダの再合成により費用が変化した。大きな改善を評価するには、生成した回路を全域の誤差検査に戻して比較する必要がある。

第 5 章

dqi

この章で学ぶこと

- 誤り訂正符号とシンドロームの役割を説明できる。
- 復号を用いて位相を付け、逆計算で作業領域を消す理由を説明できる。
- 量子ビット数、Toffoli 数と失敗率が費用に与える影響を比較できる。

5.1 エラー訂正と Reed–Solomon 符号

通信や記憶では、送った情報が雑音によって変化する。誤り訂正符号は情報に冗長性を加えて符号語に符号化し、受信語の検査で誤りを検出し、復号によって誤りを推定して訂正する。符号語どうしの最小距離が d なら、距離 $t = \lfloor (d-1)/2 \rfloor$ 以下の誤りは一意に訂正できる。距離は異なるシンボルの位置数であり、ビット数ではない。ここで扱うのは古典符号の復号を量子計算の一部として用いる方法であり、量子ハードウェアの物理エラーを訂正する量子誤り訂正とは異なる。

有限体 $\mathrm{GF}(q)$ は、加減乗算とゼロ以外による除算が閉じている q 個の元の集合である。ここで q は素数のべきであり、任意の整数を選べるわけではない。

■素数個の元を持つ例：GF(5)　$\mathrm{GF}(5) = \{0, 1, 2, 3, 4\}$ の演算は整数の和と積を 5 で割った余りで行い、$4+3=2$、$1-4=2$、$2\cdot 3=1$ となる。最後の式から $2^{-1}=3$ なので、例えば $4/2 = 4\cdot 3 = 2$ もこの集合の中で計算できる。「閉じている」とは、このように計算結果が集合の外に出ないという意味である。

■ビット列に対応する例：GF(4)　$\mathrm{GF}(4) = \{0, 1, a, a+1\}$ は $a^2 = a+1$ として作る。係数は $\mathrm{GF}(2)$ 上で計算するので、例えば $(a+1)+a=1$、$a(a+1)=1$ である。例えば $0, 1, a, a+1$ をそれぞれ $00, 01, 10, 11$ と表せば、$(a+1)+a=1$ は $11 \,\mathrm{XOR}\, 10 = 01$ に対応する。元を 2 ビットで表すと加算は XOR になり、この体は整数を 4 で割った余りの演算ではない（後者では $2\cdot 2=0$ となり、ゼロでない元に逆元が存在しない）。

$\mathrm{GF}(2^{12})$ の 1 シンボルは 12 ビットで表せ、加算はビットごとの XOR である。Reed–Solomon 符号では、次数が k 未満の情報多項式を異なる m 個の体の元で評価し、その値を符号語とする。非ゼロの位置ごとに固定の重みを掛けた一般化 Reed–Solomon 符号も同じ距離を持ち、最小距離は $m-k+1$ である。本課題では $m=4095$、$k=4025$、$d=71$、$t=35$ で、最大 35 シンボルの誤り

を訂正できる。仕様の $n = 70$ はシンドロームのシンボル数であり、符号の次元 k ではない。

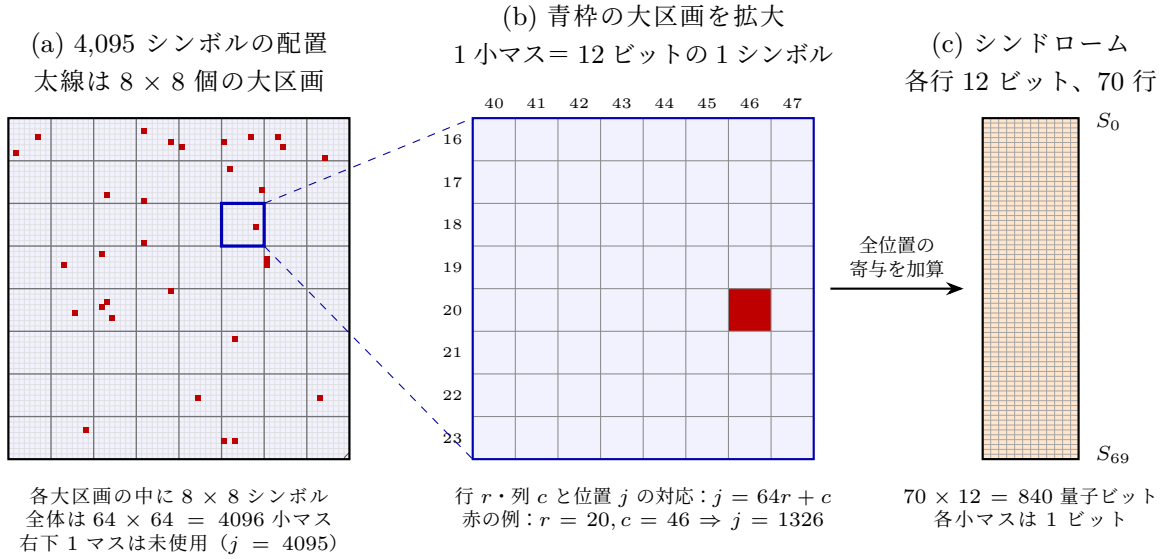
5.2　シンドローム

パリティ検査行列 H は符号語 c が $Hc = 0$ を満たすように定める。受信語 $y = c + e$ のシンドロームは $S = Hy = He$ である。誤りがなければゼロになり、元の情報そのものではなく誤りに関する情報を表す。重みが t 以下の 2 つの誤りが同じシンドロームを持つなら、その差は重み $2t < d$ の非ゼロ符号語となり矛盾する。したがってこの範囲で誤りは一意に決まる。

原始元を α とし、位置 j の誤りを e_j とする。以下の検査行列を使うと、符号は H の核であり、評価表現では位置ごとに重みを掛けた一般化 Reed–Solomon 符号となる。課題の 70 個の検査値は

$$S_i = \sum_{j=0}^{4094} e_j \alpha^{(i+1)j}, \quad i = 0, \dots, 69$$

である。1 シンボルだけに非ゼロの誤り e があり、その位置を $X = \alpha^j$ と書くと、$S_0 = eX$、$S_1 = eX^2$ である。$S_0 \neq 0$ なので体の除算で $X = S_1/S_0$、$e = S_0/X$ と復元できる。複数誤りでは各位置が和に混ざるため、後述する誤り位置多項式を用いる。



$$S_i = \sum_{j=0}^{4094} e_j \alpha^{(i+1)j} \quad (i = 0, \dots, 69)$$

e_{1326} も、全 70 シンドロームに $e_{1326}\alpha^{(i+1)1326}$ を加える。

大区画ごとに一つの S_i を作るのではない。すべての位置の有限体上の和である。

図 5.1　誤り配置の大区画・小マスとシンドロームの関係。左と中央の小マスは 1 シンボル（12 ビット）、右の小マスは 1 ビットを表す。赤は非ゼロの誤りの配置例である。64 × 64 という並べ方は説明用で、符号の位置は一次元の j で指定する。回路の入力は右のシンドロームであり、左の誤り一覧ではない。

5.3　DQI と課題の範囲

DQI（Decoded Quantum Interferometry）は、古典的な復号を量子干渉に組み込み、最適化問題のよい解を得ようとするアルゴリズムである。Optimal Polynomial Intersection（OPI）では、各 $y \in \mathrm{GF}(q)^*$ に許容集合 F_y が与えられ、次数が n 未満の多項式 Q について $Q(y) \in F_y$ となる位置を最大化する。制約行列を $B_{j,i} = \alpha^{(i+1)j}$ と書くと、シンドロームは $B^{\mathsf{T}} e$ であり、復号する符号は双対符号 $C^{\perp} = \ker B^{\mathsf{T}}$ である。Vandermonde 型の B によりこれは一般化 Reed–Solomon 符号となる [8]。DQI はこの双対符号の復号を干渉に利用する。課題は OPI 全体を解く回路ではなく、その中の位相補正である。

入力は 70 シンボル、すなわち 840 量子ビットのシンドローム S と、各位置 12 ビットの古典測定記録 M_j である。シンドローム生成は対象外で、内部で誤り e_j を復号し、$\phi(S, M) = \sum_j \mathrm{popcount}(e_j \wedge M_j) \bmod 2$ の位相を付ける。終了時には S を元に戻し、補助量子ビットをすべて $|0\rangle$ に戻す。Z_{POW} と指定外レジスタは使えない。

5.4　Euclid 法から復号へ

整数の Euclid 法では $252 = 105 \cdot 2 + 42$、$105 = 42 \cdot 2 + 21$、$42 = 21 \cdot 2 + 0$ より最大公約数は 21 となる。拡張 Euclid 法はさらに各余りを元の 2 数の整数係数による和として追跡する。体上の多項式でも、先頭係数で割って次数を下げる同様の除算ができ、係数の組合せを追跡できる。以下ではこの仕組みを鍵方程式に適用する。

符号長 $m = 4095$、訂正数 $t = 35$ とし、誤り位置を $X_k = \alpha^{j_k}$、値を e_k、誤り数を $\nu \le t$ とする。$S(x) = \sum_{i=0}^{2t-1} S_i x^i$ と書く。

5.4.1　小さい体でシンドロームから誤りを復元する

GF(5) 上で最大 1 シンボル誤りを復号する小さな例を考える。位置を $X = 1, 2, 3, 4$ とし、受け取った検査値が $S_0 = 1$、$S_1 = 2$ だったとする。求めたいのは「どの位置の値が、いくつ変わったか」であり、その答えはまだ分からないとする。シンドローム多項式は $S(x) = 1 + 2x$ である（以下すべて係数は 5 を法とする）。$\sigma(x)$ は誤り位置の逆数を根に持つ多項式、$\Omega(x)$ は誤り値を取り出すための多項式で、両者は $\sigma S \equiv \Omega \pmod{x^2}$ を満たす。まず Euclid 法でこの 2 多項式を作り、次に根と比から位置・値を読む。x^2 と $S(x)$ の多項式 Euclid 除算は

$$x^2 = (3x + 1)(1 + 2x) + 4.$$

この割り算では、最初に $x^2/(2x) = 3x$ を商に選ぶ（$2^{-1} = 3$）。$x^2 - 3x(1 + 2x) = 2x$ となり、次に商へ 1 を足すと $2x - (1 + 2x) = 4$ が残る。余り 4 は次数 $0 < t$ だから停止する。この式を並べ替えると $4 = x^2 - (3x + 1)S(x)$ であり、鍵方程式の候補は $\Omega = 4$、$\sigma = -(3x + 1) = 4 + 2x$ となる。定数項が 1 になるよう両方に $4^{-1} = 4$ を掛ければ

$$\sigma(x) = 1 + 3x = 1 - 2x, \qquad \Omega(x) = 1.$$

根は $x = 3 = 2^{-1}$ だから誤り位置は $X = 2$ と分かる。また $\sigma' = 3$ なので Forney の式より $e = -\Omega/\sigma' = -1/3 = 3$ と復元する。得られた答えは「$X = 2$ の位置に誤り値 $e = 3$」である。検

算すると $eX = 3 \cdot 2 = 1 = S_0$、$eX^2 = 3 \cdot 4 = 2 = S_1$ となり、元のシンドロームを再現する。受信語が手元にある通常の訂正なら、その位置から 3 を引く。dqi ではこの誤り値を位相補正に使う。実課題の体は $\mathrm{GF}(2^{12})$、$t = 35$ であり、この例と規模も標数も異なるが、多項式除算と鍵方程式の論理は同じである。

5.5　鍵方程式

誤り位置多項式と誤り評価多項式を

$$\sigma(x) = \prod_{k=1}^{\nu} (1 - X_k x), \qquad \Omega(x) = \sum_{k=1}^{\nu} e_k X_k \prod_{l \neq k} (1 - X_l x)$$

と定める。σ の根 X_k^{-1} が誤りの位置を、Ω が誤りの値を表す。$S(x) = \sum_k e_k X_k \sum_{i<2t} (X_k x)^i$ に $\sigma(x)$ を掛けると、$\sum_i (X_k x)^i (1 - X_k x) = 1 - (X_k x)^{2t}$ なので

$$\sigma(x)\, S(x) \equiv \Omega(x) \pmod{x^{2t}}, \qquad \deg \sigma \le t, \quad \deg \Omega < t$$

が成り立つ。これを**鍵方程式**と呼ぶ。復号は、S から鍵方程式を満たす (σ, Ω) を求めることに帰着する。

5.6　拡張 Euclid 法

拡張 Euclid 法は、2 つの多項式 r_{-1}, r_0 の最大公約式を、割り算の繰り返しで求める方法である。

$$r_{i-1} = q_i\, r_i + r_{i+1}, \qquad \deg r_{i+1} < \deg r_i .$$

同時に、各余り r_i を元の多項式の組み合わせ $r_i = u_i\, r_{-1} + v_i\, r_0$ で表す係数 u_i, v_i を、$u_{i+1} = u_{i-1} - q_i u_i$、$v_{i+1} = v_{i-1} - q_i v_i$ で更新する。

Sugiyama らは、$r_{-1} = x^{2t}$、$r_0 = S(x)$ として拡張 Euclid 法を走らせ、余りの次数が初めて t 未満になったところで止めると、v_i と r_i が定数倍を除いて σ と Ω になることを示した [9]。$r_i = u_i x^{2t} + v_i S$ を x^{2t} で割った余りを見れば、これは鍵方程式そのものである。

古典の計算機では、割り算の回数も、各商 q_i の次数も、入力ごとに違ってよい。しかし量子回路は、重ね合わせのすべての入力に同じゲート列を当てる必要がある。また、途中の商や余りを捨てずに、逆計算で消せる形で進める必要がある。

5.6.1　同期した拡張 Euclid 法

同期した拡張 Euclid 法は、入力によって進み方が違う拡張 Euclid 法を、固定回数の同じ形の反復に書き直したものである。参照回路では、1 回の反復は 4 つの小段階のどれか一つを進める。

0. 2 つの多項式の次数を比べ、割り算が要るかを判定する。
1. 割る側 B の先頭係数の位置をそろえる（係数列のシフト）。
2. B に付随する係数多項式の位置をそろえる。
3. 先頭係数の比 $q = \mathrm{lead}(B)/\mathrm{lead}(A)$ を計算し、$B \leftarrow B - q\, x^d A$ を 1 項ずつ行う。終わったら A と B を入れ替える。

今どの小段階にいるかは 2 ビットのカウンタが持ち、小段階が終わったかどうかは 1 ビットのフラグが持つ。各反復は、すべての小段階の操作を「カウンタがその値のときだけ働く」制御付きで並べたもので、入力に関係なく同じゲート列になる。次数は量子レジスタ（n_A, n_B, n_a, n_b, n_q）に持ち、比較や更新も可逆な演算で行う。復号が終わった入力では、停止カウンタが 0 でなくなり、残りの反復は何もしない。反復回数は最悪の入力に合わせて $3(n+1)+9 = 222$ に固定してある。

逆計算は、同じ 222 回の反復を逆順に、各操作の逆で実行する。途中の商は保存しない。各反復は可逆なので、終わりの状態から始めの状態（シンドローム）を復元できる。

5.6.2　Proos–Zalka 型

Proos と Zalka は、楕円曲線の離散対数を解く Shor のアルゴリズムの回路で、整数の拡張 Euclid 法を可逆にした [7]。その要点は 2 つある。

- **同期**：入力によって違う割り算の進み方を、上のように固定回数の反復と制御ビットで表す。
- **レジスタの共有**：余り r_i が小さくなるにつれて、係数 v_i は大きくなる。両者の長さの和はほぼ一定なので、1 本のレジスタに両方を詰めて置ける。

参照回路はこれを多項式に移したもので、余りと係数多項式を、長さ $n+3 = 73$ 個の係数の列（`reg_a`、`reg_b`、各係数 12 量子ビット）に一緒に詰める。`reg_a` の大部分には、入力のシンドロームレジスタ $S_0, \ldots, S_{69}$ をそのまま使う。`reg_b` は新しい作業領域で、参照回路の 1,893 量子ビットの多くはこの係数の列と次数のレジスタである。

5.7　Chien 探索

σ が求まると、誤りの位置は $\sigma(X^{-1}) = 0$ となる位置 $X = \alpha^j$ である。Chien 探索は、これを $j = 0, 1, \ldots, m-1$ のすべてについて順に調べる方法である [10]。$\sigma(x) = \sum_i \sigma_i x^i$ の各項を別々のレジスタ r_i に持ち、位置を 1 つ進めるごとに r_i に定数 α^{-i} を掛ける。

$$r_i^{(j)} = \sigma_i \alpha^{-ij}, \qquad r_i^{(j+1)} = \alpha^{-i}\, r_i^{(j)}, \qquad \sigma(\alpha^{-j}) = \sum_i r_i^{(j)}.$$

量子回路で重要なのは、$\mathrm{GF}(2^{12})$ の**定数**倍が 12 ビットの上の $\mathrm{GF}(2)$ 線形写像であり、CX だけで書けることである。項の和も XOR なので、位置を進める操作と σ の値の計算に Toffoli は要らない。参照回路では、Ω の各項にも同じように定数を掛けて、各位置での Ω の値を同時に作る。位置ごとに、σ の値が 0 かを判定するビット（`is_sigma_zero`）を立てる。

5.8　Forney の式

誤りの位置 X_k が分かると、その値は Forney の式で求まる [11]。σ の形式微分 $\sigma'(x)$ を X_k^{-1} で評価すると $\sigma'(X_k^{-1}) = -X_k \prod_{l \neq k}(1 - X_l X_k^{-1})$、$\Omega$ の定義から $\Omega(X_k^{-1}) = e_k X_k \prod_{l \neq k}(1 - X_l X_k^{-1})$ なので、

$$e_k = -\frac{\Omega(X_k^{-1})}{\sigma'(X_k^{-1})}$$

となる。標数 2 の体では $-1 = 1$ なので、符号は消える。さらに標数 2 では、形式微分で偶数次の項が消え、$x\,\sigma'(x)$ は σ の奇数次の項の和に等しい。参照回路はこれを使い、Chien 探索の各項を偶数次と奇数次に分けて足すことで、σ と（定数倍を除いた）σ' の値を同時に作る。残る定数倍は Ω 側の定数倍に畳み込んである。

位置ごとの処理は次のとおりである。

1. σ' の値の逆元を計算する（Itoh–Tsujii 法、315 Toffoli）。
2. σ の値が 0 のときだけ Ω の値を取り出し、逆元との積の各ビットと測定結果 M_j の内積で位相 $(-1)^{\mathrm{popcount}(e_j \wedge M_j)}$ を付ける。
3. 逆元と作業ビットを逆計算で消し、次の位置に進む。

誤り値 e_j そのものはレジスタに書かず、位相として直接付ける。また、$M_j = 0$ の位置は位相が変わらないので、古典ビットの条件で処理全体を飛ばす。逆元の計算は 4,095 位置のそれぞれで行うので、この段階の費用は位置の数に比例する。

5.8.1　回路の構成図

図 5.2 は、参照回路を「シンドローム」と「ラウンド（同期した拡張 Euclid 法の 1 反復）」の単位で抽象化したものである。222 ラウンドの前進でシンドロームを σ と Ω に変え、4,095 位置の走査で位相を付け、同じ 222 ラウンドを逆にたどってシンドロームに戻す。各ラウンドは同じゲート列で、カウンタの値に応じて 4 つの小段階のどれか 1 つだけが実際に働く。

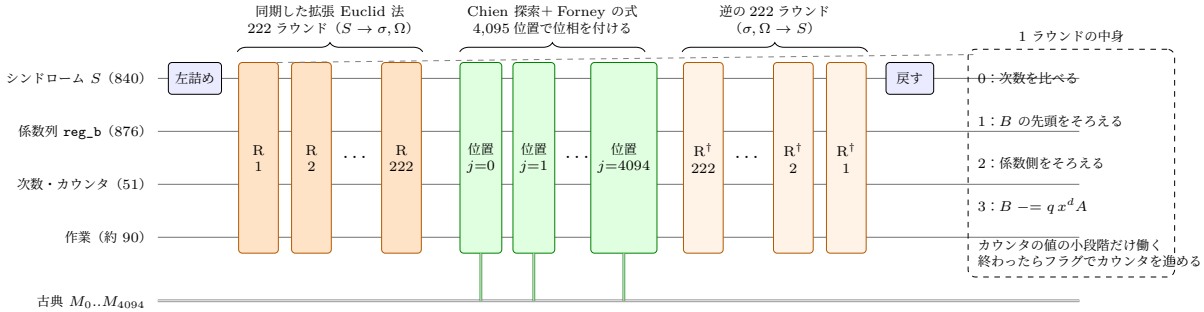


図 5.2　dqi 参照回路の抽象化した構成図。橙はラウンド（$\mathrm{R}^\dagger$ は逆）、緑は位置ごとの Chien 探索と Forney の式で、古典の測定結果 M_j（二重線）が位相の向きを決める。Toffoli は前進 222 ラウンドで 2,109,450、位置の走査で 1,444,737、逆の 222 ラウンドで 2,109,450、左詰めと戻しで各 30,912（命令列の集計）。

5.9　逆計算して S を復元する理由

通常の通信では復号した誤りを受信語から除いて訂正する。この課題では受信語の訂正も誤り一覧の出力も要求しない。復号結果を使って測定記録との内積の位相を付ける。概念的には、復号の可逆回路を U として

$$|S\rangle|0\rangle \xrightarrow{U} |S\rangle|e(S)\rangle \xrightarrow{\text{位相}} (-1)^{\phi(S,M)}|S\rangle|e(S)\rangle \xrightarrow{U^\dagger} (-1)^{\phi(S,M)}|S\rangle|0\rangle.$$

処理の流れは「S → 拡張 Euclid 法で σ, Ω → Chien 探索と Forney の式で誤り値 → M に応じた位相 → 復号の逆計算 → S とゼロの作業領域」である。図 5.2 は参照回路のこの流れを示す。実装が入力レジスタを一時的に書き換える場合も、逆計算で元の S を復元する。逆計算は通信中に生じた誤りを元に戻す操作ではなく、復号に使ったレジスタと補助量子ビットを片付ける操作である。位相は残るので、誤り位置と値を消しても目的は達成される。途中情報を残せば、シンドロームの重ね合わせと作業領域がもつれ、所望の位相オラクルとは異なる状態になる。

5.10　参照回路と費用の考え方

参照回路は同期した拡張 Euclid 法と Chien 探索、Forney の式を用いる。前進と逆計算の双方で有限体の乗算や制御付き更新を繰り返すため、復号が高コストになる。走査では各位置の根判定や逆元計算にも費用がかかる。定数倍は線形演算として CX で実装できるが、一般の体の乗算や逆元は非線形演算を要する。

スコアは小さいほどよく、$R(\varepsilon_+)Q^2T$ である。Q は量子ビット数、T はショットあたり平均 Toffoli 様演算数、ε_+ は値または位相の失敗率に対する Clopper–Pearson 片側信頼上限で、信頼度は $1-2^{-128}$、合格には $\varepsilon_+ < 1/16$ が必要である。値エラーは S または補助量子ビットが正しく戻らない場合、位相エラーはそれ以外で位相が異なる場合である。既定の検証は 102,400 ショットである。

$$R(\varepsilon) = \frac{\ln 2}{-\ln\left(\frac{1}{2} + 2\sqrt{\varepsilon}\right)}.$$

Q は 2 乗で効き、840 入力量子ビットに加える作業領域を減らす意義は大きい。ただし作業領域の節約が演算数を増やす場合は積全体で比較する。近似復号は T や Q を減らせる可能性がある一方、失敗が増えると R が悪化し、閾値を超えれば不合格になる。既定の 102,400 ショットでは失敗ゼロでも $\varepsilon_+ \simeq 8.66 \times 10^{-4}$、$R \simeq 1.19126 > 1$ であり、乱数条件により評価は変動する。

2026 年 10 月 30 日に取得した公開資料には、SomeJoint 配布の 889 量子ビット回路（出力 ID `team-1028-narrow240`）がある [6]。その公開タイトルは「third research team's 889-qubit decoder」と説明するが、デコーダ生成器のソースは確認できず、具体的な復号法は特定できない。883 量子ビット版について公開 README が述べる Schur 連分数デコーダという作者報告や、897 量子ビット版の INFO_TAG 段階名は、889 量子ビット版の復号法を確定する根拠にはならない [3, 6]。

表 5.1 は、参照回路とこの取得版を、復号方式・線形演算・作業領域・位相付与の 4 点で比べる。「確定」は回路仕様・静的命令列・公式 Verifier から直接確認できる事項、「作者報告」は公開タイトルや README の主張、「推定」は式からの算術例、「未確認」は取得資料では判断できない事項を示す。

表 5.1　参照回路（1,893 量子ビット）と 2026 年 10 月 30 日取得の 889 量子ビット公開回路

観点	参照回路（Verifier 仕様と回路構成）	SomeJoint 889Q（2026-10-30 取得）
復号方式	確定：同期した拡張 Euclid 法（222 ラウンド）で σ, Ω を求め、Chien 探索と Forney の式で誤り値を扱い、最後に逆計算する。回路構成と論文に基づく。	未確認：生成器のソースを確認できず、具体的な復号法は特定できない。静的回路の INFO_TAG にある段階名だけでは、数学的な復号方式や更新式は分からない。
線形演算	確定：GF(2^{12}) の定数倍や XOR は GF(2) 上の線形写像なので CX で実装できる。一方、一般の体乗算は非線形演算を要する。	確定（静的集計）：CX 53,275,774 個、Toffoli 様演算 456,456 個。未確認：この集計だけでは、どの線形変換を共有・合成したかは分からない。
作業領域	確定：入力 S は $70 \times 12 = 840$ 量子ビット。全体 $Q = 1{,}893$ との差 1,053 量子ビットが追加作業領域で、係数列や次数などを保持する。	確定：同じ 840 入力量子ビットに対し $Q = 889$、追加作業領域は 49 量子ビット。未確認：どの中間値をどのように詰めたか。
位相付与	確定：Chien/Forney の結果と古典測定記録 M_j から誤り値を位相へ直接反映し、誤り値レジスタを出力しない。$M_j = 0$ の位置は条件付きで処理を飛ばす。	確定（既定 salt・102,400 shots の公式検証）：値エラー 0 件、位相エラー 1 件。確定（静的集計）：PUSH_CONDITION 29,282 対。未確認：位相の計算法と個別条件の意味。

この 10 月 30 日取得版の既定条件で、公式 Verifier は平均 $T = 437{,}947.61$、静的 Toffoli 様演算数 456,456、$R = 1.19673$、費用 4.1421×10^{11} を報告した。公表参照費用 2.4439×10^{13} との比は約 59.00 倍である。参照回路の 5,725,461 は静的個数であり、889Q の T はショット平均なので、この 2 数を同じ種類の測定値として直接比較してはならない。参照費用は公開基準値との照合から再構成したもので、参照回路自体の既定 102,400 ショット実行値ではない。

推定（R, T を固定した算術例）：作業領域が少ないことの Q^2 因子への効果は、費用式から計算できる：

$$\left(\frac{889}{1{,}893}\right)^2 \simeq 0.2205.$$

つまり Q だけを見ると費用因子は約 77.95% 小さくなる。ただし実際には T と R も変わる。確定（静的命令数の算術）として、Toffoli 様演算数は $456{,}456/5{,}725{,}461 \simeq 0.0797$（約 92.0% 減）だが、参照値は静的、889Q の T は平均であり、この比を score の T 因子比として使うことはできない。両者の実際の費用差は、Verifier 条件と失敗率を含めて評価する。

この比較からは、少ない作業量子ビットと低い公開評価費用が確認できる。一方、889Q の復号アルゴリズム、内部の線形変換の構成、位相を付ける具体的な演算手順は未確認である。Euclid 法以外の復号法、線形変換の共有、位相への直接計算、測定記録に基づく条件付き実行、逆計算の省資源化は一般的な設計上の検討点として挙げられるが、これらが当該 889Q 回路に使われているとの主張ではない。

演習：量子ビットの価値

R, T を固定して Q を 1 増やすと費用が約 $2/Q$ 増えることを示せ。$Q = 883$、$T = 519{,}152$ のとき、約 1,176 Toffoli に相当することを確かめよ。

第 6 章

dfthcsos

この章で学ぶこと

- dfthcsos の課題設定、スコア $\lambda_{\mathrm{eff}} \times T \times Q$ と精度ゲートを正確に述べられるようになる。
- T と Q のうち、回路コンパイラが変えられる部分と、因子分解の形 (R, B, C, β) で固定される部分を区別できる。
- 2026 年 10 月 28 日時点の mrmeows 公開版（8.6121 倍）と、10 月 30 日取得の SomeJoint 配布版（9.8627 倍）の評価条件と費用構造を説明する。

本章は、量子化学ベンチマーク dfthcsos の数学的背景と回路評価を説明する。最終的な科学目標は FeMoCo 電子ハミルトニアンの基底状態エネルギーを推定することだが、この課題で評価するのは因子分解と回転角 lookup 回路の資源である。評価値は、ここに示す回路・係数データと、2026 年 10 月 30 日に取得した公開提出物に対するものである。

注意

表の Verifier 値は、公開された KickMix ソースからビルドした macOS 版による実行結果であり、配布 Linux バイナリでの独立再実行値ではない。公開 ANGLE 回路には測定結果で分岐する Toffoli がなく、この実行条件では乱数シードを変えても費用値は変わらない。公開作者による説明と、独立に集計した命令列の観察結果は区別して記す。

6.1　課題：FeMoCo のブロックエンコーディング

対象は窒素固定酵素の FeMo 補因子（FeMoCo）のモデルハミルトニアンで、空間軌道 $N = 76$、電子数 $\eta = 113$ である。これを DF-THC（二重因子分解テンソル超縮約）で表し、SOS（二乗和）による下界で対称性シフトを入れた形でブロックエンコードする。参加者が提出するのは、因子分解（R ランク、ブロック B、列 C の重み）と、それを読み込む `ANGLE_k` 回路の族である。

6.1.1　FeMoCo とは

FeMoCo は、窒素分子をアンモニアに還元する酵素ニトロゲナーゼの活性中心にある金属クラスターで、組成は $\mathrm{MoFe_7S_9C}$ である（図 6.1）。

- 鉄 7 個・モリブデン 1 個・硫黄 9 個と、中心の炭素 1 個からなる。
- Fe_4S_3 と $MoFe_3S_3$ の 2 つの立方体状の断片を、3 個の架橋硫黄（μ_2-S）がつなぐ。
- 中央の炭素（炭化物、μ_6-C）は、中段の鉄 6 個に囲まれている。
- 端の鉄はシステインの硫黄で、モリブデンはヒスチジンの窒素とホモクエン酸の酸素 2 個で、タンパク質に結合している。

多数の鉄の d 電子が強く相関するので、古典計算で基底状態エネルギーを精度よく求めるのが難しく、量子計算の代表的な応用例とされている。次節では、この活性中心をどの電子構造モデルで表すかを説明する。

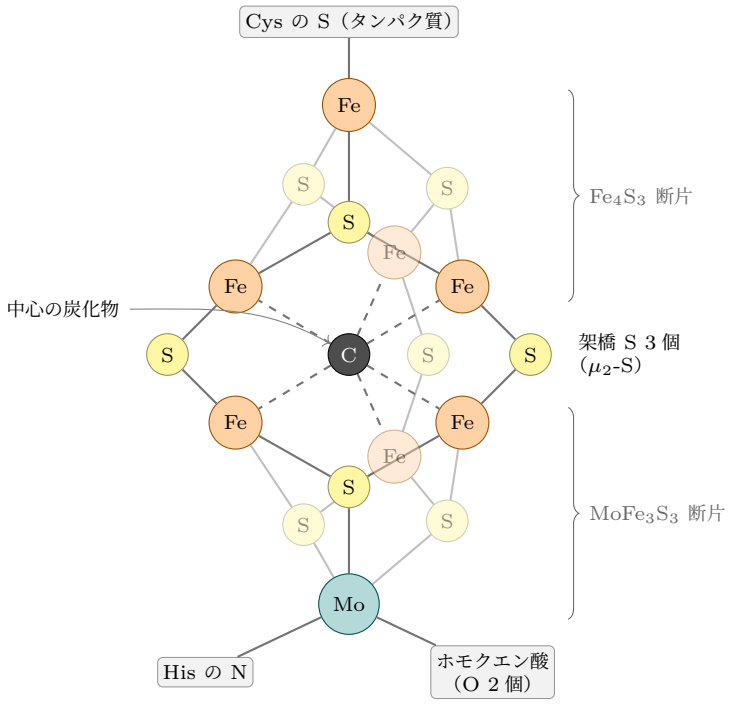


図 6.1　FeMoCo（$MoFe_7S_9C$）の模式図。原子の位置は実際の幾何ではなく、結合のつながりを示す。薄い色の原子は奥側にある。破線は中心の炭素と中段の鉄 6 個の結合。

6.1.2　FeMoCo の電子状態をモデル化する

量子化学計算では、まず分子全体の全自由度を扱うのではなく、重要な軌道を選んだ活性空間モデルを作る。

本課題の `FeMoCo76o113e` は、76 個の空間軌道と 113 個の電子からなるモデルである。電子にはスピン自由度があるため、空間軌道 1 個につきスピン軌道が 2 個あり、合計 152 スピン軌道となる。このモデルの基底状態エネルギーは科学上の目標量だが、課題が窒素固定の反応経路や活性化エネルギーを直接計算するわけではない。対象は、与えられたモデルハミルトニアンを量子回路に効率よく組み込む方法である。

6.1.3　電子ハミルトニアンと第二量子化

第二量子化では、スピン軌道 p の電子を消す演算子 a_p と、電子を作る演算子 $a_p^\dagger$ を使う。電子の運動と原子核によるポテンシャルは一体項、電子同士のクーロン反発は二体項として表され、電子ハミルトニアンは一般に

$$H = E_{\mathrm{nuc}} I + \sum_{p,q} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{p,q,r,s} g_{pqrs} a_p^\dagger a_q^\dagger a_s a_r$$

と書ける。E_{nuc} は核間反発、h_{pq} は一電子積分、g_{pqrs} は二電子積分で、これらは選んだ軌道基底と分子モデルで決まる係数である。

フェルミオン演算とは

電子のように、同種の二つの粒子を交換すると状態の振幅にマイナス符号が付く粒子をフェルミ粒子（フェルミオン）という。この性質から、同じスピン軌道には電子を二つ置けない（Pauli の排他原理）。各スピン軌道の状態は「空」$|0\rangle$ と「電子が一ついる」$|1\rangle$ で表す。フェルミオン演算とは、この占有状態を生成・消滅演算子で変える計算である。一つの軌道だけなら

$$\begin{aligned} a^\dagger |0\rangle &= |1\rangle, \quad a^\dagger |1\rangle = 0, \\ a |1\rangle &= |0\rangle, \quad a |0\rangle = 0 \end{aligned}$$

となる。右辺の 0 はゼロベクトルであり、空軌道の状態 $|0\rangle$ とは異なる。$a^\dagger a$ は電子がいるかを調べる数演算子で、$a_p^\dagger a_q$ は軌道 q から電子を取り除いて軌道 p に移す項になる。生成・消滅という言葉はこの数学的な作用を指し、粒子数を保存する電子ハミルトニアンでは、それらを電子数が変わらない積に組み合わせて使う。

複数軌道では作用の順番が重要になる。反交換関係

$$\{a_p, a_q^\dagger\} = \delta_{pq} I, \qquad \{a_p, a_q\} = 0, \qquad \{A, B\} = AB + BA$$

は、異なる軌道の演算の順を入れ替えると符号が変わることを表す。例えば二軌道の状態を $|n_0 n_1\rangle$ と書く規約では、$a_0 |11\rangle = |01\rangle$ だが $a_1 |11\rangle = -|10\rangle$ となる。この符号は振幅に付く位相であり、占有ビットを単純に反転するだけでは再現できない。Jordan–Wigner 変換は、先行軌道にある電子数の偶奇（占有パリティ）に応じた符号を、Z の列で表す方法である。例えば

$$a_p = (Z_0 \cdots Z_{p-1}) \frac{X_p + \mathrm{i} Y_p}{2}$$

とすれば、異なる軌道間の交換符号も量子ビット回路で表せる。

152 スピン軌道をそのまま使うと二体係数の添字組合せが非常に多くなるため、全項を個別に量子回路へ載せる方法は高コストになる。

DF-THC はこの係数構造を段階的に低ランク化する。二重因子分解（DF）は二体積分を少数の行列成分の和にまとめ、テンソル超縮約（THC）は各成分の軌道添字依存性を小さな因子行列で表す。課題の整数 R, B, C は、式中の添字範囲で定義される。$r = 0, \ldots, R-1$ は二重因子分解の因子添字、$c = 0, \ldots, C-1$ は各因子から作る二体生成子の添字、$b = 0, \ldots, B-1$ は各生成子の和に現れるベクトル添字である。したがって二体生成子は RC 個あり、それぞれ B 本のベクトル項を含む

(式中の $w_B I$ は恒等成分)。この記法では R が DF 因子ランク、C が各因子に対応する生成子列数、B が二乗和生成子に含むベクトル数である。これらを小さくすると回路表現や PREPARE の費用を下げられる一方、積分テンソルを近似する残差が増える可能性がある。したがって、分解サイズと再構成精度は同時に選ぶ必要がある。

6.1.4　ハミルトニアンのブロックエンコーディング（PREPARE と SELECT）

量子位相推定でエネルギーを求めるには、ハミルトニアン H を量子回路に埋め込む必要がある。標準的な方法は、H をユニタリの線形結合（LCU）$H = \sum_\ell \alpha_\ell U_\ell$（$\alpha_\ell > 0$）で表し、2 種類の回路を組み合わせることである。

$$\mathrm{PREPARE}\,|0\rangle = \sum_\ell \sqrt{\frac{\alpha_\ell}{\lambda}}\,|\ell\rangle\,, \qquad \mathrm{SELECT} = \sum_\ell |\ell\rangle\langle\ell| \otimes U_\ell,$$

$$\langle 0|\,\mathrm{PREPARE}^\dagger\,\mathrm{SELECT}\,\mathrm{PREPARE}\,|0\rangle = \frac{H}{\lambda}.$$

$\lambda = \sum_\ell \alpha_\ell$ は 1 ノルムで、位相推定の呼び出し回数は λ に比例する。$\mathrm{PREPARE}^\dagger$ は一般に補助レジスタ全体を確定的に $|0\rangle$ へ戻す操作ではない。PREPARE・SELECT・$\mathrm{PREPARE}^\dagger$ のユニタリには複数のアドレスブロックがあり、補助レジスタの $|0\rangle$ から $|0\rangle$ への行列要素だけを取ると H/λ になる。ほかの補助状態への遷移もあり、ブロックエンコーディングではそれらを含む全体がユニタリである。

この点は、ブロックエンコーディングを一度だけ使う場合に、確率的な選択として見ると分かりやすい。正規化された系の状態 $|\psi\rangle$ に全ユニタリ W を作用させると、

$$W(|0\rangle\,|\psi\rangle) = |0\rangle\,\frac{H\,|\psi\rangle}{\lambda} + |\Phi_\perp\rangle\,, \qquad (\langle 0| \otimes I)\,|\Phi_\perp\rangle = 0.$$

補助を測って 0 のショットだけを選ぶ場合

補助レジスタを実際に測って 0 が出る確率は $p = \|H\,|\psi\rangle\,\|^2/\lambda^2$ で、$p > 0$ ならその場合に系は $H\,|\psi\rangle\,/\|H\,|\psi\rangle\,\|$ へ正規化される。0 以外の結果では、その補助状態への射影成分に対応する系の状態を得る。0 が出たショットだけを採用するなら、それが**ポストセレクション**である。$p = 0$ ならその枝は起こらない。例えば $H = I + Z$、$\lambda = 2$ を 1 補助ビットの LCU で表す。PREPARE を Hadamard ゲート、SELECT を「補助が 1 のとき系に Z を作用させる」制御ゲートとし、系を $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ から始めると

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

となる。この場合は補助 0 が確率 1/2 で出て、系は $|0\rangle$ になる。補助 1 が出たら系は $|1\rangle$ になるので、二つの枝は同じ出力ではない。

ほかの補助状態へ移った成分はどうなるか

測定する前の $|\Phi_\perp\rangle$ は消えた状態ではなく、全体の重ね合わせの一部として残り、一般には系と補助がもつれている。測定して特定の補助値を得ると、その値に対応する成分へ射影される。0 以外の

枝の系の状態は W のほかのブロックによって決まり、一般には $H|\psi\rangle$ に比例しない。補助を測定結果ごと捨てれば、系は各枝を確率的に混ぜた状態になり、欲しい 0 ブロックだけの作用とは異なる。補助を無条件にリセットすることも、成功枝を選ぶことの代わりにはならない。一方、測定しなければ他の枝も干渉に利用でき、$W^\dagger$ などの後続演算で元の部分空間と結び付けられる。

エネルギー推定では各回でポストセレクションするのか

Hamiltonian simulation や位相推定で LCU を使うときは、通常この測定を毎回行って失敗枝を捨てるのではない。qubitization や quantum signal processing（QSP）、スペクトル増幅では、W と補助レジスタの反射などをコヒーレントに組み合わせ、全ユニタリのブロック構造を利用する。補助レジスタの測定・ポストセレクションを各呼び出しで行う手順とは異なる。[16, 12]

小さな例：水素分子 H_2

LCU のブロックエンコーディングを、最小の例で確かめる。水素分子（STO-3G 基底、結合長 0.735 Å）の電子ハミルトニアンは、4 つのスピン軌道をパリティ写像で量子ビットにし、対称性で 2 量子ビットに縮約すると次の形になる（係数は Ha）。

$$H = -1.05237\, II + 0.39794\, IZ - 0.39794\, ZI - 0.01128\, ZZ + 0.18093\, XX.$$

恒等項は位相推定でエネルギーをずらすだけなので除き、残りの 4 項 $H' = H + 1.05237\, I$ をブロックエンコードする。回路は図 6.2 のとおりで、アドレスの補助 2 量子ビットと系の 2 量子ビットから

表 6.1　H_2 の LCU（係数の符号はユニタリ側に含める）

アドレス ℓ	係数 c_ℓ	U_ℓ	$\alpha_\ell = \lvert c_\ell \rvert$	PREPARE の確率 α_ℓ/λ
00	$+0.39794$	$I \otimes Z$	0.39794	0.4027
01	-0.39794	$-Z \otimes I$	0.39794	0.4027
10	-0.01128	$-Z \otimes Z$	0.01128	0.0114
11	$+0.18093$	$X \otimes X$	0.18093	0.1831
$\lambda = \sum_\ell \alpha_\ell$			0.98809	1

なる。

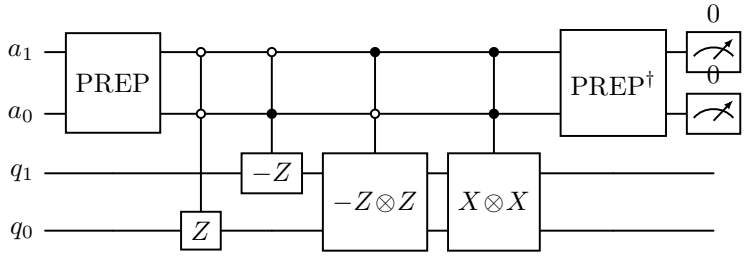
1. PREPARE が補助を $\sum_\ell \sqrt{\alpha_\ell/\lambda}\,|\ell\rangle$ にする。
2. SELECT は、補助の値 ℓ に応じて系に U_ℓ を掛ける。4 通りのアドレスごとに、開いた丸（0 で働く）と黒丸（1 で働く）の制御付き Pauli を 1 つずつ並べればよい。
3. PREPARE† を適用し、補助が $|00\rangle$ から $|00\rangle$ へ遷移するブロックを取ると、系には H'/λ が作用する。ほかの補助状態へ移る成分もユニタリ全体には残る。

実際、$P = \mathrm{PREPARE}$、$S = \mathrm{SELECT}$ とすると全回路は $W = (P^\dagger \otimes I)S(P \otimes I)$ であり、

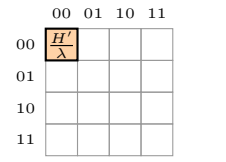
$$(\langle 00| \otimes I)W(|00\rangle \otimes I) = \sum_\ell \frac{\alpha_\ell}{\lambda} U_\ell = \frac{H'}{\lambda}.$$

P, S はユニタリなので W もユニタリである。この式から、図の 16×16 行列の左上 4×4 ブロックが H'/λ であることを直接確かめられる。表示した H を対角化すると電子エネルギーの最小固有値は約 -1.85728 Ha となり、核間反発 0.71997 Ha を加えた全エネルギーは約 -1.13731 Ha となる。

FeMoCo でも構造は同じである。違いは規模と SELECT の作り方にある（表 6.2）。H_2 ではアドレスごとに Pauli を 1 つずつ制御付きで掛けられるが、FeMoCo では項が多すぎるので、アドレスから回転角を読み込み（ANGLE 回路）、Givens 回転で「回したマヨラナ演算子」を作る。



中央の 4 つの制御付きゲートが SELECT



補助のアドレスで区切った
16×16 ユニタリ

図 6.2　H_2 の 2 量子ビットハミルトニアンのブロックエンコーディングを示す概念図。左：PREPARE・SELECT・PREPARE† の回路（白丸は 0、黒丸は 1 で働く制御）。右：補助の $|00\rangle$ 対角ブロックが H'/λ になる行列構造。この図で補助を測る表現はブロックの取り出し方を示すものであり、エネルギー推定アルゴリズム全体で毎回測定・ポストセレクションするという意味ではない。

表 6.2　H_2 と FeMoCo のブロックエンコーディングの比較

	H_2（2 量子ビット）	FeMoCo（本課題）
系	2 量子ビット	152 スピン軌道
項の数	4	生成子 $N + RC$ 個、それぞれ B 本のベクトルの和
PREPARE	4 振幅の状態準備	Outer（生成子の重み）＋ Inner（ベクトルの重み）
SELECT	制御付き Pauli 4 個	回転角の読込（ANGLE）＋ Givens 回転 ＋ γ_0 ＋ 逆回転
1 ノルム	$\lambda = 0.988$	$\Lambda \approx 300$、スペクトル増幅で $\lambda_{\text{eff}} \approx 48$

この課題の回路は、論文 [1] の付録 A に記述され、文献 [12] のスペクトル増幅法に基づく。一般の LCU にさらに SOS 分解を組み合わせる。まず、SOS 下界を引いたハミルトニアンを、作用素の二乗の和で書く。

$$H - E_{\text{SOS}} I = \sum_g O_g^\dagger O_g .$$

生成子 O_g は、一体項の正・負の固有値に対応するもの（$\pm$ 各 $N_\pm$ 個）と、二体項の $R \cdot C$ 個の二乗（r, c）である。二体の生成子は、B 本のベクトル u_b で回転したマヨラナ演算子 $\gamma_u = \sum_p u_p \gamma_p$ の積の線形結合になる。

$$O_{\text{SF},rc} = \frac{1}{\sqrt{2}} \Big(w_B^{(rc)} I + \frac{\mathrm{i}}{2} \sum_{b<B} w_b^{(rc)} \sum_\sigma \gamma_{u_b \sigma 0} \gamma_{u_b \sigma 1} \Big).$$

回転したマヨラナ演算子は、Givens 回転の木 $U(\vec{\theta})$ で固定のマヨラナ γ_0 を回して作る：$\gamma_u = U(\vec{\theta})^\dagger \gamma_0 U(\vec{\theta})$。したがって SELECT は、$N^2$ 通りの演算子から 1 つを選ぶ代わりに、「アドレスに応じた回転角を読み込み、回転し、γ_0 を掛け、回転を戻す」で済む。回転角を読み込むのが `ANGLE_k` 回路で、この課題で参加者が作る唯一の回路である。

1 回のブロックエンコーディングの順序は次のとおりである（論文の式 (A13)）。

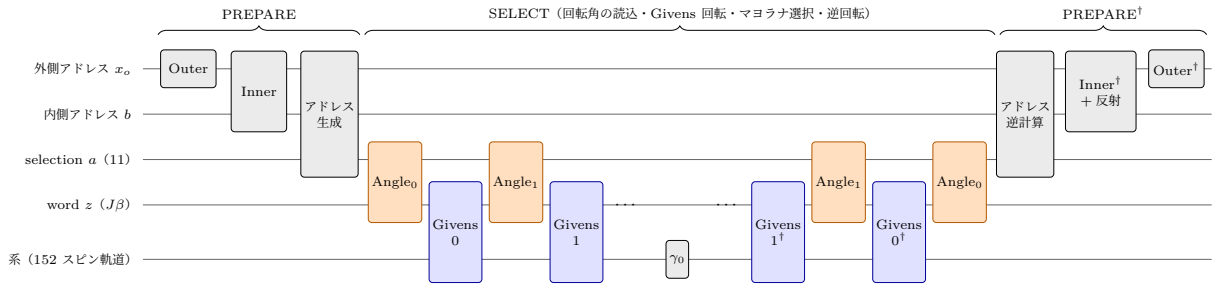
$$\text{Outer} \to [\text{デコード} \to \text{Inner} \to \text{スピン選択}] \to \{\text{Angle}_k \to \text{Givens}_k\}_{k=0}^{K-1} \to \text{マヨラナ選択}$$
$$\to \{\text{Givens}_k^\dagger \to \text{Angle}_k\}_{k=K-1}^{0} \to [\text{Inner}^\dagger \to \text{反射して二乗}] \to \text{Outer}^\dagger$$

図 6.3 に、これを回路図の形で示す。

- **Outer（外側の PREPARE）**：どの生成子 g を使うかを、重み λ_g^2 に比例する振幅で準備する。アドレスは $N + RC$ 通り。
- **Inner（内側の PREPARE）**：二体の生成子の中で、B 本のベクトルのどれ（または恒等）を使うかを、重み $|w_b|$ で準備する。
- **SELECT**：(生成子, b) からベクトルのアドレス a（$N + RB$ 通り）を作り、Angle_k で回転角を J word ずつ読み込み、Givens 回転を掛ける。K 回のバッチを終えたら γ_0 を掛け、逆順に戻す。
- **反射して二乗**：Inner の後ろで信号レジスタに反射を掛け、もう一度逆を当てることで、O_g のブロックエンコーディングから $O_g^\dagger O_g$ を作る。Outer がそれらを重み付きで足し、$(H - E_{\text{SOS}})/\Lambda - I$ のブロックエンコーディングになる。

$H - E_{\text{SOS}}$ が非負であることを使うと、スペクトル増幅により位相推定の費用は Λ ではなく、それより小さい $\lambda_{\text{eff}} = \sqrt{E_{\text{gap}}(2\Lambda - E_{\text{gap}})}$ に比例する。これがスコアの第 1 因子である。

SOS（sum of squares、二乗和）表現は、エネルギー下界 E_{SOS} を差し引いた演算子を $\sum_g O_g^\dagger O_g$ と書き、非負性を明示する方法である。ここで Λ はその表現に付随する LCU 係数の和、$E_{\text{gap}} = E_{\text{upper}} - E_{\text{SOS}}$ は上界と SOS 下界の差である。この構成では、符号を持つ一般のハミルトニアンを直接扱う場合よりも、スペクトル増幅によって実効正規化係数 λ_{eff} を小さくできる。ただし R, B, C の選択は残差、Λ、状態準備費用を同時に変えるので、単一のランクだけを最小化すればよいわけではない。



ANGLE lookup：参加者が設計・提出し、Verifier が Toffoli を数える。
Givens 回転：費用モデルで 1 回転あたり $2\beta - 1$ Toffoli と仮定する。
その他のブロック：選んだパラメータに対し、費用モデルの固定値を用いる。

図 6.3　DF-THC + SOS のブロックエンコーディング 1 回分の回路の模式図（論文の式 (A13) に基づく。スピン選択と信号レジスタは省略）。橙色の ANGLE lookup が参加者の設計・提出対象で、その他のブロックは課題の費用モデルで評価される。SomeJoint の 2026 年 10 月 28 日版では、T の内訳は ANGLES 8,666（順と逆の 2 回分）、Inner 5,463、Givens 2,250、Outer 373 である。

6.1.5　回路が読み込む角度 word

word とは何か

ここでいう **word** は、一つの数値を表すためにまとめて扱う固定長のビット列である。一般のコンピュータでいう 32 ビットや 64 ビットの word と同じ考え方だが、本課題では幅を β（`brot`）ビットに選ぶ。**角度 word** は、このビット列で回転角を離散的に表したものである。

ビットを $b_0, \ldots, b_{\beta-1}$ とし、整数として読んだ値を $w = \sum_k 2^k b_k$ とすると、公式の復号規約は

$$\theta(w) = \frac{\pi w}{2^{\beta-1}} = \frac{2\pi w}{2^{\beta}} \pmod{2\pi}.$$

したがって β ビットなら 1 周を 2^β 等分し、角度の刻みは $2\pi/2^\beta$ となる。例えば 8 ビットの `00100000` は $w = 32$ なので $\theta = \pi/4$（45 度）を表す。`11100000` は符号なしでは $w = 224$、2 の補数では -32 であり、角度は $7\pi/4 \equiv -\pi/4$（2π を法として同じ）である。このように表の整数、ビット列、回転角は同じ情報の異なる表し方である [1]。

量子回路では、1 word を β 量子ビットの **word レジスタ** に保持する。1 回に J word を読むなら、出力用には $J\beta$ 量子ビットが必要である。これは角度を数値として保存する領域であって、分子の電子状態を保存する 152 量子ビットの系レジスタとは別である。

因子分解の各ベクトルは、軌道の全域木 `tree_edges` 上の Givens 回転 75 個で表す。各回転角は β（`brot`）ビットの整数 word である。アドレス a は一体ベクトル（0〜75）と二体ベクトル（$R \cdot B$ 個）を数え、mrmeows 版では $76 + 10 \cdot 129 = 1{,}366$ 個、SomeJoint 版では $76 + 8 \cdot 143 = 1{,}220$ 個ある。`ANGLE_k` 回路は、アドレスの関数として J word ずつのバッチを word レジスタに XOR で読み込む。回路の数は $K = \lceil 75/J \rceil$ である。

- `selection` はちょうど $\lceil \log_2(76 + RB) \rceil$ 量子ビット、word レジスタは J 本 $\times\beta$ 量子ビット。
- 補助は開始・終了とも $|0\rangle$。`Z_POW` は禁止で、位相は π の整数倍だけ。
- 1 回路は 128 量子ビット以内。

6.1.6　ANGLE lookup、Givens 回転、マヨラナ演算子

ANGLE は SELECT 全体の別名ではなく、SELECT 内で選ばれたベクトルに対応する回転角を可逆に読み出す回路である。入力は選択アドレス a とゼロ初期化された word レジスタ、出力はそのアドレスに対応する J 個の β ビット角度 word で、計算に使った補助レジスタは指定された方式で初期状態へ戻す。古典側ではアドレスごとの角度 word を表として持ち、量子回路では QROM/QROAM 型 lookup を使って表の値を XOR 読み出しする。word 幅 β は角度の離散化精度を決め、J は一回の ANGLE 回路でまとめて読む回転数を決める。J を増やせば回路の呼び出し数 $K = \lceil 75/J \rceil$ が減るが、word レジスタが $J\beta$ 量子ビットを占め、128 量子ビット制約下で lookup の補助領域が狭くなる。

Givens 回転とは何か

Givens 回転は、多次元ベクトルのうち**二つの成分だけを混ぜ、ほかの成分をそのままにする平面回転**である。例えば係数の更新を

$$\begin{pmatrix} v'_p \\ v'_q \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} v_p \\ v_q \end{pmatrix}$$

と書けば、$(v_p, v_q) = (1, 0)$ は $\theta = \pi/4$ で $(1/\sqrt{2}, 1/\sqrt{2})$ に移る。二乗和 $v_p^2 + v_q^2$ は変わらない。76 成分の軌道ベクトルも、二成分の回転を順に組み合わせることで作れる。本課題では 76 軌道を結ぶ木の 75 本の辺に、それぞれ一つの回転角を指定する。

量子回路では、軌道 p, q の占有状態を混ぜる**フェルミオン Givens 回転**を使う。二軌道を $|n_p n_q\rangle$ の順に書いた一つの符号規約では

$$\begin{aligned} G(\theta)\,|10\rangle &= \cos\theta\,|10\rangle - \sin\theta\,|01\rangle\,, \\ G(\theta)\,|01\rangle &= \sin\theta\,|10\rangle + \cos\theta\,|01\rangle\,, \\ G(\theta)\,|00\rangle &= |00\rangle\,, \qquad G(\theta)\,|11\rangle = |11\rangle\,. \end{aligned}$$

例えば $|01\rangle$ から 45 度回転すれば、電子がどちらの軌道にいるかの等しい重ね合わせになる。電子数は変わらず、位置の確率振幅を混ぜる操作である。逆回転は $G(-\theta) = G(\theta)^\dagger$ で行える。係数の更新行列と状態へ作用する行列では、能動変換・受動変換や作用順により符号が変わる。ここでは係数表の復号を上の行列、演算子の変換を後述の共役規約に固定して区別する。

ANGLE lookup は、この θ を直接作用させる回路ではない。まず角度 word を可逆に読み、**そのビット列で制御した Givens 回転**を別の系レジスタに作用させる。アドレスが重ね合わせなら、word と対応する回転も同じアドレスに条件付けられ、角度を測って古典的に選び直す必要はない。

マヨラナ演算子の復習

マヨラナ演算子は、一つのフェルミオンモードの生成・消滅演算子から作る二つのエルミート演算子である：

$$\gamma_{p0} = a_p + a_p^\dagger, \qquad \gamma_{p1} = -\mathrm{i}(a_p - a_p^\dagger).$$

どちらも $\gamma = \gamma^\dagger$ かつ $\gamma^2 = I$ で、異なるマヨラナ演算子同士は反交換する。Jordan–Wigner 変換の一つの規約では、単一モードなら $\gamma_0 = X$、$\gamma_1 = Y$ となる。複数モードでは各モードの X または Y の前に先行モードの占有パリティを表す Z 文字列が付くため、異なる軌道間のフェルミオン交換符号も保たれる。一つの軌道では、$\gamma_0\,|0\rangle = |1\rangle$、$\gamma_0\,|1\rangle = |0\rangle$ と占有を反転する。γ_1 も占有を反転するが、$|0\rangle \mapsto \mathrm{i}\,|1\rangle$、$|1\rangle \mapsto -\mathrm{i}\,|0\rangle$ と位相が付く。生成だけ、あるいは消滅だけの演算はユニタリではないが、こうして組み合わせると二度作用させれば元に戻る演算になり、SELECT の部品として使える。ここでいうマヨラナは、電子の演算を扱いやすく分けた数学的表現であり、新しい物理粒子を作るという主張ではない。

マヨラナ作用とは何をすることか

「マヨラナ作用」は、系の状態 $|\psi\rangle$ に、選んだマヨラナ演算子を実際に掛けて $\gamma\,|\psi\rangle$ に変えることを指す。角度 word を読み込むことや、マヨラナ演算子の値を測定することとは異なる。固定の根軌

道に対する γ_0 は、占有を反転し、種類と占有パリティに応じた位相を付ける部品として実装できる。任意の軌道の組合せに対応する演算は、この固定部品を基底回転ではさむことで実現する。

$\|u\|_2 = 1$ の実係数ベクトルに対し $\gamma_u = \sum_p u_p \gamma_{p0}$ とする。Givens 回転を組み合わせた $U(\vec{\theta})$ が係数ベクトル u を指定するなら、系への作用は

$$|\psi\rangle \xrightarrow{U} U|\psi\rangle \xrightarrow{\gamma_0} \gamma_0 U|\psi\rangle \xrightarrow{U^\dagger} U^\dagger \gamma_0 U|\psi\rangle = \gamma_u|\psi\rangle$$

となる。つまり、まず回転した基底へ移り、固定のマヨラナ演算子を作用させ、元の基底へ戻す。途中に γ_0 が入るため、前後の回転を打ち消して恒等演算にしてよいわけではない [1]。

■二軌道の具体例　上の規約では

$$G(\theta)^\dagger \gamma_{00} G(\theta) = \cos\theta\, \gamma_{00} + \sin\theta\, \gamma_{10}.$$

45 度なら、固定の軌道 0 だけでなく、二軌道の等しい組合せに対応するマヨラナ作用になる。両軌道が空の状態に作用させると

$$\frac{\gamma_{00} + \gamma_{10}}{\sqrt{2}}|00\rangle = \frac{|10\rangle + |01\rangle}{\sqrt{2}}.$$

これは「どちらかの軌道を確率的に選んで反転する」という混合ではなく、二つの経路の振幅を足すコヒーレントな操作である。単独のマヨラナ演算子は占有数の偶奇を変えるが、電子ハミルトニアンでは必要な積や和として組み合わせる。上の例は SELECT の部品の作用であり、分子全体の時間発展そのものではない。

全体の流れは、アドレス、ANGLE lookup、角度 word、Givens 回転、固定のマヨラナ作用、逆 Givens 回転である。逆回転は電子状態を元に戻すためではなく、選んだ γ_u の作用を元の軌道表現で得るために行う。その後、ANGLE の逆処理で不要になった word を消す。参加者が提出する回路は角度 word を読み込む ANGLE lookup で、Givens 回転とマヨラナ作用を含む残りのブロックエンコーディングは費用モデルで扱う。

6.1.7　スコアと精度ゲート

$$\text{score} = \lambda_{\text{eff}} \times T \times Q \qquad (\text{小さいほどよい}), \qquad \lambda_{\text{eff}} = \sqrt{E_{\text{gap}}\,(2\Lambda - E_{\text{gap}})}.$$

Λ は LCU の 1 ノルム、$E_{\text{gap}} = E_{\text{upper}} - E_{\text{SOS}}$ は SOS 下界とのエネルギー差である。精度は次のゲートで判定し、一つでも外れると不合格になる。ここで 20 mHa は、一体・二体係数テンソルの

表 6.3　dfthcsos の係数再構成ゲート

量	意味とゲート
error_1	一体項の再構成残差（Frobenius）。≤ 20 mHa
error_2	二体項の再構成残差（Frobenius）。≤ 20 mHa
Λ	LCU の 1 ノルム。> 49.0（等号は不可）
E_{gap}	> 3.914898 Ha（等号は不可）。$0 < E_{\text{gap}} < 2\Lambda$ も必要

Frobenius ノルム残差の上限である。これは近似ハミルトニアンの係数再構成条件であり、基底状態

エネルギーを実際に計算したときの誤差が 20 mHa 以内になるという保証ではない。また、量子位相推定（QPE）そのものを実行することも、この評価には含まれない。全体の位置付けは「電子積分 → DF-THC/SOS 因子分解 → PREPARE/SELECT → ブロックエンコーディング → QPE 等のエネルギー推定」であり、評価対象は因子分解と ANGLE lookup、および関連する回路資源パラメータまでである。この課題の Verifier は提出された因子分解と ANGLE lookup 回路を検査・評価する。ブロックエンコーディング全体を実行して補助レジスタの 0 ブロックを測定・ポストセレクトする処理も、その確率的な枝からエネルギーを計算する処理も行わない。図 6.2 の測定表現は、行列ブロックの意味を説明するための概念図である。対称性シフト (S, β_1) は baseline の固定値で、提

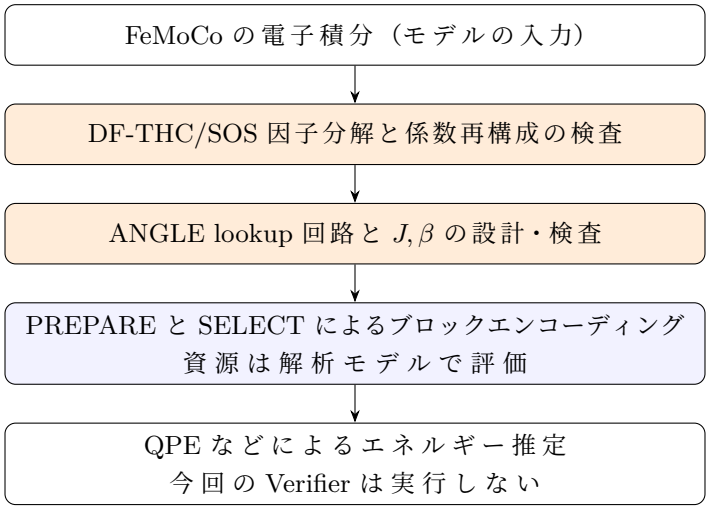


図 6.4　エネルギー推定全体における評価範囲。橙色は提出物を作成し検査する部分、青は関連する資源モデル、最下段は科学的目標につながる後続処理である。

出した値は Verifier が上書きする。したがって、論文にある BLISS 型のシフト最適化は、この競技ではスコアに効かない。

6.1.8　T と Q の内訳

$$T = T_{\mathrm{ANGLES}} + T_{\mathrm{outer}} + T_{\mathrm{inner}} + 2(N-1)(2\beta - 1),$$
$$Q = Q^{\mathrm{perm}}_{\mathrm{outer}} + Q^{\mathrm{perm}}_{\mathrm{inner}} + J\beta + \max\bigl(Q^{\mathrm{tmp}}_{\mathrm{outer}}, Q^{\mathrm{tmp}}_{\mathrm{inner}}, Q^{\mathrm{tmp}}_{\mathrm{ANGLES}}\bigr).$$

T_{ANGLES} は KickMix が数える ANGLE 回路の Toffoli（順と逆の 2 回分）である。外側・内側の状態準備（PREPARE）と Givens の費用は、公式の費用モデルが (R, B, C, N, β) だけから計算し、回路の中身には依存しない。つまり、回路コンパイラが変えられるのは T_{ANGLES}、ANGLE 回路の一時量子ビット、J、β だけである。

6.2　baseline と公開回路の再現

公開回路の比較対象には次の配布物がある。一つ目は、Kaggle データセット `mrmeows/qcc-1375-open-circuits`[3] の `circuits/dfthcsos/`（19 回路と `submission.npz`）である。因子分解の形 $R = 10$、$B = 129$ は Michael Rustad が見つけ、データセットの作者が再フィットと lookup コン

パイラを加えた（作者の記述）。再現の際は各ファイルのハッシュを同梱の `SHA256SUMS` と照合する。二つ目は、SomeJoint のノートブック [6] から 2026 年 10 月 30 日に取得した出力 `heliosli-v19`（別参加者 Dmitrii Gluzdov の表の重みを使う版）である。以下では取得日を指定して比較する。無改変の公式 Verifier の結果を示す。

表 6.4　公開回路の公式 Verifier 実行結果（macOS 版 KickMix。mrmeows は 2026 年 10 月 28 日時点の公開版、SomeJoint は各列の日付に取得）

項目	baseline	mrmeows（10/28 公開版）	SomeJoint（10/28 取得版）	SomeJoint（10/30 取得版）
(R, B, C)	(65,60,60)	(10,129,59)	(8,143,62)	(8,143,60)
アドレス数	3,976	1,366	1,220	1,220
β / J / 回路数	12 / 8 / 10	8 / 4 / 19	8 / 5 / 15	8 / 5 / 15
ショット	254,464	87,424	78,080	78,080
λ_{eff}	49.3014	47.6354	48.2713	48.6138
Λ	308.456	291.764	299.553	303.791
error_1 / error_2 (mHa)	3.185 / 18.640	19.985 / 19.976	19.813 / 19.9988	19.9995 / 19.9995
T	93,237	19,337	16,752	16,276
Q	730	423	430	430
費用	3,355,599,588	389,636,552	347,715,360	340,232,090
参照比	1	8.6121	9.6504	**9.8627**

baseline の費用は参照値と一致した。mrmeows 版は salt を変えても、単独で再実行しても、出力がバイト一致した。データセットにある一つ前の版（8.5735 倍、$C = 60$、$T = 19{,}522$）も PASS した。精度ゲートに近い。ここでの SomeJoint の余裕は 10 月 28 日取得版の値である。mrmeows 版の余裕は error_1 0.015 mHa、error_2 0.024 mHa、E_{gap} 1.6×10^{-5} Ha で、SomeJoint 版は error_2 の余裕が 1.2×10^{-6} Ha（0.006%）、E_{gap} が 4.2×10^{-6} Ha しかない。

6.2.1　費用の内訳

表 6.5　T と Q の内訳（公式値）

部分	mrmeows	SJ 10/28	SJ 10/30	何で決まるか
ANGLES（順＋逆）	10,362	8,666	8,346	回路コンパイラ（$2 \times 5{,}181$、$2 \times 4{,}333$、$2 \times 4{,}173$）
inner PREPARE	6,328	5,463	5,311	(R, B, C, β) で固定
Givens	2,250	2,250	2,250	$2 \cdot 75 \cdot (2\beta - 1)$ で固定
outer PREPARE	397	373	369	固定
T 合計	19,337	16,752	16,276	
Q	423	430	430	永続（outer ＋ inner ＋ word $J\beta$）＋共有一時 304

一時領域 304 はどちらも内側 PREPARE の QROAM が決め、ANGLE 回路の一時（85、75）は Q に効いていない。mrmeows 版での感度は、1 CCX/回路が 0.197%、1 量子ビットが 0.236%、λ_{eff} の 0.1 が 0.21% である。

10 月 28 日取得の SomeJoint 版は mrmeows 版より費用が 10.76% 低い。因子ごとの比で見ると、λ_{eff} が +1.3%、Q が +1.7% と悪化し、T が −13.4% と大きく改善した。T の減少 2,585 のうち、ANGLES が 1,696（回路が 19 本から 15 本に減った）、inner PREPARE が 865（$R \cdot B$ が 1,290 か

ら 1,144 に減った）である。つまり、Λ がやや悪い小さな分解を選び、$J = 5$ で回路の本数を減らす取引が成立している。

■10 月 30 日の更新　2026 年 10 月 30 日、SomeJoint のノートブックの出力の dfthcsos が次の版（id `heliosli-v19`、題名は別の参加者 Dmitrii Gluzdov の表の重みを使い、heliosli が配布する版）に替わっていた。公式 Verifier で PASS し、費用 340,232,090（9.8627 倍、10 月 28 日の版より -2.15%）だった。

- 分解は C が 62 から 60 に減っただけで、$R = 8$、$B = 143$、$J = 5$、15 回路、アドレス数 1,220、$Q = 430$ は同じである。
- T は 476 減った（-2.84%）。ANGLES が 320（1 回分 4,333 → 4,173）、inner PREPARE が 152、outer PREPARE が 4 減り、Givens は同じである。
- 代わりに Λ が 299.55 から 303.79 に上がり、λ_{eff} は $+0.71\%$ 悪化した。error_1・error_2 はどちらも 19.9995 mHa で、精度ゲートの余裕を Toffoli の削減に使い切っている。
- ANGLE 回路は、word の最上位ビットを流用する `R`（1 回路 80 個）がなくなり、全 15 本が 126 量子ビットになった（前の版は先頭 4 本が 121 量子ビット・305 CCX、残りが 283 CCX）。新しい版の CCX は先頭 4 本が 301〜302、残りが 258 と 279〜280 の交互である（確定）。
- 角度の word は 91,500 個のうち 58,738 個が書き換わっていた。表の重みを替えて再フィットし、lookup が安くなる word を選んだものと見られる（推定）。

baseline と mrmeows 版を比べると、T は 93,237 → 19,337（約 4.8 倍）、Q は 730 → 423（約 1.7 倍）、λ_{eff} は 49.30 → 47.64（約 1.03 倍）に下がった。T の内訳では、ANGLES が 79,480 → 10,362、内側 PREPARE が 9,461 → 6,328、Givens が 3,450 → 2,250（β を 12 から 8 へ）に減った。最も大きいのは ANGLES で、baseline は 1 回路 3,974 Toffoli の素朴な QROM を 10 回路に使っていた。

6.3　ANGLE 回路の構成

6.3.1　設計のつまみ：分解の形、J、lookup の作り

dfthcsos の参加者が選べるものは、大きく 3 つある。

- **分解の形** (R, B, C)：アドレスの数 $N + RB$、1 ノルム Λ（したがって λ_{eff}）、PREPARE の費用を決める。精度ゲートを通る因子分解が見つかる形でなければならない。
- **word の数** J（と word 幅 β）：ANGLE 回路の本数 $K = \lceil 75/J \rceil$ と、word レジスタの量子ビット $J\beta$ を決める。J を増やすと本数が減るが、1 本の中で補助に使える量子ビット $128 - \lceil \log_2(N + RB) \rceil - J\beta$ が減る。
- **lookup の作り**：アドレスから word を読み出す可逆回路の構成で、1 本あたりの CCX と補助の量子ビットを決める。

3 つは独立ではない。図 6.5 に、つまみからスコアの 3 因子へのつながりを、mrmeows 版から SomeJoint 版への変化の数値とともに示す。SomeJoint 版は、分解の形を小さくしてアドレスを減らし（Λ はやや悪化）、その分小さくなったキャッシュで $J = 5$ を 128 量子ビットに収めた。回路の

本数が 19 から 15 に減ったことが、ANGLES の T の削減の大部分を生んだ。

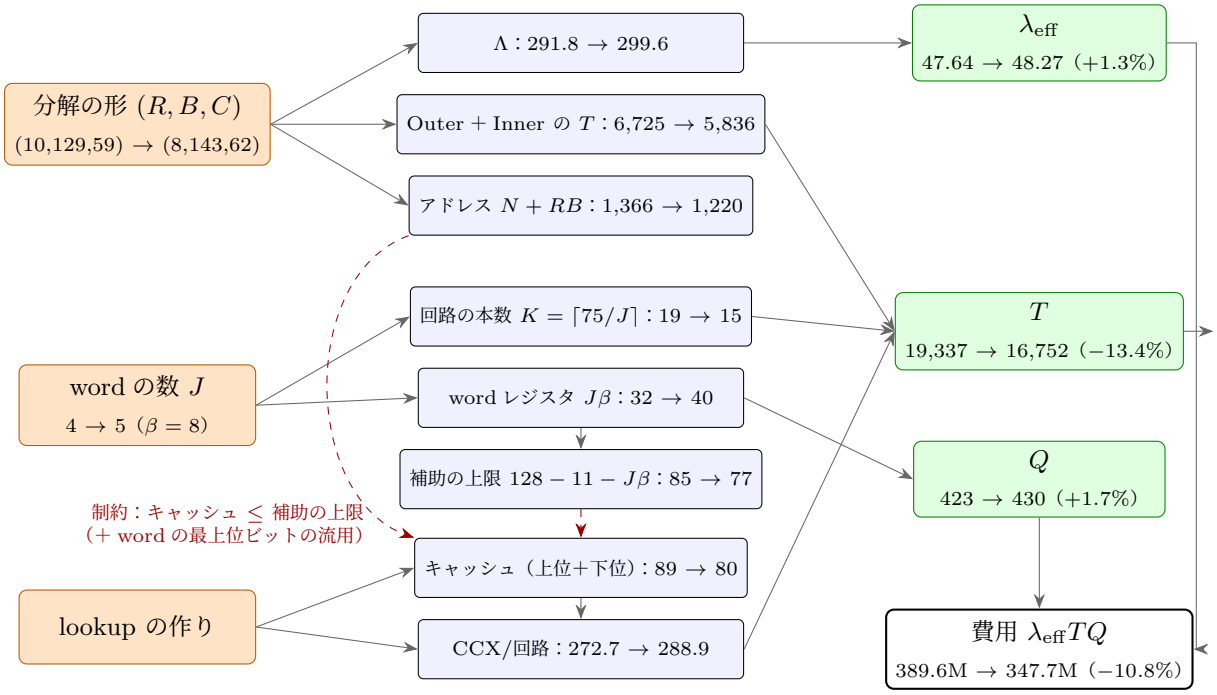


図 6.5　dfthcsos の 3 つのつまみとスコアの因子のつながり。数値は mrmeows 版 → SomeJoint 版（公式 Verifier の値と回路の命令列の集計）。赤の破線は 128 量子ビットの上限から来る制約。T には Givens の 2,250（両版で同じ）も入る。

6.3.2　lookup の作り方

ANGLE 回路は、アドレス a（11 ビット）の関数として、word の各ビット $f(a)$ を word レジスタに XOR する。f は GF(2) 上の多項式（代数標準形）で書ける：$f(a) = \bigoplus_S c_S \prod_{i \in S} a_i$。次数 1 の項は CX だけで足せるので、費用は次数 2 以上の積（単項式）をいくつ作るかで決まる。公開回路の lookup は、この積を 3 種類に分けて作る（図 6.6）。

1. **上位の単項式**：上位 7 ビット h の積を、短い積から順に（プレフィックスで閉じた集合として）1 つ 1 CCX で作り、キャッシュに置く。
2. **下位の積**：下位 4 ビット l の次数 2 以上の積は $2^4 - 1 - 4 = 11$ 個あり、2 段で作る。
3. **混合積**：上位 × 下位の積は、キャッシュの線を制御にした CCX で word のビットへ直接 XOR する。GF(2) の恒等式 $(h_1 \oplus l_2)(h_2 \oplus l_1) = h_1 l_1 \oplus h_2 l_2 \oplus h_1 h_2 \oplus l_1 l_2$ を使うと、1 つの CCX で混合積 2 項をまかなえる（残る $h_1 h_2$、$l_1 l_2$ はキャッシュから CX で足す）。

最後にキャッシュを X 測定（HMR）で消し、測定結果に応じた CZ で位相を直す。この消去に Toffoli はかからない。したがって 1 本の CCX は「上位のキャッシュ＋下位の積＋混合積」で、28 個（mrmeows 版、4×7）または 35 個（SomeJoint 版）の出力関数で混合積を共有するほど安くなる。

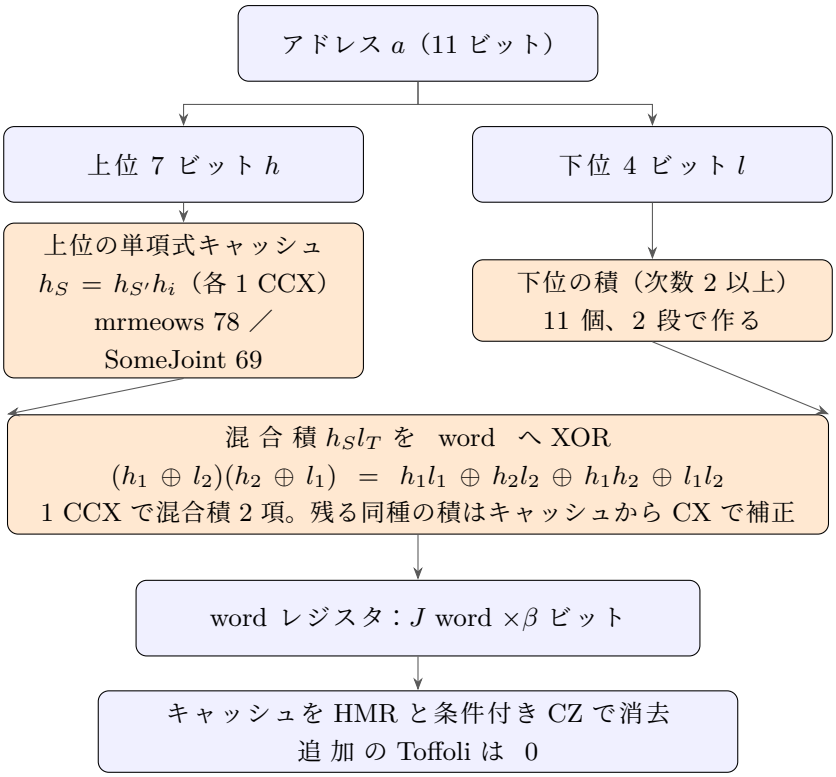


図 6.6　公開回路の ANGLE lookup の構成。上位と下位に分けた積をキャッシュし、混合積を word へ直接 XOR する。mrmeows 版の 1 回路あたりの CCX 数は平均 $78+11+183.7 = 272.7$ で、最後の項は混合積の平均である。構成に基づく上界の例は $78+11+224 = 313$。作者の説明と命令列の集計に基づく。

6.3.3　128 量子ビットの内訳

図 6.7 に、3 つの ANGLE 回路の量子ビットの内訳を示す。baseline は素朴な QROM で、補助は単項反復（アドレスの木をたどる回路）の 11 本だけだが、word 用の量子ビットが 12 ビット ×8 word で 96 本ある。mrmeows 版は 128 本を使い切り、補助 85 本をキャッシュに充てている。SomeJoint 版は word 用の量子ビットが 40 本に増えた分、補助を 75 本に減らし、足りない 5 本を word の最上位ビットで補っている。

6.3.4　mrmeows 版（$J = 4$）

mrmeows 版の ANGLE 回路は 1 本あたり 128 量子ビット（selection 11 ＋ word32 ＋ 補助 85）で、上限ちょうどである。CCX は 272〜274 個（平均 272.68、合計 5,181）で、すべて無条件なので静的な数と平均が一致する。

- アドレス 11 ビットを上位 7 ビットと下位 4 ビットに分ける。
- 上位の単項式 78 個（プレフィックスで閉じた集合、各 1 CCX）をキャッシュし、下位の AND 11 個は 2 段で計算して同時に 7 個だけ生かす。
- 残りは上位 × 下位の混合積で、GF(2) の恒等式 $(h_1+l_2)(h_2+l_1) = h_1l_1+h_2l_2+h_1h_2+l_1l_2$ を使って 1 CCX で 2 項をまかなう（作者の記述）。

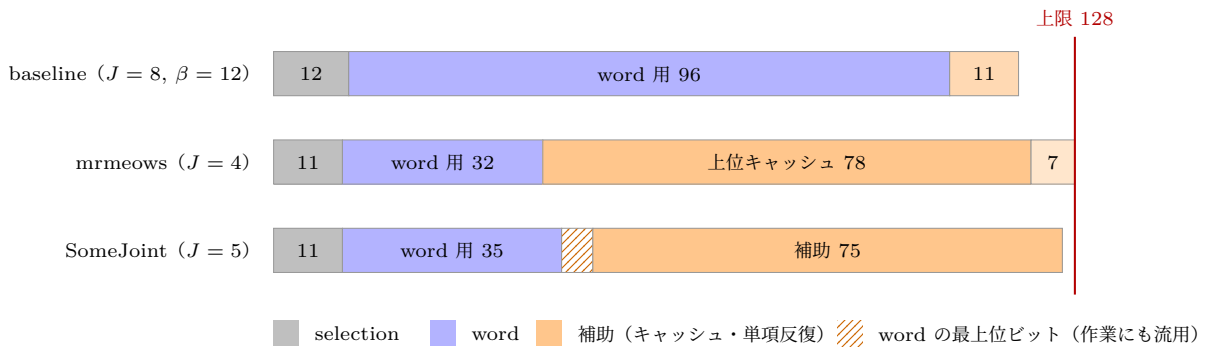


図 6.7　ANGLE 回路 1 本の量子ビットの内訳（回路の宣言と命令列の集計）。SomeJoint 版は $k = 4$〜14 の 11 本の値で、$k = 0$〜3 は 121 本。キャッシュの内訳（69 + 11）はアドレス数からの推定。

- キャッシュは X 測定（HMR）と条件付き CZ で消す。HMR は 1 回路 89 個で、78 + 11 と一致する。

補助 85 は 78 + 7 と一致し、128 量子ビットの上限がキャッシュの大きさを決めている。構成的な上界は 1 回路 313 CCX（上位 78 +下位 11 +混合 224）で、実測はそれより 12.9% 少ない。差はすべて混合積の共有分解による（出力関数 1 つあたり 8 から 6.56 へ）。

なお、データセットの README には「1 回路 274〜275 CCX」とあるが、これは 8.5735 版の値（平均 275.0）で、8.6121 版では平均 272.7 である。

6.3.5　SomeJoint 版（$J = 5$）

$J = 5$ では word レジスタが 40 量子ビットになり、selection 11 と合わせて補助に使えるのは 77 本しかない。mrmeows 版のキャッシュ（78 + 7 = 85）は入らないため、この構成では同じキャッシュ配置を $J = 5$ にそのまま使えない。SomeJoint 版は、次の 2 点でこれを回避している（回路の命令列から確定。意図は推定）。

- **アドレスが少ない**：$R \cdot B$ が 1,144 でアドレスは 1,220。上位のキャッシュは 69 個で済み、下位 11 と合わせて 80 になる。
- **word の最上位ビットを一時領域に使う**：角度 word は 8 ビットと宣言されているが、値は $[-64, 63]$（角度 $[-\pi/2, \pi/2)$）に収まり、最上位ビットは常に次のビットと等しい。回路は最上位ビットを一時的に作業用に使い、最後に `CX` で次のビットからコピーして戻す。5 word の最上位ビットと補助 75 本で 80 本になり、HMR 80 個と一致する。

CCX は $k = 4$〜14 の 11 本が 283 個（126 量子ビット）、$k = 0$〜3 の 4 本が 305 個（121 量子ビット）で、合計 4,333、平均 288.9 である。条件付きの演算はなく、静的な数と平均が一致する。題名の “phased” に当たる測定と条件付き CZ による消去は、mrmeows 版と同じ技法である。word の値が 7 ビットの範囲に収まる性質は、mrmeows 版の word にも共通していた（確定）。

6.3.6　レジスタ使用の比較

図 6.8 は、invsqrt の図 4.1 と同じ形式で、baseline と SomeJoint の 10 月 28 日取得版の ANGLE 回路の量子ビットの使い方を比べたものである。baseline の QROM は、アドレスの木をたどりながら葉ごとに word を XOR するので、Toffoli はアドレス数にほぼ等しい（1 本 3,974）。SomeJoint の 10 月 28 日取得版は、キャッシュを先に作って混合積だけを word に足すので、1 本 283〜305 CCX で済む。

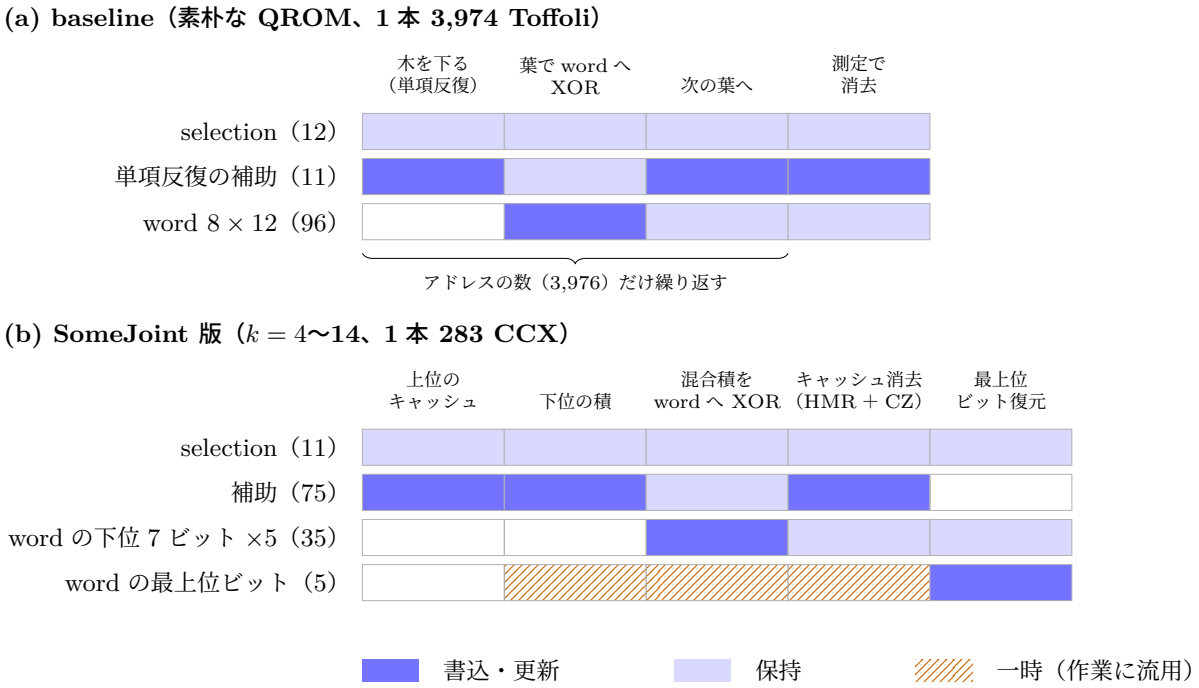


図 6.8　dfthcsos の ANGLE 回路のレジスタ使用の模式図。(a) は公式の生成器 `angle_k_reference_circuit.py` の手順、(b) は命令列の集計（HMR 80、CZ 80、word の最上位ビットの `R・HMR・CX` による復元）に基づく。(b) の段階の境界と最上位ビットを使う時期は推定を含む。mrmeows 版は (b) から最上位ビットの流用を除き、補助を 85 本にしたものに当たる。

6.4　費用分析と今後の最適化方向

因子分解のサイズを小さくすれば PREPARE の費用は減り得るが、再構成残差や Λ が増える可能性がある。$\lambda_{\mathrm{eff}} = \sqrt{E_{\mathrm{gap}}(2\Lambda - E_{\mathrm{gap}})}$ の Λ による偏微分は $E_{\mathrm{gap}}/\lambda_{\mathrm{eff}}$ なので、ギャップを固定しても正規化の改善は費用に効く。角度の word 幅 β を減らすと Givens と lookup の費用を下げられる一方、角度の刻みが変わり、係数の再フィットと残差の再検査が必要になる。J の変更は回路数と word レジスタ幅の両方を変える。128 量子ビットという ANGLE の制約と、全体モデルの Q をそれぞれ評価する必要がある。公開取得版の比較は、このような取引が実際に成立し得る例であり、特定の因子分解の最適性や下界を示すものではない。

演習：費用の再計算

表 6.4 の SomeJoint 10 月 30 日取得版について、$\lambda_{\mathrm{eff}} TQ$ を計算し、丸め誤差の範囲で 340,232,090 となることを確かめよ。mrmeows 版では、全 19 回路で各 1 CCX を削ると T は $2 \times 19 = 38$ 減る。その他の条件を固定したとき費用が約 0.197% 下がることを確かめよ。

第 7 章

公開回路の評価と検証方法

この章で学ぶこと

- 公開回路を版・検証器・パラメータとともに同定する。
- 全入力検査、確率的検査、解析モデルの保証範囲を区別する。

7.1　公開回路の再現方法

本書の数値は 2026 年 10 月 30 日取得版と、明示した 10 月 28 日取得の旧版を対象とする。mrmeows の公開データセットと SomeJoint のノートブック出力から該当版を取得する [3, 6]。ノートブックの題名や README の記録だけで同じ回路と判断せず、アーカイブと各回路の SHA-256、出力の版、取得日時を記録する。公開物が更新され旧版を取得できない場合、その数値は歴史的な報告として扱い、現在の版で再現できたと記載しない。

実行環境には KickMix、Python と依存パッケージ、公式 Verifier の版を固定する。既定設定で公式 Verifier を実行し、コマンド、salt、全出力 JSON、レジスタ宣言、提出係数データを保存する。公開回路は第三者の成果であり、回路作者と配布者は区別して記録する。

```
python -I verifiers/invsqrt/verify_submission_invsqrt.py invsqrt.kmb
python -I verifiers/dqi/verify_submission_dqi.py dqi.kmb
python -I verifiers/dfthcsos/verify.py submission_dfthcsos
```

これは競技配布物の展開先を作業ディレクトリとする呼出し例である。必要な依存関係はその版の配布手順に従う。dfthcsos には回路族と係数ファイルを含む提出ディレクトリを渡す。

7.2　全入力検証とランダム検証

invsqrt の全入力検査でも、すべての測定結果の組合せを列挙するわけではない。dqi では同じ回路でも salt により失敗件数と平均費用が変わり得る。検査分布を変えた独立テストは頑健性を調べる補助になるが、公式設定の費用とは分けて報告する。

表 7.1　既定検査の範囲。invsqrt の全入力は有効整数入力の全列挙を意味する。

課題	入力と検査	結論の範囲
invsqrt	$x = 1, \dots, 3{,}133{,}452$、各 64 ショット	有効入力は全列挙、測定分岐は標本検査
dqi	指定の誤り重み分布と一様な測定記録、102,400 ショット	失敗率の片側信頼上限
dfthcsos	ANGLE のアドレスと出力語の検査、係数再構成	lookup の正しさと残差・資源モデル

7.3　正しさと近似誤差

位相の比較には円周距離 $\min_{k\in\mathbb{Z}} |\theta - \theta_{\mathrm{target}} - k|$ を使う。値レジスタの一致、補助のゼロ復元、位相の一致をそれぞれ確認する。固定小数点では符号、切り捨て、オーバーフローも仕様の一部である。float64 の比較は境界付近で誤判定し得るので、余裕を残し、高精度演算や厳密な整数不等式で補強する。サンプルで一致した局所恒等式は、全到達状態でも一致するとは限らない。逆に、サンプル上で真理値ベクトルが線形独立なら、その関係する全入力上でも独立である。

7.4　解析的費用モデル

命令列から得る静的なゲート数、分岐を実行して得る平均、全アルゴリズムのモデル推定を区別する。dfthcsos の費用は、実際に検査する ANGLE 回路と、解析的に数える PREPARE・Givens などを組み合わせる。パラメータの変更で QROAM のブロックサイズが変わると、費用は滑らかに変化するとは限らない。線数は最大値を取る共有一時領域と永続領域から求め、各部分の最大値を無条件に足さない。

7.5　結果の信頼性と再現可能性

表 7.2　数値・主張に付ける区分。

区分	何を示すか
公式 Verifier 値	固定した検証器・設定・乱数条件での出力
厳密計算	明示した有限入力集合で整数・ビット演算により確認した性質
著者報告	公開作者の記述。独立再実行とは区別する
推定	仮定したモデルでの予測。実回路の検査が必要

本書に引き継いだ検証記録は macOS で公開ソースからビルドした KickMix を含む。競技の Linux バイナリとの同一性や、数学的な大域最適性を意味しない。公開データセット、ノートブック、論文それぞれのライセンス・引用条件を確認し、第三者のコードや図を再配布する場合はその条件に従う。本書の回路模式図は説明のためのものであり、公開コード一式の再配布ではない。

解析対象のハッシュ、保存された検証条件、探索パラメータと未記録の条件は、付録 E にまとめる。

第 8 章

3 課題の比較と今後の最適化課題

8.1　3 課題の比較

表 8.1　最適化の対象と実際に求める出力。

課題	中心となる計算	必要な出力	費用上の特徴
invsqrt	固定小数点近似と lookup	入力を保持した位相	幅と算術・回転の積
dqi	有限体上の復号	測定記録との内積の位相	幅の二乗と失敗率が効く
dfthcsos	因子分解と角度表	ANGLE の出力 word と係数データ	正規化・全体資源の積

比較例の参照比はそれぞれ約 20.29、59.00、9.86 で、幾何平均は約 22.77 である（2026 年 10 月 30 日取得版）。これらは特定版の比較であり、競技全体の現在の最良値でも理論下界でもない。

8.2　最適化で重要な共通原理

入力と補助の復元は最終出力の一部である。値の計算を完全に省略できなくても、その値を保存せず、必要な位相や選択操作へ直結すれば資源を節約できる。一方で逆計算は不要な相関を取り除くために必要であり、単に消してよい費用ではない。

非線形演算を減らすには、CX による線形変換を使い、AND や有限体乗算を共有する。共有する中間値が増えれば線数も増えるので、回路サイズ、寿命、再計算を同じ費用式で比較する。因子分解のサイズや近似精度の選択は、表の内容と回路構造を変える設計変数である。

8.3　残る研究課題

invsqrt では区間分割、係数表の符号化、Horner の精度配分を一体として設計することが課題になる。dqi では拡張 Euclid 法以外の復号、位相に必要な情報だけを求める方式、逆方向の省資源化が考えられる。dfthcsos では (R, B, C)、word 幅、1 回の読み出し word 数と lookup 合成を同時に選ぶことが重要である。どの課題でも局所探索で改良が見つからないことを、一般的な不可能性と取り違えない。

8.4　参加者への技術的示唆

最初にレジスタ契約と費用式を固定し、公開回路を同じ検証器で再現する。次に支配的な費用を分け、数式による誤差評価と小さな真理値表で候補を絞る。最後に、実際に生成した回路を公式設定で検証し、版・乱数条件・著作者を添えて比較する。改善率だけでなく、その値がどの計算範囲と検証条件で得られるかを説明できることが、再利用できる成果につながる。

付録 A

有限体 GF(2^m) の基礎

GF(2^m) は、GF(2) 上の次数 m の既約多項式 $p(z)$ を選んで

$$\mathrm{GF}(2^m) = \mathrm{GF}(2)[z]/(p(z))$$

と構成できる。元は次数 m 未満の多項式であり、その係数が m ビットに対応する。加算は XOR、乗算は多項式を掛けて $p(z)$ で割った余りである。例えば $p(z) = z^3 + z + 1$ なら $z^3 = z + 1$ なので $z^2(z+1) = z^3 + z^2 = z^2 + z + 1$ となる。$m = 12$ の体で使う既約多項式とビット順序は公式仕様に合わせなければならない。同じ抽象的な体でも基底を変えればビット列は変わり、dqi の測定記録とのビット内積にも影響する。非ゼロ元全体は乗法について巡回群を作り、原始元 α の累乗が $2^m - 1$ 個の非ゼロ元を列挙する。

付録 B

多項式 Euclid 法の計算例

GF(2) 上で $A = x^3 + 1$、$B = x^2 + 1$ とする。加算と減算が同じなので

$$A = xB + (x + 1), \qquad B = (x + 1)(x + 1) + 0$$

であり、最大公約式は $x + 1$ である。最初の余りは $x + 1 = A + xB$ と書け、これが Bézout 係数 $(1, x)$ を与える。dqi の鍵方程式では、係数体を $\mathrm{GF}(2^{12})$ とし、初期の 2 多項式を x^{70} とシンドローム多項式に置き換える。最大公約式まで進めず、余りの次数が 35 未満になった段階で係数多項式と余りを取り出す。誤りゼロの入力や次数の退化は分岐を含めて扱い、可逆回路ではそれらの分岐情報も復元する。

付録 C

用語・記号一覧

記号・用語	意味
Q	KickMix 回路の論理量子ビット数。物理誤り訂正の符号化倍率を含まない
T	本書の資源式では Toffoli 様演算数。T ゲート個数とは区別する
C（invsqrt）	回転合成を含む Toffoli 換算費用
C（dfthcsos）	因子分解の添字サイズ。上の換算費用とは別の量
$R(\varepsilon_+)$	dqi の失敗率信頼上限による繰り返し費用係数
R, B, C	dfthcsos の因子分解サイズ
β	invsqrt では位相係数、dfthcsos では角度 word 幅
J	ANGLE lookup 1 回で読み出す word 数
$\lambda, \Lambda, \lambda_{\mathrm{eff}}$	LCU の正規化、SOS 表現の正規化、スペクトル増幅後の費用係数。定義を各式で区別する
word	一つの数値を表す固定長ビット列。角度 word は整数 w で回転角を表し、1 word あたり β 量子ビットを使う
Givens 回転	二つの軌道成分を混合し、ほかの成分を変えない回転
マヨラナ作用	系の状態にマヨラナ演算子を掛けること。SELECT 内で基底回転と組み合わせる
QROM / QROAM	古典データ表を量子アドレスから読み出す回路方式
uncomputation	中間値と補助を可逆処理や測定・補正で消すこと

付録 D

Verifier とスコアの定義

提出費用を c_i、参照費用を c_i^{ref} とすると、各課題の比は $r_i = c_i^{\mathrm{ref}}/c_i$、総合比は $(r_1 r_2 r_3)^{1/3}$ である。

- invsqrt：入力と補助を復元し、全有効入力の各検査ショットで位相円周距離が $2^{-24}/\sqrt{x}$ 以下であること。費用は第 3 章の QC。
- dqi：値または位相の失敗率の片側信頼上限 ε_+ が $1/16$ 未満であること。費用は $R(\varepsilon_+)Q^2T$。試験分布と信頼度は第 5 章。
- dfthcsos：係数再構成残差の各精度条件、正規化・ギャップ条件、ANGLE のレジスタ契約と 128 量子ビット制約を満たすこと。費用は $\lambda_{\mathrm{eff}} TQ$。全体の Q と ANGLE 単体の 128 という制約を混同しない。

これらは 2026 年 10 月 30 日取得の仕様に対応する。別版の検証器を用いるときは、その版の定義と既定値を記録して比較する。

付録 E

解析対象と再現条件

本付録は、本文の数値を同定できる範囲と、再現に残る制約をまとめる。回路の検証値、命令列の静的集計、数値探索の参考値は同じ保証ではない。以下は保存された検証出力・取得記録と、確認できた数値モデルの設定を整理したもので、本書の改訂時に全回路・全探索を再実行した結果ではない。

E.1　回路と提出物の識別子

回路のハッシュは `.kmb` ファイルの SHA-256 である。dfthcsos の識別子は、公式 Verifier が回路族・因子分解・木をまとめて計算する `submission_sha256` であり、単一の係数ファイルのハッシュとは異なる。公開データセットとノートブックの取得元は文献 [3, 6]。

表 E.1: 本文で比較する提出物の識別子と保存検証条件

対象	識別子・条件
invsqrt 63Q・41 回転版（10 月 28 日取得）	`0766a111a76f241aa57fbbed466436d743be13436943ddb0628c291ca07d67d6` $x = 1, \ldots, 3{,}133{,}452$、各 64 ショット、$n_p = 10, \beta = 1, \eta = 218$、相対許容 2^{-24}。salt は `quantum-circuit-local-v1`。保存検証費用 103,546.204。本文の 103,546.17 は作者報告を丸めた値。
invsqrt busyaprime 改良版（10 月 30 日取得）	`9a8627c812833a6feca6eb82239f38d1a9a5e1eee299a9a7af9abd85e36339be` 200,540,928 ショット、$Q = 63$、一般角 41 回転。費用 102,349.236。取得・検証記録には既定条件とあるが、salt を含む完全なコマンドは当該出力 JSON に含まれない。
dqi 889Q（10 月 30 日取得）	`6349d2bc82eb1ef3a4e8c52c71a312823d756532517463b5e18eb25cc81195a8` 102,400 ショット、既定 salt（空文字列）と記録。値エラー 0・位相エラー 1、平均 Toffoli 437,947.606、費用 414,211,737,069.878。別の 889Q 版と混同しない。
dfthcsos $(R, B, C) = (8, 143, 60)$（10 月 30 日取得）	`0d78fc89bacf4d4def8d596ef6d2bd3f1058200ce663f023aca1d5d60b789c85` $\beta = 8, J = 5$、15 回路、78,080 ショット、$T = 16{,}276, Q = 430$。費用 340,232,090.430。回路単体は 126 量子ビット。salt の独立した完全記録は当該出力 JSON にない。

E.2　Verifier の版と実行環境

保存されたハッシュ一覧にある公式スクリプトの指紋を示す。invsqrt の指紋は 10 月 28 日版の実行 receipt にも含まれる。スクリプト単体のハッシュだけでは、インポート先、数値ライブラリ、KickMix バイナリまで同定したことにはならない。

公式スクリプト	保存版の SHA-256
invsqrt	`ddef520dec6b68d1cf0a6f328ddd36d79eb84570b9192cf3f7acaf073f06733a`
dqi	`f14274bc0943fb9769ddb01131be6063191a15009fe1caf1d4bdd9fd25194e65`
dfthcsos	`0775be7a528922367caebe9e4d051d6199811f1aefeaa4b793eaaa4621b8ec9d`

検証環境は公開ソースからビルドした macOS 版 KickMix である。全依存関係のバージョン固定表、ビルドしたソースの完全なコミット指紋、全実行の明示 seed は、確認できた保存記録だけではそろわない。10 月 30 日の各出力に Verifier のソースハッシュ自体は含まれないため、上表だけをもって全実行環境の同一性を保証しない。公式の seed 導出は回路・提出物のハッシュと salt に基づくが、再実行時には salt と実装版も一致させる必要がある。新しい検証では、公式配布物一式・依存関係・seed/salt・コマンドをまとめて記録する。

E.3　Walsh 打ち切りと係数探索の条件

■22 ビット Walsh の表　入力集合は $0,\dots,2^{22}-1$、評価する有効範囲は $1,\dots,3{,}133{,}452$。$\theta(0)=0$ とし、範囲外の正の整数にも $1/\sqrt{x}$ を定義する continued 拡張を使う。非正規化 FWHT の結果を 2^{22} で割り、定数以外の係数を絶対値の降順に残す。同値の係数は mask 添字順で選ぶ。乱数は使わない。anchored と global_offset の再構成は本文の定義で区別し、円周誤差を float64 で評価する。表の k は残す非定数係数の個数である。この計算は数値的な候補選別であり、公式回路の PASS や数学的な誤差証明ではない。

■区分 2 次 Horner の表　オクターブ 1 ビットと先頭の 1 の直後の $A-1$ ビットをアドレスにし、中心化した区間変数 d を使う。実係数は各区間最大 1,024 点を等間隔に選び、相対誤差に対する 40 回の再重み付き最小二乗で求める。整数化では C_1, C_2 の最近傍から各 ± 2 の 25 候補を調べ、C_0 は相対誤差の正負を均衡させる探索後の隣接整数を比較する。区間最大 512 点と最悪点を用い、最大 3 回の再探索を行った後、全有効入力で誤差を評価する。

保存された中心化・octave 設定で、目標誤差比 0.9 に対する例は

A	F	k_1	k_2	$\rho_{\max}$
7	26	12	14	0.896948
8	25	12	13	0.873305
9	25	11	12	0.896867

である。整数演算のシフトは floor 切り捨て、最後の比較は float64。区間割りを変えた別の探索結果と混同しない。全係数表と数値線形代数ライブラリの版を本書には収録していないため、ビット単位で同一のフィットが得られるという保証はない。

■ANF lookup の費用推定表　入力は上記の整数係数表の符号・絶対値をビット列へ分解したもの。生の列数は $A=8$ で 54、$A=9$ で 51 であり、公開回路の符号化後 35 列とは別の表である。パリティ積の候補を作るモデルの既定 seed は 20261027、候補探索の既定反復数は 3、各出力で試す積数の上限は 40 と確認できる。しかし、本文の全行に対応する実行時 seed・係数表・探索設定を一対一に同定する完全な記録は確認できない。したがってこれらはモデル内の参考上界であり、特定の公開回路の下界でも、第三者が同じ最良値へ必ず到達する保証でもない。

第 2・3 章のこの種の表は、計算方法と費用の取引を理解する教材として用いる。第 6 章の全体資源は ANGLE の回路評価と解析的モデルの合計、第 7 章の検証区分は上記の保証範囲に従う。記録がない条件を後から設定した値で補って、元の数値を再現したと扱うことはしない。

参考文献

[1] T. Khattar, N. C. Rubin, C. Gidney, M. Broughton, G. H. Low, R. Babbush, *Leveraging verifiable structure in quantum circuits to find more efficient compilations with agentic AI*, arXiv:2610.23832. `https://arxiv.org/abs/2610.23832`.

[2] Google Verifiable Quantum Circuit Compilation（Kaggle コンペティション）. `https://www.kaggle.com/competitions/google-verifiable-quantum-circuit-compilation`.

[3] mrmeows, *21.0 circuits, let's all reach the plateau!*（Kaggle ノートブック）および データセット `mrmeows/qcc-1375-open-circuits`. `https://www.kaggle.com/code/mrmeows/21-0-circuits-let-s-all-reach-the-plateau`.

[4] manwithacat, *Is the 63-Qubit Inverse Square Root Optimal?*（Kaggle ノートブック）. `https://www.kaggle.com/code/manwithacat/is-the-63-qubit-inverse-square-root-optimal`.

[5] A. Gu, M. Kemeny, S. Yelin, M. Lukin, *Discovering state-of-the-art quantum circuits with large language models*, arXiv:2610.22367. 実装 `https://github.com/andigu/uncompute`.

[6] SomeJoint（Agniv Banerjee ほか）, *quantum-circuits-20-96-local-gm-confluent-dqi*（Kaggle ノートブック。2026 年 10 月 28 日取得時の題名は “22.44 Local GM: Our Chemistry”。10 月 30 日にも出力を取得）. `https://www.kaggle.com/code/somejoint/quantum-circuits-20-96-local-gm-confluent-dqi`.

[7] J. Proos, C. Zalka, *Shor's discrete logarithm quantum algorithm for elliptic curves*, Quantum Inf. Comput. **3**, 317–344 (2003). arXiv:quant-ph/0301141.

[8] T. Khattar, N. Shutty, C. Gidney, A. Zalcman, N. Yosri, D. Maslov, R. Babbush, S. P. Jordan, *Verifiable quantum advantage via optimized DQI circuits*, arXiv:2510.10967 (2025).

[9] Y. Sugiyama, M. Kasahara, S. Hirasawa, T. Namekawa, *A method for solving key equation for decoding Goppa codes*, Information and Control **27**, 87–99 (1975).

[10] R. T. Chien, *Cyclic decoding procedures for Bose–Chaudhuri–Hocquenghem codes*, IEEE Trans. Inf. Theory **10**, 357–363 (1964).

[11] G. D. Forney, *On decoding BCH codes*, IEEE Trans. Inf. Theory **11**, 549–557 (1965).

[12] G. H. Low, R. King, D. W. Berry, Q. Han, A. E. DePrince III, A. F. White, R. Babbush, R. D. Somma, N. C. Rubin, *Fast quantum simulation of electronic structure by spectrum amplification*, Phys. Rev. X **15**, 041016 (2025).

[13] T. Shirakawa, H. Ueda, S. Yunoki, *Automatic quantum circuit encoding of a given arbitrary quantum state*, Phys. Rev. Research **6**, 043008 (2024). arXiv:2112.14524.

[14] J. Welch, D. Greenbaum, S. Mostame, A. Aspuru-Guzik, *Efficient quantum circuits for diagonal unitaries without ancillas*, arXiv:1306.3991.

[15] M. Amy, P. Azimzadeh, M. Mosca, *On the CNOT-complexity of CNOT-phase circuits*, arXiv:1712.01859.

[16] G. H. Low, I. L. Chuang, *Hamiltonian Simulation by Qubitization*, Quantum **3**, 163 (2019). `https://doi.org/10.22331/q-2019-07-12-163`.

[17] D. W. Berry, A. M. Childs, A. Ostrander, G. Wang, *Quantum algorithm for linear differential equations with exponentially improved dependence on precision*, Commun. Math. Phys. **356**, 1057–1081 (2017). `https://arxiv.org/abs/1701.03684`.

[18] J.-P. Liu, H. Øie Kolden, H. K. Krovi, N. F. Loureiro, K. Trivisa, A. M. Childs, *Efficient quantum algorithm for dissipative nonlinear differential equations*, PNAS **118**, e2026805118 (2021); corrected arXiv version v4 (2026). https://arxiv.org/abs/2011.03185v4.

[19] J. Boyar, R. Peralta, *Concrete Multiplicative Complexity of Symmetric Functions*, MFCS 2006 (2006). https://www.nist.gov/publications/concrete-multiplicative-complexity-symmetric-functions.

[20] R. Mirwald, C.-P. Schnorr, *The multiplicative complexity of quadratic boolean forms*, Theor. Comput. Sci. **102**(2), 307–328 (1992). https://doi.org/10.1016/0304-3975(92)90235-8.