数理解析研究所講究録
1120巻 1999年 110-119

量子計算機シミュレーションシステム

Quantum Computer Simulation System

徳永 裕己　　長井 歩　　今井 浩
Yuki TOKUNAGA　Ayumu NAGAI　Hiroshi IMAI

東京大学大学院理学系研究科情報科学専攻
〒113-0033 東京都文京区本郷 7-3-1
{tokunaga,nagai,imai}@is.s.u-tokyo.ac.jp

摘要: 自然数の因数分解の多項式時間アルゴリズムなど，量子計算の理論的有用性が示されて久しい．しかし，現在多数の量子ビットをもつ量子コンピュータは存在しない．このようなとき，汎用的に量子コンピュータを模法するシミュレータをつくり，実際のアルゴリズムの振る舞いや実際的な計算量を検証することの意義は大きい．我々は量子計算において頻繁に用いる基本的ユニタリ変換の操作の計算量が少なくなる方法を考案し，汎用的シミュレータを開発した．そしてこれを用いて，一般の自然数に対する因数分解の量子アルゴリズムを実装し，シミュレーション実験を行った．本研究ではまず汎用的シミュレータの作成手法を述べ，次に実験から得た，アルゴリズムの振る舞いや性質について考察する．

キーワード: 量子コンピュータ，量子計算，量子アルゴリズム，Shor の因数分解アルゴリズム，シミュレータ

1 はじめに

現在，量子コンピュータの実現の研究はさかんに行われているが，それはいくつかの有用性が理論的に示されたからである．1982年，R. P. Feynman は現状のコンピュータで量子力学の現象を模倣するには指数時間を要することを指摘した [8] ．これをうけ，D. Deutsch は量子力学を原理とした計算モデル（量子 Turing 機械，量子回路）を提案し [4, 5]，量子計算の研究が始まった．

1994年，P. Shor は量子計算を用いれば，自然数の因数分解を多項式時間で行えること [14, 15] を示し，注目を集めた．なぜなら，現在の RSA 公開鍵暗号系は因数分解の計算量的安全性が根拠にあったからである．よって，もし実用的な量子コンピュータが実現すると，RSA 暗号は安全性の根拠を失うことになる．

また，1996年の Grover の量子探索アルゴリズム [9] も興味深い．これは n 個の未整列なインデックスから，あるインデックスを探すアルゴリズムである．現状のコンピュータでは平均で $n/2$ 回の探索をするが，量子計算では $O(\sqrt{n})$ 回で探すことができる．これは“振幅の増幅”という手法を用いており，量子アルゴリズムの例としても非常にわかりやすく[1]，SAT など様々な問題に応用ができる．

その他，通信の面でも秘密鍵なしに安全に情報を伝送できる量子暗号というアイデアも出ており，また Communication Complexity や オートマトンにおいても最近，現状のモデルとの計算量の指数的なギャップが発見されている．

以上のような量子計算の有用性を引き起こしているものはおおまかには２つある．一つは複素数の重みがつく“重ね合わせ”を用いた並列計算である．もう一つは重みの“打ち消し合い”などによる干渉効果である．この干渉効果は量子計算に非常に特徴的である．

しかし，量子状態は非常に不安定なため，現状のところ１量子ビットをどう作成するか，そして複数量子ビットをどう安定させるかの実験段階であり，多数の量子ビットを安定させる技術は見つかっておらず，実現の見通しはたっていない．このようなとき，既存の計算機を用いて量子計算のシミュレーションをする意義は大きい．それにより，アルゴリズムが実際にどう振る舞うかという認識が得られ，新たなアルゴリズムの開発に役立つことが考えられ

[1] 今回，土村氏に Web 上にデモを作成していただいた．[16]

る．また，実際的な計算量はどのくらいになるのかという，実用上の指針が得られる．さらに，エラーがどのくらい大きくなるかをシミュレートし，許容誤差を調べることは，量子コンピュータの実現にも大きく役立つと考えられる．

そこで我々は通常のワークステーション上で動く汎用的な量子コンピュータのシミュレータを開発した．このシミュレータは量子計算のモデルである量子回路を模倣するよう構成されている．量子回路を選んだのはアルゴリズムの記述に優れているためである．シミュレータ作成においては，量子計算の基本的な操作であるユニタリ変換の計算量が少なくなる方法を考案した．またこのシミュレータを用いて，Shor の因数分解アルゴリズムを一般の自然数の入力に対して可能となるよう実装し，シミュレーション実験を行った．本論文ではまずシミュレータの作成手法について述べる．そしてシミュレーション実験の結果とそこから得られた考察について報告する．

Shor のアルゴリズムは確率的アルゴリズムであり，解を出すのに失敗することもある．しかし，シミュレーションの結果，その成功確率は理論上の保証に比べて，実際上はかなり大きいことが分かった．また，状況により成功確率が異なることもはっきりと確かめられ，いくつかの場合について解析を加えた．

本論文の構成は次の通りである．まず，2 章で量子計算の一般的な原理を説明する．3 章で量子計算のモデルである量子回路について解説する．4 章で量子計算のシミュレータの作成方法について述べる．5 章で自然数の因数分解アルゴリズムのシミュレーション結果と考察を示し，6 章で今後の課題等について述べる．

2 量子計算の原理

古典計算機と量子計算機の違いを理解するためには，まず，1 ビットを考えるとよい．古典ビットは“真”と“偽”の 2 状態のうちのどちらか一つをとる．古典的な“確率的”ビットは確率 α で真をとり，確率 β で偽をとり，$\alpha + \beta = 1$ という性質をみたす．量子ビット (qubit) は後者に非常によく似ている．量子ビットに対しては，α, β は任意の複素数をとることができ，$\|\alpha\|^2 + \|\beta\|^2 = 1$ という性質をみたす．量子ビットを観測すると，確率的ビットのように確率 $\|\alpha\|^2$ で真をとり，確率 $\|\beta\|^2$ で偽をとる．しかし，量子計算機をモデル化したとき使用可能な変換の集合は確率的計算機に比べて大きい．ここに量子計算機の能力の所以がある．

より一般的に n ビットを考える．古典 n ビットは $m = 2^n$ 個の状態のうちのどれか一つをとりうる．量子 n ビットは m 個の基底状態をとる．基底状態を $|q_1\rangle, |q_2\rangle, \ldots, |q_m\rangle$ と記す．ψ を複素数の係数をもつこれらの線形結合とする．

$$\psi = \alpha_1|q_1\rangle + \alpha_2|q_2\rangle + \ldots + \alpha_m|q_m\rangle.$$

ψ の l_2 ノルムは，

$$\|\psi\| = \sqrt{|\alpha_1|^2 + |\alpha_2|^2 + \ldots + |\alpha_m|^2}.$$

量子計算の状態は $\|\psi\| = 1$ をみたす任意の ψ をとりうる．ψ は $|q_1\rangle, |q_2\rangle, \ldots, |q_m\rangle$ の基底状態の“重ね合わせ”と呼ばれ，$\alpha_1, \ldots, \alpha_m$ は“振幅”と呼ばれる．$l_2(Q)$ を $|q_1\rangle, |q_2\rangle, \ldots, |q_m\rangle$ で張られる複素内積空間とする．

任意の複素数の振幅を用いられることは量子計算の有用性の本質となっている．例えば“負”の実数を振幅に用いることによって，“正”の実数との“打ち消しあい”が起こる．これは確率的計算機にはなかったことである．

量子計算は 2 種類の変換をする．一つはユニタリ変換である．ユニタリ変換は $l_2(Q)$ 上の線形な変換 U であり，l_2 ノルムを保存する．(これは ψ が ψ' に写されたとき $\|\psi\| = \|\psi'\| = 1$ ということである．)

二つめは観測である．観測は $\psi = \alpha_1|q_1\rangle + \alpha_2|q_2\rangle + \ldots + \alpha_m|q_m\rangle$ という重ね合わせのとき，確率 $\|\alpha_i\|^2$ で $|q_i\rangle$ を与える．($\|\psi\| = 1$ により異なる出力の確率の和は 1 であることが保証される．) 観測後，状態は $|q_i\rangle$ に写る．

3 量子回路

量子回路 [2, 5] は具体的な量子計算の操作の表現，アルゴリズムの表現に優れている．量子回路では 1 量子ビットを横線で書きワイヤと呼ぶ．時間は左から右へ流れているとする．そしてその線の上に 1 量子ビットに対するユニタリ変換を四角や丸でワイヤの上に書き，これを量子ゲートと呼ぶ．複数ビットに相互的に作用するときは相互作用するワイヤ間を縦線でつなぐ．各ビットはゲートを通過すると，そのユニタリ変換を受ける．

n 量子ビットは 2^n 次元ベクトルであるので，これに対する操作は $2^n \times 2^n$ ユニタリ行列で表す．任意の $2^n \times 2^n$ ユニタリ行列は以下の 2 種類の“基本ゲート”に分解できることが知られている [2]．量子

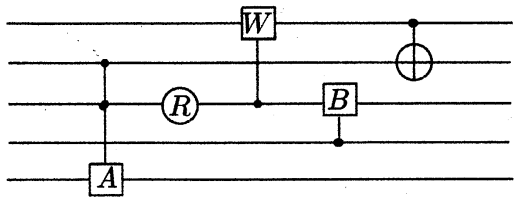

図 1: 量子回路

回路の計算時間は以下の 2 種類の基本ゲートの使用個数と定められる.

1. U ゲート
   1 量子ビットに対するユニタリ変換は任意の 2×2 ユニタリ行列で表せる. これを U ゲートと呼ぶ.

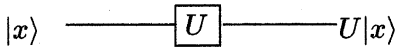

図 2: U ゲート

2. 制御 NOT ゲート (Controlled-NOT)
   制御 NOT ゲートはビット間の相互作用を行う. 制御 NOT ゲートは図3のように書く. • は制御ビットを表していて, 制御ビットが 1 のときのみ, 第 2 ビットに NOT を施す.

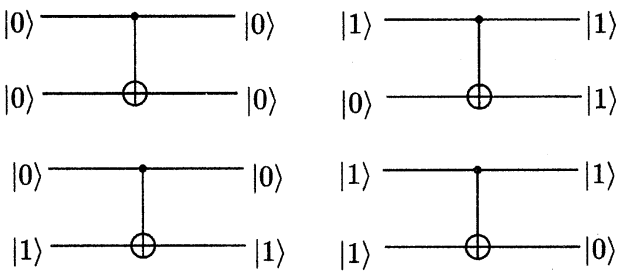

図 3: 制御 NOT

通常, 2 量子ビットの基底は

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

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

と取るので, 制御 NOT ゲートを行列で表すと以下の 4×4 行列になる.

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

また, 5 章で詳しく述べるが量子アルゴリズムにおいては

3. Walsh-Hadamard 変換
4. 離散フーリエ変換

という操作が並列, 干渉の効果の本質を担っている. 3, 4 はもちろん 1, 2 を用いて作成できるが頻繁に用いるため, パッケージ化しておくと望ましい. また,

5. 論理演算

も 1, 2 をもちいて簡単に作成できるが, これもパッケージ化した. よってこれらの 5 つの操作を合わせて本シミュレータでの "基本操作" とする.

4 汎用的シミュレータ

4.1 背景

一般的な量子コンピュータのモデルとしては量子 Turing 機械 と量子回路の二つがあるが, アルゴリズムの実装のしやすさから量子回路をシミュレートするのが実用的である.

量子コンピュータの計算過程とは n ビットあるとき, 2^n 個の基底状態の振幅が移り変わっていく様子である. よって 2^n 次元の配列ベクトルを用意し, それぞれの基底の振幅を保存する. 一般的な問題に対応するためには, 2^n 個の基底状態の振幅を保存することは避けられない. よってこのシミュレータの計算に要する記憶容量, 計算時間は $\Omega(2^n)$ の指数関数になることは避けられない.

そして計算操作はこのベクトルに $2^n \times 2^n$ ユニタリ行列をかけることで計算を表すことができる. これを単純に考えると, 量子回路で表された操作をそれぞれ $2^n \times 2^n$ ユニタリ行列に表し, 2^n 次元のベクトルにかけていけば量子計算のシミュレートができる. しかし, $2^n \times 2^n$ の行列を生成すると 2^{2n} の行列要素を保存する記憶容量が必要となる. そこで我々は量子回路の特徴を活かし, シミュレータの

1ステップの基本操作の演算子に必要な記憶容量を$O(n)$に，計算時間を$O(n2^n)$に押さえる方法を考案した．以下，これらの計算手法を記す．

4.2 計算手法

1. kビット目にUゲート(2×2ユニタリ行列U)をかける

   このゲートは$2^n \times 2^n$のユニタリ行列であらわすと，

   $$\otimes_{i=k+1}^{n-1} I \otimes U \otimes_{i=0}^{k-1} I$$

   となる．この行列は以下のような規則的な構造をした非常に疎な行列である．

   $U \otimes_{i=0}^{k-1} I = X =$

   $$\left.\begin{pmatrix} u_{11} & & u_{12} & & & \\ & u_{11} & & u_{12} & & \\ & & \ddots & & \ddots & \\ & & & u_{11} & & u_{12} \\ u_{21} & & u_{22} & & & \\ & u_{21} & & u_{22} & & \\ & & \ddots & & \ddots & \\ & & & u_{21} & & u_{22} \end{pmatrix}\right\} 2^k$$

   $\otimes_{i=k+1}^{n-1} I \otimes U \otimes_{i=0}^{k-1} I = \otimes_{i=k+1}^{n-1} I \otimes X =$

   $$\left.\begin{pmatrix} X_{(1)} & & & \\ & X_{(2)} & & \\ & & \ddots & \\ & & & X_{(2^{n-k})} \end{pmatrix}\right\} 2^n$$

   1行におけるU行列の要素は2つでそれ以外の要素はすべて0となる．（行列Uが2^{n-1}個埋め込まれた形をしている．）よってこの$2^n \times 2^n$行列を生成する必要はなく，記憶要領には，2×2行列のみを保存し，規則性を利用して，U行列の要素があたる部分のみを繰り返し，計算すれば計算時間は$O(2^n)$となる．

2. lビット目を制御ビット，mビット目を標的ビットとして制御Uゲートをかける．

   これは1ビット目が1のときのみmビット目にUゲートをかける操作である．もし，UがNOTゲートならば制御NOTゲートとなる．基本的には1と同様の形をしている．U行列のu_{11}要素がある行の行番号の2進表現のm桁目が1ならばそのU行列を計算し，行番号の2進表現のm桁目が0ならば恒等変換をする（計算をしない）．恒等変換がふえる分，制御UゲートはUゲートよりも計算時間は短い．

   (制御)^2Uゲートのような制御が2つ以上かかる場合も同様に計算できる．

3. Walsh-Hadamard変換

   各ビットに順々に

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

   をかける操作である．計算量は$O(n2^n)$．ただし，このシミュレータでは以下の高速フーリエ変換の手法を利用してパッケージ化しており，計算量は同じだが計算時間は2/3程度に減少している．

4. 離散フーリエ変換

   nビット（2^n個の状態）に対しての離散フーリエ変換の量子回路による表現は2種類提案されている．

   (a) nビットのみをつかう方法[7].

   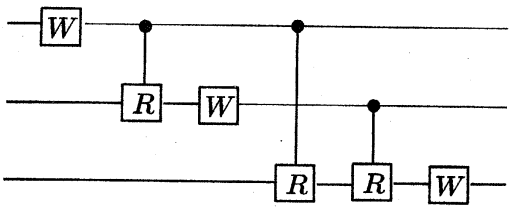

   図4: DFT1

   図4で，WはWalsh-Hadamard変換である．Rは上のk番目のビットを制御ビットとし，下のj番目のビットを標的ビットとした制御Uゲートであり，行列で以下のように表せる．

   $$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\pi i/2^{k-j}} \end{pmatrix}$$

   量子回路における計算量は$n(n+1)/2$で，シミュレータの計算量は$O(n(n+1)\cdot 2^n)$である．

   (b) 補助ビットを用いる方法．全体で$n+1$ビットを用いてnビットの離散フーリエ変換をする[12].

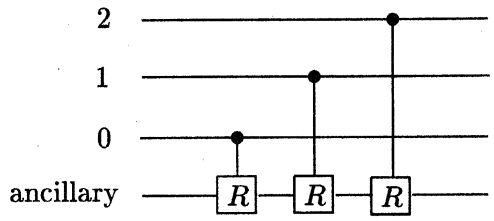

図 5: DFT2

図5で R は k 番目のビットを制御ビットとし，補助ビットを標的ビットとする制御 U ゲートであり，行列で以下のように表せる．

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\pi i/2^{n-k-1}} \end{pmatrix}$$

量子回路における計算量は n であり，シミュレータでは $O(n2^{n+1})$ である．シミュレータでは1ビット増やすとすべての操作に2倍のメモリを要するので望ましくない．

本シミュレータでは量子回路の表現とは異なるが，結果的に離散フーリエ変換の操作ができれば良いので通常の高速フーリエ変換を利用した．2^n 個の状態に対しての高速フーリエ変換の計算量は $O(n2^n)$ である．(a) と比べてオーダで n の差がでる．最近，量子回路において $O(n)$(シミュレートすると $O(n2^n)$ となる) のサイズの回路も考案されたが，本シミュレータでは Walsh-Hadamard 変換，離散フーリエ変換ともに高速フーリエ変換に対する高速化の手法をさらに用い，パッケージ化してあるため，ストレートに量子回路どおりにかけるより関数呼びだしにかかる時間等など無駄な時間がすくなくなっている．これにより 1.5 倍ほど早くなった．この比較の実験結果を表1に示す．計算機環境は 4.3節のとおりである．

5. AND OR NOT SWAP などの論理ゲート

   NOT，制御 NOT を用いて量子回路でも簡単に記述できるが論理計算としてよく用いることがあるためこれもパッケージ化して効率化してある．

6. 観測

bit	FFT(s)	DFT(s)	WH-FFT(s)	WH(s)
10	0.02	0.02	0.02	0.01
12	0.02	0.09	0.03	0.03
14	0.11	0.45	0.07	0.08
16	0.44	2.12	0.27	0.46
18	1.73	10.94	1.06	1.95
20	8.31	59.51	5.04	9.32
22	36.97	283.57	22.59	41.51
23	74.25	649.35	46.04	85.47
24	149.70	1415.61	93.68	177.54
25	345.25	3061.66	202.02	370.63

表 1: FFT と DFT の比較

それぞれの基底状態の振幅の絶対値の2乗を順に足しながら配列に蓄え，[0,1] の範囲のリストをつくる．そして，[0,1] の間の乱数を発生させて，でた値の範囲にある候補を観測値とする．

4.3 計算機環境

実験は Solaris2.6，Ultra SPARC II 360MHz，Memory 2GB にて行なった．言語は C++ を使用した．コンパイラは g++ を用い，-O2 オプションにより最適化を行った．

一つの振幅保存に対して実部，虚部の二つの倍精度浮動小数点を用いるため $2 \times 8\text{B} = 2^4\text{B}$ のメモリを使う．途中計算のために $2+\alpha$ 倍のメモリが必要である．よって，$2\text{GB}=2^{31}\text{B}$ のメモリで可能な量子ビット数は以下から 25 ビットとなる．

$$2^4 \times (2+\alpha) \times 2^{25} < 2^{31}$$

また 32bit 計算機上で実行するとメモリのアドレス空間を $2^{32}\text{B}=4\text{GB}$ までしか認識できない．よって CPU の性能による使用可能メモリ最大数は 4GB である．

5 Shor の因数分解アルゴリズムの実験

1994 年 P. Shor により自然数の因数分解を確率的多項式時間で計算する量子アルゴリズムが考えられた [14]．計算量クラスとしては因数分解の判定問題は ZQP (Zero-error Quantum Polynomial time) に属する．

[1] のシミュレーションでは簡単な場合のみに限っていた．我々は一般の自然数に対して因数分解でき

るよう実装した．因数分解する数を n として $n^2 \leq q$ となる状態数 q を用意するので $\sqrt{2^{25}} \simeq 5792$ 程度の因数分解が今回の実験では上限となる．

本章の流れは以下のとおりである．5.1節で因数分解のアルゴリズムをその実装方法を交えながら述べる．アルゴリズムの正当性については Shor の原論文 [15] を参照していただきたい．5.2節で Shor のアルゴリズムにおいて複素数の振幅がどのように変化するかをシミュレートして得たデータを視覚的に紹介する．5.3節で Shor のアルゴリズムの成功確率について理論的評価と実際的評価の比較をする．

5.1 因数分解のアルゴリズム

全体を流れに沿い，7ステップにわたって詳述するが量子計算で行うのはステップ4のみであることに注目する．

1. n が素数かどうか素数判定法を用いて調べる．素数と判定されたら出力して終了．このシミュレータでは5000程度までの因数分解しか行なえないため，Rabin の確率アルゴリズムは用いず単純に $\sqrt{n}$ までの数で割ってみる決定性アルゴリズムを用いた．

2. n が素数の自然数乗か $\sqrt{n}, \sqrt[3]{n}, \ldots, \sqrt[\lfloor \log n \rfloor]{n}$ を計算する．素数の自然数乗と判定されたら出力して終了．

3. ランダムに自然数 x $(1<x<n)$ を選び，Euclid の互除法により $\gcd(n,x)$ を計算する．$\gcd(n,x) \neq 1$ ならば $\gcd(n,x)$ を因数として出力し，n を $n/\gcd(n,x)$ に置き換えステップ1に戻る．

4. $x \pmod n$ の位数 r $(x^r \equiv 1 \pmod n$ となる最小の $r)$ を量子計算で発見する．$x^r \pmod n$ を r の関数とみれば位数が周期となっていることに注意する．Shor の量子アルゴリズムの本質となっているのはこの周期関数の周期を発見することである．ある周期関数の周期 r がわからないとき，通常は入力を1ずつ増やしていき，同じ出力を得られたとき，知ることができるがこれは r 回のステップが必要である．これを量子計算では $O(\log \log r)$ の繰り返しで高確率で知ることができる（5.3節を参照）．

   (a) $n^2 \leq q \leq 2n^2$ をみたす2のべき乗 q をとる．($n^2 \leq q$ をみたすもっとも小さい q をとる．)

   (b) $\log q$ ビットで q 状態数の量子コンピュータを生成し，初期状態を0のみ振幅1であとは0とする．Walsh-Hadamard 変換を行い等重の並列状態にする．

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle \qquad (1)$$

   (c) $x^a \pmod n$ を計算する．この計算はメモリのオーバフローを避けるため，"$x^i \equiv b \pmod n$ ならば，$x^{i+1} \equiv bx \pmod n$" という性質を用いて計算している．この結果は第2レジスタに蓄える．第2レジスタも量子ビットで表現するとさらに $\log n$ ビット必要となり，シミュレータに必要な記憶容量は n 倍になってしまう．ところが第2レジスタはメモリ的に使われていて，異なる重ね合わせ状態を増やしてはいないのでシミュレータでは量子ビットで表現する必要はない．よってこれを別メモリに蓄え，量子ビット数は増やさないことでシミュレータに必要な記憶容量を2倍に節約する．

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle |x^a \pmod n\rangle. \qquad (2)$$

   (d) 第2レジスタを観測する．観測により状態は収縮するので以下のような状態に変更する．ある k が得られたとすると，量子コンピュータの第1レジスタは $x^a \equiv k \pmod n$ となる a 以外の振幅はすべて0になる．$x^a \equiv k \pmod n$ となる a の数が A 個だとすると，それらの振幅は $\sqrt{A}$ となる．それらは位数 r を周期をして等間隔に並ぶ．

$$\frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |a_0 + jr\rangle |k\rangle. \qquad (3)$$

   (e) 離散フーリエ変換をかける．この結果状態は以下のようになる．

$$\frac{1}{\sqrt{qA}} \sum_{c=0}^{q-1} \sum_{j=0}^{A-1} e^{2\pi i (a_0 + jr)c/q} |c\rangle |k\rangle. \qquad (4)$$

   (f) 第1レジスタを観測する．

5. ステップ4-(f) で得た測定値が c であったとき c/q を連分数展開により分母が n より小さいあ

いだ展開を続け，n を越えない最大の分母を x の位数 r の候補とし，ステップ 6 に進む．本来の Shor のアルゴリズムはここで，r が位数となっているかをチェックし，位数でなければステップ 3 に戻る．r が位数となっていなくてもステップ 7 から位数を得られることがあり得るので今回の実装では先に進む方針をとった．これによって実際に成功確率があがる結果が見られた．

6. r が偶数でなければステップ 3 に戻る．

7. $\gcd(x^{r/2}-1, n)$ を Euclid の互除法により計算すると，高確率で n の因数 p が得られる．ここでも，$x^{r/2}$ は普通に計算するとオーバフローを起こす．しかし，$\gcd(x^{r/2}-1, n) = \gcd(x^{r/2} \pmod n - 1, n)$ であるので，この式の右辺のように計算すればよい．n の因数 p が得られたならば p を出力し，n を n/p に置き換えステップ 1 に戻る．そうでなければそのままステップ 3 に戻る．

5.2 振幅の変化の様子

5.1節ステップ 4 の量子計算による部分で振幅がどのように変化するかを，シミュレータで得たデータをもとにして視覚的に見ていく．実際の因数分解の例は基底状態が多くなりすぎてグラフに表すのは適当でないので，基底状態数 $q = 16$，周期 $r = 8$ となるデータをここでは用いる．このときの振幅の変化の例を図 6，7，8に表す．垂直軸が基底状態のインデックスである．水平面は複素平面になっていて，各基底状態の振幅の値を表している．実際は 16 個の点のプロットであるが，見やすいように直線でむすんである．図 6は初期状態から Walsh-Hadamard 変換を行い，等重の重ね合わせ状態になった状況である．図 7は第 2 レジスタの観測後である．等しい重みで観測したのでまったくランダムな値が選ばれ，周期の情報は得られない．ただし，値の同じ部分が周期の間隔に並んでいることが次に役立つ．図 8は離散フーリエ変換後である．これにより第 2 レジスタの値としてどれが観測されていようと，ほぼ同じような振幅の状況になるところが重要である．図 9は図 8の振幅の絶対値の 2 乗をとり観測確率を表したものである．このあと連分数近似により高確率で周期が得られる．

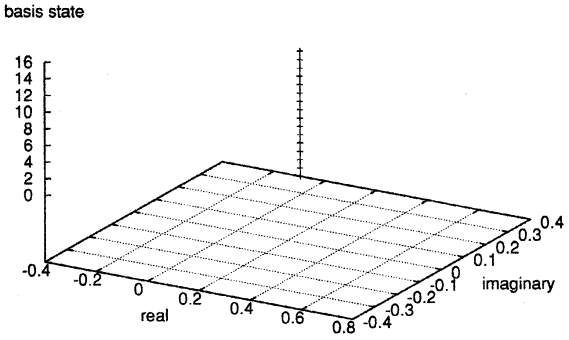

図 6: ステップ 4-(b) 等重の重ね合わせ状態

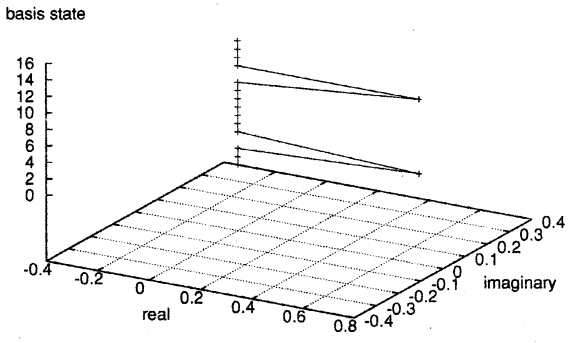

図 7: ステップ 4-(d) 第 2 レジスタ観測後

5.3 Shor のアルゴリズムの成功確率の解析

5.3.1 理論的評価

この節の評価は [7] によるものである．

1. 式 4において観測を行うと，干渉効果により，

$$-\frac{r}{2} \le rc \pmod q \le \frac{r}{2} \qquad (5)$$

となる c を高確率で観測し，このような c を観測する確率は $4/\pi^2 \simeq 0.405$ である．

2. 式 (5) を同値変形すると，

$$\exists\, d\ (0 \le d \le r-1) \quad s.t. \quad \left|\frac{c}{q} - \frac{d}{r}\right| \le \frac{1}{2q} \qquad (6)$$

となる．$q > r^2$ であるからこれは c/q を d/r で連分数近似してよい条件となっている [11]．ここで d と r が互いに素ならば r の値が得られる．d と r が互いに素となる確率は

$$\phi(r)/r > e^{-\gamma}/\log\log r \simeq 0.5615/\log\log r \qquad (7)$$

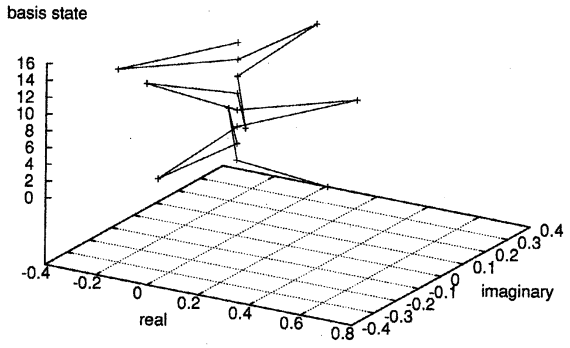

図 8: ステップ 4-(e) 離散フーリエ変換後

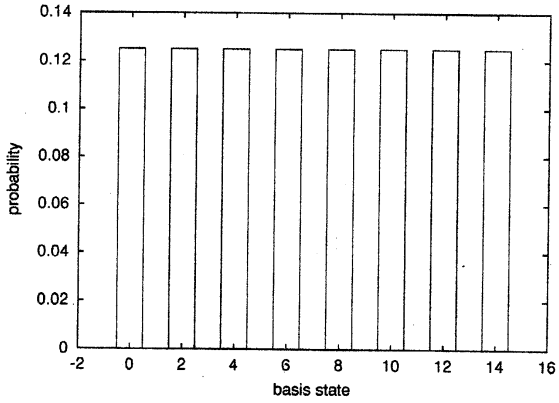

図 9: 観測確率

である．ここで $\phi(r)$ は Euler 関数である．これは図 10から n が 10000 以下ならば 0.25 以上であることは保証される．一般に，
$O(\log\log r)$ の繰り返しで十分高い確率で互いに素となる．

3. r が偶数のとき，$x^r \equiv 1 \pmod n$ は $(x^{\frac{r}{2}}-1)(x^{\frac{r}{2}}+1) \equiv 0 \pmod n$ と書き換えられる．ここで，$x^{\frac{r}{2}} \not\equiv \pm 1 \pmod n$ ならば $\gcd(x^{r/2}-1, n)$ は $1, n$ 以外の因数を与える．r が偶数，かつ，$x^{\frac{r}{2}} \not\equiv \pm 1 \pmod n$ となる確率は 1/2 以上である．

$r < n$ であるため，上記より以上のアルゴリズムは量子計算を用いると，$4/\pi^2 \times 0.5615/\log\log n \times 1/2$ より大きい確率で成功することは保証される．

例えば，$n = 221$ のとき成功確率は

$$4/\pi^2 \times 0.5615/\log\log 221 \times 1/2 \simeq 0.0675$$

より大きいことは保証される．しかし，これはかなり大まかな評価であり実際の成功確率はこれより高

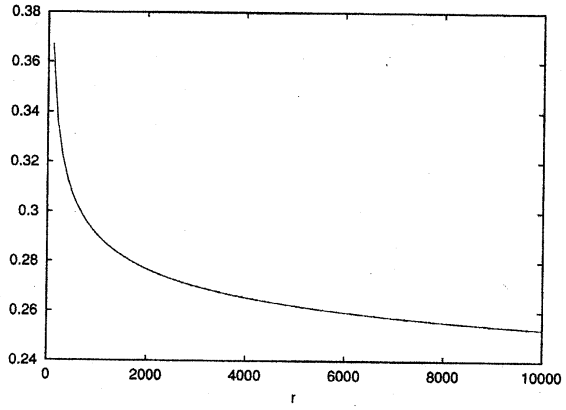

図 10: $0.5615/\log\log r$

いと考えられる．次節以降で実際的な場合について検討する．

5.3.2 実際的評価

表 2は $n = 221 = 13 \times 17$ のとき，$x = 2, \ldots, 220$ まで，それぞれ 10 回ずつ，計 2190 回シミュレートしたときの Shor のアルゴリズムが成功し因数を得た回数，Shor のアルゴリズムが失敗した回数，Euclid のアルゴリズムにより因数を得た回数である．

Shor 成功	812
Shor 失敗	1098
Euclid	280

表 2: Shor のアルゴリズムの成功回数 (1)

Shor のアルゴリズムの成功確率は，

$$\frac{\text{成功回数}}{\text{成功回数} + \text{失敗回数}} = \frac{812}{812+1098} = 0.42513$$

であった．理論的な保証確率に比べて，$n = 221$ のときは実際は約 7 倍ほど高い成功確率であった．これには以下のような要因が考えられる．

5.3.1節 3. の r が偶数，かつ，$x^{\frac{r}{2}} \not\equiv \pm 1 \pmod n$ をみたす r は，この実験において数をカウントしたところ 1476 個あった．よって，r が偶数，かつ，$x^{\frac{r}{2}} \not\equiv \pm 1 \pmod n$ となる確率は 1476 / (812 + 1098) = 0.772774 であり 1/2 より確かに大きかった．

また 5.1節，ステップ 5 において，本来の Shor のアルゴリズムのように r が位数でないときは失敗と

して実装したときは，表3のようになった．成功確率は $534/(534+1376) = 0.27958$ とより小さくなっており，ステップ5のようにする効果があったといえる．

Shor 成功	534
Shor 失敗	1376
Euclid	280

表 3: Shor のアルゴリズムの成功回数 (2)

5.3.1節 2. の $\phi(r)/r$ の部分でも実際上は大きい確率であると予測されるが，r によって変化が大きいので正確な評価は難しい．

また，256 以外においてもいくつかの値について実験を行い，実験成功確率と理論保証確率を比較した（表4）．成功確率が下がる傾向は実験においても見られた．また 437 など実験成功確率が理論保証確率の約 2.46 倍となっているときもあり，かなり良い理論保証になっているときもあることが分かった．

n	実験成功確率	理論保証確率	実験／理論
221=13 × 17	0.4251	0.0675	6.29
323=17 × 19	0.3564	0.0648	5.50
437=19 × 23	0.1556	0.0630	2.46
667=23 × 29	0.2013	0.0607	3.31
899=29 × 31	0.1862	0.0593	3.13
1147=31 × 37	0.2149	0.0582	3.69

表 4: 実験成功確率と理論保証確率の比較

5.3.3 x による分類

$n = 221 = 13 \times 17$ のとき，$x = 2,\ldots,220$ まで，10 回ずつシミュレーションした結果，x によって Shor のアルゴリズムの成功確率がはっきり異なる状況が見られた．その結果を以下に分類した．

Shor のアルゴリズムにより、高確率で (10 回中 8 回以上) 因数を得られた x は以下の 4 つに分類できる．

(1) 14, 27, 40, 53, 66, 79, 92, 105, 118, 144, 157, 183, 196, 209

(2) 18, 69, 86, 103, 131, 137, 154, 171, 188

(3) 12, 38, 116, 155, 181, 194, 207

(4) 73, 90, 99, 122, 138, 216

(1) は $x = 13k+1$ (k は自然数) の形をしており，$(13k+1)^j = 13l+1$ (j, l は自然数) である．よって r が偶数ならば，$\gcd(x^{r/2}-1, n)$ から因数 13 が得られるため，高確率で因数を得ていると考えられる．(2) は $x = 17k+1$ の形をしており，(1) と同様の理由で高確率で因数を得ていると考えられる．(3) は $x = 13k-1$ の形をしており，$(13k-1)^{2j} = 13l+1$ (j, l は自然数) である．よって r が 4 の倍数のとき，$\gcd(x^{r/2}-1, n)$ から因数 13 が得られるため，高確率で因数を得ていると考えられる．(4) はその他なんらかの要因によるものである．(1), (2), (3) は，さらに厳密な理論的限界を得るための情報となる可能性がある．

また，Shor のアルゴリズムにより，10 回とも失敗をして因数を得られなかった x は次のとおりである．これについては原因を調査中である．

21, 47, 72, 89, 98, 106, 115, 123, 132, 149, 150, 174, 179, 186, 200, 214, 215, 220

6 むすび

今回の汎用的シミュレータにより，一般的な量子アルゴリズムのシミュレートが可能となった．これを用いて，様々なアルゴリズムの振る舞いを知ることができ，新たな知見を得られるようになった．また計算量などの実際的な検証ができるようになったことの意義も大きい．

本シミュレータは必然的に入力に対して指数倍の記憶容量，計算時間を要する．これは一般的な問題に対応するためには避けられないと考えられる．よって，なるべく基本操作の計算量が小さくするよう考慮した．

本論文では述べなかったが，振幅の値が同じものが多いときなど特別な状況では BDD などを用いてデータ構造を工夫することにより記憶容量を多項式サイズにおさえることも可能であると考えられる．そうすれば多量の量子ビットのシミュレートが出来，大規模な実験が可能になる．ただし，その実装はかなり難しいことが予想される．また，今回の Shor のアルゴリズムに関しては離散フーリエ変換後の異なる振幅の個数が入力に対して指数個あるため，この手法は単純には用いえないと判断した．

今回の実験としては，Shor のアルゴリズムを実際にシミュレートして，アルゴリズムの振る舞い、性質について考察をした．本研究によって，量子

計算が実現すれば，Shorのアルゴリズムは実際上も，非常に有効であることがわかった．またアルゴリズムの性質を調べたところ，高い確率で因数を発見できる状況があることがわかったが，その状況は因数分解をする前には知ることができないためアルゴリズムの改良には用いえなかった．現状のコンピュータでShorのアルゴリズムを効率的に動作させることは，やはり難しいと考えられる．逆にいえば量子コンピュータの実現への期待が高まった．計算機科学の観点からは，様々な計算モデルを対象にして量子計算の能力を調べる必要がある．

今後のシミュレーション実験としては，Groverの量子探索アルゴリズム [9] を応用した，様々なアルゴリズム [3, 6, 10, 13] に対して検証を加えてみる予定である [17]．

また，プログラムの進行に誤差をいれてデコヒーレンスのシミュレーションをしてみるのも実際の量子コンピュータの作成の指針として大きく役に立つと考えられる．さらに，その誤り訂正のシミュレートも面白いであろう．

参考文献

[1] 渥美賢嗣，西野哲朗． 因数分解に対する量子アルゴリズムのシミュレーション．電子情報通信学会論文誌 A, Vol. J81-A, pp. 1670–1677, 1998.

[2] A. Barenco, C. Bennet, R. Cleve, D. Divincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev.*, Vol. A, No. 52, pp. 3457–3467, 1995.

[3] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Forschritte Der Physik*, Vol. 46, pp. 493–505, 1998.

[4] D. Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proc. Roy. Soc. London*, Vol. A, No. 400, pp. 97–117, 1985.

[5] D. Deutsch. Quantum computational networks. *Proc. Roy. Soc. London*, Vol. A, No. 425, pp. 73–90, 1989.

[6] C. Dürr and P. Høyer. A quantum algorithm for finding the minimum. Quantum Physics e-Print archive, http://xxx.lanl.gov/abs/quant-ph/960714, 1996.

[7] A. Ekert and R. Jozsa. Shor's quantum algorithm for factoring numbers. *Rev. Mod. Phys.*, Vol. 68, pp. 733–753, 1996.

[8] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, Vol. 21, pp. 467–488, 1982.

[9] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM Symposium on Theory of Computing*, pp. 212–219, 1996.

[10] L. K. Grover. A framework for fast quantum mechanical algorithms. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 53–62, 1998.

[11] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Oxford University Press, 5 edition, 1979.

[12] 細谷暁夫. 量子計算機, 1998. 講義ノート.

[13] A. Nayak and F. Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the 31th Annual ACM Symposium on Theory of Computing*, pp. 384–393, 1999.

[14] P. Shor. Algorithms for quantum computation: Discrete log and factoring. In *Proceedings of the 35th Annual IEEE Symposium on Foundation of Computer Science*, pp. 56–65, 1994.

[15] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, Vol. 26, pp. 1484–1509, 1997.

[16] 土村展之． JAVAによる量子探索アルゴリズムのデモ． http://www.kuamp.kyoto-u.ac.jp/labs/or/tokutei98/database/register/quantum1/index.html, 1999.

[17] 徳永裕己，小林弘忠，今井浩．Groverの量子探索アルゴリズムの応用．情報処理学会研究報告．アルゴリズム研究会, 1999年11月.