Alternative Models for Quantum Computation

by

Cedric Yen-Yu Lin

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

Signature redacted

Author ....
Department of Physics
May 22, 2015

Signature redacted

Certified by.......................
Edward H. Farhi
Professor of Physics; Director, Center for Theoretical Physics
Thesis Supervisor

Signature redacted

Accepted by ....
Nergis Mavalvala
Curtis and Kathleen Marble Professor of Astrophysics
Associate Head for Education, Physics

Alternative Models for Quantum Computation

by

Cedric Yen-Yu Lin

Submitted to the Department of Physics
on May 22, 2015, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Physics

Abstract

We propose and study two new computational models for quantum computation, and infer new insights about the circumstances that give quantum computers an advantage over classical ones.

The bomb query complexity model is a variation on the query complexity model, inspired by the Elitzur-Vaidman bomb tester. In this model after each query to the black box the result is measured, and the algorithm fails if the measurement gives a 1. We show that the bomb query complexity is asymptotically the square of the usual quantum query complexity. We then show a general method of converting certain classical algorithms to bomb query algorithms, which then give improved quantum algorithms. We apply this general method to graph problems, giving improved quantum query algorithms for single-source shortest paths and maximum bipartite matching.

Normalizer circuits are a class of restricted quantum circuits defined on Hilbert spaces associated with Abelian groups. These circuits generalize the Clifford group, and are composed of gates implementing quantum Fourier transforms, automorphisms, and quadratic phases. We show that these circuits can be simulated efficiently on a classical computer even on infinite Abelian groups (the finite case is known [1, 2]), as long as the group is decomposed into primitve subgroups. This result gives a generalization of the Gottesman-Knill theorem to infinite groups. However, if the underlying group is not decomposed (the group is a black box group) then normalizer circuits include many well known quantum algorithms, including Shor's factoring algorithm. There is therefore a large difference in computational power between normalizer circuits over explicitly decomposed versus black box groups. In fact, we show that a version of the problem of decomposing Abelian groups is complete for the complexity class associated with normalizer circuits over black box groups: any such normalizer circuit can be simulated classically given the ability to decompose Abelian groups.

Thesis Supervisor: Edward H. Farhi
Title: Professor of Physics; Director, Center for Theoretical Physics

Acknowledgments

I have many people to thank, without whom completing this thesis would be impossible.

First and foremost, I thank Eddie Farhi for being a wonderful advisor and collaborator; for giving me guidance on research yet allowing me plenty of freedom; and for teaching me how to be inquisitive, how to think about quantum computation, and how to do science.

I thank Peter Shor for being like a second advisor to me, for his frequent help and guidance in my research and being a frequent collaborator. I thank Aram Harrow and Scott Aaroson for frequent discussions and answering many questions, and also for inspiring me how to think about quantum (and classical) computation.

I thank Eddie Farhi, Aram Harrow, and David Kaiser for the time and effort they have spent on reviewing my thesis and thesis defense. I also thank Aram, Peter, and Ike Chuang for serving on my general exam committee.

I thank Robert Raußendorf and Ian Affleck for their support and collaboration at the beginning of my research career as an undergraduate, and also for introducing me to the world of quantum computation.

I thank Han-Hsuan Lin, Juan Bermejo-Vega, Maarten Van den Nest, Elizabeth Crosson, and Shelby Kimmel for being great collaborators and friends. I especially thank Han-Hsuan and Juan for years of working together, and Shebly for her invaluable help when I was applying for a postdoctoral position.

I thank Ike Chuang, Scott Aaronson, and Jon Kelner for their phenomenal classes on quantum and classical computation. In particular I thank Ike for the term project in his class, which eventually lead to Chapter 5 of this thesis.

I thank David Gosset and Andy Lutomirski for helping me get used to graduate life at MIT.

I thank Alex Arkhipov, Mohammad Bavarian, Adam Bookatz, Adam Bouland, Andrew Childs, Richard Cleve, Matt Coudron, Lior Eldar, David Gosset, Daniel Gottesman, Jeongwan Haah, Stephen Jordan, Robin Kothari, Chris Laumann, Andy Lutomirski, Ashley Montanaro, Daniel Nagaj, Anand Natarajan, Cyril Stark, Pawel Wocjan, and Henry Yuen for many useful discussions.

I thank the staff of the CTP, LNS, and the Department of Physics for their help over the years, and for making our lives so much easier. Special thanks to Scott Morley, Joyce Berggren, and Charles Suggs for keeping the CTP as nice as it is.

I thank NSERC and the ARO for supporting me financially so that I could concentrate on research.

I thank my friends for their support and friendship, and for all the good times we've had together.

Lastly, I thank my family for always being there when I need them. I thank my parents Carol and Wen-Chien, my brother Terence, and my grandparents for their unwavering support and encouragement throughout the years. I am lucky to have them in my life.

Contents

Chapter 1

Introduction

This thesis concerns the fields of quantum algorithms and quantum computational complexity. An *algorithm*, roughly speaking, is a set of instructions to be performed to solve some computational problem. *Computational complexity theory* studies the time, memory, and other resource requirements needed to solve some computational problem. By *quantum* we mean that the computers we use are quantum computers, i.e. these computers are allowed to use quantum effects, such as interference and entanglement.

We will be studying two alternative computational models in the quantum setting. In Chapter 2 we will be studying the *bomb query complexity model*, a generalization of the Elitzur-Vaidman bomb tester (see Section 2.2) to the study of complexity. This model will yield a method for converting certain classical algorithms to good quantum algorithms. In Chapters 3, 4, and 5 we will study the model of *normalizer circuits*; this restricted model of quantum circuits will allow us to better understand the quantum speedup in Shor's factoring algorithm, and perhaps give guidance on the design of future algorithms.

In this chapter we will introduce the concepts of quantum computing that we will need in this thesis; a knowledge of quantum mechanics is assumed. We will start by introducing classical computation in Section 1.1, proceed to quantum computation in Section 1.2, and describe the two most important known quantum algorithms: Shor's algorithm for factoring in Section 1.3, and Grover's search algorithm in Section 1.4. We end this chapter by outlining the rest of the thesis in Section 1.5.

Most of the material of this chapter is covered in more detail in [3, 4].

1.1 Classical Computation

The classical theory of computation is interested, broadly, in the following type of question: given a set amount of resources, what computational problems can we solve? At first sight it is not even clear that these types of questions are meaningful at all, since presumably the answer would depend on the specific machine we are using. Nevertheless we will see that models of computation can be introduced that capture the essence of all computers we know.

1.1.1 Turing and the theory of computability

In 1937, Alan Turing introduced the model of *Turing machines* [5]. In summary, a Turing machine consists of an semi-infinite tape along with a finite-sized program and memory used to manipulate the contents of the tape. The input to the Turing machine is the original contents of the tape, and the output is the contents of the tape after the Turing machine has finished executing. We will not go into very many details, but Turing made three very important observations:

1. There exists a *universal* Turing machine that is capable of simulating an arbitrary Turing machine on arbitrary input. The universal machine reads the description of a specific Turing machine $\mathcal{T}$ and input $\mathcal{I}$ to be simulated from its own input (tape), and gives the same output that one would have gotten if $\mathcal{T}$ were executed with input $\mathcal{I}$. This property of *universality* was radical at the time: given a computational task one wanted to perform, one would have used a machine specifically built to perform that task. Turing's work showed that instead of building a computer for every single task, one could instead build a general-purpose computer and change the *software* on it to perform any computation.

2. Given enough time, Turing machines appear to be able to calculate any function computable by a real-world process (e.g. done by pen-and-paper, employing a machine, measuring an observable of a physical system, etc.). This observation was made independently by Turing and his Ph. D. advisor, Alonzo Church[1]:

   > **Church-Turing thesis**: Any real-world computational process can be simulated with a Turing machine.

   The Church-Turing thesis is, in the opinion of the author, somewhat like a natural law: it cannot be proven mathematically, but can be disproved by a counterexample. No such counterexamples are known; the computers that we use nowadays can be simulated by Turing machines. Turing machines can therefore be viewed as a rigorous mathematical formalization of *algorithms.*

3. There are functions that cannot be computed by Turing machines. An example is the *Halting Problem*: given a description of a Turing machine and input, decide whether the Turing machine halts or runs forever. Turing showed that there is no algorithm (Turing machine) that solves the halting problem. This was a profound statement that came as a surprise to many, especially to David Hilbert. Hilbert had posed the challenge of finding an algorithm that could, in principle (and enough time), solve all the problems of mathematics [7]; Turing's work showed that this was impossible.

   We now know many other such problems that are unsolvable even in principle; two recent examples in physics are deciding whether a quantum many-body Hamiltonian is gapped or gapless [8, 9], and deciding whether a sequence of Stern-Gerlach-type measurements have outcomes that never occur [10].

We stress that while it is believed that Turing machines can simulate any natural computational process, we have not said anything about the resources required for such a simulation.

Finally, it will be convenient to consider a variant of the Turing machine where *randomness* is allowed. Such a *probabilistic Turing machine* would be able to flip a coin during the execution of the machine, and execute different instructions depending on the result of the coin flip. A deterministic Turing machine would still be able to simulate a probabilistic one,

[1]Church had earlier propsed the equivalent, but much less intuitive, computational model of λ-calculus [6].

by simply checking the results of the computation on all possible coin tosses, in accordance with the Church-Turing thesis. However such a simulation would of course take far more steps to execute.

1.1.2 Complexity theory

The discussion in the preceding section on Turing machines centered mainly on computability theory, i.e. studying which functions were in principle computable by an algorithm, and which functions were not. In practice, however, this is of course not all we care about, since even if a function is possible to compute in principle, in practice there may be no algorithm that does so within a reasonable time and memory. *Computational complexity theory* studies the resource requirements of computational tasks.

In computational complexity we are often interested in the *asymptotic scaling* of the time and space requirements of an algorithm as a function of the problem size or input size, and not in the precise values of the requirements itself. We will often use the following 'big-O' notation:

Definition 1. Let f and g be two nonnegative-valued functions defined on the positive integers. Then:

- Suppose there is an integer N and a constant C such that $f(n) \leq Cg(n)$ for all $n \geq N$. Then we say that $f(n) = O(g(n))$.
- Suppose there is an integer N and a constant C such that $f(n) \geq Cg(n)$ for all $n \geq N$. Then we say that $f(n) = \Omega(g(n))$.
- If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, then we say that $f(n) = \Theta(g(n))$.

When we analyze the amount of resources needed for an algorithm to run, we will usually only give a big-O upper bound for it. The actual runtime of an algorithm may very well depend on the power of the computer it is run on, which we cannot directly analyze; what we can look at is how well the algorithm scales (e.g. whether the runtime is a polynomial or exponential in the size of the problem). For example, for a $\Theta(2^n)$-time algorithm, adding a single bit to the problem size doubles the runtime, and the algorithm is therefore unlikely to be useable beyond say $n = 40$. On the other hand, algorithms with polynomial time complexity are widely used in practice.

A general rule of thumb used by computer scientists is that an algorithm runs efficiently in practice if its resource requirements (usually time) scale only as a polynomial of the size of the input. This is of course not a hard and fast rule, as an algorithm with runtime $O(n^{100})$ can hardly be called efficient while one with runtime $O(2^{0.00001n})$ will likely work well in practice. Nevertheless, distinguishing between polynomial and superpolynomial times does a sufficiently good job of distinguishing between fast and slow algorithms, and is mathematically convenient. We thus make the following definition:

Definition 2. An algorithm is *efficient* if and only if its resource requirements (usually the number of operations used) scale as a polynomial of the length of the input.

For the rest of this thesis, this is the only meaning of the word 'efficient' that we will use.

Of course, the complexity of a problem depends on the given computational model. However, in the 1960s and 1970s (at the beginning of the study of complexity theory) researchers observed and proposed the following:

> **Extended Church-Turing thesis**: Any real-world computational process can be simulated, with at most a polynomial increase in the number of steps required, by a probabilistic Turing machine.

For decades it was thought that the extended Church-Turing thesis was true. This would make probabilistic Turing machines an ideal computational model to study, since any efficient real-world algorithm would be simulable efficiently by a probabilistic Turing machine. We shall see however, in Section 1.2 (as well as Shor's algorithm in Section 1.3), that we now have good reason to think that the extended Church-Turing thesis is false, because of the classical difficulty of simulating quantum mechanics. Nevertheless probabilistic and deterministic Turing machines are worthy objects of study, since the class of problems efficiently solvable are quite robust with respect to the computational model chosen: for example, our current computers can still be modeled with probabilistic or deterministic Turing machines.

We will introduce the following *complexity classes*, classes of computational problems related by some common resource requirement. In the following, a decision problem is a yes-or-no question: given an input, decide whether or not it satisfies some property. If it does, we refer to the input string as a YES-instance; otherwise it is a NO-instance.

Definition 3. **P** is the complexity class of decision problems that can be solved with polynomial time on a deterministic Turing machine, i.e. can be solved efficiently with a classical deterministic algorithm.

Definition 4. **BPP**[2] is the complexity class of decision problems that can be solved with polynomial time with a classical *randomized, bounded error*, algorithm. To be more precise, a decision problem $\mathcal{L}$ is in **BPP** if there is a randomized algorithm $\mathcal{A}$ that always outputs 0 or 1, and always runs in polynomial time, such that the following holds:

- If a string s is a YES-instance of $\mathcal{L}$, then $\mathcal{A}$ given input s outputs 1 with probability at least $2/3$.
- If a string s is a NO-instance of $\mathcal{L}$, then $\mathcal{A}$ given input s outputs 0 with probability at least $2/3$.

Note that the time complexity are worst-case bounds; the algorithms referred to above must run in polynomial time for all inputs. In addition, the precise constants $2/3$ used in the definition of **BPP** do not matter, as long as it is strictly greater than $1/2$; this is because running a **BPP** algorithm multiple times and outputting the majority answer can amplify the success probability to any constant strictly less than 1.

P contains many natural problems, including checking whether all numbers in a list are positive, checking whether two points are connected in a graph, determining whether a system of linear equations has a solution, and so on. A natural problem that appeared to be in **BPP** but was not known to be in **P** is the problem of determining whether an integer was prime; that problem was however recently determined to be in **P** [11]. Many computer scientists now believe the conjecture that in fact **BPP** = **P**, and classical randomized algorithms are just as powerful as deterministic ones. In some sense, the complexity classes **P** and **BPP** capture the set of problems that are easy to compute on a classical computer.

Of course, not all problems are easy to compute. A simple example, that we will come back to multiple times in this thesis, is that of factoring a number. We can cast it as a

[2]**BPP** stands for bounded-error probabilistic polynomial time.

decision problem as follows: given a positive integer n and a number $\ell < n$, does n has a nontrivial factor no greater than ℓ? This problem does not seem easy to solve; the brute force method of trying all integers less than n is not efficient, since it takes time polynomial in n, which is exponential in the size of the input, the lengths of the descriptions of n and ℓ, which is $O(\log n)$. In fact, there is no known efficient classical algorithm for the factoring problem.

However, the problem of factoring has the property that if n does have a factor $k \leq \ell$, then it is easy to verify this fact as long as the *witness* k is given: we simply check that k divides n. As such, the factoring problem seems difficult to solve, but is easy to verify once a witness is given. This motivates the definition of the following complexity class:

Definition 5. **NP**[3] is the complexity class of decision problems whose YES-instances can be checked efficiently with a classical deterministic algorithm, given access to a witness string. More formally, a decision problem $\mathcal{L}$ is in **NP** if there is a classical deterministic algorithm $\mathcal{A}$ that always outputs 0 or 1, and always runs in polynomial time, such that following holds:

- If a string s is a YES-instance of $\mathcal{L}$, then there is some witness string w such that $\mathcal{A}$, given input s and w, outputs 1.
- If a string s is a NO-instance of $\mathcal{L}$, then for all witness string candidates w, $\mathcal{A}$ given inputs s and w outputs 0.

In English, this says that if s is a YES-instance, then there is a short proof w of this fact that can be efficiently verified. Conversely, if s is a NO-instance, then no purported proof w can possibly work. The factoring problem thus belongs to **NP**.

It is clear that **P** is a subset of **NP**. Most computer scientists believe that $\mathbf{P} \neq \mathbf{NP}$, i.e. that there are problems that are hard to solve but are easy to verify, given the solution. There is no proof of this, however, and proving (or disproving) that $\mathbf{P} \neq \mathbf{NP}$ is one of the seven Millenium Prize Problems selected by the Clay Institute, which carries a million-dollar prize.

1.2 Quantum Computation

Earlier in 1982, Richard Feynman had noticed the computational difficulties of simulating quantum mechanical systems on classical computers: given a system of just 30 qubits (two-dimensional quantum systems), the size of the Hilbert space is already $2^{30} \approx 10^9$, and general Hamiltonians acting on such a small system are already difficult to simulate because of the exponential dimension of the Hilbert space. To Feynman, it made sense to instead use a *quantum* computer, a computer built out of quantum mechanical components and interactions, to simulate such systems, and Feynman outlined a rudimentary design of such a computer [12]. In 1985, David Deutsch formalized the concept of a quantum computing with his introduction of *quantum Turing machines* [13], a generalization of the Turing machine that incorporates quantum mechanical interactions.[4]

These insights, that a quantum computer could possibly perform some computational tasks exponentially faster than a classical one, brought the extended Church-Turing thesis into question. This was made even more dramatic by Peter Shor's 1994 discovery [14] of

[3]Short for *nondeterministic polynomial time.*

[4]We will not pay much attention on quantum Turing machines, because quantum circuits are more common and easier to deal with, but represent an equivalent computational model.

an efficient quantum algorithm for factoring an integer, perhaps a more intuitive problem that was also difficult to solve on a classical computer. Through factoring an integer, a quantum computer could also break the widely-used RSA cryptosystem [15]; this observation generated much interest in the field of quantum computing. Shor's algorithm remains a basis for many important algorithms and results in quantum computing, including Chapter 5 of this thesis; we will look at a description of this algorithm in Section 1.3.

In this section, we will concentrate instead on giving an introduction to the basic concepts of quantum computation. We will not pay any attention to quantum Turing machines; instead we look at the equivalent, but more common and easier to deal with, quantum circuit model. For a more detailed introduction to quantum circuits, see [3, Chapter 4] or [4, Chapter 4]

A quantum circuit can be summarized as follows:

1. A quantum circuit has a quantum state that usually (but not always[5]) consists of a number of qubits, or spin-1/2 systems. The state of a single qubit can be written in the form $\alpha|0\rangle + \beta|1\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$; for a system of n qubits the general quantum state is
$$\sum_{s\in\{0,1\}^n} c_s|s\rangle, \quad \sum_s |c_s|^2 = 1 \tag{1.2.1}$$
where $\{0,1\}^n$ denotes the set of all length-n boolean (0 or 1) strings.

   Initially all qubits are set to 0, except for possibly a portion of the qubits serving as an input to the circuit.

2. The circuit will apply a sequence of *quantum gates* to the state, each gate acting only on one or two qubits.[6] A quantum gate U is a unitary operation on the Hilbert space: it satisfies $U^\dagger U = UU^\dagger = I$, and so preserves the norm of the quantum state.

3. At the end of the circuit some or all of the qubits are measured in the computational basis (the Pauli-Z basis). The classical results of the measurement serve as the output of the circuit. It will sometimes be convenient (but never necessary) to allow measurements being made in the middle of a quantum circuit as well.

Some examples of quantum gates are the Pauli gates:
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.2.2}$$

Other important gates are the Hadamard gate H, the phase gate $S = \sqrt{Z}$, and the $\pi/8$ gate T:
$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = \begin{bmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{bmatrix}, \tag{1.2.3}$$

[5]We will also use d-dimensional quantum systems, or *qudits*, in our circuits.

[6]This sequence of gates needs to be generated by a Turing machine, to prevent the sequence of gates encoding some undecidable problem.

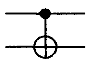
An example of a two-qubit gate is the CNOT gate:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.2.4}$$

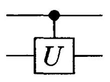
CNOT stands for *Controlled-NOT*: it is a two qubit gate that applies the NOT (or Pauli-X) operation to the second register, *controlled* on whether the first register is in the $|1\rangle$ state:

$$CNOT|0\rangle|0\rangle = |0\rangle|0\rangle, \quad CNOT|0\rangle|1\rangle = |0\rangle|1\rangle \tag{1.2.5}$$
$$CNOT|1\rangle|0\rangle = |1\rangle|1\rangle, \quad CNOT|1\rangle|1\rangle = |1\rangle|0\rangle \tag{1.2.6}$$

We refer to the first register as the control register, and the second as the target register. In the depicition of circuits, the CNOT is denoted with the following symbol:


(1.2.7)

The black dot refers to the control register, while the $\oplus$ symbol refers to the target. More generally, if U is a quantum gate then


(1.2.8)

is the controlled-U gate, where U is applied on the second register controlled on whether the first register is 1.

It turns out that the set of all one- and two-qubit gates are *universal*: they can implement any general n-qubit gate, for arbitrary n. There are also finite sets of quantum gates that are *approximately universal*, i.e. it is possible to implement arbitrary gates from them. An example set of approximately universal gates are $\{H, S, T, CNOT\}$. However, if we omit the T gate, then any quantum circuits composed of only H, S, and $CNOT$ gates can be simulated efficiently on a classical computer; this is known as the *Gottesman-Knill theorem* [16, 17], and is important in the theory of quantum error correction. We will look at a generalization of this theorem in Chapter 4.

Finally, we define the following complexity class:

Definition 6. BQP[7] is the complexity class of decision problems that can be solved with polynomial time with a *quantum, bounded error*, algorithm. To be more precise, a decision problem $\mathcal{L}$ is in **BQP** if there is an quantum algorithm $\mathcal{A}$ that always outputs 0 or 1, and always runs in polynomial time, such that the following holds:

- If a string s is a YES-instance of $\mathcal{L}$, then $\mathcal{A}$ given input s outputs 1 with probability at least 2/3.
- If a string s is a NO-instance of $\mathcal{L}$, then $\mathcal{A}$ given input s outputs 0 with probability at least 2/3.

[7]**BQP** stands for bounded-error quantum polynomial time.

Since quantum algorithms are inherently probabilistic (the output is generated by a measurement), we are mainly interested in bounded-error quantum algorithms, i.e. the algorithms are allowed to have some small chance of failure. Once again the constant 2/3 does not matter, as long as it is greater than 1/2; this probability can be boosted to any value strictly less than 1 by running multiple rounds of the algorithm and taking the majority answer.

For a list of current quantum algorithms, see the online *Quantum Algorithm Zoo* [18]. We will cover in the following sections Shor's factoring algorithm [14] and Grover's search algorithm [19], probably the two most important quantum algorithms to date.

1.3 Shor's Algorithm for Factoring

In this section we will describe Shor's algorithm for factoring [14]. The main building blocks for this algorithm – the quantum Fourier transform and phase estimation – find numerous applications in quantum algorithms. We will use quantum Fourier transforms extensively in our discussion of normalizer circuits in Chapters 3-5; moreover, we will study Shor's algorithm itself in a different light in Chapter 5.

Our description of Shor's algorithm will not follow the more "modern" approach using phase estimation as given by Kitaev [20], and will instead stay closer to Shor's original approach. The purpose of this is to make it fit more manifestly into the normalizer framework in Chapter 5.

1.3.1 The Quantum Fourier Transform

We will first describe the quantum Fourier transform (QFT). On an m-dimensional Hilbert space with basis states $|0\rangle, |1\rangle, \cdots, |m-1\rangle$, the quantum Fourier transform is the following unitary transformation:

$$|j\rangle \to \frac{1}{\sqrt{m}} \sum_{k=0}^{N-1} e^{2\pi i jk/m} |k\rangle. \tag{1.3.1}$$

This unitary gates is named after its strong resemblance to the discrete Fourier transform. Indeed, the QFT performs a Fourier transform of the coefficients of the quantum state in the standard basis: it performs the map

$$\sum_{j=0}^{m-1} x_j |j\rangle \to \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} y_k |k\rangle, \tag{1.3.2}$$

where

$$y_k = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} x_j e^{2\pi i jk/m} \tag{1.3.3}$$

is the discrete Fourier transform of $\{x_j\}$.

The quantum Fourier transform can be approximately implemented very efficiently, for any m:

Theorem 7. *For any m, the quantum Fourier transform can be implemented with error ϵ with gate complexity*

$$O\left(\log \frac{m}{\epsilon} \left(\log \log \frac{m}{\epsilon} + \log \frac{1}{\epsilon}\right)\right). \tag{1.3.4}$$

We will not prove this theorem; see [21] for a review (and references). We will instead show a weaker construction that implements the QFT *exactly* for $m = 2^n$, using $O(n^2)$ gates.

Theorem 8. *For $m = 2^n$ with integer n, the quantum Fourier transform can be implemented exactly with gate complexity $O(n^2)$.*

Proof. This proof follows [3, Chapter 5]. For shorthand, we will express numbers in binary floating point format, i.e. if $j = j_1 2^{\ell-1} + j_2 2^{\ell-2} + \cdots + j_\ell 2^0 + j_{\ell+1} 2^{-1} + \cdots + j_n 2^{\ell-n}$ then we will write $j = j_1 j_2 \cdots j_\ell . j_{\ell+1} \cdots j_n$. We proceed to analyze Eq. 1.3.1 in this notation:

$$|j\rangle \to \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i jk/2^n} |k\rangle \tag{1.3.5}$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \sum_{\ell=1}^{n} k_\ell 2^{-\ell}} |k_1 k_2 \cdots k_n\rangle \tag{1.3.6}$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{\ell=1}^{n} e^{2\pi i j k_\ell 2^{-\ell}} |k_\ell\rangle \tag{1.3.7}$$

$$= \frac{1}{2^{n/2}} \bigotimes_{\ell=1}^{n} \left[\sum_{k_\ell=0}^{1} e^{2\pi i j k_\ell 2^{-\ell}} |k_\ell\rangle \right] \tag{1.3.8}$$

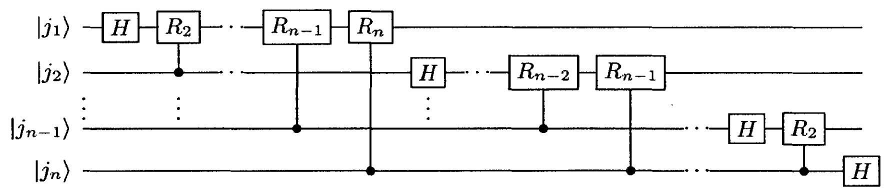
$$= \frac{1}{2^{n/2}} \bigotimes_{\ell=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-\ell}} |1\rangle \right] \tag{1.3.9}$$

$$= \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_{n-1} j_n}|1\rangle\right). \tag{1.3.10}$$

Thus the QFT maps $|j\rangle$ to a product state on n qubits. Defining

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad R_\ell = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/\ell} \end{bmatrix} \tag{1.3.11}$$

the following circuit implements the QFT:



(1.3.12)

The outputs are (ignoring normalization) $|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_{n-1} j_n}|1\rangle$ on the topmost qubit, $|0\rangle + e^{2\pi i 0.j_2 \cdots j_{n-1} j_n}|1\rangle$ on the second to top, $|0\rangle + e^{2\pi i 0.j_{n-1} j_n}|1\rangle$ on the second to bottom, and $|0\rangle + e^{2\pi i 0.j_n}|1\rangle$ on the bottomost qubit. Thus the order of the qubits need to be reversed to complete the QFT.

□

1.3.2 Shor's Algorithm for Order Finding and Factoring

Let us consider the following problem:

Problem 9 (Order Finding). Given coprime integers $a, N > 1$, $\gcd(a, N) = 1$, find the least positive integer r such that $a^r \equiv 1 \mod N$. We call this r the *order* of a modulo N.

It is not difficult to see that such an r must exist: considering the sequence $a^1, a^2, \cdots$ modulo N, eventually it must repeat itself: say $a^i \equiv a^j \mod N$, $i < j$, and this gives $a^{j-i} \equiv 1 \mod N$. What we are interested here is the smallest such integer r.

It turns out that the problem of factoring integers can be reduced to the problem of order finding. We will sketch here Shor's algorithm for order finding; this algorithm can therefore be used (with classical pre- and postprocessing) to factor integers.

In this algorithm we will need the modulo exponentiation operation: given numbers a, b, and N, it is possible to compute $a^b \mod N$ using $O(\log b)$ multiplications. To see this, note that we can compute the numbers $a, a^2, a^4, \cdots, a^{2^{\lfloor \log b \rfloor}} \mod N$ using $O(\log b)$ multiplications (each item in the list is the square of the previous one); a^b is then just the product of some subset of these numbers. We can therefore implement the controlled modulo exponentiation gate:

$$|b\rangle|c\rangle \to |b\rangle\Big|ca^b \pmod N\Big\rangle. \tag{1.3.13}$$

We now present Shor's algorithm. We will not give a complete proof of the correctness of the algorithm, and only give a sketch.

Algorithm 10 (Shor's algorithm for order finding [14]).

We will use two registers for this algorithm. The first register is L-dimensional for some $L > N^2$, while the second register is N-dimensional. We will omit all normalization of quantum states.

1. **Initialization:** Start in the state $|0\rangle|1\rangle$.
2. Apply the QFT to the first register, obtaining the equal superposition: $\sum_{j=0}^{L-1} |j\rangle|1\rangle$.
3. Apply the controlled modulo exponentiation gate to obtain $\sum_{j=0}^{L-1} |j\rangle|a^j \mod N\rangle$.
4. Measure the second register[8]. The measurement result is a value $a^b \mod N$ almost uniformly chosen from $b = 0, \cdots, r-1$. The first register becomes the state
$$\sum_{k=0}^{k_b} |kr + b\rangle \tag{1.3.14}$$
where k_b is the largest integer such that $k_b r + b < L$. The coefficients of this state are almost periodic, but not quite because of the wraparound (unless r divides L).
5. Apply the QFT to this register. If r divides L we would obtain (recall the discrete Fourier transform of periodic functions; see e.g. [22])
$$\sum_{j=0}^{r-1} e^{2\pi i j b/r}\Big|\frac{j}{r}L\Big\rangle. \tag{1.3.15}$$

[8]This step is not necessary, but is kept to simplify the analysis.

6. Measure the register to obtain j/r, for j nearly uniformly sampled from $j = 0, \cdots, r-1$. (Even if r does not divide L we would still obtain an approximation of j/r, from which we can extract the exact fraction j/r using the continued fractions algorithm; see e.g. [23, Chapter X].)

After running this algorithm for multiple times we obtain multiple samples of fractions j/r, from which we can obtain r with high probability (by taking the least common multiples of the denominators of the reduced fractions). This algorithm is efficient, in the sense that the number of operations required is $\text{poly}(\log N)$.

Finally, we sketch how we could factor a composite integer N given an algorithm for order finding. Assume N is odd; if not, 2 is a factor of N. Pick a uniformly random integer a from the set $\{1, \cdots, N-1\}$. We can find the greatest common divisor of N and a using Euclid's algorithm; if it is greater than 1, we have found a nontrivial factor of N. Otherwise a and N are coprime, and we can use the order finding algorithm to find the minimum $r > 1$ such that $a^r \equiv 1 \mod N$. If r is even, $a^r - 1 \equiv (a^{r/2} - 1)(a^{r/2} + 1) \mod N$. It can be shown that r is even and $a^{r/2} \equiv \pm 1 \mod N$ with probability at least 1/2, and so in this case $\gcd(a^{r/2} - 1, N)$ is a nontrivial factor of N that can be computed with Euclid's algorithm. Repeating this procedure multiple times allows for an arbitrarily high probability of success.

1.3.3 Hidden Subgroup Problem

Shor's algorithm solves an instance of the *hidden subgroup problem*, generally phrased as follows:

Problem 11 (Hidden Subgroup Problem (HSP)). Let G be a finitely generated group and X a finite set. $f : G \to X$ is a function that is constant on cosets of some unknown subgroup H of G, and distinct on different cosets: $f(g_1) = f(g_2)$ if and only if $g_1H = g_2H$. Assume that we have access to a black box implementing the unitary operation $O_f : |g\rangle|x\rangle \to |g\rangle|x \oplus f(g)\rangle$. Find a generating set of H.

Here we assume that any algorithm for the HSP has access to the black box O_f, but otherwise it uses no information about f. We will have more to say about black boxes in Section 1.4.

As an example, in the case of order finding we have $G = \mathbb{Z}$, $X = \{0, \cdots, N-1\} \mod N$, and $H = \{0, \pm r, \pm 2r, \cdots\}$. The function f is implemented as $f(x) = a^x \mod N$, which is constant on cosets of H and distinct on different cosets.

There is an efficient quantum algorithm for the HSP for Abelian G (apparently the algorithm is folklore), but for non-Abelian G an efficient algorithm exists only for special cases. See [21, 24] for reviews. Many, if not most, quantum algorithms that achieve superpolynomial speedup over classical algorithms are for the HSP or related variants.

We will sketch the algorithm for the HSP when G is finite Abelian. Note that this does not include the case of order finding, where $G = \mathbb{Z}$, but the similarities should be manifest.

Algorithm 12 (Algorithm for the Abelian HSP). We will use two registers for this algorithm. The first register encodes elements of G, and the second encodes elements of X. We will also need to use the *quantum Fourier transform* over G, defined analogously using the discrete Fourier transform over G; see Section 3.6 for the precise definition. We will omit all normalization of quantum states.

1. **Initialization:** Start in the state $|0\rangle|0\rangle$.

2. Create a superposition over all elements of G: $\sum_{g \in G} |g\rangle|0\rangle$.
3. Apply the black box O_f to obtain $\sum_{g \in G} |g\rangle|f(g)\rangle$.
4. Measure the second register, obtaining the value $f(g)$ for some value of g and leaving the first register in the state
$$\sum_{h \in H} |gh\rangle. \tag{1.3.16}$$
5. Apply the QFT over G to this register, to obtain the state
$$\sum_{h' \in H^{\perp}} \xi_g(h')|h'\rangle \tag{1.3.17}$$
where $H^{\perp}$ is the orthogonal subgroup of H, and ξ_g is the character corresponding to g.
6. Measure the register to a sample from the orthogonal subgroup $H^{\perp}$.

Repeating this procedure $O(\log|H|)$ times yields enough samples of $H^{\perp}$ to determine H, with high probability. Classical processing of these samples gives a generating set of H.

We will study this algorithm again in Chapter 5.

1.4 Query Complexity and Grover's Algorithm

We will introduce Grover's search algorithm [19] in this section; its key concept, amplitude amplification, forms the core of many quantum algorithms. Grover's algorithm essentially allows us to find a marked item in an unordered quantum database of N items, using only $O(\sqrt{N})$ operations. Unlike Shor's algorithm, the speedup here is only quadratic and not exponential; this is typical of most quantum algorithms for search problems.

We will also take the opportunity to introduce the query complexity model, for Grover's algorithm is most naturally presented in this model. We will make use of this model extensively in Chapter 2.

1.4.1 Black Boxes and Grover's Problem

Grover's algorithm is usually considered in the following setup. Assume that $N = 2^n$, and suppose that we have a hidden boolean function $x : \{0, \cdots, N-1\} \to \{0, 1\}$, accessible only through a black box O_x in the following manner:

$$O_x|i\rangle|r\rangle = |i\rangle|r \oplus x(i)\rangle \tag{1.4.1}$$

where i has length n (so the register containing x has dimension $N = 2^n$), r has length 1, and $\oplus$ indicates addition modulo 2. This is the most straightforward way of implementing $x(i)$ as a quantum gate, since any quantum gate is unitary and hence reversible. By saying that O_x is a black box we mean that there is no way of looking at the "inner wirings" of O_x and determining its properties that way; the only way of learning information about O_x is to apply it in a circuit.

In the set of numbers $S = \{0, \cdots, N-1\}$, we will say an element i is either marked or unmarked: $x(i) = 1$ if i is marked, and $x(i) = 0$ otherwise. We only have access to the function f through the black box O_x, and we want to find a marked item, if it exists.

Problem 13 (Grover's problem)**.** Given access to x through a black box O_x, find an i such that $x(i) = 1$, or report that no such i exists.

Grover's algorithm will show that this task can be done with $O(\sqrt{N})$ accesses to O_x and $O(\sqrt{N} \log N)$ additional quantum gates.[9] Classically $\Theta(N)$ accesses to the black box are required, so there is a quadratic speedup with a quantum computer for this problem.

Although Grover's algorithm is often referred to as a "database search algorithm", it can be misleading to think of there actually being N items in the physical world. A more intuitive visualization is to think of the elements x as being potential solutions to some combinatorial or mathematical problem, where it is easy to recognize a solution (i.e. to compute the function f), but difficult to find a solution x itself.

As an example, consider the problem of trying to factor a number k (and forget about Shor's algorithm for the moment). A brute force classical algorithm would be to check if each number $1, \cdots, \lfloor\sqrt{k}\rfloor$ is a factor of k, using $O(\sqrt{k})$ operations. Quantumly, we could implement the black box O_x, where $x(i) = 1$ if and only if i divides k, $i \in \{1, \cdots, \lfloor\sqrt{k}\rfloor\}$, and perform Grover's algorithm to find a factor of k in $O(k^{1/4})$ operations, a quadratic speedup over the classical brute force approach. Although there are better algorithms for both the classical and quantum cases for this problem, this is not the case for many other problems of interest, and often a quadratic speedup can be obtained by applying Grover's algorithm to a classical search problem when the function $x(i)$ can be easily implemented.

Finally, we note that if we keep the second register $|r\rangle$ in the state $(|0\rangle - |1\rangle)/\sqrt{2}$, one can check that

$$O_x|i\rangle\frac{|0\rangle - |1\rangle}{\sqrt{2}} = (-1)^{x(i)}|i\rangle\frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{1.4.2}$$

This means that using one call to O_x, we can implement one call to the *phase oracle* U_x, where

$$U_x|i\rangle = (-1)^{x(i)}|i\rangle \tag{1.4.3}$$

We will use the phase oracle U_x for Grover's algorithm in the next section.

1.4.2 Grover's algorithm

We will only demonstrate Grover's algorithm for the simplified case where either there is a unique i^* such that $x(i^*) = 1$, or no such i^* exists. In other words, there is at most one marked element. For the case where the number of marked elements is greater than one and possibly unknown, see [3] for a presentation (and references therein).

Assume that i^* exists. Intuitively, the algorithm will first create the state

$$|S\rangle = (|0\rangle + |1\rangle + \cdots + |N-1\rangle)/\sqrt{N}, \tag{1.4.4}$$

the equal superposition over all elements of $S = \{0, \cdots, N-1\}$. Note that $|S\rangle$ is in the two-dimensional subspace spanned by the orthogonal states $|i^*\rangle$ and

$$\left|S^{\perp}\right\rangle = \frac{1}{\sqrt{N-1}} \sum_{i \in S - \{i^*\}} |i\rangle, \tag{1.4.5}$$

[9]Grover's algorithm is often said to require $O(\sqrt{N})$ operations, but it appears to the author that this confusingly refers to the *depth* of the circuit, i.e. this counting allows quantum gates to be applied in parallel.

the equal superposition of all elements of S except for the marked element i^*. More precisely,

$$|S\rangle = \cos\theta\left|S^\perp\right\rangle + \sin\theta|i^*\rangle, \tag{1.4.6}$$

where

$$\cos\theta = \sqrt{\frac{N-1}{N}}, \quad \sin\theta = \frac{1}{\sqrt{N}} \tag{1.4.7}$$

In Grover's algorithm the quantum state will never leave this two-dimensional subspace. In fact, the algorithm will rotate the state towards the desired state $|i^*\rangle$, with each iteration being a rotation of angle 2θ. The algorithm thus needs a total number of $O(1/\theta) = O(\sqrt{N})$ of iterations.

More precisely, note that

$$U_x\left|S^\perp\right\rangle = \left|S^\perp\right\rangle U_x|i^*\rangle = -|i^*\rangle \tag{1.4.8}$$

and so in the two-dimensional subspace spanned by $|i^*\rangle$ and $\left|S^\perp\right\rangle$, U_x is the reflection about $\left|S^\perp\right\rangle$. Moreover, consider the following operation:

$$G = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n}. \tag{1.4.9}$$

This can be implemented efficiently: the operation $2|0\rangle\langle 0| - I$ gives a -1 phase shift unless the quantum state is 0. Since $H^{\otimes n}|0\rangle = |\psi\rangle$ we have that

$$G = 2|\psi\rangle\langle\psi| - I \tag{1.4.10}$$

This says that G is the reflection about the state $|\psi\rangle$. Since the states $|\psi\rangle$ and $\left|S^\perp\right\rangle$ are an angle $\theta = \arcsin(1/\sqrt{N})$ apart, the unitary operation GU_x (reflection first about $\left|S^\perp\right\rangle$, then $|\psi\rangle$) is a rotation towards $|i^*\rangle$ of angle 2θ. Grover's algorithm is simply to apply this rotation multiple times until we get close to $|i^*\rangle$:

Algorithm 14 (Grover's Algorithm). We will use a single register of dimension $N = 2^n$ (or n qubits) for the algorithm.

1. **Initialization:** Prepare the state $|\psi\rangle = H^{\otimes n}|0\rangle$.
2. Apply the operation GU_x T times to obtain the state $\cos((2T+1)\theta)\left|S^\perp\right\rangle + \sin((2T+1)\theta)|i^*\rangle$, where $(2T+1)\theta \approx \pi/2$, or remain in the state $|\psi\rangle$ if there is no marked state.
3. Measure in the computational basis, to obtain with high probability the marked state i^*, if it exists. (We can easily check whether the measurement result is the marked state by using the oracle O_x.)

This algorithm uses $O(1/\theta) = O(\sqrt{N})$ applications of GU_x, and thus $O(\sqrt{N})$ accesses to U_x. Since each application of G requires $\log N$ gates, an additional $O(\sqrt{N}\log N)$ gates are required. *Amplitude amplification* refers to this idea of amplifying the overlap between our initial state $|\psi\rangle$ and the desired state $|i^*\rangle$.

1.4.3 Query Complexity

Grover's algorithm serves as a good backdrop to introduce the notion of *query complexity*, an important concept in the study of quantum algorithms. As in the case with Grover's problem, assume that we are given access to a boolean function $x : \{0, \cdots, N-1\} \to \{0, 1\}$, implemented as a black box O_x:

$$O_x|i\rangle|r\rangle = |i\rangle|r \oplus x(i)\rangle \tag{1.4.11}$$

Instead of thinking of x as a function, it will also often be convenient to treat it as a string of length N, by simply concatenating its function values: $x = x_0x_1 \cdots x_{N-1}$, where $x_i = x(i)$.

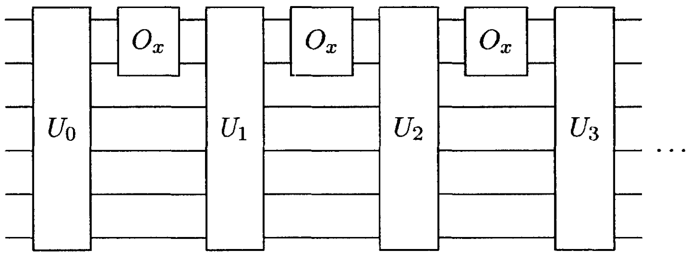
Given access to O_x, we wish to compute some function $f(x)$. In the query complexity model, we are interested only in the number of calls to O_x required, and not in the number of additional gates:

Definition 15 (Query Complexity). Given access to O_x, the *query complexity* of a function $f(x)$ is the number of calls required for an algorithm to compute O_x. This varies depending on what kind of computational resources are allowed:

- For quantum algorithms, we have the quantum query complexity, $Q_\delta(f)$, where δ is the error allowed.
- For randomized classical algorithms, we have the randomized query complexity, $R_\delta(f)$, where δ is the error allowed.
- For deterministic classical algorithms, we have the deterministic query complexity, $D(f)$.

We will also often just write $Q(f) = Q_{0.01}(f)$ and $R(f) = R_{0.01}(f)$; the number 0.01 doesn't matter if it is replaced by any number less than 1/2.

A quantum algorithm with access to a black box O_x looks like the following, where the U_t's are unitaries independent of x:



(1.4.12)

For example, consider the OR function: $OR(x) = 0$ if x is the all-zero string ($x = 000\cdots000$), and $OR(x) = 1$ otherwise. Classically there is no better way to evaluate the OR function other than checking each input one-by-one, and hence $R(OR), D(OR) = \Theta(N)$. In the quantum case, however, Grover's algorithm (when generalized for an arbitrary number of marked items) means that $Q(OR) = O(\sqrt{N})$.

Another example is that of Shor's algorithm for order finding, since it solves the following problem: given a function $x(i)$ $\{0, \cdots, N^2 - 1\} \to \{0, \cdots, N-1\}$ satisfying $x(i) = x(i + r)$ for some $r < N$, find the smallest such r (which we will call $ORDER(x)$). In this case x can still be viewed as a string by concatenating all the outputs $x(i)$ together. The query complexity of this problem is only poly$(\log N)$, unlike Grover's algorithm. The function $ORDER(x)$ is a *partial function*, since $ORDER(x)$ is defined only for strings x satisfying certain conditions ($x(i) = x(i + r)$ for some i). In contrast, OR is a *total function*, since it is defined for all strings x without preconditions on x.

Obviously query complexity is not the only thing that one should worry about when designing an algorithm, since in addition to the calls to the black box, other quantum gates are also needed to construct a quantum circuit. A query efficient algorithm is not necessarily computationally efficient; the most notorious example is perhaps the non-Abelian hidden subgroup problem, where there is an algorithm using a polynomial number of queries but exponential number of gates [25]. However, query complexity turns out to be much easier to work with: often in the design of quantum algorithms, a query efficient algorithm is first proposed, and then later made computationally efficient. In fact, recent tools have been developed to construct query efficient (but not time efficient) algorithms, such as the quantum walk formalism [26, 27, 28, 29], learning graphs [30], and bomb query complexity (Chapter 2 of this thesis).

Another reason why query complexity is important is that there are many tools for proving *lower bounds* for query complexity. For example, Bennett et al. showed in 1997 [31] that $Q(OR) = \Omega(\sqrt{N})$: the quantum query complexity for the OR function is lower bounded by $\Omega(\sqrt{N})$. Note the significance of this result: this says that any algorithms solving Grover's algorithm requires at least $\Omega(\sqrt{N})$ operations (since it requires at least $\Omega(\sqrt{N})$ black box calls). Therefore Grover's algorithm is optimal in terms of query complexity, and nearly optimal in time complexity up to a logarithmic factor.

Since Bennett et al.'s breakthrough result, there have been two main methods of proving query complexity lower bounds: the polynomial method [32], and the adversary method [33]. These methods have been applied to a wide range of problems; see the online *Quantum Algorithm Zoo* [18] for many results. In particular, the general adversary lower bound [34] has been shown to tightly characterize quantum query complexity [35, 36, 37], but calculating such a tight bound seems difficult in general. In constrast, to the knowledge of the author there is no general method of proving time complexity lower bounds, other than proving a query complexity lower bound. We will make use of the adversary method in Chapter 2.

Finally, we close this section with the following theorem, proved using the polynomial method:

Theorem 16 ([32]). *For total functions f, the query complexities $Q(f)$, $D(f)$, and $R(f)$ are polynomially related:*

$$R(f) \leq D(f) = O(Q(f)^6). \tag{1.4.13}$$

This theorem says that to look for superpolynomial improvements in query complexity, we need to look for partial functions. Shor's algorithm is such an example of a superpolynomial speedup. However, there is a long-standing conjecture that this theorem is far from optimal, and actually $D(f) = O(Q(f)^2)$ for total functions; this separation is achieved by the OR function and Grover's algorithm.

1.5 Organization of this thesis

We now describe the contents of the rest of this thesis.

Chapter 2 introduces the *bomb query* model, a variation of the quantum query model in which after each query we immediately measure the query result, and end the computation if the result is 1. Although it may appear that nothing useful can be done in this model, we show that in fact any function computable in the quantum query model can be computed in the bomb query model, but with a quadratic increase in the number of queries. This exact characterization is shown using ideas from the Elitzur-Vaidman bomb tester, and the adversary method for query complexity. We then construct a general method to convert certain classical algorithms to bomb query algorithms, and thus improved quantum algorithms; this method yields the first nontrival quantum query algorithm for maximum bipartite matching. Finally, we introduce a related model called the *projective query* model, and speculate how it may help resolve the conjecture that $R(f) = O(Q(f)^2)$ for total functions.

Chapter 3 defines the *normalizer circuit* model over Abelian groups. A normalizer circuit is a restricted quantum circuit, defined over Hilbert spaces associated with Abelian groups, in which only three types of quantum operations are allowed: quantum Fourier transforms, gates that implement group automorphisms, and gates that impart quadratic phases. This chapter focuses on only giving basic defintions and properties of such circuits; in the process we also describe the necessary group theory setting. In the case that the underlying group is infinite, it is infeasible to list all outputs of automorphisms and quadratic phase functions; we therefore develop normal form characterizations of automorphisms and quadratic phase functions to describe such gates succintly.

Chapter 4 gives the first of our two complexity results regarding normalizer circuits: if the underlying Abelian group is decomposed into primitive subgroups, then normalizer circuits can be simulated efficiently using a classical computer. (The finite case is known [1, 2]) We show this by generalizing Pauli gates to the Hilbert space associated with Abelian groups, and show that normalizer gates *normalize* the generalized Pauli group. Thus normalizer circuits are generalized Clifford circuits, and we show that an extended Gottesman-Knill theorem holds. The main new technical challenges we overcome are to efficiently represent stabilizer groups in terms of linear maps (since the stabilizer groups are infinitely generated, listing a set of generators no longer works); to compute the support of a stabilizer state; and to develop ϵ-net techniques to sample the support.

Chapter 5 gives the second of the complexity results regarding normalizer circuits: if the underlying Abelian group is not decomposed (we say such a group is a black box group), then normalizer circuits contain many algorithms for solving the Abelian hidden subgroup problem, including Shor's algorithm for factoring. We then define an extended variant of the problem of decomposing finite Abelian groups into cyclic subgroups, and show that this problem is *complete* for the compelxity class associated with these normalizer circuits. In other words, decomposing Abelian groups can be done with normalizer circuits over black box groups, and normalizer circuits over black box groups can be classically simulated given a black box to decompose Abelian groups. This yields a *no-go theorem* for finding new quantum algorithms based on normalizer circuits.

Chapter 2

Upper Bounds for Quantum Query Complexity Based on the Elitzur-Vaidman Bomb Tester

In this chapter we introduce and study the *bomb query model*, an alternative oracle model of query complexity. The bomb query model can be obtained from the usual quantum query model by adding restrictions on how the black box oracle can be applied. We show that the bomb query complexity is closely related to the usual quantum query complexity; this result will in turn inspire a general method for obtaining quantum query algorithms from classical ones.

The work presented in this chapter is joint work with Han-Hsuan Lin, and is mostly excerpted from [38].

2.1 Introduction

Our main result (Theorem 17) is that the bomb query complexity, $B(f)$, is characterized by the square of the quantum query complexity $Q(f)$:

Theorem 17.

$$B(f) = \Theta(Q(f)^2). \tag{2.1.1}$$

We prove the upper bound, $B(f) = O(Q(f)^2)$ (Theorem 19), by adapting Kwiat et al.'s solution of the Elitzur-Vaidman bomb testing problem (Section 2.2, [39]) to our model. We prove the lower bound, $B(f) = \Omega(Q(f)^2)$ (Theorem 20), by demonstrating that $B(f)$ is lower bounded by the square of the general adversary bound [34], $(\mathrm{Adv}^{\pm}(f))^2$. The aforementioned result that the general adversary bound tightly characterizes the quantum query complexity [35, 36, 37], $Q(f) = \Theta(\mathrm{Adv}^{\pm}(f))$, allows us to draw our conclusion.

This characterization of Theorem 17 allows us to give *nonconstructive* upper bounds to the quantum query complexity for some problems. For some functions f a bomb query algorithm is easily designed by adapting a classical algorithm: specifically, we show that (stated informally):

Theorem 24 (informal). *Suppose there is a classical algorithm that computes $f(x)$ in T queries, and the algorithm guesses the result of each query (0 or 1), making no more than an expected G mistakes for all x. Then we can design a bomb query algorithm that uses*

$O(TG)$ queries, and hence $B(f) = O(TG)$. By our characterization of Theorem 17, $Q(f) = O(\sqrt{TG})$.

This result inspired us to look for an explicit quantum algorithm that reproduces the query complexity $O(\sqrt{TG})$. We were able to do so:

Theorem 25. *Under the assumptions of Theorem 24, there is an explicit algorithm (Algorithm 27) for f with query complexity $O(\sqrt{TG})$.*

Using Algorithm 27, we were able to give an $O(n^{3/2})$ algorithm for the single-source shortest paths (SSSP) problem in an unweighted graph with n vertices, beating the best-known $O(n^{3/2}\sqrt{\log n})$ algorithm [40]. A more striking application is our $O(n^{7/4})$ algorithm for maximum bipartite matching; in this case the best-known upper bound was the trivial $O(n^2)$, although the time complexity of this problem had been studied in [41] (and similar problems in [42]).

Finally, in Section 2.7 we briefly discuss a related query complexity model, which we call the *projective query complexity* $P(f)$, in which each quantum query to x is immediately followed by a classical measurement of the query result. This model seems interesting to us because its power lies between classical and quantum: we observe that $P(f) \leq B(f) = \Theta(Q(f)^2)$ and $Q(f) \leq P(f) \leq R(f)$, where $R(f)$ is the classical randomized query complexity. We note that Regev and Schiff [43] showed that $P(OR) = \Theta(N)$.

Past and related work

Mitchison and Jozsa have proposed a different computational model called *counterfactual computation* [44], also based on interaction-free measurement. In counterfactual computation the result of a computation may be learnt without ever running the computer. There has been some discussion on what constitutes counterfactual computation; see for example [45, 46, 47, 48, 49, 50, 51].

There have also been other applications of interaction-free measurement to quantum cryptography. For example, Noh has proposed counterfactual quantum cryptography [52], where a secret key is distributed between parties, even though a particle carrying secret information is not actually transmitted. More recently, Brodutch et al. proposed an adaptive attack [53] on Wiesner's quantum money scheme [54]; this attack is directly based off Kwiat et al.'s solution of the Elitzur-Vaidman bomb testing problem [39].

Our Algorithm 27 is very similar to Kothari's algorithm for the oracle identification problem [55], and we refer to his analysis of the query complexity in our work.

The projective query model we detail in Section 2.7 was, to our knowledge, first considered by Aaronson in unpublished work in 2002 [56].

Discussion and outlook

Our work raises a number of open questions. The most obvious ones are those pertaining to the application of our recipe for turning classical algorithms into bomb algorithms, Theorem 24:

- Can we generalize our method to handle non-boolean input and output? If so, we might be able to find better upper bounds for the adjacency-list model, or to study graph problems with weighted edges.

- Can our explicit (through Theorem 25) algorithm for maximum bipartite matching be made more *time* efficient? The best known quantum algorithm for this task has time complexity $O(n^2 \log n)$ in the adjacency matrix model [41].

- Finally, can we find more upper bounds using Theorem 24? For example, could the query complexity of the maximum matching problem on general nonbipartite graphs be improved with Theorem 24, by analyzing the classical algorithm of Micali and Vazirani [57]?

Perhaps more fundamental, however, is the possibility that the bomb query complexity model will help us understand the relationship between the classical randomized query complexity, $R(f)$, and the quantum query complexity $Q(f)$. It is known [32] that for all total functions f, $R(f) = O(Q(f)^6)$; however, there is a long-standing conjecture that actually $R(f) = O(Q(f)^2)$. In light of our results, this conjecture is equivalent to the conjecture that $R(f) = O(B(f))$. Some more open questions, then, are the following:

- Can we say something about the relationship between $R(f)$ and $B(f)$ for specific classes of functions? For example, is $R(f) = O(B(f)^2)$ for total functions?

- Referring to the notation of Theorem 24, we have $B(f) = O(TG)$. Is the quantity G related to other measures used in the study of classical decision-tree complexity, for example the certificate complexity, sensitivity [58], block sensitivity [59], or (exact or approximate) polynomial degree? (For a review, see [60].)

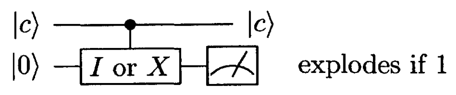
- What about other query complexity models that might help us understand the relationship between $R(f)$ and $Q(f)$? One possibility is the projective query complexity, $P(f)$, considered in Section 2.7. Regev and Schiff [43] have shown (as a special case of their results) that even with such an oracle, $P(OR) = \Theta(N)$ queries are needed to evaluate the OR function.

We hope that further study on the relationship between bomb and classical randomized complexity will shed light on the power and limitations of quantum computation.

2.2 The Elitzur-Vaidman bomb testing problem

The Elitzur-Vaidman bomb testing problem [61] is a well-known thought experiment to demonstrate how quantum mechanics differs drastically from our classical perceptions. This problem demonstrates dramatically the possibility of *interaction free measurements*, the possibility of a measurement on a property of a system without disturbing the system.

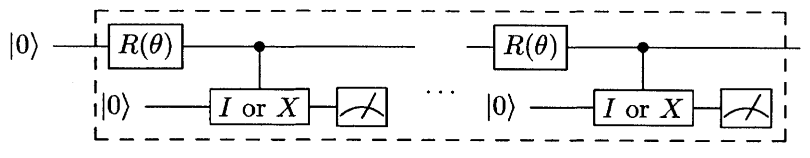
The bomb-testing problem is as follows: assume we have a bomb that is either a dud or a live bomb. The only way to interact with the bomb is to probe it with a photon: if the bomb is a dud, then the photon passes through unimpeded; if the bomb is live, then the bomb explodes. We would like to determine whether the bomb is live or not without exploding it. If we pass the photon through a beamsplitter before probing the bomb, we can implement the *controlled probe*, pictured below:



(2.2.1)

In circuit 2.2.1, the controlled gate is I if the bomb is a dud, and X if it is live. The first (control) register represents whether the photon probes the bomb or not: if the control is $|1\rangle$ the photon probes the bomb, and if the control is $|0\rangle$ the photon does not. Thus the bomb explodes only if the control is found to be in the state $|1\rangle$ and the bomb is live.

It was shown in [39] how to determine whether a bomb was live with arbitrarily low probability of explosion by making use of the quantum Zeno effect [62]. Specifically, writing $R(\theta) = \exp(i\theta X)$ (the unitary operator rotating $|0\rangle$ to $|1\rangle$ in $\pi/(2\theta)$ steps), the following circuit determines whether the bomb is live with failure probability $O(\theta)$:



$\pi/(2\theta)$ times in total

(2.2.2)

If the bomb is a dud, then the controlled probes do nothing, and repeated application of $R(\theta)$ rotates the control bit from $|0\rangle$ to $|1\rangle$. If the bomb is live, the bomb explodes with $O(\theta^2)$ probability in each application of the probe, projecting the control bit back to $|0\rangle$. After $O(1/\theta)$ iterations the control bit stays in $|0\rangle$, with only a $O(\theta)$ probability of explosion. Using $O(1/\theta)$ operations, we can thus tell a dud bomb apart from a live one with only $O(\theta)$ probability of explosion.

2.3 Bomb query complexity

In this section we introduce a new query complexity model, which we call the *bomb query complexity*. A circuit in the bomb query model is a restricted quantum query circuit, with the following restrictions on the usage of the quantum oracle:

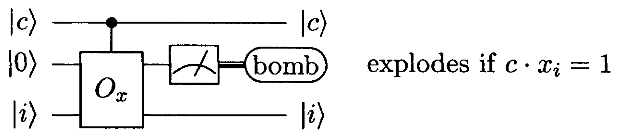
1. We have an extra control register $|c\rangle$ used to control whether O_x is applied (we call the controlled version CO_x):

$$CO_x|c, r, i\rangle = |c, r \oplus (c \cdot x_i), i\rangle. \tag{2.3.1}$$

   where $\cdot$ indicates boolean AND.

2. The record register, $|r\rangle$ in the definition of CO_x above, *must* contain $|0\rangle$ before CO_x is applied.

3. After CO_x is applied, the record register is immediately measured in the computational basis (giving the answer $c \cdot x_i$), and the algorithm *terminates immediately if a 1 is measured* (if $c \cdot x_i = 1$). We refer to this as *the bomb blowing up* or *the bomb exploding.*



(2.3.2)

Throughout this chapter we assume f has boolean input, i.e. the domain is $D \subseteq \{0,1\}^N$. We define the *bomb query complexity* $B_{\epsilon,\delta}(f)$ to be the minimum number of times the above circuit needs to be applied in an algorithm such that the following hold for all input x:

- The algorithm reaches the end without the bomb exploding with probability at least $1-\epsilon$. We refer to the probability that the bomb explodes as the *probability of explosion.*
- The total probability that the bomb either explodes or fails to output $f(x)$ correctly is no more than $\delta \geq \epsilon$.

The above implies that the algorithm outputs the correct answer with probability at least $1-\delta$.

The effect of the above circuit is equivalent to applying the following projector on $|c,i\rangle$:

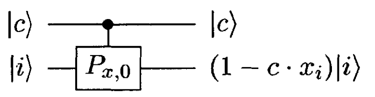
$$M_x = CP_{x,0} = \sum_{i=1}^{N} |0,i\rangle\langle 0,i| + \sum_{x_i=0} |1,i\rangle\langle 1,i| \tag{2.3.3}$$

$$= I - \sum_{x_i=1} |1,i\rangle\langle 1,i|. \tag{2.3.4}$$

$CP_{x,0}$ (which we will just call M_x in our proofs later on) is the controlled version of $P_{x,0}$, the projector that projects onto the indices i on which $x_i = 0$:

$$P_{x,0} = \sum_{x_i=0} |i\rangle\langle i|. \tag{2.3.5}$$

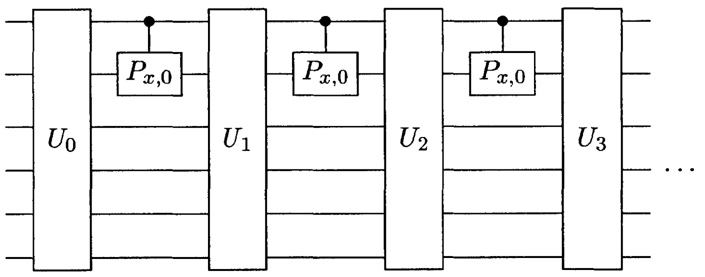
Thus Circuit 2.3.2 is equivalent to the following circuit :



(2.3.6)

In this notation, the square of the norm of a state is the probability that the state has survived to this stage, i.e. the algorithm has not terminated. The norm of $(1-c\cdot x_i)|x_i\rangle$ is 1 if $c\cdot x_i = 0$ (the state survives this stage), and 0 otherwise (the bomb blows up).

A general circuit in this model looks like the following:



It is not at all clear that gap amplification can be done efficiently in the bomb query model to improve the error δ; after all, repeating the circuit multiple times increases the chance that the bomb blows up. However, it turns out that the complexity $B_{\epsilon,\delta}(f)$ is closely related to $Q_\delta(f)$, and therefore the choice of δ affects $B_{\epsilon,\delta}(f)$ by at most a $\log^2(1/\delta)$ factor as long as $\delta \geq \epsilon$ (see Lemma 18). We therefore often omit δ by setting $\delta = 0.01$, and write $B_{\epsilon,0.01}(f)$ as $B_\epsilon(f)$. Sometimes we even omit the ϵ.

Finally, note that the definition of the bomb query complexity $B(f)$ is inherently *asymmetric* with respect to 0 and 1 in the input: querying 1 causes the bomb to blow up, while querying 0 is safe. In Section 2.5.1, we define a *symmetric* bomb query model and its corresponding query complexity, $\tilde{B}_{\epsilon,\delta}(f)$. We prove that this generalized symmetric model is asymptotically equivalent to the original asymmetric model, $\tilde{B}_{\epsilon,\delta}(f) = \Theta(B_{\epsilon,\delta}(f))$, in Lemma 21. This symmetric version of the bomb query complexity will turn out to be useful in designing bomb query algorithms.

2.4 Main result

Our main result is the following:

Theorem 17. *For all functions f with boolean input alphabet, and numbers ϵ satisfying $0 < \epsilon \leq 0.01$,*

$$B_{\epsilon,0.01}(f) = \Theta\left(\frac{Q_{0.01}(f)^2}{\epsilon}\right). \tag{2.4.1}$$

Here 0.01 can be replaced by any constant no more than $1/10$.

Proof. The upper bound $B_{\epsilon,\delta}(f) = O(Q_\delta(f)^2/\epsilon)$ is proved in Theorem 19. The lower bound $B_{\epsilon,\delta}(f) = \Omega(Q_{0.01}(f)^2/\epsilon)$ is proved in Theorem 20. □

Lemma 18. *For all functions f with boolean input alphabet, and numbers ϵ, δ satisfying $0 < \epsilon \leq \delta \leq 1/10$,*

$$B_{\epsilon,0.1}(f) = O(B_{\epsilon,\delta}(f)), \quad B_{\epsilon,\delta}(f) = O(B_{\epsilon,0.1}(f)\log^2(1/\delta)). \tag{2.4.2}$$

In particular, if δ is constant,

$$B_{\epsilon,\delta}(f) = \Theta(B_{\epsilon,0.1}(f)). \tag{2.4.3}$$

Proof. This follows from Theorem 19 and the fact that $Q_{0.1}(f) = O(Q_\delta(f))$ and $Q_\delta(f) = O(Q_{0.1}(f)\log(1/\delta))$. □

Because of this result, we will often omit the 0.01 in $B_{\epsilon,0.01}$ and write simply B_ϵ.

2.4.1 Upper bound

Theorem 19. *For all functions f with boolean input alphabet, and numbers ϵ, δ satisfying $0 < \epsilon \leq \delta \leq 1/10$,*

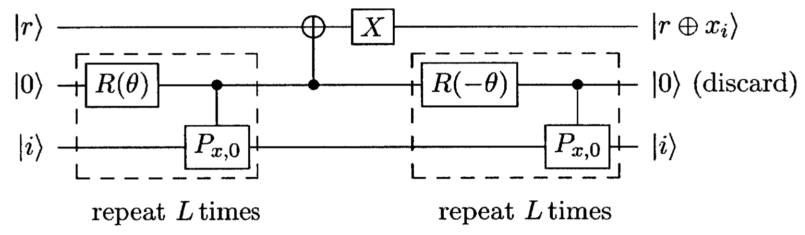
$$B_{\epsilon,\delta}(f) = O(Q_\delta(f)^2/\epsilon). \tag{2.4.4}$$

The proof follows the solution of Elitzur-Vaidman bomb-testing problem ([39], or Section 2.2). By taking advantage of the Quantum Zeno effect [62], using $O(\frac{Q(f)}{\epsilon})$ calls to M_x, we can simulate one call to O_x with probability of explosion $O(\frac{\epsilon}{Q(f)})$. Replacing all O_x queries with this construction results in a bounded error algorithm with probability of explosion $O(\frac{\epsilon}{Q(f)}Q(f)) = O(\epsilon)$.

Proof. Let $\theta = \pi/(2L)$ for some large positive integer L (chosen later), and let $R(\theta)$ be the rotation

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{2.4.5}$$

We claim that with $2L$ calls to the bomb oracle $M_x = CP_{x,0}$, we can simulate O_x by the following circuit with probability of explosion less than $\pi^2/(2L)$ and error $O(1/L)$.



(2.4.6)

In words, we simulate O_x acting on $|r, i\rangle$ by the following steps:

1. Append an ancilla qubit $|0\rangle$, changing the state into $|r, 0, i\rangle$.
2. Repeat the following L times:
   (a) apply $R(\theta)$ on the second register
   (b) apply M_x on the third register controlled by the second register.

   At this point, if the bomb hasn't blown up, the second register should contain $1 - x_i$.
3. Apply $CNOT$ on the first register controlled by the second register; this copies $1 - x_i$ to the first register.
4. Apply a NOT gate to the first register.
5. Repeat the following L times to uncompute the second (ancilla) register :
   (a) apply $R(-\theta)$ on the second register
   (b) apply M_x on the third register controlled by second register
6. Discard the second (ancilla) register.

We now calculate explicitly the action of the circuit on an arbitrary state to confirm our claims above. Consider how the circuit acts on the basis state $|r, 0, i\rangle$ (the second register being the appended ancilla). We break into cases:

- If $x_i = 0$, then $P_{x,0}|i\rangle = |i\rangle$, so the controlled projections do nothing. Thus in Step 2 the rotation $R(\theta)^L = R(\pi/2)$ is applied to the ancilla qubit, rotating it from 0 to 1. After Step 2 then, the state is $|r, 1, i\rangle$. Step 3 and 4 together do not change the state, while Step 5 rotates the ancilla back to 0, resulting in the final state $|r, 0, i\rangle$.
- If $x_i = 1$, then $P_{x,0}|i\rangle = 0$, and

$$M_x|0, i\rangle = |0, i\rangle, \quad M_x|1, i\rangle = 0 \quad (\text{for } x_i = 1) \tag{2.4.7}$$

Therefore in Step 2 and Step 5, after each rotation $R(\pm\theta)$, the projection $CP_{x,0}$ projects the ancilla back to 0:

$$M_x R(\theta)|0,i\rangle = M_x(\cos\theta|0\rangle + \sin\theta|1\rangle)|i\rangle = \cos\theta|0,i\rangle \quad (\text{for } x_i = 1) \tag{2.4.8}$$

Each application of $M_x R(\theta)$ thus has no change on the state other than to shrink its amplitude by $\cos\theta$. The CNOT in Step 3 has no effect (since the ancilla stays in 0), and Step 4 maps $|r\rangle$ to $|r \oplus 1\rangle$. Since there are $2L$ applications of this shrinkage (in Step 2 and 5), the final state is $\cos^{2L}\theta|r \oplus 1, 0, i\rangle$.

We can now combine the two cases: by linearity, the application of the circuit on a general state $\sum_{r,i} a_{r,i}|r,i\rangle$ (removing the ancilla) is

$$\sum_{r,i} a_{r,i}|r,i\rangle \to \sum_{r\in\{0,1\},x_i=0} a_{r,i}|r,i\rangle + \sum_{r\in\{0,1\},x_i=1} a_{r,i}\cos^{2L}(\theta)|r\oplus 1,i\rangle \tag{2.4.9}$$

$$= \sum_{r,i} a_{r,i}\cos^{2Lx_i}\left(\frac{\pi}{2L}\right)|r\oplus x_i, i\rangle \equiv |\psi'\rangle \tag{2.4.10}$$

Thus the effect of this construction simulates the usual quantum oracle $|r,i\rangle \to |r\oplus x_i, i\rangle$ with probability of explosion no more than

$$1 - \cos^{4L}\left(\frac{\pi}{2L}\right) \le 1 - \left(1 - \frac{\pi^2}{4L^2}\right)^{2L} \le \frac{\pi^2}{2L}. \tag{2.4.11}$$

Moreover, the difference between the output of our circuit, $|\psi'\rangle$, and the output on the quantum oracle, $|\psi\rangle = \sum_{r,i} a_{r,i}|r\oplus x_i, i\rangle$, is

$$\left\||\psi'\rangle - |\psi\rangle\right\| = \left\|\sum_{r\in\{0,1\},x_i=1} a_{r,i}(1-\cos^{2L}(\theta))|r\oplus 1,i\rangle\right\| \tag{2.4.12}$$

$$\le 1 - \cos^{2L}\frac{\pi}{2L} \le \frac{\pi^2}{4L}. \tag{2.4.13}$$

Given this construction, we can now prove our theorem. Suppose we are given a quantum algorithm that finds $f(x)$ with $Q_{\delta'}(f)$ queries, making at most $\delta' = \delta - \epsilon$ error. We construct an algorithm using bomb oracles instead by replacing each of the applications of the quantum oracle O_x by our circuit construction (2.4.6), where we choose

$$L = \left\lceil \frac{\pi^2}{2\epsilon} Q_{\delta'}(f) \right\rceil \tag{2.4.14}$$

Then the probability of explosion is no more than

$$\frac{\pi^2}{2L} Q_{\delta'}(f) \le \epsilon \tag{2.4.15}$$

and the difference between the final states, $|\psi_f\rangle$ and $\left|\psi'_f\right\rangle$, is at most

$$\left\||\psi'_f\rangle - |\psi_f\rangle\right\| \le \frac{\pi^2}{4L} Q_{\delta'}(f) \le \frac{\epsilon}{2}. \tag{2.4.16}$$

Therefore

$$\left|\langle\psi'_f|P|\psi'_f\rangle - \langle\psi_f|P|\psi_f\rangle\right| \leq \left|\langle\psi'_f|P|\psi'_f\rangle - \langle\psi_f|P|\psi'_f\rangle\right| + \left|\langle\psi'_f|P|\psi_f\rangle - \langle\psi_f|P|\psi_f\rangle\right| \tag{2.4.17}$$

$$\leq \left\||\psi'_f\rangle\right\| \left\|P\left(|\psi'_f\rangle - |\psi_f\rangle\right)\right\| + \left\|P\left(|\psi'_f\rangle - |\psi_f\rangle\right)\right\| \left\||\psi_f\rangle\right\| \tag{2.4.18}$$

$$\leq \epsilon/2 + \epsilon/2 = \epsilon \tag{2.4.19}$$

for any projector P (in particular, the projector that projects onto the classical answer at the end of the algorithm). The algorithm accumulates at most ϵ extra error at the end, giving a total error of no more than $\delta' + \epsilon = \delta$. This algorithm makes $2LQ_{\delta'}(f) < \frac{\pi^2}{\epsilon}Q^2_{\delta'}(f) + 2Q_{\delta'}(f)$ queries to the bomb oracle, and therefore

$$B_{\epsilon,\delta}(f) < \frac{\pi^2}{\epsilon}Q_{\delta-\epsilon}(f)^2 + 2Q_{\delta-\epsilon}(f) \tag{2.4.20}$$

$$= O\left(\frac{Q_{\delta-\epsilon}(f)^2}{\epsilon}\right). \tag{2.4.21}$$

From this we can derive that $B_{\epsilon,\delta}(f) = O(Q_\delta(f)^2/\epsilon)$:

$$B_{\epsilon,\delta}(f) < B_{\epsilon/2,\delta}(f) \tag{2.4.22}$$

$$= O\left(\frac{Q_{\delta-\epsilon/2}(f)^2}{\epsilon}\right), \quad \text{by 2.4.21} \tag{2.4.23}$$

$$= O\left(\frac{Q_\delta(f)^2}{\epsilon}\right), \quad \text{since } \frac{\delta}{2} \leq \delta - \frac{\epsilon}{2}. \tag{2.4.24}$$

□

2.4.2 Lower bound

Theorem 20. *For all functions f with boolean input alphabet, and numbers ϵ, δ satisfying $0 < \epsilon \leq \delta \leq 1/10$,*

$$B_{\epsilon,\delta}(f) = \Omega(Q_{0.01}(f)^2/\epsilon). \tag{2.4.25}$$

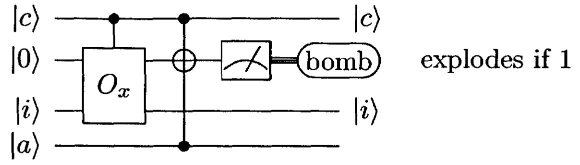
The proof of this result uses the generalized adversary bound $\text{Adv}^{\pm}(f)$ [34]: we show that $B_\epsilon(f) = \Omega(\text{Adv}^{\pm}(f)^2/\epsilon)$, and then use the known result that $Q(f) = O(\text{Adv}^{\pm}(f))$ [37]. The complete proof is given in Appendix A.1.

2.5 Generalizations and Applications

We now discuss applications of the result $B_\epsilon(f) = \Theta(Q(f)^2/\epsilon)$ that could be useful.

2.5.1 Generalizing the bomb query model

We consider modifying the bomb query model as follows. We require that the input string x can only be accessed by the following circuit:



(2.5.1)

Compare with Circuit 2.3.2; the difference is that there is now an extra register $|a\rangle$, and the bomb explodes only if both $x_i = a$ and the control bit is 1. In other words, the bomb explodes if $c \cdot (x_i \oplus a) = 1$. The three registers c, i, and a are allowed to be entangled, however. If we discard the second register afterwards, the effect of this circuit, written as a projector, is

$$\tilde{M}_x = \sum_{i\in[N],a\in\{0,1\}} |0,i,a\rangle\langle 0,i,a| + \sum_{i,a:x_i=a} |1,i,a\rangle\langle 1,i,a|. \tag{2.5.2}$$

Let $\tilde{B}_{\epsilon,\delta}(f)$ be the required number of queries to this modified bomb oracle $\tilde{M}_x$ to calculate $f(x)$ with error no more than δ, with a probability of explosion no more than ϵ. Using Theorem 17, we show that $\tilde{B}$ and B are equivalent up to a constant:

Lemma 21. *If $f : D \to E$, where $D \subseteq \{0,1\}^N$, and $\delta \leq 1/10$ is a constant, then $B_{\epsilon,\delta}(f) = \Theta(\tilde{B}_{\epsilon,\delta}(f))$.*

Proof. It should be immediately obvious that $B_{\epsilon,\delta}(f) \geq \tilde{B}_{\epsilon,\delta}(f)$, since a query in the B model can be simulated by a query in the $\tilde{B}$ model by simply setting $a = 0$. In the following we show that $B_{\epsilon,\delta}(f) = O(\tilde{B}_{\epsilon,\delta}(f))$.

For each string $x \in \{0,1\}^N$, define the string $\tilde{x} \in \{0,1\}^{2N}$ by concatenating two copies of x and flipping every bit of the second copy. In other words,

$$\tilde{x}_i = \begin{cases} x_i & \text{if } i \leq N \\ 1 - x_{i-N} & \text{if } i > N \end{cases}. \tag{2.5.3}$$

Let $\tilde{D} = \{\tilde{x} : x \in D\}$. Given a function $f : D \to \{0,1\}$, define $\tilde{f} : \tilde{D} \to \{0,1\}$ by $\tilde{f}(\tilde{x}) = f(x)$.

We claim that a $\tilde{B}$ query to x can be simulated by a B query to $\tilde{x}$. This can be seen by comparing $\tilde{M}_x$:

$$\tilde{M}_x = \sum_{i\in[N],a} |0,i,a\rangle\langle 0,i,a| + \sum_{i\in[N],a:x_i=a} |1,i,a\rangle\langle 1,i,a|. \tag{2.5.4}$$

and $M_{\tilde{x}}$:

$$M_{\tilde{x}} = \sum_{\tilde{i}\in[2N]} |0,\tilde{i}\rangle\langle 0,\tilde{i}| + \sum_{\tilde{i}\in[2N]:\tilde{x}_i=0} |1,\tilde{i}\rangle\langle 1,\tilde{i}|. \tag{2.5.5}$$

Recalling the definition of $\tilde{x}$ in 2.5.3, we see that these two projectors are exactly equal if we encode $\tilde{i}$ as (i,a), where $i \equiv \tilde{i} \mod N$ and $a = \lfloor \tilde{i}/N \rfloor$.

Since $\tilde{f}(\tilde{x}) = f(x)$, we thus have $\tilde{B}_{\epsilon,\delta}(f) = B_{\epsilon,\delta}(\tilde{f})$. Our result then readily follows; it can easily be checked that $Q(f) = Q(\tilde{f})$, and therefore by Theorem 17,

$$\begin{aligned}\tilde{B}_{\epsilon,\delta}(f) = B_{\epsilon,\delta}(\tilde{f}) &= \Theta\left(\frac{Q(\tilde{f})^2}{\epsilon}\right) \\ &= \Theta\left(\frac{Q(f)^2}{\epsilon}\right) \qquad (2.5.6)\end{aligned}$$

□

There are some advantages to allowing the projector $\tilde{M}_x$ instead of M_x. First of all, the inputs 0 and 1 in x are finally manifestly symmetric, unlike that in M_x (the bomb originally blew up if $x_i = 1$, but not if $x_i = 0$). Moreover, we now allow the algorithm to *guess* an answer to the query (this answer may be entangled with the index register i), and the bomb blows up only if the guess is wrong, controlled on c. This flexibility may allow more leeway in designing algorithms for the bomb query model, as we soon utilize.

2.5.2 Using classical algorithms to design bomb query algorithms

We now demonstrate the possibility that we can prove *nonconstructive* upper bounds on $Q(f)$ for some functions f, by creating bomb query algorithms and using that $Q(f) = \Theta(\sqrt{\epsilon B_\epsilon(f)})$. Consider for example the following classical algorithm for the OR function:

Algorithm 22 (Classical algorithm for OR). Pick some arbitrary ordering of the N bits, and query them one by one, terminating as soon as a 1 is seen. Return 1 if a 1 was queried; otherwise return 0.

We can convert this immediately to a bomb query algorithm for OR, by using the construction in the proof of Theorem 19. That construction allows us to implement the operation O_x in $O(\epsilon^{-1})$ queries, with $O(\epsilon)$ error and probability of explosion if $x_i = 1$ (but no error if $x_i = 0$). Thus we have the following:

Algorithm 23 (Bomb algorithm for OR). Query the N bits one-by-one, and apply the construction of Theorem 19 one bit at a time, using $O(1/\epsilon)$ operations each time. Terminate as soon as a 1 is seen, and return 1; otherwise return 0 if all bits are 0.

Since the algorithm ends as soon as a 1 is found, the algorithm only accumulates ϵ error in total. Thus this shows $B_\epsilon(OR) = O(N/\epsilon)$.

Note, however, that we have already shown that $Q(f) = \Theta(\sqrt{\epsilon B_\epsilon(f)})$ for boolean f. An $O(N/\epsilon)$ bomb query algorithm for OR therefore implies that $Q(OR) = O(\sqrt{N})$. We have showed the existence of an $O(\sqrt{N})$ quantum algorithm for the OR function, without actually constructing one!

We formalize the intuition in the above argument by the following theorem:

Theorem 24. *Let $f : D \to E$, where $D \subseteq \{0,1\}^N$. Suppose there is a classical randomized query algorithm $\mathcal{A}$, that makes at most T queries, and evaluates f with bounded error. Let the query results of $\mathcal{A}$ on random seed $s_\mathcal{A}$ be $x_{p_1}, x_{p_2}, \cdots, x_{p_{\tilde{T}(x)}}$, $\tilde{T}(x) \leq T$, where x is the hidden query string.*

Suppose there is another (not necessarily time-efficient) randomized algorithm $\mathcal{G}$, with random seed $s_\mathcal{G}$, which takes as input $x_{p_1}, \cdots, x_{p_{t-1}}$ and $s_\mathcal{A}$, and outputs a guess for the next *query result x_{p_t} of $\mathcal{A}$. Assume that $\mathcal{G}$ makes no more than an expected total of G mistakes (for all inputs x). In other words,*

$$\mathbf{E}_{s_\mathcal{A},s_\mathcal{G}}\left\{\sum_{t=1}^{\tilde{T}(x)}\left|\mathcal{G}(x_{p_1},\cdots,x_{p_{t-1}},s_\mathcal{A},s_\mathcal{G})-x_{p_t}\right|\right\}\leq G\quad\forall x. \tag{2.5.7}$$

Note that $\mathcal{G}$ is given the random seed $s_\mathcal{A}$ of $\mathcal{A}$, so it can predict the next query index of $\mathcal{A}$. Then $B_\epsilon(f)=O(TG/\epsilon)$, and thus (by Theorem 17) $Q(f)=O(\sqrt{TG})$.

As an example, in our simple classical example for OR we have $T = N$ (the algorithm takes at most N steps) and $G = 1$ (the guessing algorithm always guesses the next query to be 0; since the algorithm terminates on a 1, it makes at most one mistake).

Proof of theorem 24. We generalize the argument in the OR case. We take the classical algorithm and replace each classical query by the construction of Theorem 19, using $O(G/\epsilon)$ bomb queries each time. On each query, the bomb has a $O(\epsilon/G)$ chance of exploding when the guess is wrong, and no chance of exploding when the guess is correct. Therefore the total probability of explosion is $O(\epsilon/G)\cdot G = O(\epsilon)$. The total number of bomb queries used is $O(TG/\epsilon)$.

For the full technical proof, see Appendix A.2. □

2.5.3 Explicit quantum algorithm for Theorem 24

In this section we give an explicit quantum algorithm, in the setting of Theorem 24, that reproduces the given query complexity. This algorithm is very similar to the one given by R. Kothari for the oracle identification problem [55].

Theorem 25. *Under the assumptions of Theorem 24, there is an explicit quantum algorithm for f with query complexity $O(\sqrt{TG})$.*

Proof. We will construct this algorithm (Algorithm 27) shortly. We need the following quantum search algorithm as a subroutine:

Theorem 26 (Finding the first marked element in a list). *Suppose there is an ordered list of N elements, and each element is either marked or unmarked. Then there is a bounded-error quantum algorithm for finding the* **first** *marked element in the list (or determines that no marked elements exist), such that:*

- *If the first marked element is the d-th element of the list, then the algorithm uses an expected $O(\sqrt{d})$ time and queries.*
- *If there are no marked elements, then the algorithm uses $O(\sqrt{N})$ time and queries, but always determines correctly that no marked elements exist.*

This algorithm is straightforward to derive given the result in [63], and was already used in Kothari's algorithm [55]. We give the algorithm (Algorithm 97) and its analysis in Appendix A.3.

We now give our explicit quantum algorithm.

Algorithm 27 (Simulating a classical query algorithm by a quantum one).

Input. Classical randomized algorithm $\mathcal{A}$ that computes f with bounded error. Classical randomized algorithm $\mathcal{G}$ that guesses queries of $\mathcal{A}$. Oracle O_x for the hidden string x.

Output. $f(x)$ with bounded error.

The quantum algorithm proceeds by attempting to produce the list of queries and results that $\mathcal{A}$ would have made. More precisely, given a randomly chosen random seed $s_{\mathcal{A}}$, the algorithm outputs (with constant error) a list of pairs $(p_1(x), x_{p_1(x)}),\cdots,(p_{\tilde{T}(x)}(x), x_{p_{\tilde{T}(x)}(x)})$. This list is such that on random seed $s_{\mathcal{A}}$, the i-th query algorithm of $\mathcal{A}$ is made at the position $p_i(x)$, and the query result is $x_{p_i(x)}$. The quantum algorithm then determines the output of $\mathcal{A}$ using this list.

The main idea for the algorithm is this: we first assume that the guesses made by $\mathcal{G}$ are correct. By repeatedly feeding the output of $\mathcal{G}$ back into $\mathcal{A}$ and $\mathcal{G}$, we can obtain a list of query values for $\mathcal{A}$ without any queries to the actual black box. We then search for the first deviation of the string x from the predictions of $\mathcal{G}$; assuming the first deviation is the d_1-th query, by Theorem 26 the search takes $O(\sqrt{d_1})$ queries (ignoring error for now). We then know that all the guesses made by $\mathcal{G}$ are correct up to the $(d_1 - 1)$-th query, and incorrect for the d_1-th query.

With the corrected result of the first d_1 queries, we now continue by assuming again the guesses made by $\mathcal{G}$ are correct starting from the $(d_1 + 1)$-th query, and search for the location of the next deviation, d_2. This takes $O(\sqrt{d_2 - d_1})$ queries; we then know that all the guesses made by $\mathcal{G}$ are correct from the $(d_1 + 1)$-th to $(d_2 - 1)$-th query, and incorrect for the d_2-th one. Continuing in this manner, we eventually determine all query results of $\mathcal{A}$ after an expected G iterations.

We proceed to spell out our algorithm. For the time being, we assume that algorithm for Theorem 26 has no error and thus requires no error reduction.

1. Initialize random seeds $s_{\mathcal{A}}$ and $s_{\mathcal{G}}$ for $\mathcal{A}$ and $\mathcal{G}$. We will simulate the behavior of $\mathcal{A}$ and $\mathcal{G}$ on these random seeds. Initialize $d = 0$. d is such that we have determined the values of all query results of $\mathcal{A}$ up to the d-th query. Also initialize an empty list $\mathcal{L}$ of query pairs.

2. Repeat until either all query results of $\mathcal{A}$ are determined, or $100G$ iterations of this loop have been executed:

   (a) Assuming that $\mathcal{G}$ always guesses correctly starting from the $(d + 1)$-th query, compute from $\mathcal{A}$ and $\mathcal{G}$ a list of query positions $p_{d+1}, p_{d+2}, \cdots$ and corresponding results $\tilde{a}_{d+1}, \tilde{a}_{d+2}, \cdots$. This requires no queries to the black box.

   (b) Using our algorithm for finding the first marked element (Theorem 26, Algorithm 97), find the first index $d^* > d$ such that the actual query result of $\mathcal{A}$ differs from the guess by $\mathcal{G}$, i.e. $x_{p_d} \neq \tilde{a}_d$; or return that no such d^* exists. This takes $O(\sqrt{d^* - d})$ time in the former case, and $O(\sqrt{T - d})$ time in the latter.

   (c) We break into cases:

      i. If an index d^* was found in Step 2b, the algorithm decides the next mistake made by $\mathcal{G}$ is at position d^*. It thus adds the query pairs $(p_{d+1}, \tilde{a}_{d+1}), \cdots$, $(p_{d^*-1}, \tilde{a}_{d^*-1})$, and the pair $(p_{d^*}, 1 - \tilde{a}_{d^*})$, to the list $\mathcal{L}$. Also set $d = d^*$.

   ii. If no index d^* was found in Step 2b, the algorithm decides that all remaining guesses by $\mathcal{G}$ are correct. Thus the query pairs $(p_{d+1}, \tilde{a}_{d+1}), \cdots, (p_{\tilde{T}(x)}, \tilde{a}_{\tilde{T}(x)})$ are added to $\mathcal{L}$, where $\tilde{T}(x) \leq T$ is the number of queries made by $\mathcal{A}$.

3. If the algorithm found all query results of $\mathcal{A}$ in at most $100G$ iterations of step 2, use $\mathcal{L}$ to calculate the output of $\mathcal{A}$; otherwise the algorithm fails.

We now count the total number of queries. Suppose $g \leq 100G$ is the number of iterations of Step 2; if all query results have been determined, g is the number of wrong guesses by $\mathcal{G}$. Say the list of d's found is $d_0 = 0, d_1, \cdots, d_g$. Let $d_{g+1} = T$. Step 2 is executed for $g+1$ times, and the total number of queries is

$$O\left(\sum_{i=1}^{g+1} \sqrt{d_i - d_{i-1}}\right) = O\left(\sqrt{Tg}\right) = O\left(\sqrt{TG}\right) \tag{2.5.8}$$

by the Cauchy-Schwarz inequality.

We now analyze the error in our algorithm. The first source of error is cutting off the loop in Step 2: by Markov's inequality, for at least 99% of random seeds $s_{\mathcal{G}}, s_{\mathcal{G}}, \mathcal{G}$ makes no more than $100G$ wrong guesses. For these random seeds all query results of $\mathcal{A}$ are determined. Cutting off the loop thus gives at most 0.01 error.

The other source of error is the error of Algorithm 97 used in Step 2b: we had assumed that it could be treated as zero-error, but we now remove this assumption. Assuming each iteration gives error δ', the total error accrued could be up to $O(g\delta')$. It seems as if we would need to set $\delta' = O(1/G)$ for the total error to be constant, and thus gain an extra logarithmic factor in the query complexity.

However, in his paper for oracle identification [55], Kothari showed that multiple calls to Algorithm 97 can be composed to obtain a bounded-error algorithm based on span programs without an extra logarithmic factor in the query complexity; refer to [55, Section 3] for details. Therefore we can replace the iterations of Step 2 with Kothari's span program construction and get a bounded error algorithm with complexity $O(\sqrt{TG})$.

□

Note that while Algorithm 27 has query complexity $O(\sqrt{TG})$, the time complexity may be much higher. After all, Algorithm 27 proceeds by simulating $\mathcal{A}$ query-by-query, although the number of actual queries to the oracle is smaller. Whether or not we can get a algorithm faster than $\mathcal{A}$ using this approach may depend on the problem at hand.

2.6 Improved upper bounds on quantum query complexity

We now use Theorem 25 to improve the quantum query complexity of certain graph problems.

2.6.1 Single source shortest paths for unweighted graphs

Problem 28 (Single source shortest paths (SSSP) for unweighted graphs). The adjacency matrix of a directed graph n-vertex graph G is provided as a black box; a query on the pair (v, w) returns 1 if there is an edge from v to w, and 0 otherwise. We are given a fixed vertex v_{start}. Call the length of a shortest path from v_{start} to another vertex w the *distance* d_w of

w from v_{start}; if no path exists, define $d_w = \infty$. Our task is to find d_w for all vertices w in G.

In this section we shall show the following:

Theorem 29. *The quantum query complexity of single-source shortest paths in an unweighted graph is $\Theta(n^{3/2})$ in the adjacency matrix model.*

Proof. The lower bound of $\Omega(n^{3/2})$ is known [64]. We show the upper bound by applying Theorem 25 to a classical algorithm. The following well-known classical algorithm (commonly known as *breadth first search*, BFS) solves this problem:

Algorithm 30 (Classical algorithm for unweighted SSSP).

```
1. Initialize $d_w := \infty$ for all vertices $w \neq v_{start}$, $d_{v_{start}} := 0$, and $\mathcal{L} := (v_{start})$. $\mathcal{L}$ is the
   ordered list of vertices for which we have determined the distances, but whose outgoing
   edges we have not queried.

2. Repeat until $\mathcal{L}$ is empty:

   • Let $v$ be the first (in order of time added to $\mathcal{L}$) vertex in $\mathcal{L}$. For all vertices $w$
     such that $d_w = \infty$:
       – Query $(v, w)$.
       – If $(v, w)$ is an edge, set $d_w := d_v + 1$ and add $w$ to the end of $\mathcal{L}$.
   • Remove $v$ from $\mathcal{L}$.
```

We omit the proof of correctness of this algorithm (see for example [65]). This algorithm uses up to $T = O(n^2)$ queries. If the guessing algorithm always guesses that (v, w) is not an edge, then it makes at most $G = n - 1$ mistakes; hence $Q(f) = O(\sqrt{TG}) = O(n^{3/2})$.[1] □

The previous best known quantum algorithm for unweighted SSSP, to our best knowledge, was given by Furrow [40]; that algorithm has query complexity $O(n^{3/2}\sqrt{\log n})$.

We now consider the quantum query complexity of unweighted k-source shortest paths (finding k shortest-path trees rooted from k beginning vertices). If we apply Algorithm 30 on k different starting vertices, then the expected number of wrong guesses is no more than $G = k(n-1)$; however, the total number of edges we query need not exceed $T = O(n^2)$, since an edge never needs to be queried more than once. Therefore

Corollary 31. *The quantum query complexity of unweighted k-source shortest paths in the adjacency matrix model is $O(k^{1/2}n^{3/2})$, where n is the number of vertices.*

We use this idea – that T need not exceed $O(n^2)$ when dealing with graph problems – again in the following section.

[1]It seems difficult to use our method to give a corresponding result for the adjacency list model; after all, the result of a query is much harder to guess when the input alphabet is non-boolean.

2.6.2 Maximum bipartite matching

Problem 32 (Maximum bipartite matching). We are given as black box the adjacency matrix of an n-vertex bipartite graph $G = (V = X \cup Y, E)$, where the undirected set of edges E only run between the bipartite components X and Y. A *matching* of G is a list of edges of G that do not share vertices. Our task is to find a maximum matching of G, i.e. a matching that contains the largest possible number of edges.

In this section we show that

Theorem 33. *The quantum query complexity of maximum bipartite matching is $O(n^{7/4})$ in the adjacency matrix model, where n is the number of vertices.*

Proof. Once again we apply Theorem 25 to a classical algorithm. Classically, this problem is solved in $O(n^{5/2})$ time by the Hopcroft-Karp [66] algorithm (here $n = |V|$). We summarize the algorithm as follows (this summary roughly follows that of [41]):

Algorithm 34 (Hopcroft-Karp algorithm for maximum bipartite matching [66]).

1. Initialize an empty matching $\mathcal{M}$. $\mathcal{M}$ is a matching that will be updated until it is maximum.

2. Repeat the following steps until $\mathcal{M}$ is a maximum matching:

   (a) Define the *directed* graph $H = (V', E')$ as follows:

   $$\begin{aligned} V' &= X \cup Y \cup \{s,t\} \\ E' &= \{(s,x) \mid x \in X, (x,y) \notin \mathcal{M} \text{ for all } y \in Y\} \\ &\quad \cup \{(x,y) \mid x \in X, y \in Y, (x,y) \in E, (x,y) \notin \mathcal{M}\} \\ &\quad \cup \{(y,x) \mid x \in X, y \in Y, (x,y) \in E, (x,y) \in \mathcal{M}\} \\ &\quad \cup \{(y,t) \mid y \in Y, (x,y) \notin \mathcal{M} \text{ for all } x \in X\} \end{aligned} \tag{2.6.1}$$

   where s and t are extra auxilliary vertices. Note if $(s, x_1, y_1, x_2, y_2, \cdots, x_\ell, y_\ell, t)$ is a path in H from s to t, then $x_i \in X$ and $y_i \in Y$ for all i. Additionally, the edges (aside from the first and last) alternate from being in $\mathcal{M}$ and not being in $\mathcal{M}$: $(x_i, y_i) \notin \mathcal{M}$, $(y_i, x_{i+1}) \in \mathcal{M}$. Such a path is called an *augmenting path* in the literature.

   We note that a query to the adjacency matrix of E' can be simulated by a query to the adjacency matrix of E.

   (b) Using the breadth-first search algorithm (Algorithm 30), in the graph H, find the length of the shortest path, or distance, of all vertices from s. Let the distance from s to t be $2\ell + 1$.

   (c) Find a maximal set S of vertex-disjoint shortest paths from s to t in the graph H. In other words, S should be a list of paths from s to t such that each path has length $2\ell + 1$, and no pair of paths share vertices except for s and t. Moreover, all other shortest paths from s to t share at least one vertex (except for s and t) with a path in S. We describe how to find such a maximal set in Algorithm 35.

   (d) If S is empty, the matching M is a maximum matching, and we terminate. Otherwise continue:

   (e) Let $(s, x_1, y_1, x_2, y_2, \cdots, x_\ell, y_\ell, t)$ be a path in S. Remove the $\ell - 1$ edges (x_{i+1}, y_i) from $\mathcal{M}$, and insert the ℓ edges (x_i, y_i) into $\mathcal{M}$. This increases $|\mathcal{M}|$ by 1. Repeat for all paths in S; there are no conflicts since the paths in S are vertex-disjoint.

Once again, we omit the proof of correctness of this algorithm; the correctness is guaranteed by Berge's Lemma [67], which states that a matching is maximum if there are no more augmenting paths for the matching. Moreover, $O(\sqrt{n})$ iterations of Step 2 suffice [66].

We now describe how to find a maximal set of shortest-length augmenting paths in Step 2(c). This algorithm is essentially a modified version of depth-first search:

Algorithm 35 (Finding a maximal set of vertex-disjoint shortest-length augmenting paths).

Input. The directed graph H defined in Algorithm 34, as well as the distances d_v of all vertices v from s (calculated in Step 2(b) of Algorithm 34).

1. Initialize a set of paths $S := \emptyset$, set of vertices $R := \{s\}$, and a stack[2] of vertices $\mathcal{L} := (s)$. $\mathcal{L}$ contains the ordered list of vertices that we have begun, but not yet finished, processing. R is the set of vertices that we have processed. S is the set of vertex-disjoint shortest-length augmenting paths that we have found.

2. Repeat until $\mathcal{L}$ is empty:

   (a) If the vertex in the front of $\mathcal{L}$ is t, we have found a new vertex-disjoint path from s to t:

      - Trace the path from t back to s by removing elements from the front of $\mathcal{L}$ until s is at the front. Add the corresponding path to S.
      - Start again from the beginning of Step 2.

   (b) Let v be the vertex in the front of $\mathcal{L}$ (i.e. the vertex *last* added to, and still in, $\mathcal{L}$). Recall the distance from s to v is d_v.

   (c) Find w such that $w \notin R$, $d_w = d_v + 1$, and (v, w) (as an edge in H) has not been queried in this algorithm. If no such vertex w exists, remove v from $\mathcal{L}$ and start from the beginning of Step 2.

   (d) Query (v, w) on the graph H.

   (e) If (v, w) is an edge, add w to the *front* of $\mathcal{L}$. If $w \neq t$, add w to R.

3. Output S, the maximal set of vertex-disjoint shortest-length augmenting paths.

We now return to Algorithm 34 and count T and G. There is obviously no need to query the same edge more than once, so $T = O(n^2)$. If the algorithm always guesses, on a query (v, w), that there is no edge between (v, w), then it makes at most $G = O(n^{3/2})$ mistakes: in Step 2(b) there are at most $O(n)$ mistakes (see Algorithm 30), while in Step 2(c)/Algorithm 35 there is at most one queried edge leading to each vertex aside from t, and edges leading to t can be computed without queries to the adjacency matrix of H. Since Step 2 is executed $O(\sqrt{n})$ times, our counting follows.

Thus there is a quantum query algorithm with complexity $Q = O(\sqrt{TG}) = O(n^{7/4})$.

□

[2]A stack is a data structure such that elements that are first inserted into the stack are removed last.

To our knowledge, this is the first known nontrivial upper bound on the query complexity of maximum bipartite matching.[3] The time complexity of this problem was studied by Ambainis and Spalek in [41]; they gave an upper bound of $O(n^2 \log n)$ time in the adjacency matrix model. A lower bound of $\Omega(n^{3/2})$ for the query complexity of this problem was given in [68, 69].

For readers familiar with network flow, the arguments in this section also apply to Dinic's algorithm for maximum flow [70] on graphs with unit capacity, i.e. where the capacity of each edge is 0 or 1. On graphs with unit capacity, Dinic's algorithm is essentially the same as Hopcroft-Karp's, except that augmenting paths are over a general, nonbipartite flow network. (The set S in Step 2(c) of Algorithm 34 is generally referred to as a *blocking flow* in this context.) It can be shown that only $O(\min\{m^{1/2}, n^{2/3}\})$ iterations of Step 2 are required [71, 72], where m is the number of edges of the graph. Thus $T = O(n^2)$, $G = O(\min\{m^{1/2}, n^{2/3}\}n)$, and therefore

Theorem 36. *The quantum query complexity of the maximum flow problem in graphs with unit capacity is $O(\min\{n^{3/2}m^{1/4}, n^{11/6}\})$, where n and m are the number of vertices and edges in the graph, respectively.*

It is an open question whether a similar result for maximum matching in a general nonbipartite graph can be proven, perhaps by applying Theorem 25 to the classical algorithm of Micali and Vazirani [57].

2.7 Projective query complexity

We end this paper with a brief discussion on another query complexity model, which we will call the *projective query complexity*. This model is similar to the bomb query model in that the only way of accessing x_i is through a classical measurement; however, in the projective query model the algorithm does not terminate if a 1 is measured. Our motivation for considering the projective query model is that its power is intermediate between the classical and quantum query models. To the best of our knowledge, this model was first considered in 2002 in unpublished results by S. Aaronson [56].

A circuit in the projective query complexity model is a restricted quantum query circuit, with the following restrictions on the use of the quantum oracle:

1. We have an extra control register $|c\rangle$ used to control whether O_x is applied (we call the controlled version CO_x):

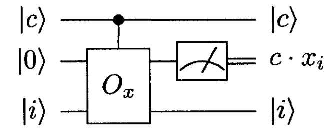
$$CO_x|c, r, i\rangle = |c, r \oplus (c \cdot x_i), i\rangle. \tag{2.7.1}$$

   where $\cdot$ indicates boolean AND.

2. The record register, $|r\rangle$ in the definition of CO_x above, *must* contain $|0\rangle$ before CO_x is applied.

3. After CO_x is applied, the record register is immediately measured in the computational basis, giving the answer $c \cdot x_i$. The result, a classical bit, can then be used to control further quantum unitaries (although only controlling the next unitary is enough, since the classical bit can be stored).

[3]The trivial upper bound is $O(n^2)$, where all pairs of vertices are queried.



(2.7.2)

We wish to evaluate a function $f(x)$ with as few calls to this *projective oracle* as possible. Let the number of oracle calls required to evaluate $f(x)$, with at most δ error, be $P_\delta(f)$. By gap amplification, the choice of δ only affects $P_\delta(f)$ by a factor of $\log(1/\delta)$, and thus we will often omit δ.

We can compare the definition in this section with the definition of the bomb query complexity in Section 2.3: the only difference is that if $c \cdot x_i = 1$, the algorithm terminates in the bomb model, while the algorithm can continue in the projective model. Therefore the following is evident:

Observation 37. *$P_\delta(f) \le B_{\epsilon,\delta}(f)$, and therefore $P(f) = O(Q(f)^2)$.*

Moreover, it is clear that the projective query model has power intermediate between classical and quantum (a controlled query in the usual quantum query model can be simulated by appending a 0 to the input string), and therefore letting $R_\delta(f)$ be the classical randomized query complexity,

Observation 38. *$Q_\delta(f) \le P_\delta(f) \le R_\delta(f)$.*

For explicit bounds on P, Regev and Schiff [43] have shown that for computing the OR function, the projective query complexity loses the Grover speedup:

Theorem 39 ([43])**.** *$P(OR) = \Omega(N)$.*

Note that this result says nothing about $P(AND)$, since the definition of $P(f)$ is asymmetric with respect to 0 and 1 in the input.[4]

We observe that there could be a separation in both parts of the inequality $Q \le P \le B$:

$$Q(OR) = \Theta(\sqrt{N}), \quad P(OR) = \Theta(N), \quad B(OR) = \Theta(N) \tag{2.7.3}$$

$$Q(PARITY) = \Theta(N), \quad P(PARITY) = \Theta(N), \quad B(PARITY) = \Theta(N^2) \tag{2.7.4}$$

In the latter equation we used the fact that $Q(PARITY) = \Theta(N)$ [32]. It therefore seems difficult to adapt our lower bound method in Section 2.4.2 to $P(f)$.

It would be interesting to find a general lower bound for $P(f)$, or to establish more clearly the relationship between $Q(f)$, $P(f)$, and $R(f)$.

[4]We could have defined a symmetric version of P, say $\tilde{P}$, by allowing an extra guess on the measurement result, similar to our construction of $\tilde{B}$ in Section 2.5.1. Unfortunately, Regev and Schiff's result, Theorem 39, do not apply to this case, and we see no obvious equivalence between P and $\tilde{P}$.

Chapter 3

Normalizer circuits over infinite-dimensional systems: an introduction

In this chapter we will study the model of *normalizer circuits* over infinite-dimensional systems. Normalizer circuits were defined in [1, 2] as a generalization of Clifford circuits to Hilbert spaces associated with (explicitly decomposed) finite Abelian groups, and it was shown that such circuits could be simulated efficiently with a classical computer. The classical simulation comes despite normalizer circuits allowing for quantum Fourier transforms, a crucial element of Shor's algorithm. We will generalize normalizer circuits to infinite-dimensional systems associated with infinite Abelian groups, and show the following:

- If the underlying Abelian group is explicitly decomposed into a product of primitive subgroups, then normalizer circuits over this group are efficiently simulable by a classical computer. This generalizes the Gottesman-Knill theorem to infinite dimensions.
- If the underlying Abelian group is not explicitly decomposed, however, normalizer circuits can implement many important quantum algorithms, including Shor's algorithm for factoring (and many algorithms for hidden subgroup problems). Thus these circuits have much more computational power than the first case.

The only difference between the two cases is whether or not the underlying Abelian group is explicitly decomposed; it appears therefore that the ppwer of Shor's algorithm comes from the classical difficulty of decomposing Abelian groups (which is easy for a quantum computer; see Section 5.2.5, or [73]). We will look at the first assertion in Chapter 4, and the second assertion in Chapter 5; in this chapter we will introduce and give basic definitions for normalizer circuits.

The results of Chapters 3-5 are joint work with Juan Bermejo-Vega and Martin Van den Nest. This chapter is mostly excerpted from [74, 75].

3.1 Introduction

Normalizer circuits [1, 2] are a family of quantum circuits that generalize the so-called Clifford circuits [17, 16, 76, 77] to Hilbert spaces associated with finite Abelian groups. A normalizer circuit is a quantum circuit on a Hilbert space indexed by the elements of a finite

Abelian group $G = \mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_n}$, i.e. the Hilbert space is spanned by states of the form $|g\rangle$, $g \in G$. Moreover, the gates in a normalizer circuit can only be of the following three types of gates, which we refer to as *normalizer gates*:

1. Quantum Fourier transforms (QFTs), over the whole group G or subgroups of G.
2. Automorphism gates, gates that compute automorphisms over G.
3. Quadratic phase gates, gates that compute quadratic functions of G.

These three types of gates can be viewed as a generalization of the Clifford gates: the quantum Fourier transform generalizes the Hadamard gate, automorphism gates generalize the CNOT gate, and quadratic phase gates generalize the $\sqrt{Z}$ gate. [1, 2] show that a generalization of the Gottesman-Knill theorem holds for normalizer circuits: every normalizer circuit over finite Abelian groups can be efficiently classically simulated.

Infinite-dimensional normalizer circuits

In this chapter we further generalize the normalizer circuit framework, by introducing normalizer circuits where the associated Abelian group G is infinite. We focus on groups of the form $G = F \times \mathbb{Z}^a$, where F is a finite Abelian group, and where $\mathbb{Z}$ denotes the *infinite* additive group of integers. The motivation for adding $\mathbb{Z}$ is that several number theoretical problems are naturally connected to problems over the integers. For example, it is well known that the factoring problem is related to the hidden subgroup problem over $\mathbb{Z}$ [78, 79, 80, 81]. Normalizer circuits over an infinite group G are composed of normalizer gates, gates of the three types listed above.

Our main motivation for allowing the integer group $\mathbb{Z}$ in the groups we study is to make a connection with the role of $\mathbb{Z}$ in many quantum algorithms for solving number theoretic problems. For example, Shor's factoring algorithm [14] can be seen as a hidden subgroup problem over $\mathbb{Z}$ (see [21, 24] or Section 5.2 for other examples). Such quantum algorithms typically make use of the quantum Fourier transform, and thus QFTs are often seen as the source of quantumness in these algorithms.

In Chapter 4, we show that if the finite portion of the Abelian group G is explicitly decomposed, i.e. the decomposition $G = \mathbb{Z}^a \times \mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_n}$ is given, then normalizer circuits over G can be efficiently simulated by a classical computer (Theorem 59). This result generalizes the finite Abelian case of [1, 2]. Our result casts some doubt on the view that QFTs are the source of quantum speedups, since normalizer ciruits are a large class of quantum circuits containing QFTs that can still be classically simulated. Our result also gives a natural framework to study continuous-variable error correcting codes for superconducting qubit systems; see the discussion in Section 4.1.

Black box normalizer circuits

We will also consider (mainly in Chapter 5) black box normalizer circuits over Abelian groups $G = \mathbb{Z}^a \times F$ that are *not* given to us in a factorized form $G = \mathbb{Z}^a \times \mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_n}$; we refer to such groups G as *black box groups*. Even though such a factorized form always exists, finiding such a decomposition is as hard as factoring [82, 83]. This seemingly trivial difference in our setting turns out to be tremendous: many quantum algorithms for Abelian hidden subgroup problems turn out to be black box normalizer circuits, and thus there is a large difference in computational power between the two cases.

In Chapter 5 we will study some well-known quantum algorithms that can be implemented as black box normalizer circuits. In particular, the Cheung-Mosca algorithm for decomposing a finite Abelian group [84, 73] can be solved with black box normalizer circuits. We show that this problem of decomposing an Abelian group is *complete* for the complexity class associated with black box normalizer circuits (Theorem 87): given an oracle to compute the decomposition of a group, any black box normalizer circuit can be efficiently classicaly simulated. An equivalent view is the following *no-go theorem* for the design of quantum algorithms: no further quantum superpolynomial speedups can be found within our normalizer circuit framework that are not already achieved by the group decomposition algorithm (Theorem 89). Section 5.1 lists a more complete summary results.

Contents of this chapter

This chapter will merely introduce and define the normalizer circuit framework; our main results are given in Chapters 4 and 5. We now discuss the contributiosn of this chapter.

In extending normalizer circuits to infinite groups G, several issues arise that are not present in the finite group setting. First of all, the quantum Fourier transform (QFT) can no longer be considered as a usual quantum gate, and instead must be interpreted as a change of basis. Roughly speaking, the QFT over a group G is a transformation which relates two bases of the Hilbert space: namely, the standard basis $\{|g\rangle\}$ and the *Fourier basis.* The Fourier basis vectors are related to the character group of G. If G were finite Abelian, it would be isomorphic to its own character group. This feature is however no longer true for infinite groups such as $\mathbb{Z}$: the character group of $\mathbb{Z}$ is (isomorphic to) a different group, the circle group $\mathbb{T} = [0, 1)$ with addition modulo 1. This group represents the addition of angles in a circle, up to a constant rescaling of 2π. We must therefore view the QFT as a change of basis, taking a state expressed in the $\mathbb{Z}$ basis to the $\mathbb{T}$ basis, and vice versa. As we will see below, this phenomenon has important consequences for the treatment of normalizer gates over G. In particular, in order to construct a closed normalizer formalism, we must consider groups of the more general form $\mathbb{Z}^a \times \mathbb{T}^b \times F$ and their associated normalizer circuits. Note that $\mathbb{T}$ is a continuous group, whereas $\mathbb{Z}$ is discrete (finitely generated).

Since G is infinite, it is also not immediately clear how quadratic functions and homomorphisms over G can be concisely specified. An important technical ingredient in our simulation is a proof that both quadratic functions and homomorphisms on G have certain concise ***normal forms***. This is a purely group-theoretic result that, aside from its importance in proving our simulation result in Chapter 4, may find interesting applications elsewhere in quantum information. For instance, our normal form can be applied to describe the relative phases of *stabilizer states*, since it was shown in [2] that such phases are quadratic[1]; as a result, one may use it to generalize part of Gross's discrete Hudson theorem [85] to our setting[2].

[1]The result in [2] is for finite dimensional systems. However, adopting the definitions in Section 4.4, it is easy to check that the proof extends step-by-step to the infinite dimensional case.

[2]Gross's theorem provides a normal form for odd-dimensional-qudit stabilizer states in terms of quadratic functions. In addition, it states that a pure state is an odd-dim qudit stabilizer state iff it has non-negative Wigner function [85]. The second statement cannot hold in our setup, due to the presence of non-local effects (cf. next section).

3.2 Outline of this chapter

Section 3.3 contains a non-technical **summary of concepts** and several examples of normalizer gates. We then review the Abelian groups we will use in Section 3.4, define Hilbert spaces associated with Abelian groups in Section 3.5, and finally introduce normalizer circuits in full detail in Sections 3.6, 3.7.

In Section 3.8 we survey some necessary notions of group and character theory. Finally, in Section 3.9 we develop a theory of matrix representations of group homomorphisms, and in Section 3.10 we develop normal forms for quadratic functions.

3.3 Summary of concepts

In this section we give a rough intuitive definition of our circuit model and provide examples of normalizer gates to illustrate their operational meaning. Our model is presented in full detail in Section 3.6, after reviewing some necessary notions of group-theory in section 3.5. The readers interested in understanding the proofs of our main results should consult these sections.

3.3.1 The setting

Normalizer gates are quantum gates that act on a Hilbert space $\mathcal{H}_G$ which has an orthonormal standard basis $\{|g\rangle\}_{g\in G}$ labeled by the elements of an Abelian group G. The latter can be finite or infinite, but it must have a well-defined integration (or summation) rule[3] so that the group has a well defined Fourier transform. We define a *normalizer circuits over G* to be any quantum circuits built of the following *normalizer gates*:

1. **Quantum Fourier transforms** implement the (classical) Fourier transform of the group $\psi(x) \rightarrow \hat{\psi}(p)$ as a quantum operation $\int \psi(x)|x\rangle \rightarrow \int \hat{\psi}(p)|p\rangle$. Here, ψ is a complex function acting on the group and $\hat{\psi}$ is its Fourier transform.

2. **Group automorphism gates** implement group automorphisms $\alpha : G \rightarrow G$ as a quantum gate: $|g\rangle \rightarrow |\alpha(g)\rangle$. Here, g and $\alpha(g)$ denote elements of G.

3. **Quadratic phase gates** are diagonal gates that multiply standard basis states with *quadratic* phases: $|g\rangle \rightarrow \xi(g)|g\rangle$. This means that $g \rightarrow \xi(g)$ is a quadratic ("almost multiplicative") function with the property $\xi(g+h) = \xi(g)\xi(h)B(g,h)$, where $B(g,h)$ is a bi-multiplicative correcting term.

3.3.2 Examples

In order to illustrate these definitions, we give examples of normalizer gates for finite groups and infinite groups of the form $\mathbb{Z}^a$ (integer lattices) and $\mathbb{T}^b$ (hypertori).

Normalizer circuits over finite groups

First we give a few examples of normalizer circuits over finite Abelian groups. (We also refer the reader to [1, 2], where these circuits have been extensively studied.)

[3]All groups we study have a well-defined Haar measure.

First consider $G = \mathbb{Z}_d^m$. The Hilbert space in this case is $\mathcal{H} = \mathcal{H}_d^m$, corresponding to a system of m qudits. The following generalizations of the CNOT and Phase gates are automorphism and quadratic phase gates, respectively [1]:

$$\mathrm{SUM}_{d,a} = \sum_{x,y\in\mathbb{Z}_d} |x, x+ay\rangle\langle x,y|, \qquad S_d = \sum_{x\in\mathbb{Z}_d} \exp\left(\tfrac{\pi \mathrm{i}}{d} x(x+d)\right)|x\rangle\langle x|$$

where $a \in \mathbb{Z}_d$ is arbitrary. The quantum Fourier transform can also be expressed similarly as a gate:

$$F_d = \frac{1}{\sqrt{d}} \sum_{x,y\in\mathbb{Z}_d} \exp\left(\tfrac{2\pi \mathrm{i} xy}{d}\right)|x\rangle\langle y|$$

For the qubit case, $d = 2$, we recover the definitions of CNOT, phase gate $S = \mathrm{diag}(1, \mathrm{i})$ and Hadamard gate, and our main result (Theorem 59) yields the (non-adaptive) Gottesman-Knill theorem [17, 16].

Another normalizer gate for qudits is the multiplication gate $M_{d,a} = \sum_{x\in\mathbb{Z}_d} |ax\rangle\langle x|$, where a is coprime to d, which is an automorphism gate. This single-qudit gate acts non-trivially only for $d \geq 3$.

As an interesting subcase, we can choose the group to be of the form $G = \mathbb{Z}_{2^n}$ with exponentially large d. Normalizer gates in this example are $M_{2^n,a}$, S_{2^n}, and $\mathcal{F}_{2^n}$, as defined above. The quantum Fourier transform in this case, $\mathcal{F}_{2^n}$, corresponds to the standard discrete QFT used in Shor's algorithm for factoring [14].

The infinite case $G = \mathbb{Z}^m$

Let us now move to an infinite case, choosing the group $G = \mathbb{Z}^m$ to be an integer lattice. Examples of automorphism gates and quadratic phase gates are, respectively,

$$\mathrm{SUM}_{\mathbb{Z},a} = \sum_{x,y\in\mathbb{Z}} |x, x+ay\rangle\langle x,y|, \qquad S_p = \sum_{x\in\mathbb{Z}} \exp\left(\pi \mathrm{i} p x^2\right)|x\rangle\langle x|$$

where a is an arbitrary integer and p is an arbitrary real number. The fact that these gates are indeed normalizer gates follows from general normal forms for matrix representations group homomorphisms (Theorem 51) and quadratic functions (Theorem 57) that we later develop.

The case of quantum Fourier transforms is more involved in this case, since the QFT over $\mathbb{Z}$ can no longer be regarded as a quantum logic gate, i.e., a unitary rotation [3]. This happens because the QFT now performs a non-canonical change of the standard integer basis $\{z\}_{z\in\mathbb{Z}}$ of the space, into a new basis $\{t\}_{t\in\mathbb{T}}$ labeled by the elements of the circle group $\mathbb{T} = [0, 1)$. The latter property is due to the fact that the QFT over $\mathbb{Z}$ is nothing but the quantum version of the discrete-time Fourier transform [22] (the inverse of the well-known Fourier series) which sends functions over $\mathbb{Z}$ to *periodic functions over the reals*. In this chapter, we will understand the QFT over $\mathbb{Z}$ as change of basis between two orthonormal bases of the Hilbert space $\mathcal{H}_{\mathbb{Z}}$. We discuss these technicalities in detail in Section 3.6.

The QFT over $\mathbb{Z}$ acts on quantum states as in the following examples (cf. Section 3.6):

State before QFT over $\mathbb{Z}$	*State after QFT over $\mathbb{Z}$*
$\lvert x\rangle$	$\int_{\mathbb{T}} dp\, \mathrm{e}^{2\pi \mathrm{i}\, px} \lvert p\rangle$
$\sum_{x\in\mathbb{Z}} \mathrm{e}^{-2\pi \mathrm{i}\, px} \lvert x\rangle$	$\lvert p\rangle$
$\sum_{x\in\mathbb{Z}} \lvert rx\rangle$	$\frac{1}{r} \sum_{\substack{k\in\mathbb{Z}:\\ k/r\in\mathbb{T}}} \lvert k/r\rangle$

These transformations can be found in standard signal processing textbooks [22].

The infinite case: $G = \mathbb{T}^m$

Finally, we choose the group $G = \mathbb{T}^m$ be an m-dimensional torus. Two examples of automorphism gates are the sum and sign-flip gates:

$$\mathrm{SUM}_{\mathbb{T},b} = \sum_{p,q\in\mathbb{T}} \lvert p, q + bp\rangle\langle p, q\rvert, \qquad M_{\mathbb{T},s} = \sum_{p\in\mathbb{T}} \lvert sp\rangle\langle p\rvert$$

where b is an arbitrary integer and $s = \pm 1$ (again, these formulas come from Theorem 51).

Unlike the previous examples we have considered, any quadratic phase gate over G is *purely multiplicative* (i.e., the bi-multiplicative function $B(g,h)$ is always trivial[4]). In the case $m = 1$, this is equivalent to saying that any such gate is of the form

$$\sum_{p\in\mathbb{T}} \exp\left(2\pi i b p\right) \lvert p\rangle\langle p\rvert$$

with b an arbitrary integer.

Lastly, we take a look at the effect of the quantum Fourier transform over $\mathbb{T}$ in some examples. Similarly to the QFT over $\mathbb{Z}$, this gate performs a non-canonical change of the standard basis (now $\{t\}_{t\in\mathbb{T}}$ changes to $\{z\}_{z\in\mathbb{Z}}$) and should be understood as a change of basis that does not correspond to a gate. The QFT over $\mathbb{T}$ is the quantum version of the Fourier series [22].

State before QFT over $\mathbb{T}$	*State after QFT over $\mathbb{T}$*
$\lvert p\rangle$	$\sum_{x\in\mathbb{Z}} \mathrm{e}^{2\pi \mathrm{i}\, px} \lvert x\rangle$
$\int_{\mathbb{T}} dp\, \mathrm{e}^{2\pi \mathrm{i}\, px} \lvert p\rangle$	$\lvert -x\rangle$
$\frac{1}{r} \sum_{\substack{k\in\mathbb{Z}:\\ k/r\in\mathbb{T}}} \lvert k/r\rangle$	$\sum_{x\in\mathbb{Z}} \lvert rx\rangle = \sum_{x\in\mathbb{Z}} \lvert -rx\rangle$

Comparing the effect of the QFT on $\mathcal{H}_{\mathbb{Z}}$ and the QFT on $\mathcal{H}_{\mathbb{T}}$, we see that the former is the "inverse" of the latter up to a change of sign of the group elements labeling the basis; concatenating the two of them yields the transformation $\lvert x\rangle$ to $\lvert -x\rangle$. This is a general

[4]This fact can be understood in the light of a later result, Theorem 57 and it is related to nonexistence of nontrivial group homomorphisms from $\mathbb{T}^m$ to $\mathbb{Z}^m$, the latter being the character group of $\mathbb{T}^m$ up to isomorphism.

phenomenon, which we shall observe again in the proof of some results (namely, Theorem 61).

3.4 Abelian groups

The most general groups we will consider in this thesis are Abelian groups of the form

$$G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbf{B}, \tag{3.4.1}$$

where a, b, $N_1, \cdots, N_c$ are arbitary integers and $\mathbf{B}$ is a finite Abelian *black box group*, which we will define in this section. (Sometimes for convenience we will also include the additive group of real numbers, $\mathbb{R}$, in the set of allowable groups.)

We will discuss each of the constitutent groups in turn.

3.4.1 $\mathbb{Z}$: the group of integers

$\mathbb{Z}$ simply refers to the group of integers under addition; it is infinite, but finitely generated (by the element 1).

3.4.2 $\mathbb{T}$: the circle group

$\mathbb{T}$, the *circle group*, refers to the group of real numbers in the interval $[0, 1)$ under addition modulo 1.[5] Unlike all the other components we will consider, it is both infinite and not finitely generated. The introduction of $\mathbb{T}$ is necessary to allow the use of quantum Fourier transforms over $\mathbb{Z}$, as we see in the next section.

3.4.3 Finite Abelian groups

Let us start by stating a very important theorem we will use for our results:

Theorem 40 (Fundamental Theorem of Finite Abelian Groups [87]**).** *Any finite Abelian group F has a decomposition into a direct product of cyclic groups, i.e.*

$$F \cong \mathbb{Z}_{d_1} \times \mathbb{Z}_{d_2} \times \cdots \times \mathbb{Z}_{d_k} \tag{3.4.2}$$

for some positive integers $d_1, \cdots, d_k$.

Here $\mathbb{Z}_d$ is the group of integers modulo d under addition.

Actually finding such a decomposition for a group F may be difficult in practice. For example, consider the set of integers modulo N that are also relatively prime to N; this set forms a group under multiplication. (This group is known as the *multiplicative group of integers modulo N*, or $\mathbb{Z}_N^\times$.) It is not known classically how to decompose $\mathbb{Z}_N^\times$ into its cyclic subgroups. For example, if $N = pq$ for p, q prime then $\mathbb{Z}_{pq}^\times \cong \mathbb{Z}_{p-1} \times \mathbb{Z}_{q-1}$, and hence decomposing $\mathbb{Z}_{pq}^\times$ is at least as hard as factoring pq or, equivalently, breaking RSA [15]. More generally, decomposing $\mathbb{Z}_N^\times$ is known to be polynomial time equivalent to factoring [88]. In the quantum case, however, Cheung and Mosca gave an algorithm [84, 73] to decompose any finite Abelian group.

[5]Since the group $\mathbb{T}^n$ is nothing but an n-dimensional torus, $\mathbb{T}$ is sometimes referred as "the torus group" [86]. $\mathbb{T}$ should not be confused with $\mathbb{T}^2$ the usual (two-dimensional) torus.

In equation (3.4.1), the factors $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$ represent an arbitrary finite Abelian group for which *the group decomposition is known.* The case where the decomposition is unknown will be covered by the black box group $\mathbf{B}$.

3.4.4 Black box groups

In this thesis, we define a *black-box group* $\mathbf{B}$ [83] to be a finite group whose elements are uniquely encoded by binary strings of a certain size n, which is the length of the encoding. The elements of the black-box group can be multiplied and inverted at unit cost by querying a black-box, or *group oracle*, which computes these operations for us. The order of a black-box group with encoding length n is bounded above by 2^n: the precise order $|\mathbf{B}|$ may not be given to us, but it is assumed that the group oracle can identify which strings in the group encoding correspond to elements of the group. When we say that a particular black-box group (or subgroup) is given (as the input to some algorithm), it is meant that a *list of generators* of the group or subgroup is explicitly provided.

From now on, all black-box groups in this thesis will be assumed to be *Abelian.* Although we only consider finite Abelian black-box groups, we stress now, that there is a subtle but crucial difference between these groups and the explicitly decomposed finite Abelian groups in [1, 2]: although, mathematically, all Abelian black-box groups have a decomposition (3.4.2), it is *computationally hard* to find one and we assume no knowledge of it. In fact, our motivation to introduce black-box groups in our setting is precisely to model those Abelian groups that cannot be efficiently decomposed with known classical algorithms that have, nevertheless, efficiently classically computable group operations. With some abuse of notation, we shall call all such groups also "black-box groups", even if no oracle is needed to define them; in such cases, oracle calls will be replaced by poly(n)-size classical circuits for computing group multiplications and inversions.

As an example, let us consider again the group $\mathbb{Z}_N^\times$. This group can be naturally modeled as a black-box group in the above sense: on one hand, for any $x, y \in \mathbb{Z}_N^\times$, xy and x^{-1} can be efficiently computed using Euclid's algorithm [89]; on the other hand, decomposing $\mathbb{Z}_N^\times$ is as hard as factoring [88]. Note, in addition, that a generating set of $\mathbb{Z}_N^\times$ can found by taking a linear number of samples[6] of $\mathbb{Z}_N^\times$.

3.5 The Hilbert space of a group

In this section we introduce Hilbert spaces associated with Abelian groups of the form

$$G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbf{B} \tag{3.5.1}$$

where $\mathbb{Z}_N$ is the additive group of integers modulo N and $\mathbb{Z}$ is the additive group of integers.

3.5.1 Finite Abelian groups

First we consider $\mathbb{Z}_N$. With this group, we associate an N-dimensional Hilbert space $\mathcal{H}_N$ having a basis $\{|x\rangle : x \in \mathbb{Z}_N\}$, henceforth called the standard basis of $\mathcal{H}_N$. A state in $\mathcal{H}_N$

[6]Sampling $\mathbb{Z}_N^\times$ can be done by sampling $\{0, \cdots, N-1\}$ uniformly and then rejecting samples that are not relatively prime to N; this takes $O(\log\log N)$ trials to succeed with high probability. A similar approach works, in general, for sampling generating-sets of uniquely-encoded finite Abelian groups [90].

is (as usual) a unit vector $|\psi\rangle = \sum \psi(x)|x\rangle$ with $\sum |\psi(x)|^2 = 1$, where $\psi(x) \in \mathbb{C}$ and where the sum is over all $x \in \mathbb{Z}_N$.

For a black box group $\mathbf{B}$, we associate a $|\mathbf{B}|$-dimensional Hilbert space $\mathcal{H}_{\mathbf{B}}$ with standard basis states $|b\rangle, b \in \mathbb{B}$.

3.5.2 The integers $\mathbb{Z}$

An analogous construction is made for the group $\mathbb{Z}$, although an important distinction with the former case is that $\mathbb{Z}$ is infinite. We consider the infinite-dimensional Hilbert space $\mathcal{H}_{\mathbb{Z}} = \ell_2(\mathbb{Z})$ with standard basis states $|z\rangle$ where $z \in \mathbb{Z}$. A state in $\mathcal{H}_{\mathbb{Z}}$ has the form

$$|\psi\rangle = \sum_{x\in\mathbb{Z}} \psi(x)|x\rangle \tag{3.5.2}$$

with $\sum |\psi(x)|^2 = 1$, where the (infinite) sum is over all $x \in \mathbb{Z}$. More generally, any normalizable sequence $\{\psi_x : x \in \mathbb{Z}\}$ with $\sum |\psi_x|^2 < \infty$ can be associated with a quantum state in $\mathcal{H}_{\mathbb{Z}}$ via normalization. Sequences whose sums are not finite can give rise to *non-normalizable* states. These states do not belong to $\mathcal{H}_{\mathbb{Z}}$ and are hence unphysical; however it is often convenient to consider such states nonetheless. Some examples are the *plane-wave states*,

$$|p\rangle := \sum_{z\in\mathbb{Z}} \overline{e^{2\pi i z p}}|z\rangle \quad p \in [0,1). \tag{3.5.3}$$

Even though the $|p\rangle$ themselves do not belong to $\mathcal{H}_{\mathbb{Z}}$, every state (3.5.2) in this Hilbert space can be re-expressed as

$$|\psi\rangle = \int_{\mathbb{T}} \mathrm{d}p\ \phi(p)|p\rangle \tag{3.5.4}$$

for some complex function $\phi : \mathbb{T} \to \mathbb{C}$, where $\mathrm{d}p$ denotes the Haar measure on $\mathbb{T}$, the circle group (real numbers with addition modulo 1). Thus the states $|p\rangle$ form an alternate basis[7] of $\mathcal{H}_{\mathbb{Z}}$ which is both infinite and continuous; we call this the Fourier basis. The Fourier basis is orthonormal in the sense that

$$\langle p|p'\rangle = \delta(p - p'), \tag{3.5.5}$$

where $\delta(\cdot)$ is the Dirac delta function.

In the following, when dealing with the Hilbert space $\mathcal{H}_{\mathbb{Z}}$, we will use both the standard and Fourier basis. This is different from the finite space $\mathcal{H}_N$ where we only use the standard basis. The motivation for this is the following: note that, similarly to $\mathcal{H}_{\mathbb{Z}}$, the space $\mathcal{H}_N$ has a Fourier basis, given by the vectors

$$|\tilde{y}\rangle := \sum_{x\in\mathbb{Z}_N} \overline{e^{2\pi i \frac{xy}{N}}}|x\rangle \quad \text{for every } y \in \mathbb{Z}_N. \tag{3.5.6}$$

Since both the Fourier basis and the standard basis have equal cardinality, there exists a unitary operation mapping $|y\rangle \to |\tilde{y}\rangle$. This operation is precisely the quantum Fourier transform over $\mathbb{Z}_N$ (see Section 3.6). Thus, instead of working with both the standard

[7]Although we use this terminology, the $|p\rangle$ do not form a "basis" in the usual sense since these states are unnormalizable and lie outside the Hilbert space $\mathcal{H}_{\mathbb{Z}}$. Rigorously, the $|p\rangle$ kets should be understood as Dirac-delta measures, or as Schwartz-Bruhat tempered distributions [91, 92]. The theory of rigged Hilbert spaces [93, 94, 95, 96] (often used to study observables with continuous spectrum) establishes that the $|p\rangle$ kets can be used as a basis for all practical purposes that concern us.

and Fourier basis, we may equivalently use the standard basis alone plus the possibility of applying this unitary rotation (which is the standard approach). In contrast, the standard and Fourier basis in the infinite-dimensional space $\mathcal{H}_{\mathbb{Z}}$ have different cardinality, since the former is indexed by the discrete group $\mathbb{Z}$ and the latter is indexed by the continuous group $\mathbb{T}$. Thus there does not exist a unitary operation which maps one basis to the other. For this reason, we will work with both bases.

3.5.3 Total Hilbert space

In general we will consider consider groups of the form (3.5.1). The associated Hilbert space has the tensor product form

$$\mathcal{H} = \mathcal{H}_{\mathbb{Z}}^{\otimes a} \otimes \mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_c} \otimes \mathcal{H}_{\mathbf{B}}. \tag{3.5.7}$$

In this thesis, we treat $\mathcal{H}$ as the underlying Hilbert space of a quantum computation with $m := a + c + 1$ computational registers. The first a registers $\mathcal{H}_{\mathbb{Z}}$ are infinite dimensional. The latter $c + 1$ registers are finite dimensional, and the last one, $\mathcal{H}_{\mathbf{B}}$, is associated with some Abelian black box group $\mathbf{B}$.

For each finite-dimensional space $\mathcal{H}_{N_i}$ (and $\mathcal{H}_{\mathbf{B}}$) we consider its standard basis as above. For each $\mathcal{H}_{\mathbb{Z}}$ we may consider either its standard basis or its Fourier basis. The total Hilbert space has thus a total 2^a possible choices that are labeled by different groups: we call these bases the ***group-element bases*** of $\mathcal{H}_G$. For example, choosing the standard basis everywhere yields the basis states

$$|x(1)\rangle \otimes \cdots \otimes |x(a)\rangle \otimes |y(1)\rangle \otimes \cdots \otimes |y(c)\rangle \otimes |b\rangle \quad x(i) \in \mathbb{Z};\ y(j) \in \mathbb{Z}_{N_j};\ b \in \mathbf{B}, \tag{3.5.8}$$

which is labeled by the elements of the group $\mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$. By choosing the Fourier basis in the a-th space we obtain in turn

$$|x(1)\rangle\otimes\cdots\otimes|x(a-1)\rangle\otimes|p\rangle\otimes|y(1)\rangle\otimes\cdots\otimes|y(c-1)\rangle\otimes|b\rangle \quad x(i) \in \mathbb{Z};\ y(j) \in \mathbb{Z}_{N_j};\ p \in \mathbb{T};\ b \in \mathbf{B}, \tag{3.5.9}$$

which is labeled by the elements of $\mathbb{Z}^{a-1} \times \mathbb{T} \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_a}$.

More generally, each of the 2^b tensor product bases of (3.5.7) is constructed as follows. Consider any Abelian group of the form

$$G' = G_1 \times \cdots \times G_a \times \mathbb{Z}_{N_1} \otimes \ldots \mathbb{Z}_{N_c} \times \mathbf{B} \quad G_i \in \{\mathbb{Z}, \mathbb{T}\}. \tag{3.5.10}$$

The associated ***group-element basis*** of $\mathcal{B}_{G'}$ of $\mathcal{H}_G$ is

$$\mathcal{B}_{G'} := \{|g\rangle := |g(1)\rangle \otimes \ldots |g(a+c+1)\rangle;\quad g = (g(1), \ldots, g(a+c+1)) \in G'\}. \tag{3.5.11}$$

The notation $G_i = \mathbb{T}$ indicates that $|g(i)\rangle$ is a Fourier state of $\mathbb{Z}$ (3.5.3). The states $|g\rangle$ are product-states with respect to the tensor-product decomposition of (3.5.7). There are 2^a possible choices of groups in (3.5.10) (all of them related via Pontryagin duality[8]) and 2^a

[8]From a mathematical point of view, all groups (3.5.10) form a family (in fact, a category) which is generated by replacing the factors G_i of the group $\mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_b} \times \mathbf{B}$ with their character groups G_i^*, and identifying isomorphic groups. Pontryagin duality [97, 98, 86, 99, 100, 101, 102] then tells us that there are 2^b different groups and bases. Note that this multiplicity is a purely *infinite-dimensional feature*, since all finite groups are isomorphic to their character groups; consequently, this feature does not play a role in the study of finite-dimensional normalizer circuits (or Clifford circuits) [1, 2].

inequivalent group-element basis of the Hilbert space.

Finally, we note that, relative to any basis $\mathcal{B}_G$, every quantum state (normalizable or not) can be expressed as

$$|\psi\rangle = \int_X \mathrm{d}g\, \psi(g)|g\rangle, \tag{3.5.12}$$

where the $\psi(g)$ are complex coefficients and where X is a subset of X (if X is a discrete set, the integral should be replaced by a sum over all elements in X).

3.6 Normalizer circuits (without black boxes)

In this section we generalize the normalizer circuits [1, 2] to general infinite-dimensional Hilbert spaces of the form

$$\mathcal{H} = \mathcal{H}_{\mathbb{Z}}^{\otimes a} \otimes \mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_c}. \tag{3.6.1}$$

Normalizer circuits over black box groups will be considered in the next section.

In previous works [1, 2], a finite Abelian group G determines both the standard basis of $\mathcal{H}_G$ and the allowed types of gates, called normalizer gates. A technical complication that arises when one tries to define normalizer gates over infinite dimensional groups of the form $\mathbb{Z}$ and $\mathbb{T}$ is that there are several group-element basis that can be associated with the Hilbert space $\mathcal{H}_G$ (see previous section). As a result, there is not a natural choice of standard basis in a normalizer circuit over an infinite Abelian group. To deal with this aspect of the Hilbert space, we allow the "standard" basis of the computation to be *time-dependent*: at every time step t in a normalizer circuit there is a designated "standard" basis $\mathcal{B}_{G_t}$, which must be a group-element basis, and is subject to change along the computation.

Designated basis $\mathcal{B}_G$. Let G' be a member of the family of 2^b groups defined in (3.5.10)—note that G thus generally contains factors $\mathbb{Z}_{N_i}$, $\mathbb{Z}$ and $\mathbb{T}$. We consider the corresponding group-element basis $\mathcal{B}_{G'} = \{|g\rangle : g \in G'\}$ as defined in (3.5.11). When we set $\mathcal{B}_{G'}$ to be the *designated basis* of the computation, it is meant that the group G that labels $\mathcal{B}_{G'}$ will determine the allowed normalizer gates (read below), and that $\mathcal{B}_{G'}$ will be the basis in which measurements are performed.

3.6.1 Normalizer gates

We now define the allowed gates of the computation, called *normalizer gates.* These can be of three types, namely automorphism gates, quadratic phase gates and quantum Fourier transforms. To define these gates, we assume that the designated basis of the computation is $\mathcal{B}_G$ for some G of the form (3.5.10).

Automorphism gates. Consider a continuous group automorphism $\alpha : G \to G$, i.e. α is an invertible map satisfying $\alpha(g+h) = \alpha(g) + \alpha(h)$ for every $g, h \in G$. Define a unitary gate U_α by its action on the basis $\mathcal{B}_G$, as follows: $U_\alpha : |h\rangle \to |\alpha(h)\rangle$. Any such gate is called an automorphism gate. Note that U_α acts as a permutation on the basis $\mathcal{B}_G$ and is hence a unitary operation.

Quadratic phase gates. A function $\chi : G \to U(1)$ (from the group G into the complex numbers of unit modulus) is called a character if $\chi(g+h) = \chi(g)\chi(h)$ for every $g, h \in G$ and

χ is continuous. A function $B : G \times G \to U(1)$ is said to be a bicharacter if it is continuous and if it is a character in both arguments. A function $\xi : G \to U(1)$ is called *quadratic* if it is continuous and if

$$\xi(g+h) = \xi(g)\xi(h)B(g,h), \quad \text{for every } g,\, h \in G \tag{3.6.2}$$

for some bicharacter $B(g,h)$. A quadratic phase gate is any diagonal unitary operation acting on $\mathcal{B}_G$ as $D_\xi : |h\rangle \to \xi(h)|h\rangle$, where ξ is a quadratic function of G.

Quantum Fourier transform. In contrast to both automorphism gates and quadratic phase gates, which leave the designated basis unchanged, the quantum Fourier transform (QFT) is a basis-changing operation: its action is *precisely* to change the designated basis $\mathcal{B}_G$ (at a given time) into another group-element basis $\mathcal{B}_{G'}$, according to certain rules described next.

Roughly speaking, the QFT realizes the unitary change of basis between standard and Fourier basis (Section 3.5). More precisely, there are different types of Fourier transforms, which act on the individual spaces $\mathcal{H}_N$ and $\mathcal{H}_\mathbb{Z}$.

- **QFT over $\mathbb{Z}_N$.** In the case of $\mathcal{H}_N$, both bases have the same cardinality (since the $\mathcal{H}_N$ is finite-dimensional) and such a change of basis can be actively performed by means of a unitary rotation; the QFT over $\mathbb{Z}_N$ (which we denote $\mathcal{F}_N$) is defined to be precisely this unitary operation. In the standard basis $|x\rangle$ with $x \in \mathbb{Z}_N$, the action of this gate on a state $|\psi\rangle = \sum \psi_x|x\rangle$ is

$$\mathcal{F}_N|\psi\rangle = \sum_{y\in\mathbb{Z}_N} \hat{\psi}(y)|y\rangle \quad \text{with } \hat{\psi}(y) := \frac{1}{\sqrt{N}} \sum_{x\in\mathbb{Z}_N} e^{2\pi i xy}\psi(x). \tag{3.6.3}$$

- **Infinite-dimensional QFTs.** Quantum Fourier transforms acting on spaces $\mathcal{H}_\mathbb{Z}$ have more exotic features than their finite dimensional counterparts. In the first place, these *QFTs are not gates* in the strict sense: since the standard basis $\{|x\rangle : x \in \mathbb{Z}\}$ and Fourier basis $\{|p\rangle : p \in \mathbb{T}\}$ have different cardinality, they cannot be rotated into each other (Section 3.5.2). Thus, the QFT is a change of basis between two orthonormal bases, but not a unitary rotation. In addition, due to this asymmetry, there are *two distinct types of QFTs*, defined as follows.

  QFT over $\mathbb{Z}$. If the standard basis ($\mathbb{Z}$ basis) is the designated basis of $\mathcal{H}_\mathbb{Z}$, states are represented as

$$|\psi\rangle = \sum_{x\in\mathbb{Z}} \psi(x)|x\rangle. \tag{3.6.4}$$

  Gates are defined according to this (integer) basis, which is also our measurement basis. When we say that *the QFT over $\mathbb{Z}$ is applied to $|\psi\rangle$*, we mean that the designated basis is changed from the standard basis to the Fourier basis. The state does not physically change[9], but normalizer gates applied after the QFT will be related to the circle group $\mathbb{T}$ (and *not* $\mathbb{Z}$); if we would measure the state right after the QFT, we would also do

[9]This somewhat metaphoric notation is chosen to be consistent with previously existing terminology [1, 2].

it in the $\mathbb{T}$ basis. As a result, the relevant amplitude-expansion of $|\psi\rangle$ is

$$|\psi\rangle = \int_{\mathbb{T}} \mathrm{d}p\ \hat{\psi}(p)|p\rangle \quad \text{with } \hat{\psi}(p) := \sum_{x\in\mathbb{Z}} \mathrm{e}^{2\pi i p x}\psi(x). \tag{3.6.5}$$

Some readers may notice, at this point, that the QFT over $\mathbb{Z}$ is nothing but the inverse Fourier series, also commonly known as the *discrete-time Fourier transform* [22].

QFT over $\mathbb{T}$. On the other hand, if the designated basis of $\mathcal{H}_{\mathbb{Z}}$ is the Fourier basis, states are represented as

$$|\psi\rangle = \int_{\mathbb{T}} \mathrm{d}p\ \psi(p)|p\rangle \tag{3.6.6}$$

When we say that *the QFT over $\mathbb{T}$ is applied to* $|\psi\rangle$, we mean that the designated basis is changed from the Fourier basis to the standard basis, and therefore we re-express the state $|\psi\rangle$ as

$$|\psi\rangle = \sum_{x\in\mathbb{Z}} \hat{\psi}(x)|x\rangle \quad \text{with } \hat{\psi}(x) := \int_{\mathbb{T}} \mathrm{d}p\ \mathrm{e}^{2\pi i p x}\psi(p). \tag{3.6.7}$$

Note that the coefficients $\hat{\psi}(x)$ in (3.6.7) are the Fourier coefficients appearing in the Fourier series [22]. Due to this fact, one can always regard $\psi(p)$ as a periodic function over the real numbers (with period 1), and identify the QFT over $\mathbb{T}$ as the quantum version of the Fourier series [22].

Lastly, note that the QFT over $\mathbb{Z}$ may only be applied (as an operation in a quantum computation) if the designated basis is the standard basis. Conversely, the QFT over $\mathbb{T}$ may only be applied if the designated basis is the Fourier basis.

We now consider the total Hilbert space $\mathcal{H}_G$ with designated basis $\mathcal{B}_G$, where $G = G_1 \times \cdots \times G_m$ with each G_i being a group of the form $\mathbb{Z}_N$, $\mathbb{Z}$ or $\mathbb{T}$. The total *QFT over G* is obtained by applying the QFT over G_i, as defined above, to all individual spaces in the tensor product decomposition of $\mathcal{H}_G$. In particular, application of the QFT over G implies that the designated basis is changed from $\mathcal{B}_G$ to $\mathcal{B}_{G'}$, where $G' = G_i' \times \ldots G_m'$ is defined as follows: if G_i has the form $\mathbb{Z}_N$ then $G_i' = G_i$: if $G_i = \mathbb{Z}$ then $G_i' := \mathbb{T}$ and, vice versa, if $G_i = \mathbb{Z}$ then $G_i' := \mathbb{T}$.

Similarly, one may perform a *partial QFT* by applying the QFT over G_i for a subset of the G_i, and leaving the other systems unchanged. The designated basis is changed accordingly on all subsystems where a QFT is applied.

3.6.2 Normalizer circuits

Roughly speaking, a normalizer circuit of size T is a quantum circuit $\mathcal{C} = U_T \cdots U_1$ composed of T normalizer gates U_i. More precisely, the definition is as follows.

A *normalizer circuit over* $G = \mathbb{Z}^{a+b} \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$ acts on a Hilbert space of the form

$$\mathcal{H}_G = \mathcal{H}_{\mathbb{Z}}^a \otimes \mathcal{H}_{\mathbb{Z}}^b \otimes (\mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_c}),$$

with arbitrary parameters a, b, c, N_i. At time $t = 0$ (before gates are applied), the designated basis of the computation is the group-element basis $\mathcal{B}_{G(0)}$ where $G(0)$ is a group from the family (3.5.10); without loss of generality, we set $G(0) = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$. The input, output, and gates of the circuit are constrained as follows:

- **Input states**. The input states are elements of the designated group basis $\mathcal{B}_{G(0)}$, i.e. $|g\rangle$ with $g \in G(0)$. The *registers*[10] $\mathcal{H}_{\mathbb{Z}}^{a}$ and $\mathcal{H}_{\mathbb{Z}}^{b}$ are initialized to be in standard-basis $|n\rangle$, $n \in \mathbb{Z}$ and Fourier-basis states $|p\rangle$, $p \in \mathbb{T}$, respectively[11]. It is assumed that Fourier basis inputs can be prepared within any finite (yet arbitrarily high) level of precision (cf. Section 4.2).

- **Gates.** At time $t = 1$, the gate U_1 is applied, which is either an automorphism gate, quadratic phase gate or a partial QFT over $G(0)$. The designated basis is changed from $\mathcal{B}_{G(0)}$ to $\mathcal{B}_{G(1)}$, for some group $G(1)$ in the family (3.5.10), according to the rules for changing the designated basis as described in Section 3.6.1.

  At time $t = 2$, the gate U_1 is applied, which is either an automorphism gate, quadratic phase gate or a partial QFT over $G(1)$. The designated basis is changed from $\mathcal{B}_{G(1)}$ to $\mathcal{B}_{G(2)}$, for some group $G(2)$.

  The gates $U_3, \dots, U_t$ are considered similarly. We denote by $\mathcal{B}_{G(t)}$ the designated basis after application of U_t (for some group $G(t)$ in the family (3.5.10)), for all $t = 3, \dots, T$. Thus, after all gates have been applied, the designated basis is $\mathcal{B}_{G(T)}$.

- **Measurement.** After the circuit, a measurement in the final designated basis $\mathcal{B}_{G(T)}$ is performed.

For *finite* Abelian groups and their associated Hilbert spaces (i.e. the Hilbert space has the form $\mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_a}$), the above definitions of normalizer circuits and normalizer gates specialize to the previously defined notion of normalizer circuits over finite Abelian groups, as done in [1, 2].

3.6.3 Classical encodings of normalizer gates

We now show how to give classical descriptions of normalizer gates and circuits. In the finite Abelian case this was straightforward [1, 2]: a description of a normalizer gate simply consisted of a list of its effects when applied to certain computational basis states. In the infinite dimensional case this approach does not work, because the Hilbert space is not finitely generated. Instead we will show that there exist efficient *standard encodings* of normalizer gates.

First, by saying that our encodings are *efficient*, we mean that the number of bits needed to store a description of a normalizer gate scales as $O(\text{poly}\, m, \text{polylog}\, N_i)$, where m is the total number of registers of the Hilbert space (3.6.1) and N_i are the local dimensions of the finite dimensional registers (the memory size of each normalizer gate in these encodings is given in table 3.1). This polynomial (as opposed to exponential) scaling in m is crucial in our setting, since normalizer gates may act nontrivially on all m registers of the Hilbert space (3.6.1)—this is an important difference between our computational model (based on normalizer gates) and the standard quantum circuit model [3], where a quantum circuit is always given as a sequence of one- and two-qubit gates.

Our standard encodings are as follows:

(i) A partial quantum Fourier transform $\mathcal{F}_i$ over G_i (the ith factor of G) is described by the index i indicating the register where the gate acts non-trivially.

[10] In a quantum computation we call each physical subsystem in (3.6.1) a "register".

[11] The results in this thesis can be easily generalized to input states that are stabilizer states (Section 4.4), given that we know the stabilizer group of the state.

(ii) An automorphism gate U_α is described by what we call a *matrix representation* A of the automorphism α (Definition 48): an $m \times m$ real matrix A that specifies the action of the map α. Such a matrix representation can be given in a normal form; see Theorem 51.

(iii) A quadratic phase gate D_ξ is described by an $m \times m$ real matrix M and an m-dim real vector v. The pair (M, v) specifies the action of the quadratic function ξ associated to D_ξ. Here we exploit a normal form for quadratic functions given later in Theorem 57.

A normalizer circuit is specified as a list of normalizer gates given to us in their standard encodings.

In our work, we assume that all maps α and ξ can be represented *exactly* by rational matrices and vectors A, M, v, which are explicitly given to us[12].

The efficiency of our standard encodings relies strongly on results presented in sections 3.9 and 3.10. In Section 3.9, we develop a (classical) **theory of matrix representations** of group homomorphisms, proving their existence (Lemma 50) and providing a normal form that characterizes the structure of these matrices (Theorem 51). In Section 3.10, we develop analytic normal forms for bicharacter functions (Lemmas 52, 53) and quadratic functions (Theorem 57). These results are also main contributions of our work.

We would like to highlight, in particular, that our **normal form for quadratic functions** (Theorem 57) should be of interest to a quantum audience. It was recently shown in [2] that quadratic functions over an Abelian group describe the quantum wave-functions of the so-called *stabilizer states*. Hence, our normal form can be used to characterize the complex amplitudes of such states[13].

Lastly, we mention that we allow the matrices A, M and the vector v in (i-iii) to contain *arbitrarily large* and *arbitrarily small* coefficients. This degree of generality is necessary in the setting we consider, since we allow *all* normalizer gates to be valid components of a normalizer circuit. However, the presence of infinite groups in (3.5.10) implies that that there exists an infinite number of normalizer gates (namely, of automorphism and quadratic gates, which follows from the our analysis in sections 3.9 and 3.10). This is in contrast with the settings considered in [1, 2], where both the group (3.5.1) and the associated set of normalizer gates are finite. As a result, the arithmetic precision needed to store the coefficients of A, M, v in our standard encodings becomes a variable of the model (just like in the standard problem of multiplying two integer matrices).

3.7 Normalizer circuits over black box groups

In this section we define normalizer circuits over black box groups: these circuits will act over Hilbert spaces of the form in 3.5.7:

[12]Some automorphisms and quadratic functions can only be represented by matrices with irrational entries (cf. the normal forms in sections 3.9,3.10). Restricting ourselves to study the rational ones allows us to develop *exact simulation algorithms*. We believe irrational matrices (even with transcendental entries) could also be handled by taking into account floating-point errors. We highlight that our stabilizer formalism and all of our normal forms are developed ***analytically***, and hold even if transcendental numbers appear in the matrix representations of α and ξ. (It is an good question to explore whether an exact simulation results may hold for matrices with algebraic coefficients.)

[13]As mentioned in the introduction, the result in [2] is for stabilizer states over finite dimensional Hilbert spaces but it can be easily generalized.

Input element	Description needs to specify...	Bits needed
Input state $\lvert g\rangle$	Element $g(i)$ of infinite group $\mathbb{Z}$, $\mathbb{T}$	variable
	Element $g(j)$ of finite group $\mathbb{Z}_{N_j}$	$\log N_j$
Normalizer circuit $\mathcal{C}$	Quantum Fourier transform $\mathcal{F}_i$	$\log m$
	Automorphism gate U_α	$m^2\|A\|_{\mathrm{b}}$
	Quadratic phase gate D_ξ	$m^2\|M\|_{\mathrm{b}} + m\|v\|_{\mathrm{b}}$

Table 3.1: The encoding size of a normalizer circuit. $\|X\|_{\mathrm{b}}$ denotes the number of bits used to store one single coefficient of X, which is always assumed to a rational matrix/vector. Formulas in column 3 are written in Big Theta notation and do not include constant factors (which are anyway small).

$$\mathcal{H} = \mathcal{H}_{\mathbb{Z}}^{\otimes a} \otimes \mathcal{H}_{\mathbb{T}}^{\otimes b} \otimes \mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_c} \otimes \mathcal{H}_{\mathbf{B}}. \tag{3.7.1}$$

where $\mathbf{B}$ is a finite Abelian black box group. The only additional ingredient compared with the previous section is the addition of the black box group $\mathbf{B}$, and it is on this that we will focus on. Let $m = a + b + c$.

We will only consider settings where *quantum Fourier transforms never act on the black box portion $\mathcal{H}_{\mathbf{B}}$ of the total system.* This is a natural restriction; in particular, (to our knowledge) in all existing quantum algorithms that do use QFTs, these QFTs act on systems of the form $\mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_c}$. Thus the only gates that act on the black box register $\mathcal{H}_{\mathbf{B}}$ are automorphism gates and quadratic phase gates.

Since the decomposition of $\mathbf{B}$ is unknown, we cannot hope for automorphism gates and quadratic phase gates to be specified as per the normal forms of Theorems 51 and 57. Therefore we assume that automorphism gates and quadratic phase gates are specified as *efficiently computable rational functions.* This limits the class of *classical functions* that we consider.

1. **Rational.**[14] An automorphism (or an arbitrary function) $\alpha : G \to G$ is rational if it returns rational outputs for all rational inputs. A quadratic function ξ is rational if it can be written in the form $\xi(g) = \exp(2\pi i\, q(g))$ where q is a rational function from G into $\mathbb{R}$ modulo $2\mathbb{Z}$.

2. **Efficiently computable.** α and q can be computed by polynomial-time uniform family of classical circuits $\{\alpha_i\}$, $\{q_i\}$. All α_i, q_i are $\mathrm{poly}(m, i)$ size classical circuits that query the *black-box group oracle* at most $\mathrm{poly}(m, i)$ times: their inputs are strings of rational numbers whose numerators and denominators are represented by i classical bits (their size is $O(2^i)$). For any rational element $g \in G$ that can be represented with so many bits (if G contains factors of the form $\mathbb{T}$ these are approximated by fractions), it holds that $\alpha_i(g) = \alpha(g)$ and $q_i(g) = q(g)$.

[14]We expect this assumption not to be essential, but it simplifies our proofs by allowing us to use exact arithmetic operations. Our stabilizer formalism can still be applied if the functions α, ξ are not rational, and we expect some version of the simulation result (Theorem 87) to hold even when trascendental numbers are involved (taking carefully into account precision errors). It is an good question to explore whether an exact simulation result may hold for algebraic numbers [103].

In certain cases (see Section 5.2) we will consider groups like $\mathbb{Z}_N^\times$ which, strictly speaking, are not black-box groups (because polynomial time algorithms for group multiplication for them are available and there is no need to introduce oracles). In those cases, the queries to the black-box group oracle (in the above model) are substituted by some efficient subroutine.

We add a third restriction to the above.

3. **Precision bound.** For any q or α that acts on an *infinite* group a bound n_{out} is given so that for every i, the number of bits needed to specify the numerators and denominators in the output of q_i or α_i exactly is at most $i + n_{\text{out}}$. The bound n_{out} is independent of i and indicates how much the input of each function may grow or shrink along the computation of the output[15]. This bound is used to correctly store the output of maps $\alpha : \mathbb{Z}^a \to \mathbb{Z}^a$, $\alpha' : \mathbb{Z}^a \to \mathbb{T}^a$ and to detect whether the output of a function $\alpha'' : \mathbb{T}^b \to \mathbb{T}^b$ might get truncated modulo 1.

The allowed automorphism gates U_α and quadratic phase gates D_ξ are those associated with efficiently computable rational functions α, ξ. We ask these unitaries to be efficiently implementable as well[16], by poly(m, i, n_{out})-size quantum circuits comprising at most poly(m, i, n_{out}) quantum queries of the group oracle. The variable i denotes the bit size used to store the labels g of the inputs $|g\rangle$ and bounds the precision level d of the normalizer computation, which we set to fulfill $\log d \in O(i + n_{\text{out}})$. The complexity of a normalizer gate is measured by the number of gates and (quantum) oracle queries needed to implement them.

In Section 5.2, we will see particular examples of efficiently computable normalizer gates. We will repeatedly make use of automorphism gates of the form

$$U_\alpha |k_1, \ldots, k_m, x\rangle \longrightarrow \left|k_1, \ldots, k_m, b_1^{k_1} \cdots b_m^{k_m} x\right\rangle$$

where k_i are integers and b_j, x are elements of some black-box group $\mathbf{B}$. These gates are allowed in our model, since there exist well-known efficient classical circuits for modular exponentiation given access to a group multiplication oracle [89]. In this case, a precision bound can be easily computed: since the infinite elements k_i do not change in size and all the elements of $\mathbf{B}$ are specified with strings of the same size, the output of α can be represented with as many bits as the input and we can simply take $n_{\text{out}} = 0$ (no extra bits are needed).

Many examples of efficiently computable normalizer gates were given in [1, 2], for decomposed finite group $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$. It was also shown in [1] that all normalizer gates over such groups can be efficiently implemented.

Finally, a normalizer circuit over G is simply a circuit composed of normalizer gates over the black box group G, defined analogously to Section 3.6.2: the input state is a basis state, normalizer circuits are applied at each step of the circuit (changing the *designated basis* of the underlying group if the applied gate is a QFT), and at the end of the computation a measurement is made in the current designated basis.

[15]For infinite groups there is no fundamental limit to how much the output of α or q may grow/shrink with respect to the input (this follows from the normal forms in Theorems 51 and 57.). The number n_{out} parametrizes the precision needed to compute the function. Similarly to 2., this assumption might me weakened if a treatment for precision errors is incorporated in the model.

[16]Recall that, in finite dimensions, the gate cost of implementing a classical function α as a quantum gate is at most the classical cost [3] and that computing q efficiently is enough to implement ξ using phase kick-back tricks [104]. We expect these results to extend to infinite dimensional systems of the form $\mathcal{H}_{\mathbf{Z}}$.

Precision requirements

In the model of quantum circuits above, input states and final measurements in the Fourier-basis $\{|p\rangle, p \in \mathbb{T}\}$ of $\mathcal{H}_\mathbb{Z}$ can never be implemented with perfect accuracy, a limitation that stems from the fact that the $|p\rangle$ states are *unphysical.* This can be quickly seen in two ways: first, in the $\mathbb{Z}$ basis, these states are infinitely-spread plane-waves $|p\rangle = \overline{\sum e^{2\pi i z p}|z\rangle}$; second, in the $\mathbb{T}$ basis, they are infinitely-localized Dirac-delta pulses. Physically, preparing Fourier-basis states or measuring in this basis *perfectly* would require infinite energy and lead to infinite precision issues in our computational model.

In the algorithms we study in this thesis (namely, the order-finding algorithm in Theorem 76), Fourier states over $\mathbb{Z}$ can be substituted with ***realistic physical approximations.*** The degree of *precision* used in the process of Fourier state preparation is treated as a *computational resource.* We model the precision used in a computation as follows.

Since our goal is to use the Fourier basis $|p\rangle$, $p \in \mathbb{T}$, to represent information in a computation, we require the ability to store and retrieve information in this continuous-variable basis. Our assumption is that for any set X with cardinality $d = |X|$, we can divide the continuous circle-group $\mathbb{T}$ spectrum into d equally sized sectors of length $1/d$ and use them to represent the elements of X. More precisely, to each element of X we assign a number in $\mathbb{Z}_d$. The element $x_i \in X$ with index $i \in \mathbb{Z}_d$ is then represented by any state of the subspace $V_{i,d} = \text{span}\{|\frac{i}{d} + \Delta\rangle$ with $|\Delta| < \frac{1}{2d}\}$. We call the latter states *d-approximate Fourier states* and refer to d as the *precision level* of the computation. We assume that these states can be prepared and distinguished to any desired precision d in the following way:

1. **State preparation assumption.** Inputs $|\psi_i\rangle$ with at least $\frac{2}{3}$ fidelity to some element of $V_{i,d}$ can be prepared for any $i \in \mathbb{Z}_d$.

2. **Distinguishability assumption.** The subspaces $V_{i,d}$ can be reliably distinguished.

Note that d determines how much information is stored in the Fourier basis.

Definition 41 (Efficient use of precision[17]). A *quantum algorithm* that uses d-approx. Fourier states to solve a computational problem with input size n is said to use an *efficient* amount of precision if and only $\log d$ is upper bounded by some polynomial of n. Analogously, an algorithm that stores information in the standard basis $\{|m\rangle$, $m \in \mathbb{Z}\}$ is said to be *efficient* if the states with m larger than some threshold $\log(m_{\max}) \in O(\text{poly}\, n)$ do not play a role in the computation.

3.8 Group and Character Theory

3.8.1 Elementary Abelian groups

A group of the form

$$G = \mathbb{Z}^a \times \mathbb{R}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbb{T}^d \tag{3.8.1}$$

[17]Note that this definition is not necessary to define normalizer circuits but to discuss the physicality of the model. We point out that there might be better ways to model precision than ours (which may, e.g., lead to tighter bounds or more efficient algorithms), but our simple model is enough to derive our main results. We advance that, even if these precision requirements turned out to be high in practice, there exist efficient discretized *qubit* implementations of all the infinite-dimensional quantum algorithms that we study later in the thesis (cf. theorem 79).

will be called an *elementary Abelian group*. We will often use the shorthand notation $F = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$ for the finite subgroup in the above decomposition. Note that the Hilbert space formalism introduced in Section 3.5 does not refer to groups of the form $\mathbb{R}^b$ (i.e. direct products of the group of real numbers). However for some of our calculations in later sections, it will be convenient to include these types of groups in the analysis.

An elementary Abelian group of the form $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{T}$ or $\mathbb{Z}_N$ is said to be primitive. Thus every elementary Abelian group can be written as $G = G_1 \times \cdots \times G_m$ with each G_i primitive; we will often use this notation. We will also use the notation $G_\mathbb{Z}$, $G_\mathbb{R}$, G_F, $G_\mathbb{T}$ to denote elementary Abelian groups that are, respectively, integer lattices $\mathbb{Z}^a$, real lattices $\mathbb{R}^b$, finite groups F and tori $\mathbb{T}^d$. We will also assume that the factors G_i of G are arranged so that $G = G_\mathbb{Z} \times G_\mathbb{R} \times G_F \times G_\mathbb{T}$.

Next we introduce the notion of group characteristic char(G) for primitive groups:

$$\mathrm{char}(\mathbb{Z}) := 0, \quad \mathrm{char}(\mathbb{R}) := 0, \quad \mathrm{char}(\mathbb{Z}_N) := N, \quad \mathrm{char}(\mathbb{T}) := 1. \tag{3.8.2}$$

Alternatively, we can also define the characteristic as follows: char(G) is the number that coincides with (a) the order of 1 in G if 1 has finite order (which is the case for $\mathbb{Z}_N$ and $\mathbb{T}$); (b) zero, if 1 has infinite order in G (which is the case for $\mathbb{Z}$ and $\mathbb{R}$).

Consider an elementary Abelian group $G = G_1 \times \cdots \times G_m$ where c_i is the characteristic of G_i. Each element $g \in G$ can be represented as an m-tuple $g = (g_1, \ldots, g_m)$ of real numbers. If $x = (x_1, \ldots, x_m)$ is an arbitrary m-tuple of real numbers, we say that x is congruent to g, denoted by $x \equiv g \pmod{G}$, if

$$x_i \equiv g_i \pmod{c_i} \quad \text{for every } i = 1, \ldots, m. \tag{3.8.3}$$

For example, every string of the form $x = (\lambda_1 c_1, \ldots, \lambda_m c_m)$ with $\lambda_i \in \mathbb{Z}$ is congruent to $0 \in G$.

3.8.2 Characters

Definition 42 (Character [97, 105]**).** Let G be an elementary Abelian group. A character of G is a continuous homomorphism χ from G into the group $U(1)$ of complex numbers with modulus one. Thus $\chi(g+h) = \chi(g)\chi(h)$ for every $g, h \in G$.

If G is an elementary Abelian group, the set of all of its characters is again an elementary Abelian group, called the dual group of G, denoted by $\widehat{G}$. Moreover, $\widehat{G}$ is isomorphic to another elementary Abelian group, according to the following rule:

$$G = \mathbb{R}^a \times \mathbb{T}^b \times \mathbb{Z}^c \times F \qquad \longrightarrow \qquad \widehat{G} \cong \mathbb{R}^a \times \mathbb{Z}^b \times \mathbb{T}^c \times F. \tag{3.8.4}$$

Thus, in particular, $\widehat{\mathbb{R}}$ is isomorphic to $\mathbb{R}$ itself and similarly $\widehat{F}$ is isomorphic to F itself; these groups are called autodual. On the other hand, $\widehat{\mathbb{Z}}$ is isomorphic to $\mathbb{T}$ and, conversely, $\widehat{\mathbb{T}}$ is isomorphic to $\mathbb{Z}$. We also note from the rule (3.8.4) that the dual group of $\widehat{G}$ is isomorphic to G itself. This is a manifestation of the Pontryagin-Van Kampen duality [97, 98, 86].

We now give explicit formulas for the characters of any primitive Abelian group.

- The characters of $\mathbb{R}$ are

$$\chi_x(y) := \exp(2\pi i x y), \quad \text{for every } x,\, y \in \mathbb{R}. \tag{3.8.5}$$

Thus each character is labeled by a real number. Note that $\chi_x \chi_{x'} = \chi_{x+x'}$ for all $x, x' \in \mathbb{R}$. The map $x \to \chi_x$ is an isomorphism from $\mathbb{R}$ to $\widehat{\mathbb{R}}$, so that $\mathbb{R}$ is autodual.

- The characters of $\mathbb{Z}_N$ are

$$\chi_x(y) := \exp\left(\frac{2\pi i}{N}\, xy\right), \quad \text{for every } x,\, y \in \mathbb{Z}_N. \tag{3.8.6}$$

  Thus each character is labeled by an element of $\mathbb{Z}_N$. As above, we have $\chi_x\chi_{x'} = \chi_{x+x'}$ for all $x, x' \in \mathbb{Z}_N$. The map $x \to \chi_x$ is an isomorphism from $\mathbb{Z}_N$ to $\widehat{\mathbb{Z}}_N$, so that $\mathbb{Z}_N$ is autodual.

- The characters of $\mathbb{Z}$ are

$$\chi_p(m) := \exp(2\pi i p m), \quad \text{for every } p \in \mathbb{T},\ m \in \mathbb{Z}, \tag{3.8.7}$$

  Each character is labeled by an element of $\mathbb{T}$. Again we have $\chi_p\chi_{p'} = \chi_{p+p'}$ for all $p, p' \in \mathbb{T}$ and the map $p \to \chi_p$ is an isomorphism from $\mathbb{T}$ to $\widehat{\mathbb{Z}}$.

- The characters of $\mathbb{T}$ are

$$\chi_m(p) := \exp(2\pi i p m), \quad \text{for every } p \in \mathbb{T},\ m \in \mathbb{Z}; \tag{3.8.8}$$

  Each character is labeled by an element of $\mathbb{Z}$. Again we have $\chi_m\chi_{m'} = \chi_{m+m'}$ for all $m, m' \in \mathbb{Z}$ and the map $m \to \chi_m$ is an isomorphism from $\mathbb{Z}$ to $\widehat{\mathbb{T}}$.

If G is a general elementary Abelian group, its characters are obtained by taking products of the characters described above. More precisely, if A and B are two elementary Abelian groups, the character group of $A \times B$ consists of all products $\chi_A\chi_B$ with $\chi_A \in \widehat{A}$ and $\chi_B \in \widehat{B}$, and where $\chi_A\chi_B(a, b) := \chi_A(a)\chi_B(b)$ for every $(a, b) \in A \times B$. To obtain all characters of a group G having the form (3.8.4), we denote

$$G^* := \mathbb{R}^a \times \mathbb{Z}^b \times \mathbb{T}^c \times F. \tag{3.8.9}$$

Considering an arbitrary element

$$\mu = (r_1, \ldots, r_a, z_1, \ldots, z_b, t_1, \ldots, t_c, f_1, \ldots, f_d) \in G^*, \tag{3.8.10}$$

the associated character is given by the product

$$\chi_\mu := \chi_{r_1} \cdots \chi_{r_a}\ \chi_{z_1} \cdots \chi_{z_b}\ \chi_{t_1} \cdots \chi_{t_c}\ \chi_{f_1} \cdots \chi_{f_d} \tag{3.8.11}$$

where the individual characters $\chi_{r_i}, \chi_{z_j}, \chi_{t_k}, \chi_{f_l} \cdots$ of $\mathbb{R}, \mathbb{Z}, \mathbb{T}$ and $\mathbb{Z}_{N_l}$ are defined above. The character group of G is given by

$$\widehat{G} = \{\chi_\mu : \mu \in G^*\}. \tag{3.8.12}$$

The rule (3.8.4) immediately implies that $(G^*)^* = G$. This implies that the character group of G^* is $\{\chi_g : g \in G\}$, where χ_g is defined in full analogy with (3.8.11). Two elementary but important features are the following:

Lemma 43. *For every $g \in G$ and $\mu \in G^*$ we have*

$$\chi_\mu(g) = \chi_g(\mu). \tag{3.8.13}$$

Lemma 44. *For every $\mu, \nu \in G^*$ and every $g \in G$ it follows that*

$$\chi_{\mu+\nu}(g) = \chi_\mu(g)\chi_\nu(g) \tag{3.8.14}$$

Both Lemmas 43 and 44 follow from inspection of the characters of $\mathbb{R}, \mathbb{Z}, \mathbb{T}$ and $\mathbb{Z}_N$ defined in (3.8.5)-(3.8.8). The lemmas also reflect the strong duality between G and G^*.

Finally, the definition of every character function $\chi_a(b)$ as given in (3.8.5)-(3.8.8), which is in principle defined for a in $\mathbb{R}, \mathbb{Z}_N, \mathbb{Z}, \mathbb{T}$ and b in $\mathbb{R}, \mathbb{Z}_N, \mathbb{T}, \mathbb{Z}$, respectively, can be readily extended to the entire domain of real numbers, yielding functions $\chi_x(y)$ with $x, y \in \mathbb{R}$. Consequently, the character functions (3.8.11) of general elementary Abelian groups $G = G_1 \times \cdots \times G_m$ can also be extended to a larger domain, giving rise to functions $\chi_x(y)$ where $x, y \in \mathbb{R}^m$. With this extended notion, we have the following basic property:

Lemma 45. *Let $g \in G$ and $\mu \in G^*$. For every $x, y \in \mathbb{R}^m$ such that $x \equiv g$ (mod G) and $y \equiv \mu$ (mod G^*), we have*

$$\chi_y(x) = \chi_\mu(g). \tag{3.8.15}$$

The proof is easily given for primitive groups, and then extended to general elementary Abelian groups.

3.8.3 Simplifying characters via the bullet group

Let $G = G_1 \times \cdots \times G_m$ be an elementary Abelian group where each G_i is of primitive type. Recall that in Section 3.8.2 we have introduced the definition of the elementary Abelian group G^*, which is isomorphic to the character group of G. Here we define another group $G^\bullet$ (which is Abelian but not elementary), called the bullet group of G. The bullet group is isomorphic to G^* (and hence also to the character group of G) and is mainly introduced for notational convenience, as working with $G^\bullet$ will turn out to simplify calculations. The bullet group is defined to be $G^\bullet = G_1^\bullet \times \cdots \times G_m^\bullet$, where

$$\begin{aligned} \mathbb{Z}_N^\bullet &:= \left\{0, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N} \bmod 1\right\}, \\ \mathbb{R}^\bullet &:= \mathbb{R}^* = \mathbb{R}; \quad \mathbb{Z}^\bullet := \mathbb{Z}^* = \mathbb{T}; \quad \mathbb{T}^\bullet := \mathbb{T}^* = \mathbb{Z}. \end{aligned} \tag{3.8.16}$$

Thus the only difference between the groups G^* and $G^\bullet$ is in the $\mathbb{Z}_{N_i}$ components. The groups G^* and $G^\bullet$ are isomorphic via the "bullet map"

$$\mu \in G^* \to \mu^\bullet := (\mu_1^\bullet, \ldots, \mu_m^\bullet) \in G^\bullet, \tag{3.8.17}$$

where $\mu_i^\bullet := \mu_i/N$ if $\mu_i \in \mathbb{Z}_N$ and $\mu_i^\bullet = \mu_i$ if μ_i belongs to either $\mathbb{R}, \mathbb{Z}$ or $\mathbb{T}$. The bullet map is easily seen to be an isomorphism.

One of the main purposes for introducing the bullet group is to simplify calculations with characters. In particular, for every $g \in G$ and $\mu \in G^*$ we have

$$\chi_\mu(g) = \exp\left(2\pi\mathrm{i} \sum_{i=1}^m \mu_i^\bullet g_i\right). \tag{3.8.18}$$

3.8.4 Annihilators

Let G be an elementary Abelian group. Let X be any subset of G. The *annihilator* $X^\perp$ is the subset

$$X^\perp := \{\mu \in G^* : \chi_\mu(x) = 1 \text{ for every } x \in X\}. \tag{3.8.19}$$

We can define the annihilator $Y^\perp$ of a subset $Y \subseteq G^*$ analogously as

$$Y^\perp := \{x \in G : \chi_\mu(x) = 1 \text{ for every } \mu \in Y\}. \tag{3.8.20}$$

By combining the two definitions it is possible to define double annihilator sets $X^{\perp\!\!\perp} := (X^\perp)^\perp$, which is a subset of the initial group G, for every set $X \subseteq G$. Similarly, $Y^{\perp\!\!\perp} \subseteq G^*$ for every $Y \subseteq G^*$. The following lemma states that X and $X^{\perp\!\!\perp}$ are, in fact, identical sets *iff* X is closed as a set, and related to each other in full generality:

Lemma 46 ([98]). *Let X be an arbitrary subset of an elementary Abelian group G. Then the following assertions hold:*

(a) The annihilator $X^\perp$ is a closed subgroup of G^* (and $X^{\perp\!\!\perp}$ is a closed subgroup of G).

(b) $X^{\perp\!\!\perp}$ is the smallest closed subgroup of G containing X.

(c) If Y is a subset of G such that $X \subseteq Y$ then $X^\perp \supseteq Y^\perp$ and $X^{\perp\!\!\perp} \subseteq Y^{\perp\!\!\perp}$.

We mention that in the quantum computation literature (see e.g. [78, 21, 1, 2]) the annihilator $H^\perp$ of a subgroup H is more commonly known as the *orthogonal subgroup* of H.

Let $\alpha : G \to H$ be a continuous group homomorphism between two elementary Abelian groups G and H. Then [97, proposition 30] there exists a unique continuous group homomorphism $\alpha^* : H^* \to G^*$, which we call the **dual homomorphism** of α, defined via the equation

$$\chi_{\alpha^*(\mu)}(g) = \chi_\mu(\alpha(g)). \tag{3.8.21}$$

Note that $\alpha^{**} = \alpha$ by duality.

3.9 Homomorphisms and matrix representations

3.9.1 Homomorphisms

Let $G = G_1 \times \ldots \times G_m$ and $H = H_1 \times \ldots \times H_n$ be two elementary finite Abelian groups, where G_i, H_j are primitive subgroups. As discussed in Section 3.8.1, we assume that the G_i and H_j are ordered so that $G = G_\mathbb{Z} \times G_\mathbb{R} \times G_F \times G_\mathbb{T}$ and $H = H_\mathbb{Z} \times H_\mathbb{R} \times H_F \times H_\mathbb{T}$.

Consider a continuous group homomorphism $\alpha : G \to H$. Let $\alpha_{\mathbb{Z}\mathbb{Z}} : G_\mathbb{Z} \to G_\mathbb{Z}$ be the map obtained by restricting the input and output of α to $H_\mathbb{Z}$. More precsiely, for $g \in G_\mathbb{Z}$ consider the map

$$(g, 0, 0, 0) \in G \to \alpha(g, 0, 0, 0) \in H \tag{3.9.1}$$

and define $\alpha_{\mathbb{Z}\mathbb{Z}}(g)$ to be the $G_\mathbb{Z}$-component of $\alpha(g, 0, 0, 0)$. The resulting map $\alpha_{\mathbb{Z}\mathbb{Z}}$ is a continuous homomorphism from $\mathbb{Z}$ to $\mathbb{Z}$. Analogously, we define the continuous group homomorphisms $\alpha_{XY} : G_Y \to H_X$ with $X, Y = \mathbb{Z}, \mathbb{R}, \mathbb{T}, F$. It follows that, for any $g = (z, r, f, t) \in G$, we have

$$\alpha(g) = \begin{pmatrix} \alpha_{\mathbb{Z}\mathbb{Z}}(z) + \alpha_{\mathbb{Z}\mathbb{R}}(r) + \alpha_{\mathbb{Z}F}(f) + \alpha_{\mathbb{Z}\mathbb{T}}(t) \\ \alpha_{\mathbb{R}\mathbb{Z}}(z) + \alpha_{\mathbb{R}\mathbb{R}}(r) + \alpha_{\mathbb{R}F}(f) + \alpha_{\mathbb{R}\mathbb{T}}(t) \\ \alpha_{F\mathbb{Z}}(z) + \alpha_{F\mathbb{R}}(r) + \alpha_{FF}(f) + \alpha_{F\mathbb{T}}(t) \\ \alpha_{\mathbb{T}\mathbb{Z}}(z) + \alpha_{\mathbb{T}\mathbb{R}}(r) + \alpha_{\mathbb{T}F}(f) + \alpha_{\mathbb{T}\mathbb{T}}(t) \end{pmatrix} \leftrightarrow \begin{pmatrix} \alpha_{\mathbb{Z}\mathbb{Z}} & \alpha_{\mathbb{Z}\mathbb{R}} & \alpha_{\mathbb{Z}F} & \alpha_{\mathbb{Z}\mathbb{T}} \\ \alpha_{\mathbb{R}\mathbb{Z}} & \alpha_{\mathbb{R}\mathbb{R}} & \alpha_{\mathbb{R}F} & \alpha_{\mathbb{R}\mathbb{T}} \\ \alpha_{F\mathbb{Z}} & \alpha_{F\mathbb{R}} & \alpha_{FF} & \alpha_{F\mathbb{T}} \\ \alpha_{\mathbb{T}\mathbb{Z}} & \alpha_{\mathbb{T}\mathbb{R}} & \alpha_{\mathbb{T}F} & \alpha_{\mathbb{T}\mathbb{T}} \end{pmatrix} \begin{pmatrix} z \\ r \\ f \\ t \end{pmatrix} \tag{3.9.2}$$

α is therefore naturally identified with the the 4×4 "matrix of maps" given in the r.h.s of (3.9.2).

The following lemma (see e.g. [106] for a proof) shows that homomorphisms between elementary Abelian groups must have a particular block structure.

Lemma 47. *Let $\alpha : G \to H$ be a continuous group homomorphism. Then α has the following block structure*

$$\alpha \leftrightarrow \begin{pmatrix} \alpha_{\mathbb{Z}\mathbb{Z}} & 0 & 0 & 0 \\ \alpha_{\mathbb{R}\mathbb{Z}} & \alpha_{\mathbb{R}\mathbb{R}} & 0 & 0 \\ \alpha_{F\mathbb{Z}} & 0 & \alpha_{FF} & 0 \\ \alpha_{\mathbb{T}\mathbb{Z}} & \alpha_{\mathbb{T}\mathbb{R}} & \alpha_{\mathbb{T}F} & \alpha_{\mathbb{T}\mathbb{T}} \end{pmatrix} \tag{3.9.3}$$

where 0 denotes the trivial group homomorphism.

The lemma shows, in particular, that there are no non-trivial continuous group homomorphisms between certain pairs of primitive groups: for instance, continuous groups cannot be mapped into discrete ones, nor can finite groups be mapped into zero-characteristic groups.

3.9.2 Matrix representations

Definition 48 (Matrix representation). Consider elementary Abelian groups $G = G_1 \times \cdots \times G_m$ and $H = H_1 \times \cdots \times H_n$ and a group homomorphism $\alpha : G \to H$. A *matrix representation of* α is an $n \times m$ real matrix A satisfying the following property:

$$\alpha(g) \equiv Ax \pmod H \quad \text{for every } g \in G \text{ and } x \in \mathbb{R}^m \text{ satisfying } x \equiv g \pmod G \tag{3.9.4}$$

Conversely, a real $n \times m$ matrix A is said to define a group homomorphism if there exists a group homomorphism α satisfying (3.9.4).

It is important to highlight that in the definition of matrix representation we impose that the identity $\alpha(g) = Ax \pmod H$ holds in a very general sense: the output of the map must be equal for inputs x, x' that are *different* as strings of real numbers but correspond to the *same* group element g in the group G. In particular, all strings that are congruent to zero in G must be mapped to strings congruent to zero in H. Though these requirements are (of course) irrelevant when we only consider groups of zero characteristic (like $\mathbb{Z}$ or $\mathbb{R}$), they are crucial when quotient groups are involved (such as $\mathbb{Z}_N$ or $\mathbb{T}$).

As a simple example of a matrix representation, we consider the bullet map[18], which is an isomorphism from G^* to $G^\bullet$. Define the diagonal $m \times m$ matrix Υ with diagonal entries

[18]Strictly speaking, definition 48 cannot be applied to the bullet map, since $G^\bullet$ is not an elementary Abelian group. However the definition is straightforwardly extended to remedy this.

defined as

$$\Upsilon(i,i) = \begin{cases} 1/N_i & \text{if } G_i = \mathbb{Z}_{N_i} \text{ for some } N_i, \\ 1 & \text{otherwise.} \end{cases} \tag{3.9.5}$$

It is easily verified that Υ satisfies the following property: for every $\mu \in G^*$ and $x \in \mathbb{R}^m$ satisfying $x \equiv \mu \pmod{G^*}$, we have

$$\mu^\bullet \equiv \Upsilon x \pmod{G^\bullet}. \tag{3.9.6}$$

Note that, with the definition of Υ, equation (3.8.18) implies

$$\chi_\mu(g) = \exp\left(2\pi\mathrm{i}\sum_{i=1}^{m}\mu^\bullet(i)g(i)\right) = \exp\left(2\pi\mathrm{i}\,\mu^T\Upsilon g\right). \tag{3.9.7}$$

Looking at equation (3.9.6) coefficient-wise, we obtain a relationship $\mu^\bullet(i) \equiv \frac{x(i)}{N_i} \pmod 1$ for each factor G_i of the form $\mathbb{Z}_{N_i}$; other factors are left unaffected by the bullet map. From this expression it is easy to derive that Υ^{-1} is a matrix representation of the inverse of the bullet map[19], i.e. the group isomorphism $\mu^\bullet \to \mu \pmod{G^*}$.

The next lemma (see Appendix B.1 for a proof) summarizes some useful properties of matrix representations.

Lemma 49 (Properties of matrix representations). *Let G, H, J be elementary Abelian groups, and $\alpha : G \to H$ and $\beta : H \to J$ be group homomorphisms with matrix representations A, B, respectively. Then it holds that*

(a) BA is a matrix representation of the composed homomorphism $\beta \circ \alpha$;
(b) The matrix $A^* := \Upsilon_G^{-1}A^T\Upsilon_H$ is a matrix representation of the dual homomorphism α^*, where Υ_X denotes the matrix representation of the bullet map $X^* \to X^\bullet$.

As before, let $G = G_1 \times \cdots \times G_m$ be an elementary Abelian group with each G_i of primitive type. Let

$$e_i = (0, \ldots, 0, 1, 0, \ldots, 0) \tag{3.9.8}$$

denote the i-th canonical basis vector of $\mathbb{R}^m$. If we regard $g \in G$ as an element of $\mathbb{R}^m$, we may write $g = \sum g(i)e_i$. Note however that e_i may not belong to G itself. In particular, if $G_i = \mathbb{T}$ then $e_i \notin \mathbb{T}$ (since $1 \notin \mathbb{T}$ in the representation we use, i.e. $\mathbb{T} = [0,1)$).

Lemma 50 (Existence of matrix representations). *Every group homomorphism $\alpha : G \to H$ has a matrix representation A. As a direct consequence, we have $\alpha(g) \equiv \sum_i g(i)Ae_i$ (mod H), for every $g = \sum_i g(i)e_i \in G$.*

The last property of Lemma 50 is remarkable, since the coefficients $g(i)$ are real numbers when G_i is of the types $\mathbb{R}$ and $\mathbb{T}$. We give a proof of the lemma in Appendix B.1.

We finish this section by giving a normal form for matrix representations and characterizing which types of matrices constitute valid matrix representations as in definition 48.

[19]We ought to highlight that the latter is by no means a general property of matrix representations. In fact, in many cases, the matrix-inverse A^{-1} (if it exists) of a matrix representation A of a group isomorphism is not a valid matrix representation of a group homomorphism. (This happens, for instance, for all group automorphisms of the group $\mathbb{Z}_N$ that are different from the identity.) In Theorem 51 we characterize which matrices are valid matrix representations. Also, in Section 4.5.2 we discuss the problem of computing matrix representations of group automorphisms.

Theorem 51 (Normal form of a matrix representation). *Let $G = G_1 \times \cdots \times G_m$ and $H = H_1 \times \cdots \times H_n$ be elementary Abelian groups. Let c_j, c_j^*, d_i and d_i^* denote the characteristic of G_j, G_j^*, H_i and H_i^*, respectively. Define* **Rep** *to be the subgroup of all $n \times m$ real matrices that have integer coefficients in those rows i for which H_i has the form $\mathbb{Z}$ or $\mathbb{Z}_{d_i}$. A real $n \times m$ matrix A is a valid matrix representation of some group homomorphism $\alpha : G \to H$ iff A is an element of* **Rep** *fulfilling two (dual) sets of consistency conditions:*

$$c_j A(i,j) = 0 \mod d_i, \qquad d_i^* A^*(i,j) = 0 \mod c_j^*, \tag{3.9.9}$$

for every $i = 1, \ldots, n$, $j = 1, \ldots, m$, and being A^* the $m \times n$ matrix defined in Lemma 49(b). Equivalently, A must be of the form

$$A := \begin{pmatrix} A_{\mathbb{ZZ}} & 0 & 0 & 0 \\ A_{\mathbb{RZ}} & A_{\mathbb{RR}} & 0 & 0 \\ A_{F\mathbb{Z}} & 0 & A_{FF} & 0 \\ A_{\mathbb{TZ}} & A_{\mathbb{TR}} & A_{\mathbb{T}F} & A_{\mathbb{TT}}. \end{pmatrix} \tag{3.9.10}$$

with the following restrictions:

1. *$A_{\mathbb{ZZ}}$ and $A_{\mathbb{TT}}$ are arbitrary integer matrices.*
2. *$A_{\mathbb{RZ}}$, $A_{\mathbb{RR}}$ are arbitrary real matrices.*
3. *$A_{F\mathbb{Z}}$, A_{FF} are integer matrices: the first can be arbitrary; the coefficients of the second must be of the form*
$$A(i,j) = \alpha_{i,j} \frac{d_i}{\gcd(d_i, c_j)} \tag{3.9.11}$$
where $\alpha_{i,j}$ can be arbitrary integers[20].
4. *$A_{\mathbb{TZ}}$, $A_{\mathbb{TR}}$ and $A_{\mathbb{T}F}$ are real matrices: the first two are arbitrary; the coefficients of the third are of the form $A(i,j) = \alpha_{i,j}/c_j$ where $\alpha_{i,j}$ can be arbitrary integers*[21].

The result is proven in Appendix B.1. For the normalizer circuits we consider, we will not need to use the results regarding $\mathbb{R}$.

3.10 Quadratic functions

In this section we study the properties of quadratic functions over arbitrary elementary groups of the form $G = \mathbb{R}^a \times \mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$. Most importantly, we give normal forms for quadratic functions and bicharacters. We list results without proof, since all techniques used throughout the section are classical. Yet, we highlight that the normal form in Theorem 59 should be of quantum interest, since it can be used to give a normal form for stabilizer states over elementary groups.

All results in this section are proven in **Appendix B.2**.

[20]Since $A_{F\mathbb{Z}}$, A_{FF} multiply integer tuples and output integer tuples modulo $F = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$, for some N_is, the coefficients of their ith rows can be chosen w.l.o.g. to lie in the range $[0, N_i)$ (by taking remainders).

[21]Due to the periodicity of the torus, the coefficients of $A_{\mathbb{TZ}}$, $A_{\mathbb{T}F}$ can be chosen to lie in the range $[0, 1)$.

3.10.1 Definitions

Let G be an elementary Abelian group. Recall from Section 3.6 that a bicharacter of G is a continuous complex function $B : G \times G \to U(1)$ such that the restriction of B to either one of its arguments is a character of G. Recall that a quadratic function $\xi : G \to U(1)$ is a continuous function for which there exists a bicharacter B such that

$$\xi(g+h) = \xi(g)\xi(h)B(g,h) \quad \text{for all } g,h \in G. \tag{3.10.1}$$

We say that ξ is a B-representation.

A bicharacter B is said to be symmetric if $B(g,h) = B(h,g)$ for all $g,h \in G$. Symmetric bicharacters are natural objects to consider in the context of quadratic functions: if ξ is a B-representation then B is symmetric since

$$B(g,h) = \xi(g+h)\overline{\xi(g)\xi(h)} = \xi(h+g)\overline{\xi(h)\xi(g)} = B(h,g). \tag{3.10.2}$$

3.10.2 Normal form of bicharacters

The next lemmas characterize bicharacter functions.

Lemma 52 (Normal form of a bicharacter). *Given an elementary Abelian group G, then a function $B : G \times G \to U(1)$ is a bi-character iff it can be written in the normal form*

$$B(g,h) = \chi_{\beta(g)}(h) \tag{3.10.3}$$

where β is some group homomorphism from G into G^*.

This result generalizes Lemma 5(a) in [1]. The next lemma gives a explicit characterization of symmetric bicharacter functions.

Lemma 53 (Normal form of a symmetric bicharacter). *Let B be a symmetric bicharacter of G in the form (3.10.3) and let A be a matrix representation of the homomorphism β. Let Υ denote the default matrix representation of the bullet map $G^* \to G^\bullet$ as in (3.9.5), and $M = \Upsilon A$. Then*

(a) $B(g,h) = \exp(2\pi i\, g^T M h)$ *for all $g,h \in G$.*
(b) *M is a matrix representation of the homomorphism* $G \xrightarrow{\beta} G^* \xrightarrow{\bullet} G^\bullet$.
(c) If $x,y \in \mathbb{R}^m$ and $g,h \in G$ are such that $x \equiv g$ (mod G) and $y \equiv h$ (mod G), then

$$B(g,h) = \exp(2\pi i\, x^T M y). \tag{3.10.4}$$

(d) The matrix M is symmetric modulo integer factors, i.e. $M = M^T \bmod \mathbb{Z}$.
(e) The matrix M can be efficiently symmetrized: i.e. one can compute in classical polynomial time a symmetric matrix $M' = M'^T$ that also fulfills (a)-(b)-(c).

3.10.3 Normal form of quadratic functions

Our final goal is to characterize all quadratic functions. This is achieved in Theorem 57. To show this result a few lemmas are needed.

Lemma 54. *Two quadratic functions ξ_1,ξ_2 that are B-representations of the same bicharacter B must be equal up to multiplication by a character of G, i.e. there exists $\mu \in G^*$ such that*

$$\xi_1(g) = \chi_\mu(g)\xi_2(g), \quad \textit{for every } g \in G. \tag{3.10.5}$$

We highlight that a much more general version of Lemma 54 was proven in [107], using projective representation theory[22].

Our approach now will be to find a method to construct a quadratic function that is a B-representation for any given bicharacter B. Given one B-representation, Lemma 54 tells us how all other B-representation look like. We can exploit this to characterize all possible quadratic functions, since we know how symmetric bicharacters look (Lemma 53).

The next lemma shows how to construct B-representations canonically.

Lemma 55. *Let be a bicharacter B of G. Consider a symmetric real matrix M such that $B(g,h) = \exp(2\pi i\, g^TMh)$. Then the following function is quadratic and a B-representation:*

$$Q(g) := e^{\pi i\,(g^TMg + C^Tg)}, \tag{3.10.6}$$

where C is an integer vector dependent on M, defined component-wise as $C(i) = M(i,i)c_i$, where c_i denotes the characteristic of the group G_i.

Finally, we arrive at the following important result:

Theorem 56 (Normal form of a quadratic function, short). *Let G be an elementary Abelian group. Then a function $\xi : G \to U(1)$ is quadratic if and only if*

$$\xi(g) = e^{\pi i\,(g^TMg + C^Tg + 2v^Tg)} \tag{3.10.7}$$

where C, v, M are, respectively, two vectors and a matrix that satisfy the following:

- *v is an element of the bullet group $G^\bullet$;*
- *M is the matrix representation of a group homomorphism from G to $G^\bullet$; and*
- *C is an integer vector dependent on M, defined component-wise as $C(i) = M(i,i)c_i$, where c_i is the characteristic of the group G_i.*

Combined with Theorem 51 and Lemma 53, this gives the following normal form theorem:

Theorem 57 (Normal form of a quadratic function, extended).
Let $G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbb{T}^b$ be an elementary Abelian group. Recall $\mathbb{Z}_N^\bullet = \{0, 1/N, \cdots, (N-1)/N\}$ to be a group under addition modulo 1, and $G^\bullet = \mathbb{T}^a \times \mathbb{Z}_{N_1}^\bullet \times \cdots \times \mathbb{Z}_{N_c}^\bullet \times \mathbb{Z}^b$. Then a function $\xi : G \to U(1)$ is quadratic if and only if

$$\xi(g) = e^{\pi i\,(g^TMg + C^Tg + 2v^Tg)} \tag{3.10.8}$$

where C, v, M are, respectively, two vectors and a matrix that satisfy the following:

[22]Precisely, the authors show (see [107, Theorem 1]) that if D_1, D_2 are finite-dimensional irreducible unitary projective representations of a locally compact Abelian group G, possessing the same factor system ω, then there exists a unitary transformation U and a character χ_h such that $U^{-1}D_1(g)U = \chi_h(g)D_2(g)$. In our setup quadratic functions are one-dimensional projective irreps of G and bicharacters are particular examples of factor systems.

- *v is an element of the bullet group $G^\bullet$;*
- *M is the matrix representation of a group homomorphism from G to $G^\bullet$, which necessarily has the form*

$$M := \begin{pmatrix} M_{\mathbb{T}\mathbb{Z}} & M_{\mathbb{T}F} & M_{\mathbb{T}\mathbb{T}} \\ M_{F^\bullet\mathbb{Z}} & M_{F^\bullet F} & 0 \\ M_{\mathbb{Z}\mathbb{Z}} & 0 & 0 \end{pmatrix} \tag{3.10.9}$$

  with the following restrictions:

  - *$M_{\mathbb{Z}\mathbb{Z}}$ and $M_{\mathbb{T}\mathbb{T}}$ are arbitrary integer matrices.*
  - *$M_{F^\bullet\mathbb{Z}}$ and $M_{\mathbb{T}F}$ are rational matrices, the former with the form $M(i,j) = \alpha_{i,j}/N_i$ and the latter with the form $M(i,j) = \alpha_{i,j}/N_j$, where $\alpha_{i,j}$ are arbitrary integers, and N_i is the order of the i-th cyclic subgroup $\mathbb{Z}_{N_i}$.*
  - *$M_{F^\bullet F}$ is a rational matrix with coefficients of the form*

$$M(i,j) = \frac{\alpha_{i,j}}{\gcd(N_i, N_j)} \tag{3.10.10}$$

    where $\alpha_{i,j}$ are arbitrary integers, and N_i is the order of the i-th cyclic subgroup $\mathbb{Z}_{N_i}$.
  - *$M_{\mathbb{T}\mathbb{Z}}$ is an arbitrary real matrix.*

  The entries of $M_{F^\bullet\mathbb{Z}}$, $M_{\mathbb{T}F}$, $M_{F^\bullet F}$, and $M_{\mathbb{T}\mathbb{Z}}$ can be assumed to lie in the interval $[0,1)$. Moreover, M can be assumed to be symmetric, i.e. $M_{\mathbb{Z}\mathbb{Z}}^T = M_{\mathbb{T}\mathbb{T}}$, $M_{F^\bullet\mathbb{Z}}^T = M_{\mathbb{T}F}$, $M_{F^\bullet F}^T = M_{F^\bullet F}$, and $M_{\mathbb{T}\mathbb{Z}}^T = M_{\mathbb{T}\mathbb{Z}}$.

- *C is an integer vector dependent on M, defined component-wise as $C(i) = M(i,i)c_i$, where c_i is the characteristic of the group G_i. (Recall that* $\mathrm{char}(\mathbb{Z}) = 0$, $\mathrm{char}(\mathbb{T}) = 1$, *and* $\mathrm{char}(\mathbb{Z}_N) = N$.*)*

As discussed in the introduction, Theorem 57 may be used to extend part of the so-called "discrete Hudson theorem" of Gross [85], which states that the phases of odd-qudit stabilizer states are quadratic.

The normal form in Theorem 57 can be very useful to perform certain calculations within the space of quadratic functions, as illustrated by the following lemma.

Lemma 58. *Let $\xi_{M,v}$ be the quadratic function (3.10.7) over G. Let A be the matrix representation of a continuous group homomorphism $\alpha : G \to G$. Then the composed function $\xi_{M,v} \circ \alpha$ is also quadratic and can be written in the normal form (3.10.7) as $\xi_{M',v'}$, with*

$$M' := A^T M A, \qquad v' := A^T v + v_{A,M}, \qquad v_{A,M} := A^T C_M - C_{A^T M A}, \tag{3.10.11}$$

where C_M is the vector C associated with M in (3.10.7).

This lemma will be used to prove the main simulation result, Theorem 59 (in the proof of Lemma 70.)

Chapter 4

Classical simulation of normalizer circuits for infinite-dimensional systems

In this chapter we study normalizer circuits over explicitly decomposed infinite Abelian groups, as defined in Chapter 3. We show that such normalizer circuits can be efficiently simulated on a classical computer.

The results of this chapter is joint work with Juan Bermejo-Vega and Martin Van den Nest. This chapter is mostly excerpted from [74].

4.1 Introduction

In the preceding chapter we have introduced and defined the model of normalizer circuits over infinite Abelian groups. We now proceed to prove the first of our two main complexity theoretic results: normalizer circuits over explicitly decomposed groups can be efficiently simulated classically.

To achieve an efficient classical simulation of normalizer circuits over $\mathbb{Z}^a \times \mathbb{T}^b \times F$ (**Theorem 59**), we develop new ***stabilizer formalism*** techniques which extend the stabilizer formalism for finite Abelian groups developed in [1, 2] (which was in turn a generalization of the well-known stabilizer formalism for qubit/qudit systems [17, 16, 76, 108, 109, 110, 111]) . The stabilizer formalism is a paradigm where a quantum state may be described by means of a group of unitary (Pauli) operators (the Pauli stabilizer group) which leave the state invariant. An appealing feature of the stabilizer formalism for finite-dimensional systems is that each stabilizer group is finite and fully determined by specifying a small list of group generators. This list of generators thus forms a concise representation of the corresponding quantum state. Furthermore, if a Clifford gate is applied to the state, the list of generators transforms in a transparent way which can be efficiently updated. Performing such updates throughout the computation yields a stabilizer description of the output state. Finally, one can show that the statistics arising from a final measurement can be efficiently reproduced classically by suitably manipulating the stabilizer generators of the final state.

One major complication arises from the fact that the underlying Abelian group is not finitely generated: when we define Pauli stabilizer groups later in this chapter, we will see that the stabilizer groups are no longer finite nor even finitely generated. This means that the paradigm of representing the group in terms of a small list of generators no longer applies, and a different method needs to be used. We will show how *stabilizer groups can be efficiently represented in terms of certain linear maps*; these maps have concise matrix representations, which will form efficient classical descriptions of the associated stabilizer group.

A crucial ingredient in the last step of our simulation is a polynomial-time classical algorithm that computes the ***support*** of a stabilizer state, given a stabilizer group that describes it. This algorithm exploits a classical reduction of this problem to solving systems of ***mixed real-integer linear equations*** [112], which can be efficiently solved classically. To find this reduction, we make crucial use of the aforementioned normal forms and our infinite-group stabilizer formalism (Section 4.6).

Lastly, we mention a technical issue that arises in the simulation of the final measurement of a normalizer computation: the basis in which the measurement is performed may be *continuous* (stemming again from the fact that G contains factors of $\mathbb{T}$). As a result, precision issues need to be taken into account in the simulation. For this purpose, we develop ***ε-net techniques*** to sample the support of stabilizer states.

Relationship to previous work

In the particular case when G is finite, our results completely generalize the results in [1] and some of the results in [2]. Our normal forms for quadratic functions/stabilizer states improve those in [2]. However, the simulations in [2] are stronger in the finite group case, allowing non-adaptive Pauli measurements in the middle of a normalizer computation. It should be possible to extend the techniques in [2] to our regime, which we leave for future investigation. Normal forms for qudit stabilizer states based on quadratic functions were also given in [109].

Prior to our work, an infinite-dimensional stabilizer formalism best known as "the *continuous variable* (CV) *stabilizer formalism*" was developed for systems that can be described in terms of harmonic oscillators [113, 114, 115, 116, 117, 118], which can be used as "continuous variable" carriers of quantum information. The CV stabilizer formalism is used in the field of continuous-variable quantum information processing [113, 114, 115, 116, 117, 118, 119, 120, 121], being a key ingredient in current schemes for CV quantum error correction [115, 122] and CV measurement-based quantum computation with CV cluster states [123, 124, 125, 122]. A CV version of the Gottesman-Knill theorem [116, 117] for simulations of Gaussian unitaries (acting on Gaussian states) has been derived in this framework.

We stress that, although our infinite-group stabilizer formalism and the CV stabilizer formalism share some similarities, they are physically and mathematically inequivalent and should not be confused with each other. Ours is applied to describe Hilbert spaces of the form $\mathcal{H}_{\mathbb{Z}}^{\otimes a}\otimes\mathcal{H}_{\mathbb{T}}^{\otimes b}\otimes\mathcal{H}_{N_1}\otimes\cdots\otimes\mathcal{H}_{N_c}$ with a basis $|g\rangle$ labeled by the elements of $\mathbb{T}^a\times\mathbb{Z}^b\times\mathbb{Z}_{N_1}\times\cdots\times\mathbb{Z}_{N_c}$: the last c registers correspond to finite-dimensional "discrete variable" systems; the first $a+b$ registers can be thought of infinite-dimensional "**rotating-variable**" systems that are best described in terms of **quantum rotors**[1]. In the CV formalism [116], in contrast, the Hilbert space is $\mathcal{H}_{\mathbb{R}}^{m}$ with a standard basis labeled by $\mathbb{R}^m$ (explicitly constructed as a product basis of position and momentum eigenstates of m harmonic oscillators). Due to these differences, the available families of normalizer gates and Pauli operators in each

[1]The quantum states of a **quantum fixed-axis rigid rotor** (a quantum particle that can move in a circular orbit around a fixed axis) live in a Hilbert space with position and momentum bases labeled by $\mathbb{T}$ and $\mathbb{Z}$: the position is given by a continuous angular coordinate and the angular momentum is quantized in ±1 units (the sign indicates the direction in which the particle rotates [126]).

framework (see sections 3.6, 4.3 and [116]) are simply inequivalent.

Furthermore, dealing with continuous-variable stabilizer groups as in [115, 116, 118] is sometimes simpler, from the simulation point of view, because the group $\mathbb{R}^m$ is also a finite-dimensional **vector space** with a *finite basis*. In our setting, in turn, G is *no longer* a vector space but a **group** that may well be *uncountable* yet having *neither a basis nor a finite generating set*; on top of that, our groups contain *zero divisors* and are not *compact*. These differences require new techniques to track stabilizer groups as they *inherit* all these rich properties. For further reading on these issues we refer to the discussion in [2], where the differences between prime-qudit stabilizer codes [17, 16, 77] (which can described in terms of fields and vector spaces) and stabilizer codes over arbitrary spaces $\mathcal{H}_{d_1} \otimes \cdots \otimes \mathcal{H}_{d_n}$ (which are associated to a finite Abelian group) are explained in detail.

Finally, we mention some related work on the classical simulability of Clifford circuits based on different techniques other than stabilizer groups: see [127] for simulations of qubit non-adaptive Clifford circuits in the Schrödinger picture based on the stabilizer-state normal form of [108]; see [128, 129] for phase-space simulations of odd-dimensional qudit Clifford operation exploiting a local hidden variable theory based on the discrete Wigner function of [130, 85, 131]; lastly, see [132] for phase-space simulations of qubit CSS-preserving Clifford operations based on a Wigner function for rebits.

It should be insightful at this point to discuss briefly whether the latter results may extend to our setup. In this regard, it seems plausible to us that efficient simulation schemes for normalizer circuits analog to those in [127] might exist and may even benefit from the techniques developed in the present work (specifically, our normal forms, as well as those given in [2]). Within certain limitations, it might also be possible to extend the results in [128, 129] and [132] to our setting. We are aware that local hidden variable models for the full-fledged normalizer formalism studied here cannot exist due to the existence of stabilizer-type Bell inequalities [133, 134, 135], which can be violated already within the qubit stabilizer formalism (the $G = \mathbb{Z}_2^n$ normalizer formalism). Consequently, in order to find a hypothetical non-negative quasi-probability representation of normalizer circuits with properties analogous to those of the standard discrete Wigner function of [85, 131], one would necessarily need to specialize to restricted normalizer-circuit models[2] with, e.g., fewer types of gates, input states or measurements; this is part of the approach followed in [132], where the positive Wigner representation for qubit CSS-preserving Clifford elements is given.

Currently, there are no good candidate Wigner functions for extending the results of [128, 129] or [132] to systems of the form $\mathcal{H}_{\mathbb{Z}}^{\otimes a}$: the proposed ones (see [136, 137, 138] and references therein) associate negative Wigner values to Fourier basis states (which are allowed inputs in our formalism and also in [128, 129, 132]) that we introduce in Section 3.5; for one qubit, these are the usual $|+\rangle$, $|-\rangle$ states. The existence of a non-negative Wigner representation for this individual case has not yet been ruled out by Bell inequalities, to the best of our knowledge.

[2]Note that this might not be true for all quasi-probability representations. The locality of the hidden variable models given in [130, 131, 128, 132] comes both from the positivity of the Wigner function and an additional factorizability property (cf. [131] and [132, p.5, Property 4]): in principle, classical simulation approaches that sample non-negative quasi-probability distributions without the factorizability property are well-defined and could also work, even if they do not lead to local hidden variable models.

Discussion and outlook

Finally, we discuss some open questions suggested by our work as well as a few potential avenues for future research.

Due to the presence of Hilbert spaces of the form $\mathcal{H}_{\mathbb{Z}}$, our stabilizer formalism over infinite groups yields a natural framework to study continuous-variable error correcting codes for quantum computing architectures based on superconducting qubits. Consider, for instance, the so-called *0-π qubits* [139, 140]. These are encoded qubits that, in our formalism, can be written as eigenspaces of groups of (commuting) generalized Pauli operators associated to $\mathbb{Z}$ and $\mathbb{T}$ (cf. sections 4.3-4.4 and also the definitions in [139, 140]). Hence, we can interpret them as instances of generalized *stabilizer codes*[3] over the groups $\mathbb{Z}$ and $\mathbb{T}$. We believe that it should be possible to apply the simulation techniques in this chapter (e.g., our generalized Gottesman-Knill theorem) to the study of fault-tolerant quantum-computing schemes that employ this form of generalized stabilizer codes: we remind the reader that the standard Gottesman-Knill theorem [17, 16] is often applied in fault-tolerant schemes for quantum computing with traditional qubits, in order to delay recovery operations and track the evolution of Pauli errors (see, for instance, [141, 142, 143, 144]).

Also in relation with quantum error correction, it would interesting to improve our stabilizer formalism in order to describe *adaptive Pauli measurements*; this would extend the results in [2] for finite Abelian groups.

In connection with previous works, it would be interesting to study normalizer circuits over the group $\mathbb{R}^m$, to understand how they compare to the Gaussian formalism and to analyze in a greater level of detail the full hybrid scenario $\mathcal{H}_{\mathbb{R}}^{\otimes a} \otimes \mathcal{H}_{\mathbb{Z}}^{\otimes b} \otimes \mathcal{H}_{\mathbb{T}}^{\otimes c} \otimes \mathcal{H}_{N_1} \otimes \cdots \otimes \mathcal{H}_{N_d}$. We have left this question to future investigation. However, we highlight that the formalism of normalizer circuits we present can be extended transparently to study this case, by considering additional Hilbert spaces of the form $\mathcal{H}_{\mathbb{R}}^{\otimes m}$ with associated groups $\mathbb{R}^m$; the stabilizer formalism over groups that we develop, our normal forms and simulation techniques can be applied to study that context with little (or zero) modifications.

Another question that remains open after our work is whether normalizer circuits over finite Abelian groups constitute all possible Clifford operations (see a related conjecture in [2], for finite dimensional systems).

Lastly, we mention that an important ingredient underlying the consistency of our normalizer/stabilizer formalism is the fact that the groups associated to the Hilbert space fulfill the so-called **Pontryagin-Van Kampen duality** [97, 98, 86, 99, 100, 101, 102]. From a purely mathematical point of view, it is possible to associate a family of normalizer gates with every group in such class, which accounts for all possible Abelian groups that are locally compact Hausdorff (often called LCA groups). Some LCA groups are notoriously complex objects and remain unclassified to date. Hilbert spaces associated to them can exhibit exotic properties, such as *non-separability*, and may not always be in correspondence with natural quantum mechanical systems. In order to construct a physically relevant model of quantum circuits, we have restricted ourselves to groups of the form $\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$, can be naturally associated to known quantum mechanical systems. We believe that our results can be easily extended to all groups of the form $\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbb{R}^d$, which we call "*elementary*", and form a well-studied class of groups known as "*compactly generated Abelian Lie groups*" [98]. Some examples of LCA groups that are not elementary are the p-adic numbers $\mathbb{Q}_p$ and the adele ring $\mathbb{A}_F$ of an algebraic number field F [86].

[3]A generalized notion of stabilizer code over an Abelian group was introduced in [2].

Outline of this chapter

In Section 4.2 we state the **main result** (Theorem 59). The remaining sections we develop the techniques we use to prove the main result.

In Section 4.3 we study the properties of Pauli operators over Abelian groups. In Section 4.4 we present stabilizer group techniques based on these operators. Section 4.5 is a brief digression for the study systems of linear equations over groups (including systems of mixed real-integer linear equations) and algorithms to solve these. Finally, in 4.6, we combine the techniques we have developed to prove our main result.

4.2 Main result

In our main result (Theorem 59 below) we show that any polynomial-size normalizer circuit (cf. Section 3.6.2 for definitions) associated to any group of the form (3.5.10) can be simulated efficiently classically. Before stating the result, we will rigorously state what it is meant in our work by an efficient classical simulation of a normalizer circuit, in terms of computational complexity.

In short, the **computational problem** we consider is the following: given a classical description of a normalizer quantum circuit, its input quantum state and the measurement performed at the end of the computation (see Section 3.6.2 for details on our computational model), our task is to sample the probability distribution of final measurement outcomes with a classical algorithm. Any classical algorithm to solve this problem in polynomial time (in the bit-size of the input) is said to be *efficient.*

Theorem 59 (Main result). *Let C be any normalizer circuit over any group $G = \mathbb{Z}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$ as defined in Section 3.6.2. Let C act on a input state $|g\rangle$ in the designated standard basis at time zero, and be followed by a final measurement in the designated basis at time T. Then the output probability distribution can be sampled classically efficiently using epsilon-net methods.*

We remind the reader that, in Theorem 59, both standard and Fourier basis states of $\mathcal{H}_{\mathbb{Z}}$ are allowed inputs (cf. Section 3.6.2).

In Theorem 59, the state $|g\rangle$ is described by the group element g, which is encoded as a tuple of m rational[4] numbers of varying size (see Table 3.1, row 1). The memory needed to store the normalizer gates comprising C is summarized in table **??**, row 2. By "*classically efficiently*" it is meant that there exists a classical algorithm (**Theorem 73**) to perform the given task whose worst-time running time scales *polynomially* in the input-size (namely, in the number of subsystems m, the number of normalizer gates of C) and of all other variables listed in the "bits needed" column of table 3.1), and *polylogarithmically* in the parameters $\frac{1}{\varepsilon}$, Δ that specify the number of points in a (Δ, ε)-*net* (read below) and their geometrical arrangement.

Sampling techniques

We finish this section by saying a few words about the (Δ, ϵ)-net methods used in the proof of Theorem 59. These techniques are fully developed in sections 4.6 and 4.6.3.

We shall show later (Lemma 63) that the final quantum state $|\psi\rangle$ generated by a normalizer circuit is always a uniform quantum superposition in any of the computational basis

[4]In this thesis we do not use floating point arithmetic.

we consider (3.5.11): if G is the group associated to our designated basis, and if X is the set of $x \in G$ such that $\psi(x) \neq 0$, then $|\psi(x)| = |\psi(y)|$ for all $x, y \in X$. As a result the final distribution of measurement outcomes is always a flat distribution over some set X.

Moreover, we show in Section 4.6.3 that X is always isomorphic to a group of the form $K \times \mathbb{Z}^{\mathbf{r}}$ where K is compact, and that such isomorphism can be efficiently computed: as a result, we see that, although X is not compact, the non-compact component of X has an Euclidean geometry. Our sampling algorithms are based on this fact: to sample X in an approximate sense, we construct a subset $\mathcal{N}_{\Delta,\varepsilon} \subset X$ of the form

$$\mathcal{N}_{\Delta,\varepsilon} = \mathcal{N}_\varepsilon \oplus \mathcal{P}_\Delta, \tag{4.2.1}$$

where $\mathcal{N}_\varepsilon$ is an ε-net (definition 71) of the compact component K of X and $\mathcal{P}_\Delta$ is a $\mathbf{r}$-dimensional parallelotope contained in the Euclidean component $\mathbb{Z}^{\mathbf{r}}$, centered at 0, with edges of length $2\Delta_1, \ldots, 2\Delta_{\mathbf{r}}$. We call $\mathcal{N}_{\Delta,\varepsilon}$ a (Δ, ε)-net (definition 72). The algorithm in Theorem 59 can efficiently construct and sample such sets for any ε and $\Delta := \Delta_1, \ldots, \Delta_{\mathbf{r}}$ of our choice: its worse running-time is $O(\operatorname{polylog} \frac{1}{\varepsilon}, \operatorname{polylog} \Delta_i)$, as a function of these parameters. We refer the reader to Section 4.6 and Theorem 73 for more details.

Treatment of finite-squeezing errors

It follows from the facts that we have just dicussed that when G is not a compact group (i.e. G contains $\mathbb{Z}$ primitive factors) the support X of the quantum state $|\psi\rangle$ can be an unbounded set. In such case, it follows from the fact that $|\psi\rangle$ is a uniform superposition that the quantum state is *unphysical* and that the physical preparation of such a state requires infinite energy; in the continuous-variable quantum information community, states like $|\psi\rangle$ are often called *infinitely squeezed states* [145]. In a physical implementation, these states can be replaced by physical finitely-squeezed states, whose amplitudes will decay towards the infinite ends[5] of the support set X. This leads to finite-squeezing errors, compared to the ideal scenario.

In this thesis, we consider normalizer circuits to work perfectly in the ideal infinite-squeezing scenario. Our simulation algorithm in Theorem 59 samples the ideal distribution that one would obtain in the infinite precision limit, neglecting the presence of finite-squeezing errors. This is achieved with the (Δ, ϵ)-net methods described above, which we use to discretize and sample the manifold X that supports the ideal output state $|\psi\rangle$; the output of this procedure reveals the information encoded in the wavefunction of the state.

4.3 Pauli operators over Abelian groups

In this section we introduce Pauli operators over groups of the form $G = \mathbb{Z}^a \times \mathbb{T}^b \times F$ (note that we no longer include factors of $\mathbb{R}^d$ because these groups are not related to the Hilbert spaces that we study), discuss some of their basic properties and finally show that normalizer gates map any Pauli operator to another Pauli operator. The latter property is a generalization of a well known property for qubit systems, namely that Clifford operations map the Pauli group to itself.

Note on terminology. Throughout the rest of this thesis, sometimes we use the symbol $\mathcal{H}_{\mathbb{T}}$ as a second name for the Hilbert space $\mathcal{H}_{\mathbb{Z}}$. Whenever this notation is used, we make

[5]The particular form of the damping depends on the implementation. These effects vanish in the limit of infinite squeezing.

implicit that we are working on the Fourier basis of $\mathcal{H}_{\mathbb{Z}}$, which is labeled by the circle group $\mathbb{T}$. Sometimes, this basis will be called the $\mathbb{T}$ standard basis or just $\mathbb{T}$ basis.

4.3.1 Definition and basic properties

Consider an Abelian group of the form $G = \mathbb{Z}^a \times \mathbb{T}^b \times F$ and the associated Hilbert space $\mathcal{H}_G$ with the associated group-element basis $\{|g\rangle : g \in G\}$ as defined in Section 3.5 . We define two types of unitary gates acting on $\mathcal{H}_G$, which we call the *Pauli operators of G*. The first type of Pauli operators are the X-type operators $X_G(g)$ (often called *shift operators* in generalized harmonic analysis):

$$X_G(g)\psi(h) := \psi(h-g), \quad \text{for every } g, h \in G, \tag{4.3.1}$$

where the $\psi(h)$ are the coefficients of some quantum state $|\psi\rangle$ in $\mathcal{H}_G$. These operators can also be written via their action on the standard basis, which yields a more familiar definition:

$$X_G(g)|h\rangle = |g+h\rangle, \quad \text{for every } g, h \in G. \tag{4.3.2}$$

In representation theory, the map $g \to X_G(g)$ is called the *regular representation* of the group G. The second type of Pauli operators are the Z-type operators $Z_G(\mu)$:

$$Z_G(\mu)|g\rangle := \chi_\mu(g)|g\rangle, \quad \text{for every } g \in G,\ \mu \in G^*. \tag{4.3.3}$$

We define a *generalized Pauli operator* of G to be any unitary operator of the form

$$\sigma := \gamma Z_G(\mu) X_G(g) \tag{4.3.4}$$

where γ is a complex number with unit modulus. We will call the duple (μ, g) and the complex number γ, respectively, the *label* and the *phase* of the Pauli operator σ. Furthermore we will regard the label (μ, g) as an element of the Abelian group $G^* \times G$. The above definition of Pauli operators is a generalization of the notion of Pauli operators over finite Abelian groups as considered in [1, 2], which was in turn a generalization of the standard notion of Pauli operators for qubit systems. An important distinction between Pauli operators for finite Abelian groups and the current setting is that the $Z_G(\mu)$ are labeled by $\mu \in G^*$. For finite Abelian groups, we have $G^* = G$ and consequently the Z-type operators are also labeled by elements of G.

Using the definition of Pauli operators, it is straightforward to verify the following commutation relations, which hold for all $g \in G$ and $\mu \in G^*$:

$$\begin{aligned} X_G(g)X_G(h) &= X_G(g+h) = X_G(h)X_G(g) \\ Z_G(\mu)Z_G(\nu) &= Z_G(\mu+\nu) = Z_G(\nu)Z_G(\mu) \\ Z_G(\mu)X_G(g) &= \chi_\mu(g)X_G(g)Z_G(\mu) \end{aligned} \tag{4.3.5}$$

It follows that the set of generalized Pauli operators of G form a group, which we shall call the *Pauli group* of G.

4.3.2 Evolution of Pauli operators

The connection between normalizer gates and the Pauli group is that the former "preserve" the latter under conjugation, as we will show in this section. This property will be a generalization of the well known fact that the Pauli group for n qubits is mapped to itself under the conjugation map $\sigma \to U\sigma U^\dagger$, where U is either a Hadamard gate, CNOT gate or $(\pi/2)$-phase gate[17, 16]—these gates being normalizer gates for the group $G = \mathbb{Z}_2 \times \ldots \mathbb{Z}_2$ [1, 2]. More generally, it was shown in [1] that normalizer gates over any finite Abelian group G map the corresponding Pauli group over G to itself under conjugation. In generalizing the latter result to Abelian groups of the form $G = \mathbb{Z}^a \times \mathbb{T}^b \times F$, we will however note an important distinction. Namely, normalizer gates over G will map Pauli operators over G to Pauli operators over a group G' which is, in general, *different* from the initial group G. This feature is a consequence of the fact that the groups $\mathbb{Z}^a$ and $\mathbb{T}^b$ are no longer autodual (whereas all finite Abelian groups are). Consequently, as we have seen in Section 3.6.1, the QFT over G (or any partial QFT) will change the group that labels the designated basis of $\mathcal{H}$ from G to G'. We will therefore find that the QFT maps Pauli operators over G to Pauli operators over G'. In contrast, such a situation does not occur for automorphism gates and quadratic phase gates, which do not change the group G that labels the designated basis.

Before describing the action of normalizer gates on Pauli operators (Theorem 61), we provide two properties of QFTs.

Lemma 60 (Fourier transforms diagonalize shift operators). *Consider a group of the form $G = \mathbb{Z}^a \times \mathbb{T}^b \times F$. Then the X-type Pauli operators of G and the Z-type operator of G^* are related via the quantum Fourier transform $\mathcal{F}_G$ over G:*

$$Z_{G^*}(g) = \mathcal{F}_G X_G(g) \mathcal{F}_G^\dagger. \tag{4.3.6}$$

Proof. We show this by direct evaluation of the operator $X_G(h)$ on the Fourier basis states (Section 3.5 definition). Using the definitions introduced in Section 3.8.2 we can write the vectors in the Fourier basis of G in terms of character functions (definition 42): letting $|\mu\rangle$ be the state

$$|\mu\rangle = \int_G \mathrm{d}h\, \overline{\chi_\mu(h)} |h\rangle = \int_G \mathrm{d}h\, \chi_{-\mu}(h) |h\rangle, \tag{4.3.7}$$

then the Fourier basis of G is just the set $\{|\mu\rangle,\ \mu \in G^*\}$. Now it is easy to derive

$$\begin{aligned} X_G(g)|\mu\rangle &= X_G(g)\left(\int_G \mathrm{d}h \overline{\chi_\mu(h)}|h\rangle\right) = \int_G \mathrm{d}h \overline{\chi_\mu(h)}|g+h\rangle = \int_G \mathrm{d}h' \overline{\chi_\mu(h'-g)}|h'\rangle \\ &= \overline{\chi_\mu(-g)}\left(\int_G \mathrm{d}h' \chi_\mu(h')|h'\rangle\right) = \chi_g(\mu)|\mu\rangle = Z_{G^*}(g)|\mu\rangle. \end{aligned} \tag{4.3.8}$$

In the derivation we use lemmas 43, 44 and equation (4.3.3) applied to the group G^*. □

The next theorem shows that normalizer gates are generalized Clifford operations, i.e. they transform Pauli operators into Pauli operators under conjugation and, therefore, they *normalize* the group of all Pauli operators within the group of all unitary gates[6].

Theorem 61 (Normalizer gates are Clifford). *Consider a group of the form $G = \mathbb{Z}^a \times \mathbb{T}^b \times F$. Let U be a normalizer gate of the group G. Then U corresponds to an isometry from $\mathcal{H}_G$ to $\mathcal{H}_{G'}$ for some suitable group G', as discussed in section 3.6. Then the conjugation map $\sigma \to U\sigma U^\dagger$ sends Pauli operators of G to Pauli operators of G'. We say that U is a Clifford operator.*

[6]It is usual in quantum information theory to call the normalizer group of the n-qubit Pauli group "the Clifford group" because of a "tenuous relationship" [146, Gottesman] to Clifford algebras.

Proof. We provide an explicit proof for Pauli operators of type $X_G(g)$ and $Z_G(\mu)$. This is enough to prove the lemma due to (4.3.5). As before, $G = G_1 \times \cdots \times G_m$ where the G_i are groups of primitive type.

We break the proof into three cases.

- If U is an automorphism gate $U_\alpha : |h\rangle \rightarrow |\alpha(h)\rangle$ then

$$U_\alpha X_G(g) U_\alpha^\dagger |h\rangle = |\alpha(\alpha^{-1}(h) + g)\rangle = |h + \alpha(g)\rangle = X_G(\alpha(g))|h\rangle, \tag{4.3.9}$$

$$U_\alpha Z_G(\mu) U_\alpha^\dagger |h\rangle = \chi_\mu(\alpha^{-1}(h))|h\rangle = \chi_{\alpha^{*^{-1}}(\mu)}(h)|h\rangle = Z_G\left(\alpha^{*^{-1}}(\mu)\right)|h\rangle. \tag{4.3.10}$$

- If U is a quadratic phase gate D_ξ associated with a quadratic function ξ then

$$\begin{aligned} D_\xi X_G(g) D_\xi^\dagger |h\rangle &= \xi(g+h)\overline{\xi(h)}|g+h\rangle = \xi(g)B(g,h)|g+h\rangle \\ &= \xi(g)\chi_{\beta(g)}(h)|g+h\rangle = \xi(g)X(g)Z(\beta(g))|h\rangle, \end{aligned} \tag{4.3.11}$$

  where, in the second line, we use Lemma 52. Moreover $D_\xi Z_G(\mu) D_\xi^\dagger = Z_G(\mu)$ since diagonal gates commute.

- If U is the Fourier transform F_G on the $\mathcal{H}_G$ then

$$F_G X_G(g) F_G^\dagger = Z_{G^*}(g), \qquad F_G Z_G(\mu) F_G^\dagger = X_{G^*}(-\mu). \tag{4.3.12}$$

  The first identity is the content of Lemma 60. The second is proved in a similar way:

$$\begin{aligned} Z_G(\mu)|\nu\rangle &= Z_G(\mu)\left(\int_G \mathrm{d}h \chi_{-\nu}(h)|h\rangle\right) = \int_G \mathrm{d}h \chi_{-\nu}(h)\chi_\mu(h)|h\rangle = \int_G \mathrm{d}h \chi_{-(\nu-\mu)}(h)|h\rangle \\ &= |\nu - \mu\rangle = X_{G^*}(-\mu)|\nu\rangle, \end{aligned} \tag{4.3.13}$$

  where we apply (4.3.7), Lemma 44 and (4.3.2,4.3.3). These formula also apply to partial Fourier transforms $\mathcal{F}_{G_i}$, since Pauli operators decompose as tensor products.

□

4.4 Stabilizer states

In this section we develop a stabilizer framework to simulate normalizer circuits over infinite Abelian groups of the form $G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$. As explained in Sections 3.1 and 4.1, our techniques generalize methods given in [1, 2] (which apply to groups of the form $F = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$) and is closely related to the (more general) monomial stabilizer formalism [147].

4.4.1 Definition and basic properties

A *stabilizer group* $\mathcal{S}$ over G is any group of *commuting* Pauli operators of G with a nontrivial $+1$ common eigenspace. Here we are interested in stabilizer groups where the $+1$ common eigenspace is one-dimensional, i.e. there exists a state $|\psi\rangle$ such that $\sigma|\psi\rangle = |\psi\rangle$ for all $\sigma \in \mathcal{S}$, and moreover $|\psi\rangle$ is the unique state (up to normalization) with this property. Such states are called stabilizer states (over G). This terminology is an extension of the already established stabilizer formalism for finite-dimensional systems [17, 16, 76, 77, 1, 2].

We stress here that stabilizer states $|\psi\rangle$ are allowed to be unnormalizable states; in other words, we do not require $|\psi\rangle$ to belong to the physical Hilbert space $\mathcal{H}_G$. In a more precise language, stabilizer states may be tempered distributions in the Schwartz-Bruhat space $\mathcal{S}_G^\times$ [91, 92]. This issue arises only when considering infinite groups, i.e. groups containing $\mathbb{Z}$ or $\mathbb{T}$. An example of a non-physical stabilizer state is the Fourier basis state $|p\rangle$ (3.5.3) (we argue below that this is indeed a stabilizer state). Note that not all stabilizer states for $G = \mathbb{T}$ must be unphysical; an example of a physical stabilizer state within $\mathcal{H}_G$ is

$$\int_{\mathbb{T}} dp|p\rangle. \tag{4.4.1}$$

The stabilizer group of this state is $\{X_{\mathbb{T}}(p) : p \in \mathbb{T}\}$, which can be alternatively written as $\{Z_{\mathbb{Z}}(p) : p \in \mathbb{T}\}$ (Lemma 60). Similar examples of stabilizer states within and outside $\mathcal{H}_G$ can be given for $G = \mathbb{Z}$. Note, however, that in this case the standard basis states $|x\rangle$ with $x \in \mathbb{Z}$ (which are again stabilizer states) *do* belong to $\mathcal{H}_{\mathbb{Z}}$.

Next we show that all standard basis states are stabilizer states.

Lemma 62. *Consider $G = \mathbb{Z}^a \times \mathbb{T}^b \times F$ with associated Hilbert space $\mathcal{H}_G$ and standard basis states $\{|g\rangle : g \in G\}$. Then every standard basis state $|g\rangle$ is a stabilizer state. Its stabilizer group is*

$$\{\overline{\chi_\mu}(g) Z_G(\mu) : \mu \in G^*\}. \tag{4.4.2}$$

The lemma implies that the Fourier basis states and, in general, any of the allowed group-element basis states (11) are stabilizer states.

Proof. Let us first prove the theorem for $g = 0$, and show that $|0\rangle$ is the unique state that is stabilized by $\mathcal{S} = \{Z_G(\mu) : \mu \in G^*\}$. It is easy to check that a standard-basis state $|h\rangle$ with $h \in G$ is a common $+1$-eigenstate of $\mathcal{S}$ if and only if $\chi_\mu(h) = 1$ for all $\mu \in G^*$ or, equivalently, iff h belongs to $G^\perp$, the annihilator of G. It is known that $G^\perp$ coincides with the trivial subgroup $\{0\}$ of G^* [98, corollary 20.22], and therefore $|0\rangle$ is the unique standard-basis state that is also a common $+1$ eigenstate of $\mathcal{S}$. Since all unitary operators of $\mathcal{S}$ are diagonal in the standard basis, $|0\rangle$ is the unique common $+1$ eigenstate of $\mathcal{S}$.

For arbitrary $|g\rangle = X_G(g)|0\rangle$, the stabilizer group of $|g\rangle$ is $X_G(g)\mathcal{S}X_G(g)^\dagger$, which equals $\{\overline{\chi_\mu}(g) Z_G(\mu) : \mu \in G^*\}$ (see equation (4.3.5)).

□

Let $|\psi\rangle$ be a stabilizer state with stabilizer group $\mathcal{S}$. We define the following sets, all of which are easily verified to be Abelian groups:

$$\begin{aligned}
\mathbb{L} &:= \{(\mu, g) \in G^* \times G \ : \ \mathcal{S} \text{ contains a Pauli operator of the form } \gamma Z(\mu)X(g)\}; \\
\mathbb{H} &:= \{g \in G \ : \ \mathcal{S} \text{ contains a Pauli operator of the form } \gamma Z(\mu)X(g)\}; \\
\mathbb{D} &:= \{\mu \in G^* \ : \ \mathcal{S} \text{ contains a Pauli operator of the form } \gamma Z(\mu)\}
\end{aligned} \tag{4.4.3}$$

The groups $\mathbb{L}$, $\mathbb{D}$ and $\mathbb{H}$ contain information about the labels of the operators in $\mathcal{S}$. We highlight that, although $\mathbb{D}$ and $\mathbb{H}$ are subsets of very different groups (namely G and G^*, respectively), they are actually closely related to each other by the relation

$$\mathbb{H} \subseteq \mathbb{D}^\perp \quad \text{(or, equivalently, } \mathbb{D} \subseteq \mathbb{H}^\perp\text{)}, \tag{4.4.4}$$

which follows from the commutativity of the elements in $\mathcal{S}$ and the definition of orthogonal complement (recall Section 4.3.1).

Finally, let $\mathcal{D}$ be the subgroup of all diagonal Pauli operators of $\mathcal{S}$. It is easy to see that, by definition, $\mathcal{D}$ and $\mathbb{D}$ are isomorphic to each other.

4.4.2 Support of a stabilizer state

We show that the support of a stabilizer state $|\psi\rangle$ (the manifold of points where the wavefunction $\psi(x)$ is not zero) can be fully characterized in terms of the label groups $\mathbb{H}$, $\mathbb{D}$.

Our next result characterizes the the structure of this wavefunction.

Lemma 63. *Every stabilizer state $|\psi\rangle$ over G is a uniform quantum superposition over some subset of the form $s + \mathbb{H}$, where $\mathbb{H}$ is a subgroup of G. Equivalently, any stabilizer state $|\psi\rangle$ can be writen in the the form*

$$|\psi\rangle = \int_{\mathbb{H}} \mathrm{d}h\, \psi(h)|s+h\rangle \tag{4.4.5}$$

where all amplitudes have equal absolute value $|\psi(h)| = 1$. We call the $s + \mathbb{H}$ the support *of the state.*

This lemma generalizes Corollary 1 in [1].

Proof. Let $|\psi\rangle$ be an arbitrary quantum state $|\psi\rangle = \int_X \mathrm{d}g\, \psi(g)|g\rangle$. The action of an arbitrary Pauli operator $U = \gamma Z_G(\mu) X_G(h) \in \mathcal{S}$ on the state is

$$U|\psi\rangle = \gamma \int_X \mathrm{d}g\, \chi_\mu(g+h)\psi(g)|g+h\rangle = \int_X \mathrm{d}g\, \psi(g)|g\rangle = |\psi\rangle. \tag{4.4.6}$$

Recall the definition of $\mathbb{H}$ in (4.4.3). Comparing the two integrals in (4.4.6), and knowing that $|\chi_\mu(x)| = 1$ for every $x \in G$, we find that the absolute value of ψ cannot change if we shift this function by an element of $\mathbb{H}$; in other words,

$$\textit{for every } g \in X \textit{ it holds } |\psi(g)| = |\psi(g+h)| \textit{ for every } h \in \mathbb{H}. \tag{4.4.7}$$

Now let $Y \subset X$ denote the subset of points $y \in X$ for which $\psi(y) \neq 0$. Eq. (4.4.7) implies that Y is a disjoint union of cosets of $\mathbb{H}$, i.e.

$$Y = \bigcup_{\iota \in I} s_\iota + \mathbb{H}, \tag{4.4.8}$$

where I is a (potentially uncountable) index set, and that $|\psi\rangle$ is of the form

$$|\psi\rangle = \int_Y \mathrm{d}y\, \psi(y)|y\rangle = \int_I \mathrm{d}\iota\, \alpha(\iota)|\phi_\iota\rangle, \tag{4.4.9}$$

where the states $|\phi_\iota\rangle$ are *non-zero* linearly-independent uniform superpositions over the cosets $x_\iota + \mathbb{H}$:

$$|\phi_\iota\rangle = \int_{\mathbb{H}} \mathrm{d}h\, \phi_\iota(h)|s_\iota + h\rangle \tag{4.4.10}$$

and $|\phi_\iota(h)| = 1$ for every h. Putting together (4.4.9) and (4.4.10) we conclude that, for any $U \in \mathcal{S}$, the condition $U|\psi\rangle = |\psi\rangle$ is fulfilled if and only if $U|\phi_\iota\rangle = |\phi_\iota\rangle$ for every $|\phi_\iota\rangle$: this holds because U leaves invariant the mutually-orthogonal vector spaces $\mathcal{V}_\iota := \mathrm{span}\{|s_\iota + h\rangle : h \in \mathbb{H}\}$. Consequently, every state $|\phi_\iota\rangle$ is a (non-zero) common $+1$ eigenstate of all operators in $\mathcal{S}$. Finally, since we know that $|\psi\rangle$ is the *unique* $+1$ common eigenstate of $\mathcal{S}$, it follows from (4.4.9, 4.4.10) that I has exactly one element and $Y = s + \mathbb{H}$; as a result, $|\psi\rangle$ is a uniform superposition of the form (4.4.10). This proves the lemma. □

Lemma 64. *An element $x \in G$ belongs to the support $s + \mathbb{H}$ of a stabilizer state $|\psi\rangle$ if and only if*

$$D|x\rangle = |x\rangle \quad \textit{for all } D \in \mathcal{D}. \tag{4.4.11}$$

Equivalently, using that every D is of the form $D = \gamma_\mu Z_{G^*}(\mu)$ for some $\mu \in \mathbb{D}$ and that γ_μ is determined given μ,

$$\mathrm{supp}(|\psi\rangle) = \{x \in G \ : \ \chi_\mu(x) = \overline{\gamma_\mu} \textit{ for all } \mu \in \mathbb{D}\}. \tag{4.4.12}$$

Lemma 64 was proven for finite groups in [1], partially exploiting the monomial stabilizer formalism (MSF) developed in [147]. Since the MSF framework has not been generalized to finite dimensional Hilbert spaces, the techniques in [147, 1] can no longer be applied in our setting[7]. Our proof works in infinite dimensions and even in the case when the Pauli operators (4.3.1,4.3.3) have unnormalizable eigenstates.

Proof. Write $|\psi\rangle$ as in (4.4.5) integrating over $X := s + \mathbb{H}$. Then, the "if" condition follows easily by evaluating the action of an arbitrary diagonal stabilizer operator $D = \gamma_\mu Z_{G^*}(\mu)$ on a the stabilizer state $|\psi\rangle$: indeed, the condition

$$D|\psi\rangle = |\psi\rangle \quad \Longleftrightarrow \quad \int_X \mathrm{d}x \; (\gamma_\mu \chi_\mu(x))\, \psi(x)|x\rangle = \int_X \mathrm{d}x\, \psi(x)|x\rangle, \tag{4.4.13}$$

holds only if $\gamma_\mu \chi_\mu(x) = 1$, which is equivalent to $D|x\rangle = |x\rangle$ (here, we use implicitly that $\psi(x) \neq 0$ for all integration points).

Now we prove the reverse implication. Take $x \in G$ such that $D|x\rangle = |x\rangle$ for all $D \in \mathcal{D}$. We want to show that $|x\rangle$ belongs to the set $s + \mathbb{H}$. We argue by contradiction, showing that $x \notin s + \mathbb{H}$ implies that there exists a nonzero common $+1$ eigenstate $|\phi\rangle$ of all $\mathcal{S}$ that is not proportional to $|\psi\rangle$, which cannot happen.

We now show how to construct such a $|\phi\rangle$.

Let $Y := \{\xi(\mu, g) Z_{G^*}(\mu) X_G(g)\}$ be a system of representatives of the factor group $\mathcal{S}/\mathcal{D}$. For every $h \in \mathbb{H}$, we use the notation V_h to denote a Pauli operator of the form $\xi(\nu_h, h) Z_{G^*}(\nu_h) X_G(h)$. It is easy to see that the set of all such V_h forms an equivalence class in $\mathcal{S}/\mathcal{D}$, so that there is a one-to-one correspondence between $\mathbb{H}$ and $\mathcal{S}/\mathcal{D}$. Therefore, if to every $h \in \mathbb{H}$ we associate a $U_h \in Y$ (in a unique way), written as $U_h := \xi(\nu_h, h) Z_{G*}(\nu_h) X(h)$, then we have that:

(a) any Pauli operator $V \in \mathcal{S}$ can be written as $V = U_x D$ for some $U_x \in Y$ and $D \in \mathcal{D}$;

(b) $U_g U_h = U_{g+h} D_{g,h}$ for every U_g, $U_h \in Y$ and some $D_{g,h} \in \mathcal{D}$.

With this conventions, we take ϕ to be the state

$$|\phi\rangle := \left(\int_Y \mathrm{d}U\, U\right)|x\rangle = \left(\int_{\mathbb{H}} \mathrm{d}h\, U_h\right)|x\rangle = \int_{\mathbb{H}} \mathrm{d}h\, \xi(\nu_h, h)\chi_{\nu_h}(x+h)|x+h\rangle \mathrm{d}h. \tag{4.4.14}$$

[7] We believe that the MSF formalism in [147] should be easy to extend to infinite dimensional systems if one looks at monomial stabilizer groups with normalizable eigenstates. However, dealing with monomial operators with unnormalizable eigenstates—which can be the case for (4.3.1,4.3.3)—seems to be notoriously harder.

The last equality in (4.4.14) shows that $|\phi\rangle$ is a uniform superposition over $x+\mathbb{H}$. As a result, $|\phi\rangle$ is non-zero. Moreover, $|\phi\rangle$ linearly independent from $|\psi\rangle$ if we assume $x \notin \mathrm{supp}(\psi)$, since this implies that $\mathrm{supp}(\phi) = x + \mathbb{H}$ and $\mathrm{supp}(\psi) = s + \mathbb{H}$ are disjoint. Lastly, we prove that $|\phi\rangle$ is stabilized by all Pauli operators in $\mathcal{S}$. First, for any diagonal stabilizer D we get

$$D|\phi\rangle = D\left(\int_Y \mathrm{d}U\, U\right)|x\rangle = \left(\int_Y \mathrm{d}U\, U\right)D|x\rangle = \left(\int_Y \mathrm{d}U\, U\right)|x\rangle, \tag{4.4.15}$$

due to commutativity and the promise that $D|x\rangle = |x\rangle$. Also, any stabilizer of the form U_x from the set of representatives Y fulfills

$$U_x|\phi\rangle = U_x\left(\int_{\mathbb{H}} \mathrm{d}h\, U_h\right)|x\rangle = \left(\int_{\mathbb{H}} \mathrm{d}h\, U_x U_h\right)|x\rangle = \left(\int_{\mathbb{H}} \mathrm{d}h\, U_{x+h}\right)D_{x,h}|x\rangle \tag{4.4.16}$$

$$= \left(\int_{\mathbb{H}} \mathrm{d}h'\, U_{h'}\right)|x\rangle = |\phi\rangle \tag{4.4.17}$$

Hence, using property (a) above, it follows that any arbitrary stabilizer V stabilizes $|\phi\rangle$ as well. □

Corollary 1. *The sets* $\mathbb{H}$ *and* $\mathrm{supp}(|\psi\rangle) = s + \mathbb{H}$ *are closed.*

Proof. It follows from (4.4.12) that $\mathrm{supp}(|\psi\rangle)$ is of the form $x_0 + \mathbb{D}^\perp$. Putting this together with (4.4.5) in Lemma 63 it follows that $\mathbb{H} = \mathbb{D}^\perp$. Since any annihilator is closed (Lemma 46), $\mathbb{H}$ is closed. Since the group operation of G is a continuous map[8], $s + \mathbb{H}$ is closed too. □

4.5 Systems of linear equations over Abelian groups

We take a brief digression from the theory of stabilizers over Abelian groups, and consider certain systems of linear equations. These systems will be important to us when we try to sample from the support of a stabilizer state.

Let $\alpha : G \to H$ be a continuous group homomorphism between elementary Abelian groups G, H and let A be a rational matrix representation of α. We consider systems of equations of the form

$$\alpha(x) \equiv Ax \equiv b \pmod{H}, \quad \text{where } x \in G, \tag{4.5.1}$$

which we dub *systems of linear equations over (elementary) Abelian groups.* In this section we develop algorithms to find solutions of such systems.

Systems of linear equations over Abelian groups form a large class of problems, containing, as particular instances, standard systems of linear equations over real vectors spaces,

$$\mathbf{Ax} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{R}^{n\times m}, \mathbf{x} \in \mathbb{R}^m, \mathbf{b} \in \mathbb{R}^n, \tag{4.5.2}$$

as well as systems of linear equations over other types of vector spaces, such as $\mathbb{Z}_2^n$, e.g.

$$\mathbf{By} = \mathbf{c}, \quad \mathbf{B} \in \mathbb{Z}_2^{n\times m}, \mathbf{y} \in \mathbb{Z}_2^m, \mathbf{c} \in \mathbb{Z}_2^n. \tag{4.5.3}$$

[8]This is a fundamental property of topological groups. Consult e.g. [98, 86] for details.

In (4.5.2) the matrix $\mathbf{A}$ defines a linear map from $\mathbb{R}^m$ to $\mathbb{R}^n$, i.e. a map that fulfills $\mathbf{A}(a\mathbf{x}+b\mathbf{y}) = \mathbf{A}(a\mathbf{x}) + \mathbf{A}(b\mathbf{y})$, for every $a, b \in \mathbb{R}$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and is, hence, compatible with the vector space operations; analogously, $\mathbf{B}$ in (4.5.3) is a linear map between $\mathbb{Z}_2$ vector spaces.

We dub systems (4.5.1) "linear" to highlight this resemblance. Yet the reader must beware that, in general, the groups G and H in problem (4.5.1) are *not* vector spaces (primitive factors of the form $\mathbb{Z}$ or $\mathbb{Z}_d$, with non-prime d, are rings yet *not* fields; the circle $\mathbb{T}$ is not even a ring, as it lacks a well-defined multiplication operation[9]), and that the map A is a group homomorphism between groups, but *not* a linear map between vector spaces.

Indeed, there are interesting classes of problems that fit in the class (4.5.1) and that are not systems of linear equations over vectors spaces. An example are the systems of linear equations over finite Abelian groups studied in [2]. Another example are systems of mixed real-integer linear equations [112, 148], that we introduce later in this section (equation 4.5.5).

Input of the problem We only consider systems of the form (4.5.1) where the matrix A is *rational.* In other words, we always assume that the group homomorphism α has a rational matrix representation A; the latter is given to us in the input of our problem. Exact integer arithmetic will be used to store the rational coefficients of A; floating point arithmetic will never be needed in our work.

Of course, not all group homomorphisms have rational matrix representations (cf. Theorem 51). However, for the applications we are interested in it is enough to study this subclass.

General solutions of system (4.5.5) Since A is a homomorphism, it follows that the set G_{sol} of all solutions of (4.5.1) is either empty or a coset of the kernel of A:

$$G_{\mathrm{sol}} = x_0 + \ker A \tag{4.5.4}$$

The main purpose of this section is to devise efficient algorithms to solve system (4.5.1) when A, b are given as input, in the following sense: we say that we have *solved* system (4.5.1) if we manage to find a *general solution* of (4.5.1) as defined next.

Definition 65 (General solution of system (4.5.1)). A *general solution* of a system of equations $Ax \equiv b \pmod{H}$ as in (4.5.1) is a tuple $(x_0, \mathcal{E})$ where x_0 is a particular solution of the system and $\mathcal{E}$ is a continuous group homomorphism (given as a matrix representation) from an auxiliary group $\mathcal{X} := \mathbb{R}^\alpha \times \mathbb{Z}^\beta$ into G, whose image $\operatorname{im}\mathcal{E}$ is the kernel of A.

Although it is not straightforward to prove, general solutions of systems of the form (4.5.1) *always* exist. This is shown in Appendix C.1.

4.5.1 Algorithm for finding a general solution of (4.5.1)

Observe that if we know a general-solution $(x_0, \mathcal{E})$ of system (4.5.1) then we can conveniently write the set of all solutions simply as $G_{\mathrm{sol}} = x_0 + \operatorname{im}\mathcal{E}$. This expression suggests us a simple heuristic to sample random elements in G_{sol}— which will be an important step in our proof of our main classical simulation result—based on the following approach:

[9]Recall that $\mathbb{T}$ is a quotient group of $\mathbb{R}$ and that the addition in $\mathbb{T}$ is well-defined group operation between equivalence classes. It is, however, not possible to define a multiplication ab for $a, b \in \mathbb{T}$ operation between equivalence classes: different choices of class representatives yield different results.

(1) Choose a random element $v \in \mathcal{X}$ using some efficient classical procedure. This step should be feasible since this group has a simple structure: it is just the product of a conventional real Euclidean space $\mathbb{R}^a$ and an integer lattice $\mathbb{Z}^b$.

(2) Apply the map $v \rightarrow x_0 + \mathcal{E}(v)$, yielding a probability distribution on G_{sol}.

A main contribution of our work is a deterministic classical algorithm that finds a general solution of any system of the form (4.5.1) in polynomial time. This is the content of the next theorem, which is one of our main technical results.

Theorem 66 (General solution of system (4.5.1)). *Let A, b define a system of linear equations (over elementary Abelian groups) of form (4.5.1), with the group G as solution space and image group H. Let m and n denote the number of direct-product factors of G and H respectively and let c_i, d_j denote the characteristics of G_i and d_j. Then there exist efficient, deterministic, exact classical algorithms to solve the following tasks in worst-case time complexity $O(\mathrm{poly}(m, n, \log \|A\|_{\mathbf{b}}, \log \|b\|_{\mathbf{b}}, \log c_i, \log d_j)$:*

1. *Decide whether system (4.5.1) admits a solution.*

2. *Find a general solution $(x_0, \mathcal{E})$ of (4.5.1).*

A rigorous proof of this theorem is given in Appendix C.2. The main ideas behind it are discussed next.

In short, we show that the problem of finding a general solution of a system of the form (4.5.1) reduces in polynomial time to the problem of finding a general solution of a so-called *system of mixed real-integer linear equations* [112].

$$A'x' + B'y' = c, \quad \text{where } x' \in \mathbb{Z}^a, y' \in \mathbb{R}^b, \tag{4.5.5}$$

where A' and B' are rational matrices and c is a rational vector. Denoting by $\mathbb{R}^{b'}$ the given space in which c lives, we see that, in our notation, $\begin{pmatrix} A & B \end{pmatrix} w = c$, where $w \in \mathbb{Z}^a \times \mathbb{R}^b$ is a particular instance of a system of linear equations over elementary locally compact Abelian groups that are products of $\mathbb{Z}$ and $\mathbb{R}$. Systems (4.5.5) play an important role within the class of problems (4.5.1), since any efficient algorithm to solve the former can be adapted to solve the latter in polynomial time.

The second main idea in the proof of Theorem 66 is to apply an existing (deterministic) algorithm by Bowman and Burdet [112] that computes a general solution to a system of the form (4.5.5). Although Bowman and Burdet did not prove the efficiency of their algorithm in [112], we show in Appendix C.3 that it can be implemented in polynomial-time, completing the proof of the theorem.

4.5.2 Computing inverses of group automorphisms

In Section 3.9.2 we discussed that computing a matrix representation of the inverse α^{-1} of a group automorphism α cannot be done by simply inverting a (given) matrix representation A of α. However, the algorithm given in Theorem 66 can be adapted to applied to solve this problem.

Lemma 67. *Let $\alpha : G \rightarrow G$ be a continuous group automorphism. Given any matrix representation A of α, there exists efficient classical algorithms that compute a matrix representation X of the inverse group automorphism α^{-1}.*

A proof (and an algorithm) is given in Appendix C.4.

4.6 Proof of Theorem 59

In this section we prove our main result (Theorem 59). As anticipated, we divide the proof in three parts. In Section 4.6.1, we show that the evolution of the quantum state during a normalizer computation can be tracked efficiently using **stabilizer groups** (which we introduced in the previous section). In Section 4.6.2 we show how to compute the support of the final quantum state by reducing the problem to solving systems of **linear equations over an Abelian group**, which can be reduced to systems of mixed real-integer linear equations [112] and solved with the classical algorithms presented in Section 4.5. Finally, in Section 4.6.3, we show how to simulate the final measurement of a normalizer computation by developing **net techniques** (based, again, on the algorithms of Section 4.5) to sample the support of the final state.

4.6.1 Tracking normalizer evolutions with stabilizer groups

As in the celebrated Gottesman-Knill theorem [17, 16] and its existing generalizations [77, 109, 111, 1, 2], our approach will be to track the evolution of the system in a stabilizer picture. Since we know that the initial state $|0\rangle$ is a stabilizer state (Lemma 62) and that normalizer gates are Clifford operations (Lemma 68), it follows that the quantum state at every time step of a normalizer computation is a stabilizer state. It is thus tempting to use stabilizer groups of Abelian-group Pauli operators to classically describe the evolution of the system during the computation; this approach was used in [1, 2] to simulate normalizer circuits over finite Abelian groups. (We remind the reader at this point that qubit and qudit Clifford circuits are particular instances of normalizer circuits over finite Abelian groups [2].)

However, complications arise compared to all previous cases where normalizer circuits are associated to a finite group G. We discuss these issues next.

Stabilizer groups are infinitely generated. A common ingredient in all previously known methods to simulate Clifford circuits and normalizer circuits over finite Abelian groups can no longer be used in our setting: traditionally[10], simulation algorithms based on stabilizer groups keep track of a list of (polynomially many) generators of a stabilizer group, which can be updated to reflect the action of Clifford/normalizer gates. In our set-up, this is a *futile approach* because *stabilizer groups over infinite Abelian groups can have an **infinite** number of generators.* Consider for example the state $|0\rangle$ with $G = \mathbb{Z}$, which has a continuous stabilizer group $\{Z_G(p)|p \in \mathbb{T}\}$ (Lemma 62); the group that describes the labels of the Pauli operators is the circle group $\mathbb{T}$, which cannot be generated by a finite number of elements (since it is uncountable).

Fourier transforms change the group G. In previous works [1, 2], the group G associated to a normalizer circuits is a parameter that does not change during the computation. In Section 4.3.2 we discussed that our setting is different, as Fourier transforms can change the group that labels the designated basis (Theorem 61, Eq. 4.3.12; this reflects that groups (3.4.1,3.5.10) are not autodual.

[10]As discussed in section "Relationship to previous work", there are a few simulation methods [127, 128, 129] for Clifford circuits that are not based on stabilizer-groups, but they are more limited than stabilizer-group methods: the Schrödinger-picture simulation in [127] is for non-adaptive qubit Clifford circuits; the Wigner-function simulation in [128, 129] is for odd-dimensional qudit Clifford circuits (cf. also Section 4.1).

In this section we will develop new methods to track the evolution of stabilizer groups, that deal with the issues mentioned above.

From now on, unless stated otherwise, we consider a normalizer circuit $\mathcal{C}$ comprising T gates. The input is the $|0\rangle$ state of a group G, which we denote by $G(0)$ to indicate that this group occurs at time $t = 0$. The stabilizer group of $|0\rangle$ is $\{Z_G(\mu) : \mu \in G(0)^*\}$. The quantum state at any time t during the computation will have the form $|\psi(t)\rangle = \mathcal{C}_t|0\rangle$ where $\mathcal{C}_t$ is the normalizer circuit containing the first t gates of $\mathcal{C}$. This state is a stabilizer state over a group $G(t)$. The stabilizer group of $|\psi(t)\rangle$ is $\mathcal{S}(t) := \{\mathcal{C}_t Z_{G^*}(\mu)\mathcal{C}_t^\dagger,\ \mu \in G(0)^*\}$.

Throughout this section, we always assume that normalizer gates are given in the standard encodings defined in Section 4.2.

Tracking the change of group G

First, we show how to keep track of how the group G that labels the designated basis changes along the computation. Let $G = G_1 \times \cdots \times G_m$ with each G_i of primitive type. Define now the larger group $\Gamma := G^* \times G$. Note that the labels (μ, g) of a Pauli operator $\gamma Z_G(\mu) X_G(g)$ can be regarded as an element of Γ, so that the transformations of these labels in Theorem 61 can be understood as transformations of this group. We show next that the transformations induces on this group by normalizer gates are *continuous group isomorphisms*, that can be stored in terms of matrix representations. This will give us a method to keep track of G and G^* at the same time. Studying the transformation of Γ as a whole (instead of just G) will be useful in the next section, where we consider the evolution of Pauli operators.

First, note that both automorphism gates and quadratic phase gates leave G (and thus Γ) unchanged (Theorem 61). We can keep track of this effect by storing the $2m \times 2m$ identiy matrix I_{2m} (the matrix clearly defines a group automorphism of Γ). Moreover, (4.3.12) shows that Fourier transforms just induce a signed-swap operation on the factors of Γ. We can associate a $2m \times 2m$ matrix S_i to this operation, defined as follows: S_i acts non-trivially (under multiplication) only on the factors G_i^* and G_i; in the subgroup $G_i^* \times G_i$ formed by these factors S_i acts as

$$(\mu(i), g(i)) \in G_i^* \times G_i \quad \longrightarrow \quad (g(i), -\mu(i)) \in G_i \times G_i^*. \tag{4.6.1}$$

By construction, $\Gamma' = S_i\Gamma$. Manifestly, S_i defines a group isomorphism $S_i : \Gamma \to \Gamma'$.

Lastly, let $G(t)$ denote the underlying group at time step t of the computation. Define $\Gamma(t) := G^*(t) \times G(t)$ and let $V_1, \ldots, V_t$ be the matrices associated to the first t gates describing the transformations of Γ. Then, we have $\Gamma(t) = V_t V_{t-1} \cdots V_1 \Gamma(0)$, so that it is enough to store the matrix $V_t V_{t-1} \cdots V_1$ to keep track of the group $\Gamma(t)$.

Tracking Pauli operators

We deal next with the fact that we can no longer store the "generators" of a stabilizer group. We will exploit a crucial mathematical property of our stabilizer groups: for any stabilizer group $\mathcal{S}$ arising along the course of a normalizer circuit, we will show that there always exists a classical description for $\mathcal{S}$ consisting of a triple (Λ, M, v) where Λ and M are real matrices and v is a real vector. If we have $G = \mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$ with $m = a + b + c$, then all elements of the triple (Λ, M, v) will have $O(\operatorname{poly} m)$ entries. As a result, we can use these triples to describe the stabilizer state $|\psi\rangle$ associated to $\mathcal{S}$ efficiently classically. Moreover, we shall show (lemmas 68, 70) that the description (Λ, M, v) can be *efficiently transformed* to track the evolution of $|\psi\rangle$ under the action of a normalizer circuit.

Let $\Gamma(t)$ be the group $G^*(t) \times G(t)$. Recalling the definition of the group $\mathbb{L}$ in (4.4.3), we denote by $\mathbb{L}(t) \subseteq \Gamma(t)$ this group at time t. We want to keep track of this group in a way that does not involve storing an infinite number of generators. As a first step, we consider the initial standard basis state $|0\rangle$, where

$$\mathbb{L}(0) = \{(\mu, 0) : \mu \in G(0)^*\}. \tag{4.6.2}$$

A key observation is that this group can be written as the image of a continuous group homomorphism

$$\Lambda_0 : (\mu, g) \in \Gamma(0) \to (\mu, 0) \in \Gamma(0); \tag{4.6.3}$$

it is easy to verify $\mathbb{L}(0) = \operatorname{im} \Lambda_0$. Therefore, in order to keep track of the (potentially uncountable) set $\mathbb{L}(0)$ it is enough to store a $2m \times 2m$ matrix representation of Λ_0 (which we denote by the same symbol):

$$\Lambda_0 = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \tag{4.6.4}$$

Motivated by this property, we will track the evolution of the group $\mathbb{L}(t)$ of Pauli-operator labels by means of a matrix representation of a group homomorphism: $\Lambda_t : \Gamma(0) \to \Gamma(t)$ whose image is precisely $\mathbb{L}(t)$. The following lemma states that this approach works.

Lemma 68 (Evolution of Pauli labels). *There exists a group homomorphism Λ_t from $\Gamma(0)$ to $\Gamma(t)$ satisfying*

$$\mathbb{L}(t) = \operatorname{im} \Lambda_t. \tag{4.6.5}$$

Moreover, a matrix representation of Λ_t can be computed in classical polynomial time, using $O(\mathrm{poly}(m, t))$ basic arithmetic operations.

Proof. We show this by induction. As discussed above, at $t = 0$ we choose Λ_0 as in (4.6.4). Now, given the homomorphism Λ_t at time t, we show how to compute Λ_{t+1} for every type of normalizer gate. The proof relies heavily on the identities in the proof of Theorem 61. We also note that the equations below are for groups of commuting Pauli operators but they can be readily applied to any single Pauli operator just by considering the stabilizer group it generates.

- Automorphism gate U_α: Let A be a matrix representation of α; then equations (4.3.9)-(4.3.10) imply

$$\Lambda_{t+1} = \begin{pmatrix} A^{*^{-1}} & 0 \\ 0 & A \end{pmatrix} \Lambda_t. \tag{4.6.6}$$

  The matrix $A^{*^{-1}}$ can be computed efficiently due to lemmas 67 and 49.(b).

- Quadratic phase gate D_ξ: suppose that ξ is a B-representation for some bicharacter B (recall Section 3.10). Let M be a matrix representation of the homomorphism β that appears in Lemma 52. Then (4.3.11) implies

$$\Lambda_{t+1} = \begin{pmatrix} I & M \\ 0 & I \end{pmatrix} \Lambda_t. \tag{4.6.7}$$

- Partial Fourier transform $\mathcal{F}_{G_i}$: recalling (4.3.12), we simply have

$$\Lambda(t+1) = S_i \Lambda(t), \tag{4.6.8}$$

with

$$S_i = \left(\begin{array}{ccc|ccc} \mathbf{1} & & & \mathbf{0} & & \\ & 0 & & & 1 & \\ & & \mathbf{1} & & & \mathbf{0} \\ \hline \mathbf{0} & & & \mathbf{1} & & \\ & -1 & & & 0 & \\ & & \mathbf{0} & & & \mathbf{1} \end{array}\right) \tag{4.6.9}$$

where the $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ subblock in S_i corresponds to the i-th entries of G^* and G. □

We now show how the phases of the Pauli operators in $\mathcal{S}(t)$ can be tracked.

Suppose that there exists $(\mu, g) \in \mathbb{L}$ and complex phases γ and β such that both

$$\sigma := \gamma Z(\mu) X(g) \quad \text{and } \tau := \beta Z(\mu) X(g) \tag{4.6.10}$$

belong to $\mathcal{S}$. Then $\sigma^\dagger \tau$ must also belong to $\mathcal{S}$, where $\sigma^\dagger \tau = \overline{\gamma}\beta I$ with I the identity operator. But this implies that $\overline{\gamma}\beta|\psi\rangle = |\psi\rangle$, so that $\gamma = \beta$. This shows that the phase of σ is uniquely determined by the couple $(\mu, g) \in \mathbb{L}$. We may thus define a function $\gamma : \mathbb{L} \to U(1)$ such that

$$\mathcal{S} = \{\gamma(\mu, g) Z(\mu) X(g) : \ (\mu, g) \in \mathbb{L}\}. \tag{4.6.11}$$

Lemma 69. *The function γ is a quadratic function on $\mathbb{L}$.*

Proof. By comparing the phases of two stabilizer operators $\sigma_1 = \gamma(\mu_1, g_1) Z_G(\mu_1) X_G(g_1)$ and $\sigma_2 = \gamma(\mu_2, g_2) Z_G(\mu_2) X_G(g_2)$ to the phase $\gamma((g_1, h_1) + (g_2, h_2))$ of their product operator $\sigma_2\sigma_1$, we obtain

$$\gamma((\mu_1, g_1) + (\mu_2, g_2)) = \gamma(\mu_1, g_1)\gamma(\mu_2, g_2)\overline{\chi_{\mu_2}(g_1)}, \tag{4.6.12}$$

which implies that γ is quadratic. □

Although it does not follow from Lemma 69, in our setting, the quadratic function γ will always be *continuous.* As a result, we can apply the normal form given in Theorem 57 to describe the phases of the Pauli operators of a stabilizer group. Intuitively, γ must be continuous in our setting, since this is the case for the allowed family of input states (Lemma 62) and normalizer gates continuously transform Pauli operators under conjugation; this is rigorously shown using induction in the proof of Theorem 70.

We will use that these phases of Pauli operators are described by quadratic functions on $\mathbb{L}(t)$ (recall Lemma 69). In particular, Theorem 57 shows that every quadratic function can be described by means of an $m \times m$ matrix M and a m-dimensional vector v. For the initial state $|0\rangle$, we simply set both M, v to be zero. The next lemma shows that M, v can be efficiently updated during any normalizer computation.

Lemma 70 (Evolution of Pauli phases). *At every time step t of a normalizer circuit, there exists a $2m \times 2m$ rational matrix M_t and a m-dimensional rational vector v_t such that the quadratic function describing the phases of the Pauli operators in $\mathcal{S}(t)$ is ξ_{M_t,v_t} (as in Theorem 57). Moreover, M_t and v_t can be efficiently computed classically with $O(\mathrm{poly}(m,n))$ basic arithmetic operations.*

Proof. The proof is similar to the proof of Lemma 68. We act by induction. At $t = 0$ we just take M_0 to be the zero matrix and v_0 to be the zero vector. Then, given M_t and v_t at time t, we show how to compute M_{t+1}, v_{t+1}. In the following, we denote by $\mathbf{A}$ the matrix that fulfills $\Lambda_{t+1} = \mathbf{A}\Lambda_t$ in each case of Lemma 68 and write $(\mu', g') = \mathbf{A}(\mu, g)$ for every $(\mu, g) \in \Gamma_t$. Finally, let ξ_t and ξ_{t+1} denote the quadratic phase functions for $\mathcal{S}(t)$ and $\mathcal{S}(t+1)$, respectively.

- **Automorphism gate U_α.** Let A, $A^{*^{-1}}$ be matrix representations of α, $\alpha^{*^{-1}}$. Using (4.3.9, 4.3.10) we have

$$\xi_t(\mu, g) Z_G(\mu) X_G(g) \xrightarrow{U_\alpha} \xi_t(\mu, g) Z_G(\mu') X_G(g') \tag{4.6.13}$$

  with $(\mu', g') = \mathbf{A}(\mu, g)$ and $\mathbf{A} = \begin{pmatrix} A^{*^{-1}} & 0 \\ 0 & A \end{pmatrix}$. The matrix $A^{*^{-1}}$ can be computed using lemmas 67 and 49.(b). The phase $\xi_t(\mu, g)$ of the Pauli operator can be written now as a function ξ_{t+1} of (μ', g') defined as

$$\xi_{t+1}(\mu', g') := \xi_t(\mathbf{A}^{-1}(\mu', g')) = \xi_t(\mu, g). \tag{4.6.14}$$

  The function is manifestly quadratic. By applying Lemma 58 we obtain

$$M_{t+1} = \mathbf{A}^{-T} M_t \mathbf{A}^{-1}, \qquad v_{t+1} = \mathbf{A}^{-T} v_t + v_{\mathbf{A}^{-1}, M_t}, \tag{4.6.15}$$

  where $v_{\mathbf{A}^{-1},M_t}$ is defined as $v_{A,M}$ in Lemma 58.

- **Partial Fourier transform $\mathcal{F}_{G_i}$.** The proof is analogous using that $\mathbf{A} = S_i$. Since the Fourier transform at the register ith exchanges the order of the X and Z Pauli operators acting on the subsystem $\mathcal{H}_{G_i}$ (4.3.12), we locally exchange the operators locally using (4.3.5), gaining an extra phase. Assume for simplicity that $i = 1$ and re-write $G = G_1 \times \cdots \times G_m$ as $G = A \times B$; let $g = (a, b)$ and $\mu = (\alpha, \beta)$. Then $\mathcal{F}_{G_1}$ acts trivially on $\mathcal{H}_{G'}$ and we get

$$\xi_t(\mu, g) Z_{G_1}(\alpha) X_{G_1}(a) \otimes U \xrightarrow{\mathcal{F}_{G_1}+\text{reorder}} \left(\xi_t(\mu, g)\chi_{(\alpha,0)}(a, 0)\right) Z_{G_1^*}(a) X_{G_1^*}(-\alpha) \otimes U.$$

  In general, for arbitrary i, we gain a phase factor $\overline{\chi_{(0,\ldots,\mu(i),\ldots,0)}((0,\ldots,g(i),\ldots,0))}$. Using the change of variables $(\mu', g') = \mathbf{A}(\mu, g) = S_i(\mu, g)$, we define ξ_{t+1} to be function that carries on the accumulated phase of the operator. For arbitrary i we obtain

$$\xi_{t+1}(\mu', g') := \xi_t(\mu, g)\, \chi_{(0,\ldots,\mu(i),\ldots,0)}((0,\ldots,g(i),\ldots,0)). \tag{4.6.16}$$

  The character $\chi_{(0,\ldots,\mu(i),\ldots,0)}((0,\ldots,g(i),\ldots,0))$ can be written as a quadratic function

$\xi_{M_F,v_F}(\mu,g)$ with $v_F=0$ and

$$M_F := \left(\begin{array}{c|c} \mathbf{0} & \begin{matrix} 0 & & \\ & \Upsilon_G(i,i) & \\ & & 0 \end{matrix} \\ \hline \begin{matrix} 0 & & \\ & \Upsilon_G(i,i) & \\ & & 0 \end{matrix} & \mathbf{0} \end{array}\right), \tag{4.6.17}$$

where $\Upsilon_G(i,i)$ is the ith diagonal element of Υ_G (3.8.18). Applying Lemma 58 we obtain

$$M_{t+1} = \mathbf{A}^{-T}(M_t + M_F)\,\mathbf{A}^{-1}, \qquad v_{t+1} = \mathbf{A}^{-T}v_t + v_{\mathbf{A}^{-1},M_t+M_F}. \tag{4.6.18}$$

- **Quadratic phase gate D_ξ.** Let $\xi = \xi_{M_Q,v_Q}$ be the quadratic function implemented by the gate and M_β be the matrix representation of β as in (52). We know from Lemma 53 that $M_Q = \Upsilon_G M_\beta$. Using (4.3.11) and reordering Pauli gates (similarly to the previous case) we get

$$\xi_t(\mu,g)Z_G(\mu)X_G(g) \xrightarrow{D_\xi+\text{reorder}} \left(\xi_t(\mu,g)\xi_{M_Q,v_Q}(g)\overline{\chi_{\beta(g)}(g)}\right) Z_G(\mu+\beta(g))X_G(g)$$

The accumulated phase can be written as a quadratic function $\xi_{M',v'}$ with

$$M' := M_t + \begin{pmatrix} 0 & 0 \\ 0 & M_Q \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & 2M_Q \end{pmatrix}, \qquad v' := v + \begin{pmatrix} 0 \\ v_Q \end{pmatrix} \tag{4.6.19}$$

Using Lemma 58 and $\mathbf{A} = \begin{pmatrix} I & M_\beta \\ 0 & I \end{pmatrix}$ (from the proof of Lemma 68) we arrive at:

$$M_{t+1} = \mathbf{A}^{-T}M'\mathbf{A}^{-1}, \qquad v' = \mathbf{A}^{-T}v' + v'_{\mathbf{A}^{-1},M'} \tag{4.6.20}$$

□

Combining lemmas 68 and 70, we find that the triple (Λ_t, M_t, v_t), which constitutes a classical description of the stabilizer state $|\psi(t)\rangle$, can be efficiently computed for all t. This yields a poly-time algorithm to compute the description (Λ_T, M_T, v_T) of the output state $|\psi_T\rangle$ of the circuit. Henceforth we continue to work with this final state and drop the reference to T throughout. That is, the final state is denoted by $|\psi\rangle$, which is a stabilizer state over G with stabilizer is $\mathcal{S}$. The latter is described by the triple (Λ, M, v), the map from $\Gamma(0)$ to Γ is described by Λ, etc.

4.6.2 Computing the support of the final state

Given the triple (Λ, M, v) describing the final state $|\psi\rangle$ of the computation, we now consider the problem of determining the support of $|\psi\rangle$. Recall that the latter has the form $x + \mathbb{H}$ where the label group $\mathbb{H}$ was defined in (4.4.3) and $x \in G$ is any element satisfying conditions (4.4.11). Since $\mathbb{L} = \Lambda\Gamma(0)$ and Λ is given, a description of $\mathbb{H}$ is readily obtained: the $m \times 2m$ matrix $P = (0\ I)$ is a matrix representation of the homomorphism $(\mu, g) \in \Gamma \to g \in G$. It easily follows that $\mathbb{H} = P\Lambda\Gamma(0)$. Thus the matrix $P\Lambda$ yields an efficient description for $\mathbb{H}$. To compute an x in the support of $|\psi\rangle$, we need to solve the equations (4.4.11). In the case of finite groups G, treated in previous works [1, 2], the approach consisted of first computing a (finite) set of generators $\{D_1, \dots, D_r\}$ of $\mathcal{D}$. Note that $x \in G$ satisfies (4.4.11) if and only if $D_i|x\rangle = |x\rangle$ for all i. This gives rise to a finite number of equations. In [1, 2] it was subsequently showed how such equations can be solved efficiently. In contrast with such a finite group setting, here the group G, and hence also the group $\mathcal{D}$, can be continuous, so that $\mathcal{D}$ can in general not be described by a finite list of generators. Consequently, the approach followed for finite groups does no longer work. Next we provide an alternative approach to compute an x in the support of $|\psi\rangle$ in polynomial time.

Computing $\mathcal{D}$

We want to solve the system of equations (4.4.12). Our approach will be to reduce this problem to a system of linear equations over a group of the form (4.5.1) and apply the algorithm in Theorem 66 to solve it. To compute $\mathcal{D}$ it is enough to find a compact way to represent $\mathbb{D}$, since we can compute the phases of the diagonal operators using the classical description (Λ, M, v) of the stabilizer group. To compute $\mathbb{D}$ we argue as follows. An arbitrary element of $\mathbb{L}$ has the form Λu with $u \in \Gamma(0)$. Write Λ in a block form

$$\Lambda = \begin{pmatrix} \Lambda_1 \\ \Lambda_2 \end{pmatrix} \tag{4.6.21}$$

so that $\Lambda u = (\Lambda_1 u, \Lambda_2 u)$ with $\Lambda_1 u \in G^*$ and $\Lambda_2 u \in G$. Then

$$\mathbb{D} = \{\Lambda_1 u : u \text{ satisfies } \Lambda_2 u \equiv 0 \bmod G.\}$$

The equation $\Lambda_2 u \equiv 0 \bmod G$ is of the form (4.5.1). This means we can compute in polynomial time a description for $\mathbb{D}$ of the form

$$\mathbb{D} = \{\mathcal{E}_{\mathbb{D}} w : \ w \in \mathbb{R}^a \times \mathbb{Z}^b\}, \tag{4.6.22}$$

where $\mathcal{E}_{\mathbb{D}}$ is a group homomorphism $\mathcal{E}_{\mathbb{D}} : \mathbb{R}^a \times \mathbb{Z}^b \to G^*$ whose image is precisely $\mathbb{D}$.

Computing the support $x_0 + \mathbb{H}$

Recalling the support equations (4.4.12) and the fact that $|\psi\rangle$ is described by the triple (Λ, M, v), we find that x_0 belongs to the support of $|\psi\rangle$ if and only if

$$\xi_{M,v}(\mu, 0)\chi_\mu(x_0) = 1, \quad \text{for all } \mu \in \mathbb{D}. \tag{4.6.23}$$

We will now write the elements $\mu \in \mathbb{D}$ in the form $\mu = \mathcal{E}_{\mathbb{D}} w$ where w is an arbitrary element in $\mathbb{R}^a \times \mathbb{Z}^b$. We further denote $\mathcal{E} := \begin{pmatrix} \mathcal{E}_{\mathbb{D}} \\ 0 \end{pmatrix}$. We now realize that

- $\xi_{M,v}(\mathcal{E}_{\mathbb{D}} w, 0)$, as a function of w only, is a quadratic function of $\mathbb{R}^a \times \mathbb{Z}^b$, since $\xi_{M,v}$ is quadratic and $\mathcal{E}_{\mathbb{D}}$ is a homomorphism. Furthermore
$$\xi_{M,v}(\mathcal{E}_{\mathbb{D}} w, 0) = \xi_{M',v'}(w) \qquad \text{with } M' := \mathcal{E}^T M \mathcal{E},\ v' := \mathcal{E}^T v. \tag{4.6.24}$$
- $\chi_{\mathcal{E}_{\mathbb{D}} w}(x_0)$, as a function of w only, is a character function of $\mathbb{R}^a \times \mathbb{Z}^b$ which can be written as χ_{ϖ} with $\varpi := \mathcal{E}_{\mathbb{D}}{}^*(x_0)$.

It follows that x_0 satisfies (4.6.23) if and only if the quadratic function $\xi_{M',v'}$ is a character and coincides with χ_{ϖ}. Using Lemma 53 and Theorem 57, we can write these two conditions equivalently as:

$$w_1^T M' w_2 = 0 \pmod{\mathbb{Z}}, \quad \text{for all } w_1, w_2 \in \mathbb{R}^a \times \mathbb{Z}^b \tag{4.6.25}$$
$$\mathcal{E}_{\mathbb{D}}^*(x_0) = \mathcal{E}^T v \pmod{\mathbb{R}^a \times \mathbb{T}^b}. \tag{4.6.26}$$

The first equation does not depend on x_0 and it must hold by promise: we are guaranteed that the support is not empty, so that the above equations must admit a solution. The second equation is a system of linear equations over groups of the form given in Section 4.5, and it can be solved with the techniques given in that section.

4.6.3 Sampling the support of a state

Back in Section 4.5 we formulated a fairly simple heuristic to sample the solution space of a linear system of equations over elementary Abelian groups (4.5.1,4.5.4) that exploited our ability to compute general solutions of such systems (Theorem 66). Unfortunately, this straightforward method does not yield an efficient algorithm to sample such solution spaces, which would allow us to efficiently simulate classically quantum normalizer circuits. In the first place, the heuristic neglects two delicate mathematical properties of the groups under consideration, namely, that they are *continuous* and *unbounded.* Moreover, the second step of the heuristic involves the transformation of a given probability distribution on a space $\mathcal{X}$ by the application of a non-injective map $\mathcal{E} : \mathcal{X} \to G$; this step is prone to create a wild number of *collisions*, about which the heuristic gives no information.

In this section we will present an *efficient classical algorithm* to sample solution space of systems of linear equations over groups. Our algorithm applies appropriate techniques to tackle the previously mentioned issues. The algorithm relies on a subroutine to construct and sample from a certain type of epsilon net that allows the *collision-free* sampling from a subgroup of an elementary group, where the subgroup is given as the image of a homomorphism. This algorithm is sufficient for our purposes, since the solution-space (4.5.4) of (4.5.1) is exactly $x_0 + \operatorname{im} \mathcal{E}$ for some homomorphism $\mathcal{E}$ and group element x_0.

Input of the problem and assumptions

Again let G be of the form

$$G = \mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \tag{4.6.27}$$

with $m = a+b+c$. We are given a matrix representation $\mathcal{E}$ of a group homomorphism from $\mathbb{R}^\alpha \times \mathbb{Z}^\beta$ to G such that H is the image of $\mathcal{E}$. The matrix $\mathcal{E}$ and the numbers α, β provide a description of the subgroup H.

Throughout the entire section, H is assumed to be **closed** (in the topological sense). The word "subgroup" will be used as a synonym of "closed subgroup". This is enough for our purposes, since the subgroup $\mathbb{H}$ that defines the support of a stabilizer state (and we aim to sample) is always closed (corollary 1).

Norms

There exists a natural notion of 2-norm for every group of the form $G := \mathbb{Z}^a \times \mathbb{R}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbb{T}^d$ analogous to the standard 2-norm $\|\cdot\|_2$ of a real Euclidean space (we denote the group 2-norm simply by $\|\cdot\|_G$): given $g = (g_\mathbb{Z}, g_\mathbb{R}, g_F, g_\mathbb{T}) \in G$,

$$\|g\|_G := \left\|(g_\mathbb{Z}, g_\mathbb{R}, g_F^{\circlearrowleft}, g_\mathbb{T}^{\circlearrowleft})\right\|_2 \tag{4.6.28}$$

where $g_F^{\circlearrowleft}$ (resp. $g_\mathbb{T}^{\circlearrowleft}$) stands for any integer tuple $x \in \mathbb{Z}^a$ (resp. real tuple $y \in \mathbb{R}^d$) that is congruent to g_F (resp. $g_\mathbb{T}$) and has minimal two norm $\|\cdot\|_2$. The reader should note that, although $g_F^{\circlearrowleft}$, $g_\mathbb{T}^{\circlearrowleft}$ may not be uniquely defined, the value of $\|g\|_G$ is always unique.

The following relationship between norms will later be useful:

$$\text{if} \quad \|g\|_2 \le \tfrac{1}{2} \quad \text{then} \quad \|g\|_G = \|g\|_2 \le \tfrac{1}{2}; \tag{4.6.29}$$

or, in other words, if an element $g \in G$ has small $\|\cdot\|_2$ norm as a tuple of real numbers, then its norm $\|g\|_G$ as a group element of G is also small and equal to $\|g\|_2$.

Net techniques

Groups of the form (4.6.27) contain subgroups that are *continuous* and/or *unbounded* as sets. These properties must be taken into account in the design of algorithms to sample subgroups.

We briefly discuss the technical issues—absent from the case of finite G as in [1, 2] —that arise, and present net techniques to tackle them.

The first issue to confront, related to continuity, is the presence of discretization errors due to finite precision limitations, for no realistic algorithm can sample a continuous subgroup H exactly. Instead, we will sample some distinguished discrete subset $\mathcal{N}_\varepsilon$ of H that, informally, "discretizes" H and that can be efficiently represented in a computer. More precisely, we choose $\mathcal{N}_\epsilon$ to be a certain type of ε-net:

Definition 71 (ε-net[11]). An ε-net $\mathcal{N}$ of a subgroup H is a finitely generated subgroup of H such that for every $h \in H$ there exists $n \in \mathcal{N}$ with $\|h - n\|_G \le \varepsilon$.

The second issue in our setting is the unboundedness of certain subgroups of *G by itself.* We must carefully define a notion of sampling for such sets that suits our needs, dealing with the fact that uniform distributions over unbounded sets (like $\mathbb{R}$ or $\mathbb{Z}$) cannot be interpreted as well-defined probability distributions; as a consequence, one cannot simply "sample" $\mathcal{N}$ or H uniformly. However, in order to simulate the distribution of measurement outcomes of a *physical* normalizer quantum computation (where the initial states $|g\rangle$ can only be prepared

[11]Our definition of ε-net is based on the ones used in [149, 150, 151, 152]. We adopt an additional non-standard convention, that $\mathcal{N}$ must be a subgroup, because it is convenient for our purposes.

approximately) it is enough to sample uniformly some bounded compact region of H with finite volume V. We can approach the infinite-precision limit by choosing V to be larger and larger, and in the $V \to \infty$ limit we will approach an exact quantum normalizer computation.

We will slightly modify the definition of ϵ-net so that we can sample H in the sense described above. For this, we need to review some structural properties of the subgroups of groups of the form (4.6.27)

It is known that any arbitrary closed subgroup H of an elementary group G of the form (4.6.27) is isomorphic to an elementary group also of the form (4.6.27) (see [98] theorem 21.19 and proposition 21.13). As a result, any subgroup H is of the form $H = H_{\text{comp}} \oplus H_{\text{free}}$ where H_{comp} is a *compact* Abelian subgroup of H and H_{free} is either the trivial subgroup or an *unbounded* subgroup that does not contain non-zero finite-order elements (it is *torsion-free*, in group theoretical jargon). By the same argument, any ε-net $\mathcal{N}_\varepsilon$ of H decomposes in the same way

$$\mathcal{N}_\varepsilon := \mathcal{N}_{\varepsilon,\text{comp}} \oplus \mathcal{N}_{\varepsilon,\text{free}}. \tag{4.6.30}$$

where $\mathcal{N}_{\varepsilon,\text{comp}}$ is a finite subgroup of H_{comp} and $\mathcal{N}_{\varepsilon,\text{free}}$ is a finitely generated torsion-free subgroup of H_{free}. The fundamental theorem of finitely generated Abelian groups tells us that $\mathcal{N}_{\varepsilon,\text{free}}$ is isomorphic to a group of the form $\mathbb{Z}^{\mathbf{r}}$ (a *lattice* of rank $\mathbf{r}$) and, therefore, it has a *$\mathbb{Z}$-basis* [153]: i.e. a set $\{\mathcal{b}_1, \ldots, \mathcal{b}_{\mathbf{r}}\}$ of elements such that every $n \in \mathcal{N}_{\varepsilon,\text{free}}$ can be written in one and only one way as a linear combination of basis elements with *integer* coefficients:

$$\mathcal{N}_{\varepsilon,\text{free}} = \left\{ n = \sum_{i=1}^{\mathbf{r}} n_i \mathcal{b}_i, \text{ for some } n_i \in \mathbb{Z} \right\}. \tag{4.6.31}$$

In view of equation (4.6.27) we introduce a more general notion of nets that is adequate for sampling this type of set.

Definition 72. Let $\mathcal{N}_\varepsilon$ be an ε-net of H and let $\{\mathcal{b}_1, \ldots, \mathcal{b}_{\mathbf{r}}\}$ be a prescribed basis of $\mathcal{N}_{\varepsilon,\text{free}}$. Then, we call a ***$(\Delta, \varepsilon)$-net*** any finite subset $\mathcal{N}_{\Delta,\varepsilon}$ of $\mathcal{N}_\varepsilon$ of the form

$$\mathcal{N}_{\Delta,\varepsilon} = \mathcal{N}_{\varepsilon,\text{comp}} \oplus \mathcal{P}_\Delta, \tag{4.6.32}$$

where $\mathcal{P}_\Delta$ denotes the parallelotope contained in $\mathcal{N}_{\varepsilon,\text{free}}$ with vertices $\pm\Delta_1\mathcal{b}_1, \ldots, \pm\Delta_{\mathbf{r}}\mathcal{b}_{\mathbf{r}}$,

$$\mathcal{P}_\Delta := \left\{ n = \sum_{i=1}^{\mathbf{r}} n_i \mathcal{b}_i, \text{ where } n_i \in \{0, \pm1, \pm2, \ldots, \pm\Delta_i\} \right\}. \tag{4.6.33}$$

The index of $\mathcal{P}_\Delta$ is a tuple of positive integers $\Delta := (\Delta_1, \ldots, \Delta_{\mathbf{r}})$ that specifies the lengths of the edges of $\mathcal{P}_\Delta$.

Notice that $\mathcal{N}_{\Delta,\varepsilon} \to \mathcal{N}_\varepsilon$ in the limit where the edges Δ_i of $\mathcal{P}_\Delta$ become infinitely long and that the volume covered by $\mathcal{N}_{\Delta,\varepsilon}$ increases monotonically as a function of the edge-lengths. Hence, any algorithm to construct and sample (Δ, ε)-nets of H can be used to sample H in the sense we want. Moreover, the next theorem (a main contribution of our work) states that there exist classical algorithms to sample the subgroup H through (Δ, ε)-nets *efficiently.*

Theorem 73. *Let H be an arbitrary closed subgroup of an elementary group $G = \mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$. Assume we are given a matrix-representation $\mathcal{E}$ of a group homomorphism $\mathcal{E} : \mathbb{R}^\alpha \times \mathbb{Z}^\beta \to G$ such that H is the image of $\mathcal{E}$. Then, there exist classical algorithms to* *sample H through (Δ,ε)-nets using $O(\mathrm{poly}(m,\alpha,\beta,\log N_i,\log\|\mathcal{E}\|_{\mathbf{b}},\log\frac{1}{\varepsilon},\log\Delta_i))$ time and bits of memory.*

Again, $\log\|\mathcal{E}\|_{\mathbf{b}}$ denotes the maximal number of bits needed to store a coefficient of $\mathcal{E}$ as a fraction. The proof is the content of the next section, where we devise a classical algorithm with the advertised properties.

Proof of Theorem 73: an algorithm to sample subgroups

We denote by $\mathcal{E}_{\mathbb{TR}}$ the block of $\mathcal{E}$ with image contained in $\mathbb{T}^a$ and with domain $\mathbb{R}^\alpha$. Define a new set $\mathcal{L} := (\varepsilon_1\mathbb{Z})^\alpha \times \mathbb{Z}^\beta$, which is a subgroup of $\mathbb{R}^a \times \mathbb{Z}^b$, and let $\mathcal{N} := \mathcal{E}(\mathcal{L})$ be the image of $\mathcal{L}$ under the action of the homomorphism $\mathcal{E}$.

In first place, we show that by setting ε_1 to be smaller than $2\varepsilon/(\alpha\sqrt{a}|\mathcal{E}|)$, we can ensure that $\mathcal{N}$ is an ε-net of H for any ε of our choice. We will use that $\mathcal{L}$ is, by definition, a $(\frac{\varepsilon_1\sqrt{\alpha}}{2})$-net of $\mathbb{R}^\alpha \times \mathbb{Z}^\beta$. (This follows from the fact that, for every $x \in \mathbb{R}^\alpha$ there exists $x' \in (\varepsilon_1\mathbb{Z})^\alpha$ such that $|x(i) - x'(i)| \leq \varepsilon_1/2$, so that $\|x - x'\|_2 \leq \varepsilon_1\sqrt{\alpha}/2$). Of course, we must have that $\mathcal{N}$ must be an ε-net of H for some value of ε. To bound this ε we will use the following bound for the operator norm of the matrix $\mathcal{E}_{\mathbb{TR}}$:

$$\|\mathcal{E}_{\mathbb{TR}}\|_{\mathrm{op}}^2 \leq \alpha a|\mathcal{E}_{\mathbb{TR}}|^2 \leq \alpha a|\mathcal{E}|^2. \tag{4.6.34}$$

The first inequality in (4.6.34) follows from Schur's bound on the maximal singular value of a real matrix. This bound implies that, if two elements $\chi := (x,z) \in \mathcal{X}$ and $\chi' := (x',z) \in \mathcal{L}$ are $\varepsilon_1\sqrt{\alpha}/2$-close to each other, then

$$\|\mathcal{E}\chi - \mathcal{E}\chi'\|_2 \leq \|\mathcal{E}_{\mathbb{TR}}\|_{\mathrm{op}}\|x - x'\|_2 \leq \tfrac{1}{2}\alpha\sqrt{a}|\mathcal{E}|\varepsilon_1 \tag{4.6.35}$$

(In the first inequality, we apply the normal form in Theorem 51.) Finally, by imposing $\frac{\alpha\sqrt{a}}{2}|\mathcal{E}|\varepsilon_1 \leq \varepsilon \leq \frac{1}{2}$, we get that $\|\mathcal{E}(\chi - \chi')\|_G \leq \varepsilon$ due to property (4.6.29); it follows that $\mathcal{N}$ is an ε-net if $\varepsilon_1 \leq 2\varepsilon/(\alpha\sqrt{a}|\mathcal{E}|)$ for every $\varepsilon \leq \frac{1}{2}$.

Assuming that ε_1 is chosen so that $\mathcal{N}$ is an ε-net, our next step will be to devise an algorithm to construct and sample an (Δ,ε)-net $\mathcal{N}_\Delta \subset \mathcal{N}$. The key step of our algorithm will be a subroutine that computes a nicely-behaved classical representation of the quotient group $Q = \mathcal{L}/\ker\mathcal{E}$ and a matrix representation of the group isomorphism $\mathcal{E}_{\mathrm{iso}} : Q \to \mathcal{N}$ (we know that these groups are isomorphic due to the first isomorphism theorem). We will use the computed representation of Q to construct a (Δ,ε)-net $Q_\Delta \subset Q$ and sample elements form it; then, by applying the map $\mathcal{E}_{\mathrm{iso}}$ to the sampled elements, we will effectively sample a (Δ,ε)-net $\mathcal{N}_\Delta \subset \mathcal{N}$; and, moreover, in a clean *collision free* fashion.

To simplify conceptually certain calculations to come, we will change some notation. Note that $\mathcal{L}$ is isomorphic to the group $\mathcal{L}' := \mathbb{Z}^{\alpha+\beta}$ and that $\varepsilon_1 I_\alpha \oplus I_\beta$ is a matrix representation of this isomorphism. We will work with the group $\mathcal{L}'$ instead of $\mathcal{L}$; accordingly, we will substitute $\mathcal{E}$ with the map $\mathcal{E}' := \mathcal{E}(\varepsilon_1 I_\alpha \oplus I_\beta)$ and Q with $Q' := \mathcal{L}'/\ker\mathcal{E}'$.

Our subroutine to compute a representation of Q' begins by applying algorithm in Theorem 66 to obtain a $(\alpha+\beta)\times\gamma$ matrix representation A of a group homomorphism $A : \mathbb{Z}^\gamma \to \mathcal{L}'$ such that $\mathrm{im}\,A = \ker\mathcal{E}'$ (where $\gamma = \alpha + \beta + m$). (Theorem 51 ensures that real factors do not appear in the domain of A because there are no non-trivial continuous group homomorphisms from products of $\mathbb{R}$ into products of $\mathbb{Z}$.) We can represent these maps in a diagram:

$$\mathbb{Z}^\gamma \xrightarrow{A} \mathbb{Z}^{\alpha+\beta} \xrightarrow{\mathcal{E}'} \mathcal{N} \tag{4.6.36}$$

The worst-case time complexity needed to compute A with the algorithm in theorem 66 is polynomial in the variables m, α, β, $\log N_i$, $\log \|\mathcal{E}\|_{\mathbf{b}}$, and $\log \frac{1}{\varepsilon}$.

Next, we compute two integer unimodular matrices U, V such that $A = USV$ and S is in Smith normal form (SNF). This can be done, again, in $\mathrm{poly}(m, \alpha, \beta, \log N_i, \mathrm{size}(\mathcal{E}), \log \frac{1}{\varepsilon})$ time with existing algorithms to compute the SNF of an integer matrix (see e.g. [154] for a review). Each matrix V, S, U is the matrix of representation of some new group homomorphism, as illustrated in the following diagram.

$$\begin{tikzcd}
\mathbb{Z}^{\gamma} \arrow[r, "A"] \arrow[d, "V"] & \mathbb{Z}^{\alpha+\beta} \arrow[r, "\mathcal{E}'"] & \mathcal{N} \\
\mathbb{Z}^{\gamma} \arrow[r, "S"] & \mathbb{Z}^{\alpha+\beta} \arrow[u, "U"] &
\end{tikzcd} \tag{4.6.37}$$

Since V, U are invertible integer matrices the maps $V : \mathbb{Z}^{\alpha} \to \mathbb{Z}^{\alpha}$ and $U : \mathbb{Z}^{\alpha+\beta} \to \mathbb{Z}^{\alpha+\beta}$ are continuous group isomorphisms and, hence, have trivial kernels. As a result, $\operatorname{im} S = \operatorname{im} U^{-1}AV^{-1} = \operatorname{im} U^{-1}A = U^{-1}(\operatorname{im} A) = U^{-1}(\ker \mathcal{E}')$, which shows that $\ker \mathcal{E}'$ is isomorphic to $\operatorname{im} S$ via the isomorphism U^{-1}. These facts together with Lemma 49.(a) show that $\mathcal{E}_{\text{iso}} := \mathcal{E}'U$ is a matrix representation of a *group isomorphism* from the group $Q' := \mathcal{L}'/\operatorname{im} S$ into $\mathcal{N}$.

Finally, we show that Q' can be written explicitly as a direct product of primitive groups of type $\mathbb{Z}$ and $\mathbb{Z}_d$, thereby computing a finite set of generators of Q' that we can immediately use to construct (Δ, ε)-nets $\mathcal{N}_{\Delta}$. We make crucial use of the fact that S is Smith normal form, i.e.

$$S = \left(\begin{array}{cccc|c} s_1 & & & & \\ & s_2 & & & \\ & & \ddots & & 0 \\ & & & s_{(\alpha+\beta)} & \end{array}\right) = \left(\begin{array}{cccc|c} I_{\mathbf{a}} & & & & \\ & \sigma_1 & & & \\ & & \ddots & & 0 \\ & & & \sigma_{\mathbf{b}} & \\ & & & & 0 \end{array}\right), \tag{4.6.38}$$

where the coefficients σ_i are strictly positive. It follows readily that $\operatorname{im} S = \mathbb{Z}^{\mathbf{a}} \times \sigma_1\mathbb{Z} \times \cdots \sigma_{\mathbf{b}}\mathbb{Z} \times \{0\}^{\mathbf{c}}$, and therefore

$$Q' = \mathbb{Z}^{a+b}/\operatorname{im} S = \{0\}^{\mathbf{a}} \times \mathbb{Z}_{\sigma_1} \times \cdots \times \mathbb{Z}_{\sigma_{\mathbf{b}}} \times \mathbb{Z}^{\mathbf{c}}. \tag{4.6.39}$$

As Q' and $\mathcal{N}$ are *isomorphic* the columns of the matrix $\mathcal{E}_{\text{iso}} = \mathcal{E}'U$ form a generating set of $\mathcal{N}$. Moreover, since $\mathcal{E}_{\text{iso}}$ acts isomorphically on (4.6.39), the subgroup $\mathcal{N}$ must be a direct sum of cyclic subgroups generated by the columns of $\mathcal{E}_{\text{iso}}$:

$$\mathcal{N} = \langle f_1 \rangle \oplus \cdots \oplus \langle f_{\mathbf{b}} \rangle \oplus \langle b_1 \rangle \oplus \cdots \oplus \langle b_{\mathbf{c}} \rangle, \tag{4.6.40}$$

where f_i, b_j stand for the $(\mathbf{a} + i)$th and the $(\mathbf{b} + j)$th column of $\mathcal{E}_{\text{iso}}$. Equation (4.6.39) also tells us that the f_is must generate the compact subgroup $\mathcal{N}_{\text{comp}}$ and that the b_j form a $\mathbb{Z}$-basis of $\mathcal{N}_{\text{free}}$.

The last two observations yield an efficient straightforward method to construct and sample (Δ, ε)-nets within $\mathcal{N}$. First, set $\{f_i\}$ (resp. $\{b_j\}$) to be the default generating-set (resp. default basis) of $\mathcal{N}_{\text{comp}}$ and $\mathcal{N}_{\text{free}}$; then, select a parallelotope $\mathcal{P}_{\Delta}$ of the form (4.6.33) with some desired $\Delta = (\Delta_1, \ldots, \Delta)$. This procedures specifies a net $\mathcal{N}_\Delta = \mathcal{N}_{\text{comp}} \oplus \mathcal{P}_\Delta$ that can be efficiently represented with $O(\text{poly}(m, \alpha, \beta, \log N_i, \log \|\mathcal{E}\|_{\mathbf{b}}, \log \frac{1}{\varepsilon} \log \Delta_i))$ bits of memory (by keeping track of the generating-sets of $\mathcal{N}$ and the numbers Δ_i). Moreover, we can efficiently sample $\mathcal{N}_\Delta$ uniformly and collision-freely by generating random strings of the form

$$\sum_{i=1}^{\mathbf{b}} \chi_i f_i + \sum_{j=1}^{\mathbf{c}} y_i b_j, \tag{4.6.41}$$

where $\chi_i \in \mathbb{Z}_{\sigma_i}$ and $y_j \in \{0, \pm 1, \ldots, \pm \Delta_j\}$.

Chapter 5

The computational power of normalizer circuits over black-box groups

In this chapter we study normalizer circuits over black box Abelian groups, as defined in Chapter 3. We show that such normalizer circuits contain many important quantum algorithms, including Shor's factoring. Moreover, we show that the extended group decomposition problem is *complete* for the associated complexity class, meaning that all normalizer circuits over black box Abelian groups can be classically simulated given an oracle to solve the decomposition problem.

The results of this chapter is joint work with Juan Bermejo-Vega and Martin Van den Nest. This chapter is mostly excerpted from [75].

5.1 Introduction

We will study normalizer circuits over black box Abelian groups in this chapter. The difference between this setting and that of the preceding chapter is that the underlying group need not be explicitly decomposed into primitive subgroups. This is a subtle yet tremendously important difference: although such a decomposition *always* exists for any finite Abelian group [87], finding just one is regarded as a *hard computational problem*; indeed, it is provably at least as hard as *factoring*[1]. Our ***motivation*** to adopt the notion of black-box group is to study Abelian groups for which the group multiplication can be performed in classical polynomial-time while no efficient classical algorithm to decompose them is known. A key example is $\mathbb{Z}_N^\times$, the multiplicative group of integers modulofootnote 1, which plays an important role in Shor's factoring algorithm [14]. With some abuse of notation, we call any such group also a "black-box group"[2].

[1]Knowing $\mathbf{B} \cong \mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_m}$ implies that the order of the group $|G| = d_1 d_2 \cdots d_m$. Hardness results for computing orders [83, 82] imply that the problem is provably hard for classical computers in the black-box setting. For groups $\mathbb{Z}_N^\times$, computing $\varphi(N) := |\mathbb{Z}_N^\times|$ (the Euler totient function) is equivalent to factoring [88].

[2]It will always be clear from context whether the group multiplication is performed by an oracle at unit cost or by some well-known polynomial-time classical algorithm. Most results will be stated in the black-box setting though.

Statement of results

This chapter focuses on understanding the potential uses and limitations of black-box normalizer circuits. Our results (listed below) give a precise characterization of their **computational power**. On one hand, we show that several famous quantum algorithms, including shor's celebrated *factoring algorithm*, can be implemented with black-box normalizer circuits. On the other hand, we apply our former simulation results in Chapter 4 to set upper limits to the class of problems that these circuits can solve, as well as to draw practical implications for quantum algorithm design.

Our main results are now summarized:

1. **Quantum algorithms.** We show that many of the best known quantum algorithms are particular instances of normalizer circuits over black-box groups, including Shor's celebrated factoring and discrete-log algorithms; it follows that black-box normalizer circuits can achieve ***exponential quantum speed-ups***. Namely, the following algorithms are examples of black-box normalizer circuits.

    - **Discrete logarithm.** Shor's discrete-log quantum algorithm [14] is a normalizer circuit over $\mathbb{Z}_{p-1}^2 \times \mathbb{Z}_p^\times$ (Theorem 74, Section 5.2.1).
    - **Factoring.** We show that a hybrid infinite-finite dimensional version of Shor's factoring algorithm [14] can be implemented with normalizer circuit over $\mathbb{Z} \times \mathbb{Z}_N^\times$. We prove that there is a close relationship between *Shor's original algorithm* and our version: Shor's can be understood as a discretized qubit implementation of ours (theorems 76, 79). We also discuss that the ***infinite group*** $\mathbb{Z}$ plays a key role in our "infinite Shor's algorithm", by showing that it is impossible to implement Shor's modular-exponentiation gate efficiently, even approximately, with finite-dimensional normalizer circuits (Theorem 80). Last, we further *conjecture* that only normalizer circuits over infinite groups can factorize (Conjecture 81).
    - **Elliptic curves.** The generalized Shor's algorithm for computing discrete logarithms over an elliptic curve [155, 104, 156] can be implemented with black-box normalizer circuits (Section 5.2.3); in this case, the black-box group is the group of integral points E of the elliptic curve instead of $\mathbb{Z}_p^\times$.
    - **Group decomposition.** Cheung-Mosca's algorithm for decomposing black-box finite Abelian groups [84, 73] is a combination of several types of black-box normalizer circuits. In fact, we discuss a new *extended Cheung-Mosca's algorithm* that finds even more information about the structure of the group and it is also based on normalizer circuits (Section 5.2.5).
    - **Hidden subgroup problem.** Deutsch's [157], Simon's [158] and, in fact, all quantum algorithms that solve Abelian hidden subgroup problems [159, 160, 20, 161, 78, 79, 80, 81], are normalizer circuits over groups of the form $G \times \mathcal{O}$, where G is the group that contains the hidden subgroup H and $\mathcal{O}$ is a group isomorphic to G/H (Section 5.2.4). The group $\mathcal{O}$, however, is not a black-box group due to a small technical difference between our oracle model we use and the oracle setting in the HSP.
    - **Hidden kernel problem.** The group $\mathcal{O} \cong G/H$ in the previous section becomes a black-box group if the oracle function in the HSP is a homomorphism between black-box groups: we call this subcase the *hidden kernel problem* (HKP). The

difference does not seem to be very significant, and can be eliminated by choosing different oracle models (Section 5.2.4). However, we will never refer to Simon's or to general Abelian HSP algorithms as "black-box normalizer circuits", in order to be consistent with our and pre-existing terminology.

Note that it follows from the above that black-box normalizer circuits can render insecure widespread public-key cryptosystems, namely Diffie-Hellman key-exchange [162], RSA [15] and elliptic curve cryptography [163, 164].

2. **Group decomposition is *as hard as* simulating normalizer circuits.** Another main contribution of this chapter is to show that the group decomposition problem (suitably formalized) is, in fact, ***complete*** for the complexity class **Black-Box Normalizer**, of problems efficiently solvable by probabilistic classical computers with oracular access to black-box normalizer circuits. Since normalizer circuits over decomposed groups are efficiently classically simulable ([1, 2], and Chapter 4 of this thesis), this result suggests that the computational power of normalizer circuits originate *precisely* in the classical hardness of learning the structure of a black-box group.

   We obtain this last result by proving a significantly ***stronger theorem*** (Theorem 87), which states that any black-box normalizer circuit can be efficiently simulated *step by step* by a classical computer if an efficient subroutine for decomposing finite Abelian groups is provided.

3. **A no-go theorem for new quantum algorithms.** In this chapter, we provide a negative answer to the question "*can new quantum algorithms based on normalizer circuits be found?*": by applying the latter simulation result, we conclude that any new algorithm not in our list can be efficiently simulated step-by-step using the extended Cheung-Mosca algorithm and classical post-processing. This implies (Theorem 89) that new *exponential* speed-ups cannot be found without changing our setting (we discuss how the setting might be changed in the discussion 5.1). This result says nothing about polynomial speed-ups.

4. **Universality of short normalizer circuits.** A practical consequence of our no-go theorem is that all problems in the class **Black Box Normalizer** can be solved using short normalizer circuits with a *constant* number of normalizer gates. (We may still need polynomially many runs of such circuits, along with classical processing in between, but each individual normalizer circuit is short.) We find this observation interesting, in that it explains a very curious feature present in all the quantum algorithms that we study [14, 155, 104, 156, 84, 73, 157, 158, 159, 160, 20, 161, 78, 79, 80, 81] (Section 5.2): they all contain at most a constant number of *quantum Fourier transforms* (actually at most two).

5. **Other complete problems.** As our last contribution in this series, we identify another two complete problems for the class **Black Box Normalizer** (Section 5.5): these are the (aforementioned) Abelian *hidden kernel problem*, and the problem of finding a general-solution to a *system of linear equations over black-box groups* (the latter are related to the systems of linear equations over groups studied in Section 4.5.

A link between Clifford circuits and Shor's algorithm

The results in this chapter together with those previously obtained in [1, 2] and Chapter 4 of this thesis demonstrate the existence of a precise connection between Clifford circuits and Shor's factoring algorithm. At first glance, it might be hard to digest that two types of quantum circuits that seem to be so far away from each other might be related at all. Indeed, classically simulating Shor's algorithm is widely believed to be an intractable problem (at least as hard as factoring), while a zoo of classical techniques and efficient classical algorithms exist for simulating and computing properties of Clifford circuits [17, 16, 76, 77, 108, 110, 109, 165, 127, 111, 166]. However, we will see that these circuits are just different types of normalizer circuits. In other words, they are both *members of a common family of quantum operations.*

Remarkably, this correspondence between Clifford and Shor, rather than being just a mere mathematical curiosity, also has some sensible consequences for the theory of quantum computing. One that follows from Theorem 87, our simulation result, is that all algorithms studied in this chapter (Shor's factoring and discrete-log algorithms, Cheung-Mosca's, etc.) have a *rich hidden structure* which enables simulating them classically with a stabilizer picture approach "à la Gottesman-Knill" [17, 16]. This structure lets us track the evolution of the quantum state of the computation *step by step* with a very special algorithm, which, despite being inefficient, exploits *completely different* algorithmic principles than the naive brute-force approach: i.e., writing down the coefficients of the initial quantum state and tracking their quantum mechanical evolution through the gates of the circuit[3]. Although the stabilizer-picture simulation is *inefficient* when black-box groups are present (i.e., it does not yield an efficient classical algorithm for simulating Shor's algorithm), the mere existence of such an algorithm reveals how much mathematical structure these quantum algorithms have in common with Clifford and normalizer circuits.

In retrospect, and from an applied point of view, it is also rather satisfactory that one can gracefully exploit the above connection to draw practical implications for quantum algorithm design: in our work, we have actively used our knowledge of the hidden "Clifford-ish" mathematical features of the Abelian hidden subgroup problem algorithms in deriving results 2, 3, 4 and 5 (in the list given in the previous section).

Finally, we regard it a memorable curiosity that replacing decomposed groups with black-box groups not only renders the simulation methods in Chapter 4 inefficient (this is, in fact, something to be expected, due to the existence of hard computational problems related to black-box groups), but also suddenly bridges the gap between Clifford/normalizer circuits and important quantum algorithms such as Shor's and Simon's algorithms.

[3]Note that throughout this manuscript we always work at a high level of abstraction (algorithmically speaking), and that the "steps" in a normalizer-based quantum algorithm are always counted at the logic level of normalizer gates, disregarding smaller gates needed to implement them. In spite of this, we find the above simulability property of black-box normalizer circuits to be truly fascinating. To get a better grasp of its significance, we may perform the following thought experiment. Imagine, we would repeatedly concatenate black-box normalizer circuits in some intentionally complex geometric arrangement, in order to form a gargantuan, intricate "Shor's algorithm" of monstrous size. Even in this case, our simulation result states that if we can decompose Abelian groups (say, with an oracle), then we can efficiently simulate the evolution of the circuit, normalizer-gate after normalizer-gate, independently of the number of Fourier transforms, automorphism and quadratic-phase gates involved in the computation (the overhead of the classical simulation is always at most polynomial in the input-size).

Relationship to previous work

Up to our best knowledge, neither normalizer circuits over black-box groups, nor their relationship with Shor's algorithm or the Abelian hidden subgroup problem, have been previously investigated. Normalizer circuits over explicitly-decomposed finite groups $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_a}$ were studied in [1, 2], while an extension of the formalism to infinite groups of the form $\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_a}$ is given in Chapter 4.

Clifford circuits over qubits and qudits (which can be understood as normalizer circuits over groups of the form $\mathbb{Z}_2^m$ and $\mathbb{Z}_d^m$) have been extensively investigated in the literature [17, 16, 76, 77, 108, 110, 109, 111, 127, 166]. Certain generalizations of Clifford circuits that are not normalizer circuits have also been studied: [110, 167, 168, 127, 166] consider Clifford circuits supplemented with some non-Clifford ingredients; a different form of Clifford circuits based on projective normalizers of unitary groups were investigated in [169].

The hidden subgroup problem (HSP) has played a central role in the history of quantum algorithms and has been extensively studied before our work. The Abelian HSP (see Section 1.3.3), which is also a central subject of this chapter, is related to most of the best known quantum algorithms that were found in the early days of the field [157, 158, 159, 160, 20, 161, 78, 79, 80, 81]. Its best-known generalization, the non-Abelian HSP, has also been heavily investigated due to its relationship to the graph isomorphism problem and certain shortest-vector-lattice problems [25, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187] (see also the reviews [21, 188, 189] and references therein).

The notion of black-box group, which is a key concept in our setting, was first considered by Babai and Szméredi in [83] and have since been extensively studied in classical complexity theory [190, 191, 192, 193, 82]. In general, black-box groups may not be Abelian and do not need to have uniquely represented elements [83]; in the present work, we only consider Abelian uniquely-encoded black-box groups.

In quantum computing, black-box groups were previously investigated in the context of quantum algorithms, both in the Abelian [84, 73, 194] and the non-Abelian group setting [195, 175, 183, 196, 197, 182, 198, 199]. Except for a few exceptions (cf. [195, 194]) most quantum results have been obtained for uniquely-encoded black-box groups.

Aside from generalizations of Clifford circuits [1, 2, 17, 16, 76, 77, 108, 110, 109, 165, 127, 111, 166] (which includes normalizer circuits), many other classes of restricted quantum circuits have been studied in the literature. Some examples (by no means meant to be an exhaustive list) are nearest-neighbor matchgate circuits [200, 201, 202, 168, 203, 204, 205, 206, 207], the one-clean qubit model [208, 209, 210, 211, 212, 213, 214, 215], circuit models based on Gaussian or linear-optical operations [119, 116, 117, 216, 128, 129, 217], commuting circuits [218, 219, 220, 151], low-entangling[4] circuits [222, 223] , low-depth circuits [224, 225], tree-like circuits [225, 226, 227, 228, 229], low-interference circuits [230, 231] and a few others [232, 233].

Discussion and outlook

We finish our introduction by discussing a few potential avenues for finding new quantum algorithms as well as some open questions suggested by our work.

In this chapter, we provide a strict no-go theorem for finding new quantum algorithms with black-box normalizer circuits, as we define them. There are, however, a few possible

[4]Here entanglement is measured with respect to the Schmidt-rank measure (low-entangling circuits with respect to continuous entanglement measures are universal for quantum computation [221]).

ways to modify our setting leading to scenarios where one could bypass these results and, indeed, find new interesting quantum algorithms. We now discuss some.

One enticing possibility would be to study possible extensions of the normalizer circuit framework to non-Abelian groups, in connection with non-Abelian hidden subgroup problems [25, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187]. We have not addressed this question in the present work. In this direction, the classical simulability of non-Abelian quantum Fourier transforms was studied in [234].

A second possibility would be to consider more general types of normalizer circuits than ours, by ***extending the class of Abelian groups*** they can be associated with. However, looking at more general *decomposed* groups does not look particularly promising: we believe that the methods of this thesis can be extended, e.g., to simulate normalizer circuits over groups of the form $\mathbb{R}^a \times \mathbb{Z}^b \times \mathbb{T}^c \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_d} \times \mathbf{B}$, with additional $\mathbb{R}$ factors (cf. our discussion in Section 3.9). On the other hand, allowing more general types of groups to act as *black-boxes* looks rather promising to us: one may, for instance, attempt to extend the notion of normalizer circuits to act on Hilbert spaces associated with multi-dimensional infrastructures [235, 236], which may, informally, be understood as "infinite black-box groups"[5] We expect, in fact, that known quantum algorithms for finding hidden periods and hidden lattices within real vector spaces [237, 238, 239, 240] and/or or infrastructures [235, 236] (e.g., Hallgren's algorithm for solving Pell's equation [237, 238]) could be at least partially interpreted as generalized normalizer circuits in this sense. Addressing this question would require a careful treatment of precision errors that appear in such algorithms due to the presence of transcendental numbers, which play no role in the present work[6]. Some open questions in this quantum algorithm subfield have been discussed in [236].

A third possible direction to investigate would be whether different models of normalizer circuits could be constructed over ***algebraic structures that are not groups***. One could, for instance, consider sets with *less algebraic structure*, like semi-groups. In this regard, we highlight that a quantum algorithm for finding discrete logarithms over finite semigroups was recently given in [241]. Alternatively, one could study also *sets* with more structure than groups, such as *fields*, whose study is relevant to Van Dam-Seroussi's quantum algorithm for estimating Gauss sums [242].

Lastly, we mention some open questions suggested by our work.

In this thesis, we have not investigated the computational complexity of black-box normalizer circuits *without* classical post-processing. There are two facts which suggest that power of black-box normalizer circuits alone might, in fact, be significantly smaller. The first is the fact that the complexity class of problems solvable by Clifford circuits alone is $\oplus\mathbf{L}$ [110], believed to be a strict subclass of $\mathbf{P}$. The second is that normalizer circuits seem to be incapable of implementing most classical functions coherently even with constant accuracy (this has been rigorously shown in finite dimensions [1, 2]).

Finally, one may study whether considering more general types of inputs, measurements

[5]An n-dimensional infrastructure $\mathcal{I}$ provides a classical presentation for an n-dimensional hypertorus group $\mathbb{R}^n/\Lambda \cong \mathbb{T}^n$, where Λ is an (unknown) period lattice Λ. The elements of this continuous group are represented with some classical structures known as *f-representations*, which are endowed with an operation that allows us to compute within the torus. Although one must deal carefully with non-trivial technical aspects of infinite groups in order to properly define and compute with f-representations (cf. [235, 236] and references therein), one may intuitively understand infrastructures as "generalized black-box hypertoruses". We stress, though, that it is not standard terminology to call "black-box group" to an infinite group.

[6]No such treatment is needed in this chapter, since we study quantum algorithms for finding hidden structures in *discrete* groups.

or adaptive operations might change the power of black-box normalizer circuits. Allowing, for instance, input product states has potential to increase the power of these circuits, since this already occurs for standard Clifford circuits [167, 166]. Concerning measurements, we believe that allowing, e.g. adaptive Pauli operator measurements (in the sense of [2]) is unlikely to give any additional computational power to black-box normalizer circuits: in the best scenario, this could only happen in infinite dimensions, since adaptive normalizer circuits over finite Abelian groups are also efficiently classically simulable with stabilizer techniques [2]. With more general types of measurements, it should be possible to recover full quantum universality, given that qubit cluster-states (which can be generated by Clifford circuits) are a universal resource for measurement-based quantum computation [243, 244]. The possibility of obtaining intermediate hardness results if non-adaptive yet also non-Pauli measurements are allowed (in the lines of [216] or [166, theorem 7]) remains also open.

5.2 Quantum algorithms

5.2.1 The discrete logarithm problem over $\mathbb{Z}_p^\times$

In this section we consider the discrete-logarithm problem studied by Shor [14]. For any prime number p, let $\mathbb{Z}_p^\times$ be the multiplicative group of non-zero integers modulo p. An instance of the discrete-log problem over $\mathbb{Z}_p^\times$ is determined by two elements a, $b \in \mathbb{Z}_p^\times$, such that a generates the group $\mathbb{Z}_p^\times$. Our task is to find the smallest non-negative integer s that is a solution to the equation $a^s = b \bmod p$; the number is called the discrete logarithm $s = \log_a b$.

We now review Shor's algorithm [14, 24] for this problem and prove our first result.

Theorem 74 (Discrete logarithm). *Shor's quantum algorithm for the discrete logarithm problem over $\mathbb{Z}_p^\times$ is a black-box normalizer circuit over the group $\mathbb{Z}_{p-1}^2 \times \mathbb{Z}_p^\times$.*

Theorem 74 shows that black box normalizer circuits over *finite* Abelian groups can efficiently solve a problem for which no efficient classical algorithm is known. In addition, it tells us that black-box normalizer circuits can render widespread public-key cryptosystems vulnerable: namely, they break the Diffie-Helman key-exchange protocol [162], whose security relies in the assumed classical intractability of the discrete-log problem.

Proof. Let us first recall the main steps in Shor's discrete log algorithm.

Algorithm 75 (Shor's algorithm for the discrete logarithm).

Input. Positive integers a, b, where $\mathbb{Z}_p^\times = \langle a \rangle$.

Output. The least nonnegative integer s such that $a^s \equiv b \pmod p$.

We will use three registers indexed by integers, the first two modulo $p-1$ and the last modulo p. The first two registers will correspond to the additive group $\mathbb{Z}_{p-1}$, while the third register will correspond to the multiplicative group $\mathbb{Z}_p^\times$. Two important ingredients of the algorithm will be the unitary gates $U_a : |s\rangle \to |sa\rangle$ and $U_b : |s\rangle \to |sb\rangle$.

1. **Initialization:** Start in the state $|0\rangle|0\rangle|1\rangle$.
2. Create the superposition state $\frac{1}{p-1}\sum_{x,y=0}^{p-1} |x\rangle|y\rangle|1\rangle$, by applying the standard quantum Fourier transform on the first two registers.

3. Apply the unitary U defined by $U|x\rangle|y\rangle|z\rangle = |x\rangle|y\rangle|za^x b^y\rangle$, to obtain the state

$$\frac{1}{p-1}\sum_{x,y=0}^{p-1} |x\rangle|y\rangle|a^x b^y\rangle$$

   This is equivalent to applying the controlled-U_a^x gate between the first and third registers, and the controlled-U_b^y between the second and third registers.

4. Measure and discard the third register. This step generates a so-called coset state

$$\frac{1}{\sqrt{p-1}}\sum_{k=0}^{p-1} |\gamma + ks, -k\rangle,$$

   where γ is some uniformly random element of $\mathbb{Z}_{p-1}$ and s is the discrete logarithm.

5. Apply the quantum Fourier transform over $\mathbb{Z}_{p-1}$ to the first two registers, to obtain

$$\frac{1}{\sqrt{p-1}}\sum_{k'=0}^{p-1} \mathrm{e}^{2\pi\mathrm{i}\frac{k'\gamma}{p-1}} |k', k's\rangle,$$

6. Measure the system in the standard basis to obtain a pair of the form $(k', k's) \bmod p$ uniformly at random.

7. Classical post-processing. By repeating the above process n times, one can extract the discrete logarithm s from these pairs with exponentially high probability (at least $1 - 2^{-n}$), in classical polynomial time.

Note that the Hilbert space of the third register precisely corresponds to $\mathcal{H}_{\mathbf{B}}$ if we choose the black-box group to be $\mathbf{B} = \mathbb{Z}_p^\times$. It is now easy to realize that Shor's algorithm for discrete log is a normalizer circuit over $\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \times \mathbb{Z}_p^\times$: steps 2 and 4 correspond to applying partial QFTs over $\mathbb{Z}_{p-1}$, and the gate U applied in state 3 is a group automorphism over $\mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \times \mathbb{Z}_p^\times$. □

We stress that, in the proof above, there is no known efficient classical algorithm for solving the group decomposition problem for the group $\mathbb{Z}_p^\times$ (as we define it in Section 5.2.5): although, by assumption, we know that $\mathbb{Z}_p^\times = \langle a\rangle \cong \mathbb{Z}_{p-1}$, this information does not allow us to convert elements from one representation to the other, since this requires solving the discrete-logarithm problem itself. In other words, we are unable to compute classically the *group isomorphism* $\mathbb{Z}_p^\times \cong \mathbb{Z}_{p-1}$. In our version of the group decomposition problem, we require the ability to *compute* this group isomorphism. For this reason, we treat the group $\mathbb{Z}_p^\times$ as a *black-box group.*

5.2.2 Shor's factoring algorithm

In this section we will show that normalizer circuits can efficiently compute the order of elements of (suitably encoded) Abelian groups. Specifically, we show how to efficiently solve the order finding problem for every (finite) Abelian black-box group $\mathbf{B}$ [83] with normalizer circuits. Due to the well-known classical reduction of the factoring problem to the problem of computing orders of elements of the group $\mathbb{Z}_N^\times$, our result implies that black-box normalizer circuits can efficiently factorize large composite numbers, and thus break the widely used RSA public-key cryptosystem [15].

We briefly introduce the **order finding problem** over a black-box group $\mathbf{B}$, that we always assume to be finite and Abelian. In addition, we assume that the elements of the black-box group can be uniquely encoded with n-bit strings, for some known n. The task we consider is the following: given an element a of $\mathbf{B}$, we want to compute the order $|a|$ of a (the smallest positive integer r with the property[7] $a^r = 1$). Our next theorem states that this version of the order finding problem can be efficiently solved by a quantum computer based on normalizer circuits.

Theorem 76 (Order finding over B). *Let $\mathbf{B}$ be a finite Abelian black-box group with an n-qubit encoding and $\mathcal{H}_\mathbf{B}$ be the Hilbert space associated with this group. Let V_a be the unitary that performs the group multiplication operation on $\mathcal{H}_\mathbf{B}$: $V_a|x\rangle = |ax\rangle$. We denote by c-$V_a$ the unitary that performs V_a on $\mathcal{H}_\mathbf{B}$ controlled on the value of an ancillary register $\mathcal{H}_\mathbb{Z}$:*

$$|m, x\rangle \xrightarrow{c-U_a} |m, a^m x\rangle, \qquad \text{for any } m \text{ in } \mathbb{Z}.$$

Assume that we can query an oracle that implements c-V_a in one time step for any $a \in \mathbf{B}$. Then, there exists a hybrid version of Shor's order-finding algorithm, which can compute the order $|a|$ of any $a \in \mathbf{B}$ efficiently, using normalizer circuits over the group $\mathbb{Z} \times \mathbf{B}$ and classical post-processing. The algorithm runs in polynomial-time, uses an efficient amount of precision and succeeds with high probability.

In Theorem 76, by "efficient amount of precision" we mean that instead of preparing Fourier basis states of $\mathcal{H}_\mathbb{Z}$ or measuring on this (unphysical) basis, it is enough to use realistic physical approximations of these states (cf. Section 3.7).

Proof. For simplicity, we assume that a generating set of $\mathbf{B}$ with $O(n)$ elements is given (otherwise we could generate one efficiently probabilistically by sampling elements of $\mathbf{B}$).

We divide the proof into two steps. In the first part, we give an infinite-precision quantum algorithm to randomly sample elements from the set $\mathrm{Out}_a = \{\frac{k}{|a|} : k \in \mathbb{Z}\}$ that uses normalizer circuits over the group $\mathbb{Z} \times \mathbf{B}$ in polynomially many steps. In this first algorithm, we assume that Fourier basis states of $\mathcal{H}_\mathbb{Z}$ can be prepared perfectly and that there are no physical limits in measurement precision; the outcomes $k/|a|$ will be stored with floating point arithmetic and with finite precision. The algorithm allows one to extract the period $|a|$ efficiently by sampling fractions $k/|a|$ (quantumly) and then using a continued fraction expansion (classically).

In the second part of the proof, we will remove the infinite precision assumption.

Our first algorithm is essentially a variation of Shor's algorithm for order finding [14] with one key modification: whereas Shor's algorithm uses a large n-qubit register $\mathcal{H}_2^n$ to estimate the eigenvalues of the unitary V_a, we will replace this multiqubit register with a single *infinite* dimensional Hilbert space $\mathcal{H}_\mathbb{Z}$. The algorithm is *hybrid* in the sense that it involves both continuous- and discrete-variable registers. The key feature of this algorithm is that, at every time step, the implemented gates are *normalizer gates*, associated with the groups $\mathbb{Z} \times \mathbb{Z}_N^\times$ and $\mathbb{T} \times \mathbb{Z}_N^\times$ (which are, themselves, related via the partial Fourier transforms $\mathcal{F}_\mathbb{Z}$ and $\mathcal{F}_\mathbb{T}$). The algorithm succeeds with constant probability.

Algorithm 77 (Hybrid order finding with infinite precision).

[7]Since $\mathbf{B}$ is finite, the order $|a|$ is a well-defined number.

Input. A black box (finite abelian) group $\mathbf{B}$, and an element $a \in \mathbf{B}$.
Output. The order s of a in $\mathbf{B}$, i.e. the least positive integer s such that $a^s = 1$.

We will use multiplicative notation for the black box group $\mathbf{B}$, and additive notation for all other subgroups.

1. **Initialization:** Initialize $\mathcal{H}_{\mathbb{Z}}$ on the Fourier basis state $|0\rangle$ with $0 \in \mathbb{T}$, and $\mathcal{H}_{\mathbf{B}}$ on the state $|1\rangle$, with $1 \in \mathbf{B}$. In our formalism, we will regard $|0, 1\rangle$ as a standard-basis state of the basis labeled by $\mathbb{T} \times \mathbf{B}$.

2. Apply the Fourier transform $\mathcal{F}_{\mathbb{T}}$ to the register $\mathcal{H}_{\mathbb{Z}}$. This changes the designated basis of this register to be the one labeled by the group $\mathbb{Z}$. The state $|0\rangle$ in the new basis is an infinitely-spread comb of the form $\sum_{m \in \mathbb{Z}} |m\rangle$.

3. Let the oracle V_a act jointly on $\mathcal{H}_{\mathbb{Z}} \times \mathcal{H}_{\mathbf{B}}$; then the state is mapped in the following manner:
$$\sum_{m\in\mathbb{Z}} |m\rangle|1\rangle \xrightarrow{c\text{-}V_a} \sum_{m\in\mathbb{Z}} |m, a^m\rangle. \tag{5.2.1}$$
Note that, in our formalism, the oracle c-V_a can be regarded as an automorphism gate U_α. Indeed, the gate implements a classical invertible function on the group $\alpha(m, x) = (m, a^m x)$. The function is, in addition, a continuous[8] group automorphism, since
$$\begin{aligned}\alpha\left((m,x)(n,y)\right) = \alpha(m+n, xy) &= (m+n, (a^{m+n})(xy)) \\ &= (m+n, (a^m x)(a^n y)) = (m, a^m x)(n, a^n y) \\ &= \alpha(m,x)\alpha(n,y).\end{aligned} \tag{5.2.2}$$

4. Measure and discard the register $\mathcal{H}_{\mathbf{B}}$. Say we obtain a^s as the measurement outcome. Note that the function a^m is periodic with period $r = |a|$, the order of the element. Due to periodicity, the state after measuring a^s will be of the form
$$\left(\sum_{j\in\mathbb{Z}} |s + jr\rangle\right) |a^s\rangle. \tag{5.2.3}$$
After dicarding $\mathcal{H}_{\mathbf{B}}$ we end up in a periodic state $\sum |s + jr\rangle$ which encodes $r = |a|$.

5. Apply the Fourier transform $\mathcal{F}_{\mathbb{Z}}$ to the register $\mathcal{H}_{\mathbb{Z}}$. We work again in the Fourier basis of $\mathcal{H}_{\mathbb{Z}}$, which is labelled by the circle group $\mathbb{T}$. The periodic state $\sum |s + jr\rangle$ in the dual $\mathbb{T}$ basis reads [22]
$$\sum_{k=0}^{r-1} e^{2\pi i \frac{sk}{r}} |\tfrac{k}{r}\rangle \tag{5.2.4}$$

6. Measure $\mathcal{H}_{\mathbb{Z}}$ in the Fourier basis (the basis labeled by $\mathbb{T}$). Since we that the initial state of the computation is as close to $|0\rangle$ as we wish, the wavefunction of the final state (5.2.4) is *sharply peaked* around values $p \in \mathbb{T}$ of the form k/r. As a result, a high resolution measurement will let us sample these numbers (within some floating-point precision window Δ) nearly uniformly at random.

[8]This is vacuously true: since the group $G := \mathbb{Z} \times \mathbf{B}$ is discrete, *any* functtion $f : G \to G$ is continuous.

7. **Classical postprocessing:** Repeat Steps 1-7 a few times to obtain randomly sampled multiples $\{k_i/r\}_i$. Afterwards by using a (classical) continued-fraction expansion algorithm [3, 4], the order r can be extracted. This can be done, for instance, with an algorithm from [245] that obtains r with *constant* probability after sampling two numbers $\frac{k_1}{r}$, $\frac{k_2}{r}$, if the measurement resolution is high enough: $\Delta \leq 1/2r^2$ is enough for our purposes.

There is a strong manifest similarity between Algorithm 77 and Shor's factoring algorithm: the quantum Fourier transforms $\mathcal{F}_{\mathbb{T}}$ in our algorithm $\mathcal{F}_{\mathbb{Z}}$ play the role of the discrete Fourier transorm $\mathcal{F}_{2^n}$, and $c\text{-}V_a$ acts as the modular exponentation gate [14]. In fact, one can regard Algorithm 77 as a "hybrid" version of Shor's algorithm combining both continuous and discrete variable registers. The remarkable feature of this version of Shor's algorithm is that the quantum part of the algorithm 1-6 is a normalizer computation.

Algorithm 77 is efficient if we just look at the number of gates it uses. However, the algorithm is *inefficient* in that it uses infinitely-spread Fourier states $|p\rangle = \sum_{m\in\mathbb{Z}} e^{-2\pi i p m}|m\rangle$ (which are unphysical and cannot be prepared with finite computational resources) and arbitrarily precise measurements. We finish the proof of Theorem 76 by giving an improved algorithm that does not rely on unphysical requirements.

Algorithm 78 (Hybrid order finding with finite precision).

1-2 **Initialization:** Initialize $\mathcal{H}_{\mathbf{B}}$ to $|1\rangle$. The register $\mathcal{H}_{\mathbb{Z}}$ will begin in an *approximate* Fourier basis state $\left|\tilde{0}\right\rangle = \frac{1}{\sqrt{2M+1}}\sum_{-M}^{+M}|m\rangle$, i.e. a square pulse of length $2M+1$ in the integer basis, centered at 0. This step simulates steps 1-2 in Algorithm 77.

3-4 Repeat steps 3-4 of Algorithm 77. The state after obtaining the measurement outcome a^s is now different due to the finite "length" of the comb $\sum_{m=0}^{M}|m\rangle$; we obtain

$$|\psi\rangle = \frac{1}{\sqrt{L}}\sum_{-L_a}^{L_b}|s+jr\rangle, \tag{5.2.5}$$

where $L = L_a+L_b+1$ and s is obtained nearly uniformly at random from $\{0,\ldots,r-1\}$. The values L_a, L_b are positive integers of of the form $\lfloor M/r\rfloor - \epsilon$ with $-2\leq\epsilon\leq 0$ (the particular value of ϵ depends on s, but it is irrelevant in our analysis). Consequently, we have $L = 2\lfloor M/r\rfloor - (\epsilon_a+\epsilon_b)$.

5 Apply the Fourier transform $\mathcal{F}_{\mathbb{Z}}$ to the register $\mathcal{H}_{\mathbb{Z}}$. The wavefunction of the final state $\hat{\psi}$ is the Fourier transform of the wavefunction ψ of (5.2.5). We compute $\hat{\psi}$ using formula (3.6.5):

$$\begin{aligned}\hat{\psi}(p) &= \sum_{x\in\mathbb{Z}} e^{2\pi i p x}\psi(x) = \frac{1}{\sqrt{L}}\sum_{-L_a}^{L_b} e^{2\pi i p(s+jr)} = \frac{1}{\sqrt{L}}\left(e^{2\pi i p s}\right)\frac{e^{2\pi i p r(L_b+1)} - e^{-2\pi i p r L_a}}{e^{2\pi i p r}-1}\\ &= \frac{e^{2\pi i p\left(s+\frac{L_b-L_a}{2}\right)}}{\sqrt{L}}\frac{\sin(\pi L p r)}{\sin(\pi p r)} = \frac{e^{2\pi i p\left(s+\frac{L_b-L_a}{2}\right)}}{\sqrt{L}}D_{L,r}(p)\end{aligned} \tag{5.2.6}$$

(to derive the equation, we apply the summation formula of the geometric series and re-express the result in terms of the Dirichlet kernel [99]

$$D_{L,r}(p) = \frac{\sin(\pi Lpr)}{\sin(\pi pr)}. \tag{5.2.7}$$

6 **Measure** $\mathcal{H}_{\mathbb{Z}}$ in the Fourier basis. We show now that, if the resolution is high enough, then the probability distribution of measurement outcomes will be "polynomially close" to the one obtained in the infinite precision case (5.2.4). Intuitively, this is a consequence of the fact that in the limit $M \to \infty$ (when the initial state becomes an infinitely-spread comb), we have also $L \to \infty$ and that the function $D_L, r(p)$ converges to a train $\sum_{k=0}^{r-1} \delta_{k/r}(p)$ of Dirac measures [99]. In addition, for a high finite value of M, we find that the probability of obtaining some outcome p within a $\Delta = \frac{1}{Lr}$ window of a fraction $\frac{k}{r}$ is also high.

$$\Pr(|p - \tfrac{k}{r}| \leq \tfrac{\Delta}{2}) = \frac{1}{L}\int_{-\frac{\Delta}{2}}^{+\frac{\Delta}{2}} \frac{\sin^2(\pi Lpr)}{\sin^2(\pi pr)}\,\mathrm{d}p \geq \frac{\Delta}{L}\frac{\sin^2\left(\frac{\pi}{2}\right)}{\sin^2\left(\frac{\pi}{2L}\right)} \geq \frac{4}{\pi^2 r}, \tag{5.2.8}$$

where we use the mean value theorem and the bound $\sin(x)^2 \leq x^2$. It follows that with *constant* probability (larger than $4/\pi^2 \approx 0.41$) the measurement will output some outcome $\frac{\Delta}{2}$-close to a number of the form k/r. (A tighter lower bound of 2/3 for the success probability can be obtained by evaluating the integral numerically.)

Lastly, note that although the derivation of (5.2.8) implicitly assumes that the finial measurement is infinitely precise, it is enough to implement measurements with resolution close to Δ. Due to the peaked shape of the final distribution (5.2.8), it follows that $\mathbb{T}heta(\frac{1}{M})$ resolution is enough if our task is to sample $\frac{\Delta}{2}$-estimates of these fractions nearly uniformly at random; this scaling is *efficient* as a function of M (cf. Section 3.7).

7 **Classical postprocessing:** We now set M (the length of the initial comb state) to be large enough so that $\frac{\Delta}{2} = \frac{1}{2Lr} \leq \frac{1}{2r^2}$; taking $\log M = O(\text{poly}\, n)$ is enough for our purposes. With such an M, the measurement step 6 will output a number p that is $\frac{1}{2r^2}$ close to a $\frac{k}{r}$ with high probability, which can be increased to be arbitrarily close to 1 with a few repetitions. We then proceed as in step 7 of Algorithm 77 to compute the order r. □

Shor's algorithm as a normalizer circuit

Our discussion in the previous section reveals strong a resemblance between our hybrid normalizer quantum algorithm for order finding and Shor's original quantum algorithm for this problem [14]: indeed, both quantum algorithms employ remarkably similar circuitry. In this section we show that this resemblance is actually more than a mere fortuitous analogy, and that, in fact, one can understand Shor's original order-finding algorithm as a discretized version of our finite-precision hybrid algorithm for order finding 77.

Theorem 79 (Shor's algorithm as a normalizer circuit). *Shor's order-finding algorithm [14] provides an efficient discretized implementation of our hybrid normalizer Algorithm 78.*

Note that the theorem does not imply that all possible quantum algorithms for order finding are normalizer circuits (or discretized versions of some normalizer circuit). What it shows is that the one first found by Shor in [14] does exhibit such a structure.

Proof. Our approach will be to show explicitly that the evolution of the initial quantum state in Shor's algorithm is analogous to that of the initial state in Algorithm 78 if we discretize the computation. Recall that Shor's algorithm implements a quantum phase estimation [20] for the unitary V_a. Let D be the dimension of the Hilbert space used to record such phase. We assume D to be odd[9] and write $D = 2M + 1$. Then Shor's algorithm can be written as follows:

1. Initialize the state $|0, 1\rangle$ on the Hilbert space $\mathcal{H}_D \times \mathcal{H}_{\mathbb{Z}_N^\times}$.

2. Apply the discrete Fourier transform $\mathcal{F}_{\mathbb{Z}_D}$ on $\mathcal{H}_D$ to obtain
$$\sum_{m=0}^{D-1} |m\rangle|1\rangle = \sum_{-M}^{M} |m\rangle|1\rangle. \tag{5.2.9}$$
So far, we have simulated step 1 in Algorithm 78 by constructing the same periodic state. These first two steps are also clearly analogous to steps 1-2 in Algorithm 77.

3-4 Apply the modular exponentiation gate U_{me}, which is the following unitary [14]
$$U_{\mathrm{me}}|m, x\rangle = |m, a^m x\rangle, \tag{5.2.10}$$
to the state. Measure the register $\mathcal{H}_{\mathbb{Z}_N^\times}$ in the standard basis. We obtain, again, a quantum state of the form (5.2.5), with $L \leq D$.

6 We apply the discrete Fourier transform $\mathcal{F}_{\mathbb{Z}_D}$ to the register $\mathcal{H}_{\mathbb{Z}_D}$ again. We claim now that the output state will be a discretized version of (5.2.6) due to a remarkable **mathematical correspondence** between Fourier transforms. Note that any quantum state $|\psi\rangle$ of the infinite-dimensional Hilbert space $\mathcal{H}_{\mathbb{Z}}$ can be regarded as a quantum state of $\mathcal{H}_D$ given that the support of $|\psi\rangle$ is limited to the standard basis states $|0\rangle, |\pm 1\rangle, \ldots, |\pm M\rangle$. Let us denote the latter state $|\psi_D\rangle$ to distinguish both. Then, we observe a correspondence between letting $\mathcal{F}_{\mathbb{Z}}$ act on $|\psi\rangle$ and letting $\mathcal{F}_{\mathbb{Z}_D}$ act on $|\psi_D\rangle$.
$$\hat{\psi}(p) = \sum_{x=-M}^{x=+M} \mathrm{e}^{2\pi i p x}\psi(x) \qquad \longleftrightarrow \qquad \hat{\psi}_D(k) = \sum_{x=-M}^{x=+M} \mathrm{e}^{2\pi i \frac{kx}{D}}\psi_D(x) \tag{5.2.11}$$
The correspondence (equation 5.2.11) tells us that, since we have $\psi(x) = \psi_D(x)$, it follows that the Fourier transformed function $\hat{\psi}_D(k)$ is precisely the function $\hat{\psi}(p)$ evaluated at points of the form $p = \frac{k}{D}$. The final state can be written as
$$\sum_{k=0}^{D-1} \hat{\psi}\left(\tfrac{k}{D}\right)|k\rangle. \tag{5.2.12}$$
which is, indeed, a discretized version of (5.2.6).

[9]This choice is not essential, neither in Shor's algorithm nor in Algorithm 78, but it simplifies the proof.

7-8 The last steps of Shor's algorithm are identical to 7-8 in Algorithm 78, with the only difference being that the wavefunction (5.2.12) is now a discretization of (5.2.6). The probability of measuring a number k such that $\frac{k}{D}$ is close to a multiple of the form $\frac{k'}{r}$ will again be high, due to the properties of the Dirichlet kernel (5.2.7). Indeed, one can show (see, e.g. [24]) with an argument similar to (5.2.8) that, by setting $D = N^2$, the algorithm outputs with constant probability and almost uniformly a fraction $\frac{k}{D}$ among the two closest fraction to some value of the form k/r (see e.g. [14] for details). The period r can be recovered, again, with a continued fraction expansion. □

Normalizer gates over ∞ groups are necessary to factorize

At this point, it is a natural question to ask whether it is necessary at all to replace the Hilbert space $\mathcal{H}_2^n$ with an infinite-dimensional space $\mathcal{H}_{\mathbb{Z}}$ with an integer basis in order to be able to factorize with normalizer circuits. We discuss in this section that, in our view, this is a **key indispensable ingredient** of our proof.

We begin our discussion by showing rigorously, in the black-box set-up, that no quantum algorithm for factoring based on *modular exponentation* gates (controlled V_a rotations) can be efficiently implemented with normalizer circuits over finite Abelian groups, in a strong sense.

Theorem 80. *Let $\mathcal{H}_M$ be the Hilbert space with basis $\{|0\rangle, \ldots, |M-1\rangle\}$ and dimension M. Let $\mathbf{B}$ be an Abelian black-box group with associated Hilbert space $\mathcal{H}_{\mathbf{B}}$. Consider the composite Hilbert space $\mathcal{H} = \mathcal{H}_M \times \mathcal{H}_{\mathbf{B}}$ and define U_{me} to be the unitary gate on $\mathcal{H}$ defined as $U_{\mathrm{me}}|m, x\rangle = |m, a^m x\rangle$, where $a, x \in \mathbf{B}$ and $m \in \mathbb{Z}_M$. Then, unless M is a multiple of the order of a, there does not exist any normalizer circuit over $\mathcal{H}$ (even of exponential size) satisfying $\|C - U_{\mathrm{me}}\|_{\mathrm{op}} \leq 1 - 2^{-1/2}$.*

We prove the theorem in Appendix D.1. We highlight that a similar result was proven in [1, theorem 2]: that normalizer circuits over groups of the form $\mathbb{Z}_{2^n} \times \mathbb{Z}_N$ also fail to approximate the modular exponentiation. Also, we point out that it is easy to see that the converse of Theorem 80 is also true: if $|a|$ divides M, then an argument similar to (D.1.1) shows that $(m, x) \to (m, a^m x)$ is a group automorphism of $\mathbb{Z}_M \times \mathbf{B}$, and the gate U_{mf} automatically becomes a normalizer automorphism gate.

The main implication of Theorem 80 is that finite-group normalizer circuits *cannot* implement nor approximate the quantum modular exponentiation gate between $\mathcal{H}_{\mathbf{B}}$, playing the role of the target system, and some ancillary control system, *unless* a multiple $M = \lambda|a|$ of the order of a is known in advance. Yet the problem of finding multiples of orders is *at least as hard as factoring and order-finding*: for $\mathbf{B} = \mathbb{Z}_N^{\times}$, a subroutine to find multiples of orders can be used to efficiently compute classically a multiple of the order of the group $\varphi(N)$, where φ is the Euler totient function, and it is known that factoring is polynomial-time reducible to the problem of finding a single multiple of the form $\lambda\varphi(N)$ [88].

We arrive to the conclusion that, unless we are in the trivial case where we know how to factorize in advanced, a factoring algorithm based on finite-group normalizer gates cannot comprise controlled-V_a rotations. We further conjecture that any other approach based on finite-group normalizer gates cannot work either.

Conjecture 81. Unless factoring is contained in BPP, there is no efficient quantum algorithm to solve the factoring problem using only normalizer circuits over finite Abelian groups (even when these are allowed to be black-box groups) and classical pre- and post-processing.

We back up our conjecture with two facts. On one hand, Shor's algorithm for factoring [14] (to our knowledge, the only quantum algorithm for factoring thus far) uses a modular exponentiation gate to estimate the phases of the unitary V_a, and these gates are hard to implement with finite-group normalizer circuits due to Theorem 80. On the other hand, the reason why this does works for the group $\mathbb{Z}$ seems to be, in our view, intimately related to the fact that the order-finding problem can be naturally casted as an instance of the Abelian ***hidden subgroup problem*** over $\mathbb{Z}$ (see also Section 5.2.4). Note that, although one can always cast the order-finding problem as an HSP over any finite group $\mathbb{Z}_{\lambda\varphi(N)}$ for an integer λ, this formulation of the problem is unnatural in our setting, as it requires (again) the prior knowledge of a multiple of $\varphi(N)$, which we could use to factorize and find orders classically without the need of a quantum computer [88].

5.2.3 Elliptic curves

In the previous sections we have seen that black-box normalizer circuits can compute discrete logarithm in $\mathbb{Z}_p^\times$ and break the Diffie-Hellman key exchange protocol. In the proof, we showed that Shor's algorithm for this problem decomposes naturally in terms of normalizer gates over $\mathbb{Z}_p^\times$, treated as a black-box group.

It is known that Shor's algorithm can be adapted in order to compute discrete logarithms over arbitrary black-box groups. In particular, this can be done for the group of solutions E of an elliptic curve [155, 104, 156], thereby rendering elliptic curve cryptography (ECC) vulnerable. Efficient unique encodings and fast multiplication algorithms for these groups are known, so that they can be formally treated as black-box groups. In this section, we show that a quantum algorithm given by Proos and Zalka [155] to compute discrete logarithms over elliptic curves can be implemented with black-box normalizer circuits.

Basic notions

To begin, we review some rudiments of the theory of elliptic curves. For simplicity, our survey focuses only on the particular types of elliptic curves that were studied in [155], over fields with characteristic different than 2 and 3. Our discussion applies equally to the (more general) cases considered in [104, 156], although the definition of the elliptic curve group operation becomes more cumbersome in such settings[10]. For more details on the subject, in general, we refer the reader to [24, 246].

Let $p > 3$ be prime and let K be the field defined by endowing the set $\mathbb{Z}_p$ with the addition and multiplication operations modulo p. An *elliptic curve* E over the field K s a finite Abelian group formed by the solutions $(x, y) \in K \times K$ to an equation

$$C : y^2 = x^3 + \alpha x + \beta \tag{5.2.13}$$

together with a special element O called the "point at infinity"; the coefficients α, β in this equation live in the field K. The discriminant $\Delta := -16(4\alpha^3 + 27\beta^2)$ is nonzero, ensuring that the curve is non-singular. The elements of E are endowed with a commutative group operation. If $P \in E$ then $P + O = O + P = P$. The inverse element $-P$ of P is obtained by the reflection of P about the x axis. Given two elements $P = (x_P, y_P)$ and $Q = (x_Q, y_Q) \in E$,

[10]Correspondingly, the complexity of performing group multiplications in [104, 156] is greater.

the element $P+Q$ is defined via the following rule:

$$P+Q=\begin{cases} O & \text{if } P=(x_P,y_P)=(x_Q,-y_Q)=-Q, \\ -R & \text{otherwise (read below).} \end{cases} \tag{5.2.14}$$

In the case $P \neq Q$, the point R is computed as follows:

$$\begin{aligned} x_R &= \lambda^2 - x_P - x_Q \\ y_R &= y_P - \lambda(x_P - x_R) \end{aligned} \qquad \lambda := \begin{cases} \frac{y_Q-y_P}{x_Q-x_P} & \text{if } P \neq Q \\ \frac{3x_P^2+\alpha}{2y_P} & \text{if } P = Q \text{ and } y_P \neq 0 \end{cases}$$

R can also be defined, geometrically, to be the "intersection between the elliptic curve and the line through P and Q" (with a minus sign) [24].

It is not hard to check form the definitions above that the elliptic-curve group E is finite and Abelian; from a computational point of view, the elements of E can be stored with $n \in O(\log|K|)$ bits and the group operation can be computed in $O(\operatorname{poly} n)$ time. Therefore, the group E can be treated as a **black box group**.

Finally, the **discrete logarithm problem** (DLP) over an elliptic curve is defined in a way analogous to the $\mathbb{Z}_p^\times$ case, although now we use additive notation: given $a, b \in E$ such that $xa = b$ for some integer x; our task is to find the least nonnegative integer s with that property. The elliptic-curve DLP is believed to be intractable for classical computers and can be used to define cryptosystems analog to Diffe-Hellman's [24].

Finding discrete logarithms over elliptic curves with normalizer circuits

In this section we review Proos-Zalka's quantum approach to solve the DLP problem over an elliptic curve [155]; their quantum algorithm is, essentially, a modification of Shor's algorithm to solve the DLP over $\mathbb{Z}_p^\times$, which we covered in detail in Section 5.2.1.

Our **main contribution** in this section is that Proos-Zalka's algorithm can be implemented with normalizer circuits over the group $\mathbb{Z} \times \mathbb{Z} \times E$. The proof reduces to combining ideas from sections 5.2.1 and 5.2.2 and will be sketched in less detail.

Algorithm 82 (Proos-Zalka's [155]).

Input. An elliptic curve with associated group E (the group operation is defined as per (5.2.14)), and two points $a, b \in E$. It is promised that $sa = b$ for some nonnegative integer s.

Output. Find the least nonnegative integer s such that $sa = b$.

1. We use a register $\mathcal{H}_E$, where E is the group associated with the elliptic curve (5.2.13), and two ancillary registers $\mathcal{H}$ of dimension $N = 2^n$, associated with the group $A = \mathbb{Z}_N \times \mathbb{Z}_N$. The computation begins in the state $|0,0,O\rangle$, where $(0,0) \in A$ and $O \in E$.

2. Apply Fourier transforms to the ancillas to create the superposition $\sum_{(x,y)\in A} |x,y,O\rangle$.

3. Apply the following unitary transfomation:

$$\sum_{(x,y)\in A} |x,y,O\rangle \xrightarrow{c\text{-}U} \sum_{(x,y)\in A} |x,y,xa+yb\rangle. \tag{5.2.15}$$

4. Apply Fourier transforms to the ancillas again, and then measure the register $\mathcal{H}_E$, obtaining an outcome of the form (x', y'). These outcomes contain enough information to extract the number s, with similar post-processing techniques to those used in Shor's DLP algorithm.

Algorithm 82 is not a normalizer circuit over $\mathbb{Z}_N \times \mathbb{Z}_N \times E$. Similarly to the factoring case, the algorithm would become a normalizer circuit if the classical transformation in step 3 was an automorphism gate; however, for this to occur, N needs to be a common multiple of the orders of a and b (the validity of these claims follows with similar arguments to those in Section 5.2.2). In view of our results in sections 5.2.1 and 5.2.2, one can easily come up with two approaches to implement Algorithm 75 using normalizer gates.

(a) The first approach would be to use our normalizer version of Shor's algorithm (Theorem 76) to find the orders of the elements a and b: normalizer gates over $\mathbb{Z} \times E$ would be used in this step. Then, the number N in Algorithm 82 can be set so that all the gates involved become normalizer gates over $\mathbb{Z}_N \times \mathbb{Z}_N \times E$.

(b) Alternatively, one can choose not to compute the orders by making the ancillas infinite dimensional, just as we did in Algorithm 77. The algorithm becomes a normalizer circuit over $\mathbb{Z}\times\mathbb{Z}\times E$: as in Algorithm 77, the ancillas are initialized to the zero Fourier basis state, and the discrete Fourier transforms are replaced by QFTs over $\mathbb{T}$ (in step 2) and $\mathbb{Z}$ (in step 4). A finite precision version of the algorithm can be obtained in the same fashion as we derived Algorithm 77. Proos-Zalka's original algorithm could, again, be interpreted as a discretization of the resulting normalizer circuit.

5.2.4 The hidden subgroup problem

All problems we have considered this far—finding discrete logarithms and orders of Abelian group elements—fit inside a general class of problems known as hidden subgroup problems over Abelian groups [78, 79, 80, 81]. Most quantum algorithms discovered in the early days of quantum computation solve problems that can be recasted as Abelian HSPs, including Deutsch's problem [157], Simon's [158], order finding and discrete logarithms [14], finding hidden linear functions [159], testing shift-equivalence of polynomials [160], and Kitaev's Abelian stabilizer problem [20, 161].

In view of our previous results, it is natural to ask how many of these problems can be solved within the normalizer framework. In this section we show that a well-known quantum algorithm that solves the Abelian HSPs (in full generality) can be modeled as a normalizer circuit over an Abelian group $\mathcal{O}$. Unlike previous cases, the group involved in this computation cannot be regarded as a black-box group, as it will not be clear how to perform group multiplications of its elements. This fact reflects the presence of oracular functions with unknown structure are present in the algorithm, to which the group $\mathcal{O}$ is associated; thus, we call $\mathcal{O}$ an *oracular group*. We will discuss, however, that this latter difference does not seem to be very substantial, and that the Abelian HSP algorithm can be naturally regarded as a normalizer computation.

The quantum algorithm for the Abelian HSP

In the *Abelian hidden subgroup* problem we are given a function $f : G \to X$ from an Abelian finite[11] group G to a finite set X. The function f is constant on cosets of the form $g + H$, where H is a subgroup "hidden" by the function; moreover, f is different between different cosets. Given f as a black-box, our task is to find such a subgroup H.

The Abelian HSP is a hard problem for classical computers, which need to query the oracle f a superpolynomial amount of times in order to identify H [24]. In contrast, a quantum computer can determine H in polynomial time $O(\text{polylog}\,|G|)$, and using the same amount of queries to the oracle. We describe next a celebrated quantum algorithm for this task [78, 79, 84]. The algorithm is efficient given that the group G is explicitly given[12] in the form $G = \mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_m}$ [84, 73, 81].

Algorithm 83 (Abelian HSP).

Input. An explicitly decomposed finite abelian group $G = \mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_m}$, and oracular access to a function $f : G \to X$ for some set X. f satisfies the promise that $f(g_1) = f(g_2)$ iff $g_1 = g_2 + h$ for some $h \in H$, where $H \subseteq G$ is some fixed but unknown subgroup of G.

Output. A generating set for H.

1. Apply the QFT over the group G to an initial state $|0\rangle$ in order to obtain a uniform superposition over the elements of the group $\sum_{g\in G} |g\rangle$.
2. Query the oracle f in an ancilla register, creating the state

$$\frac{1}{\sqrt{|G|}} \sum_{g\in G} |g, f(g)\rangle \tag{5.2.16}$$

3. The QFT over G is applied to the first register, which is then measured.
4. After repeating 1-3 polynomially many times, the obtained outcomes can be postprocessed classically to obtain a generating set of H with exponentially high probability (we refer the reader to [21] for details on this classical part).

We now claim that the quantum part of Algorithm 83 is a *normalizer circuit*, of a slightly more general kind than the ones we have already studied. The normalizer structure of the HSP-solving quantum circuit is, however, remarkably well-hidden compared to the other quantum algorithms that we have already studied. It is indeed a surprising fact that there is *any* normalizer structure in the circuit, due to the presence of an oracular function, whose inner structure appears to be completely unknown to us!

Theorem 84 (The Abelian HSP algorithm is a normalizer circuit.). *In any Abelian hidden subgroup problem, the subgroup-hiding property of the oracle function f induces a group structure $\mathcal{O}$ in the set X. With respect to this hidden "linear structure", the function f becomes a group homomorphism, and the HSP-solving quantum circuit becomes a normalizer circuit over $G \times \mathcal{O}$.*

The proof is the content of the next two sections.

[11]In this section we assume G to be finite for simplicity. For a case where G is infinite, we refer the reader back to Section 5.2.2, where we studied the order finding problem (which is a HSP over $\mathbb{Z}$).

[12]If the group G is not given in a factorized form, the Abelian HSP may still be solved by applying Cheung-Mosca's algorithm to decompose G (see next section).

Unweaving the hidden-subgroup oracle

The key ingredient in the proof of the theorem (which is the content of the next section) is to realize that the oracle f cannot fulfill the subgroup-hiding property without having a hidden homomorphism structure, which is also present in the quantum algorithm.

First, we show that f induces a ***group structure*** on X. Without loss of generality, we assume that the function f is surjective, so that $\mathrm{im} f = X$. (If this is not true, we can redefine X to be the image of f.) Thus, for every element $x \in X$, the preimage $f^{-1}(x)$ is contained in G, and is a coset of the form $f^{-1}(x) = g_x + H$, where H is the hidden subgroup and $f(g_x) = x$. With these observations in mind, we can define a group operation in X as follows:

$$x \cdot y = \tilde{f}\left(f^{-1}(x) + f^{-1}(y)\right). \tag{5.2.17}$$

In (5.2.17) we denote by $\tilde{f}$ the function $\tilde{f}(x+H) = f(x)$ that sends cosets $x+H$ to elements of X. The subgroup-hiding property guarantees that this function is well-defined; moreover, f and $\tilde{f}$ are related via $f(x) = \tilde{f}(x+H)$. The addition operation on cosets $f^{-1}(x) = g_x + H$ and $f^{-1}(y) = g_y + H$ is just the usual group operation of the quotient group G/H [87]:

$$f^{-1}(x) + f^{-1}(y) = (g_x + H) + (g_y + H) = (g_x + g_y) + H. \tag{5.2.18}$$

By combining the two expressions, we get an explicit formula for the group multiplication in terms of coset representatives: $x \cdot y = f(g_x + g_y)$. It is routine to check that this operation is associative and invertible, turning X into a group, which we denote by $\mathcal{O}$. The neutral element of the group is the string e in X such that $e = f(0) = f(H)$, which we show explicitly:

$$x \cdot e = e \cdot x = \tilde{f}\left(f^{-1}(x) + f^{-1}(e)\right) = \tilde{f}\left(f^{-1}(x) + H\right) = x \tag{5.2.19}$$

The group $\mathcal{O}$ is manifestly finite and Abelian—the latter property is due to the fact that the addition (5.2.18) is commutative.

Lastly, it is straightforward to check that the oracle f ***is a group homomorphism*** from G to $\mathcal{O}$: for any g, $h \in G$ let $x := f(g)$ and $y := f(h)$, we have

$$\begin{aligned} f(g+h) &= \tilde{f}\left(g+h+H\right) = \tilde{f}\left((g+H)+(h+H)\right) = \tilde{f}\left(f^{-1}(x) + f^{-1}(y)\right) && (5.2.20)\\ &= x \cdot y = f(g) \cdot f(h). && (5.2.21)\end{aligned}$$

It follows from the first isomorphism theorem in group theory [87] that $\mathcal{O}$ is isomorphic to the quotient group G/H via the map $\tilde{f}$.

The HSP quantum algorithm is a normalizer circuit

We will now analyze the role of the different quantum gates used in Algorithm 83 and see that they are examples of normalizer gates over the group $G \times \mathcal{O}$, where $\mathcal{O}$ is the oracular group that we have just introduced.

The Hilbert space underlying the computation can be written as $\mathcal{H}_G \otimes \mathcal{H}_{\mathcal{O}}$ with the standard basis $\{|g, x\rangle : g \in G,\, x \in \mathcal{O}\}$. associated with this group. We will initialize the ancillary registers to the state $|e\rangle$, where $e = f(0)$ is the neutral element of the group; the total state at step 1 will be $|0, e\rangle$. The Fourier transforms in steps 1 and 3 are just partial QFTs over the group G, which are normalizer gates. The quantum state at the end of step 1 is $\sum |g, e\rangle$.

Next, we look now at step 2 of the computation:

$$\sum |g,e\rangle \quad \longrightarrow \quad \frac{1}{\sqrt{|G|}} \sum_{g\in G} |g, f(g)\rangle. \tag{5.2.22}$$

This step can be implemented by a normalizer automorphism gate defined as follows. Let $\alpha : G \times \mathcal{O} \to G \times \mathcal{O}$ be the function $\alpha(g,x) = (g, f(g)\cdot x)$. Using the fact that $f : G \to \mathcal{O}$ is a group homomorphism (5.2.20), it is easy to check that α is a group automorphism of $G \times \mathcal{O}$. Then the evolution at step 2 corresponds to the action of the automorphism gate U_α:

$$U_\alpha \sum_g |g,e\rangle = \sum_g |\alpha(g,e)\rangle = \sum_g |g, f(g)\cdot e\rangle = \sum_g |g, f(g)\rangle. \tag{5.2.23}$$

Of course, our choice to begin the computation in the state $|0,e\rangle$ and to apply U_α in step 3 is only one possible way to implement the first three steps of the algorithm. We could have alternatively initialized the computation on some $|0,0\rangle$ state and used a slightly different gate $U_{\text{add}} = \sum |g, x + f(g)\rangle\langle g, x|$ in step 3. The latter sequence of gates can however be regarded as an exact gate-by-gate simulation of the former, so that it is perfectly licit to call the algorithm a normalizer computation—at least up to steps 3 and 4.

Finally, note that in the last step of the algorithm we measure the $\mathcal{H}_G$ in the standard basis like a normalizer computation. Therefore, every step in the quantum Algorithm 83 corresponds to one of the allowed operations in a normalizer circuit over $G\times\mathcal{O}$. This finishes the proof of Theorem 84.

The oracular group $\mathcal{O}$ is not a black-box group (but almost)

We ought to stress, at this point, that although Theorem 84 shows that the Abelian HSP quantum algorithm is a normalizer computation over an Abelian group $G \times \mathcal{O}$, the oracular group $\mathcal{O}$ is not a black-box group (as defined in Section 3.4.4), since it is not clear how to compute the group operation (5.2.17), due to our lack of knowledge about the oracular function which defines the multiplication rule. Yet, even in the absence of an efficiently computable group operation, we regard it natural to call the Abelian HSP quantum algorithm a normalizer circuit over $G \times \mathcal{O}$. Our reasons are multi-fold.

First, there is a manifest strong similarity between the quantum circuit in Algorithm 83 and the other normalizer circuits that we have studied in previous sections, which suggests that normalizer operations naturally capture the logic of the Abelian HSP quantum algorithm.

Second, it is in fact possible to argue that, although $\mathcal{O}$ is not a black-box group, it behaves *effectively* as a black-box group in the quantum algorithm. Observe that, although it is true that one cannot generally compute $x\cdot y$ for arbitrary x, $y \in \mathcal{O}$, it is indeed always possible to multiply any element x by the neutral element e, since the computation is trivial in this case: $x\cdot e = e\cdot x = x$. Similarly, in the previous section, it is not clear at all how to implement the unitary transformation $U_\alpha|g,x\rangle = |g, f(g)\cdot x\rangle$ for arbitrary inputs. However, for the restricted set of inputs that we need in the quantum algorithm (which is just the state $|e\rangle$), it is trivial to implement the unitary, for in this case $U_\alpha|g,e\rangle = |g, f(g)\rangle$; since quantum queries to the oracle function are allowed (as in step 2 of the algorithm), the unitary can be simulated by such process, regardless of how it is implemented. Consequently, the circuit *effectively* behaves as a normalizer circuit over a black-box group.

Third, although the oracular model in the black-box normalizer circuit setting is slightly different from the one used in the Abelian HSP they are still *remarkably close* to each other. To see this, let x_i be the elements of X defined as $x_i := f(e_i)$ where e_i is the bit string containing a 1 in the ith position and zeroes elsewhere. Since the e_is form a generating set of G, the x_is generate the group $\mathcal{O}$. Moreover, the value of the function f evaluated on an element $g = \sum g(i)e_i$ is $f(g) = x_1^{g(1)}x_2^{g(2)}\cdots x_m^{g(m)}$, since f is a group homomorphism. It follows from this expression that the group homomorphism is *implicitly multiplying* elements of the group $\mathcal{O}$. We cannot use this property to multiply elements of $\mathcal{O}$ ourselves, since everything happens at the hidden level. However, this observation shows that the assuming that f is computable is *tightly related* to the assumption that we can multiply in $\mathcal{O}$, although slightly weaker. (See also the next section.)

Finally, we mention that this very last feature can be exploited to extend several of our main results, which we derive in the black-box setting, to the more-general "HSP oracular group setting" (although proofs become more technical). For details, we refer the reader to sections 5.3-5.5 and appendix D.3.

A connection to a result by Mosca and Ekert

Prior to our work, it was observed by Mosca and Ekert [80, 84] that f must have a hidden homomorphism structure, i.e. that f can be decomposed as $\mathcal{E} \circ \alpha$ where α is a group homomorphism between G and another Abelian group $Q \cong G/H$, and $\mathcal{E}$ is a one-to-one hiding function from Q to the set X. In this decomposition, $\mathcal{E}$ hides the homomorphism structure of the oracle.

Our result differs from Mosca-Ekert's in that we show that X *itself* can always be viewed as a group, with a group operation that is induced by the oracle, with no need to know the decomposition $\mathcal{E} \circ \alpha$.

It is possible to relate both results as follows. Since both Q and $\mathcal{O}$ are isomorphic to G/H, they are also mutually isomorphic. Explicitly, if β is an isomorphism from Q to G/H (this map depends on the particular decomposition $f = \mathcal{E}\circ\alpha$), then Q and $\mathcal{O}$ are isomorphic via the map $\tilde{f} \circ \beta$.

5.2.5 Decomposing finite Abelian groups

As mentioned earlier, there is a quantum algorithm for decomposing Abelian groups, due to Cheung and Mosca [84, 73]. In this section, we will introduce this problem, and present a quantum algorithm that solves it, which uses only black-box normalizer circuits supplemented with classical computation. The algorithm we give is based on Cheung-Mosca's, but it reveals some additional information about the structure of the black-box group. We will refer to it as the ***extended Cheung-Mosca's algorithm.***

The group decomposition problem

In this section, we define the **group decomposition** problem as follows. The input of the problem is a list of generators $\alpha = (\alpha_1, \cdots, \alpha_k)$ of some Abelian black-box group $\mathbf{B}$. Our task is to return a *group-decomposition table* for $\mathbf{B}$. A group-decomposition table is a tuple (α, β, A, B, c) consisting of the original string α and four additional elements:

(a) A new generating set $\beta = \beta_1, \ldots, \beta_\ell$ with the property $\mathbf{B} = \langle\beta_1\rangle \oplus \cdots \oplus \langle\beta_\ell\rangle$. We will say that these new generators are *linearly independent.*

(b) An integer vector c containing the orders of the linearly independent generators β_i.

(c) Two integer matrices A, B that relate the old and new generators as follows:

$$\left(\beta_1,\ldots,\beta_\ell\right) = \left(\alpha_1,\ldots\alpha_k\right) A, \qquad \left(\alpha_1,\ldots\alpha_k\right) = \left(\beta_1,\ldots,\beta_\ell\right) B. \tag{5.2.24}$$

This last equation should be read in multiplicative notation (as in e.g. [247]), where "vectors" of group elements are right-multiplied by matrices as follows: given the ith column a_i of A (for the left hand case), we have $\beta_i = (\alpha_1,\ldots,\alpha_k)a_i = \alpha_1^{a_i(1)}\cdots\alpha_k^{a_i(k)}$.

Our definition of the group decomposition is more general than the one given in [84, 73]. In Cheung and Mosca's formulation, the task is to find just β and c. The algorithm they give also computes the matrix A in order to find the generators β_i (cf. the next section). What is completely new in our formulation is that we ask in addition for the matrix B.

Note that a **group-decomposition table** (α,β,A,B,c) contains a lot of information about the group structure of $\mathbf{B}$. First of all, the tuple elements (a-b) tell us that $\mathbf{B}$ is isomorphic to a decomposed group $G = \mathbb{Z}_{c_1}\times\cdots\times\mathbb{Z}_{c_k}$. In addition, the matrices A and B provide us with an efficient method to re-write linear combinations of the original generators α_i as linear combinations of the new generators β_j (and vice-versa). Indeed, equation (5.2.24) implies

$$\begin{aligned}\alpha_1^{x_1}\cdots\alpha_k^{x_k} &= \left(\alpha_1,\ldots\alpha_k\right) x = \left(\beta_1,\ldots,\beta_\ell\right)(Bx), &&\text{for any } x\in\mathbb{Z}^k,\\ \beta_1^{y_1}\cdots\beta_1^{y_\ell} &= \left(\beta_1,\ldots,\beta_\ell\right) y = \left(\alpha_1,\ldots\alpha_k\right)(Ay), &&\text{for any } y\in\mathbb{Z}^\ell.\end{aligned}$$

It follows that, for any given x, the integer string $y = Bx$ (which can be efficiently computed classically) fulfills the condition $\alpha_1^{x_1}\cdots\alpha_1^{x_k} = \beta_1^{y_1}\cdots\beta_1^{y_\ell}$. (A symmetric argument proves the opposite direction.)

As we discussed earlier in the introduction, the group decomposition problem is *provably hard* for classical computers within the black-box setting, and it is at least *as hard as* Factoring (or Order Finding) for matrix groups of the form $\mathbb{Z}_N^\times$ (the latter being polynomial-time reducible to group decomposition). It can be also shown that group decomposition is also at least as hard as computing discrete logarithms, a fact that we will use in the proof of theorems 87, 89:

Lemma 85 (Multivariate discrete logarithms). *Let $\beta_1,\ldots,\beta_\ell$ be generators of some Abelian black-box group $\mathbf{B}$ with the property $\mathbf{B} = \langle\beta_1\rangle\oplus\cdots\oplus\langle\beta_\ell\rangle$. Then, the following generalized version of the discrete-logarithm problem is polynomial time reducible to group decomposition: for a given $\beta\in\mathbf{B}$, find an integer string x such that $\beta_1^{x_1}\cdots\beta_\ell^{x_\ell} = \beta$.*

Proof. Define a new set of generators for $\mathbf{B}$ by adding the element $\beta_{\ell+1} = \beta$ to the given set $\{\beta_i\}$. The array $\alpha' := (\beta_1,\ldots,\beta_\ell+1)$ defines an instance of Group Decomposition. Assume that a group decomposition table $(\alpha',(\beta_1',\ldots,\beta_m'),A',B',c')$ for this instance of the problem is given to us. We can now use the columns b_i' of the matrix B' to re-write the previous generators β_i in terms of the new ones:

$$\beta_i = (\beta_1,\ldots,\beta_{\ell+1})e_i = \left(\beta_1',\ldots,\beta_m'\right)(B'e_i) = \left(\beta_1',\ldots,\beta_m'\right) b_i' = \beta_1'^{b_i'(1)}\cdots\beta_m'^{b_i'(m)}. \tag{5.2.25}$$

Here, e_i denotes the integer vector with $e(i) = 1$ and $e(j) = 0$ elsewhere. Conditions (a-b) imply that the columns b_i' can be treated as elements of the group $G = \mathbb{Z}_{c_1'}\times\cdots\times\mathbb{Z}_{c_m'}$. Using this identification, the original discrete logarithm problem reduces to finding an integer string

$x \in G$ such that $b'_{\ell+1} = (b'_1, \ldots, b'_\ell)x = \sum x(i)b'_i$ (now in additive notation). The existence of such an x can be easily proven using that the elements $\beta_1, \ldots, \beta_\ell$ generate $\mathbf{B}$: the latter guarantees the existence of an x such that

$$\beta_{\ell+1} = (\beta_1, \ldots, \beta_\ell)x = \left(\beta'_1, \ldots, \beta'_m\right)(b'_1, \ldots, b'_\ell)x = \left(\beta'_1, \ldots, \beta'_m\right) b'_{\ell+1}, \tag{5.2.26}$$

which implies $(b'_1, \ldots, b'_\ell)x \equiv \left(\beta'_1, \ldots, \beta'_m\right) b'_{\ell+1} \bmod (c'_1, \ldots, c'_m)$. By finding such an x, we can solve the multivariate discrete problem, since $\beta_1^{x_1} \cdots \beta_\ell^{x_\ell} = \beta_1'^{b'_{\ell+1}(1)} \cdots \beta_m'^{b'_{\ell+1}(m)} = \beta_{\ell+1} = \beta$, due to (5.2.25). Finally, note that we can find x efficiently with existing deterministic classical algorithms for Group Membership in finite Abelian groups (cf. lemma 3 in [2]). $\square$

We highlight that, in order for the latter result to hold, it seems critical to use our formulation of group decomposition instead of Cheung-Mosca's. Consider again the discrete-log problem over the group $\mathbb{Z}_p^\times$ (recall Section 5.2.1). This group $\mathbb{Z}_p^\times$ is cyclic of order $p-1$ and a generating element a is given to us as part of the input of the discrete-log problem. Although it is not known how to solve this problem efficiently, Cheung-Mosca's group decomposition problem (find some linearly independent generators and their orders) can be solved effortlessly in this case, by simply returning a and $p-1$, since $\langle a \rangle = \mathbb{Z}_p^\times \cong \mathbb{Z}_{p-1}$. The crucial difference is that Cheung-Mosca's algorithm returns a factorization $\mathbb{Z}_{c_1} \times \cdots \times \mathbb{Z}_{c_\ell}$ of $\mathbf{B}$, but it cannot be used to convert elements between the two representations efficiently (one direction is easy; the other requires computing discrete logarithms). In our formulation, the matrices A, B provide such a method.

Quantum algorithm for group decomposition

We now present a quantum algorithm that solves the group decomposition problem.

Algorithm 86 (Extended Cheung-Mosca's algorithm).

Input. A list of generators $\alpha = (\alpha_1, \ldots, \alpha_k)$ of an Abelian black-box group $\mathbf{B}$[13].

Output. A group decomposition table (α, β, A, B, c).

1. Use the order finding algorithm (comprising normalizer circuits over $\mathbb{Z} \times \mathbf{B}$ and classical postprocessing) to obtain the orders d_i of the generators α_i. Then, compute (classically) and store their least common multiplier $d = \mathrm{lcm}(d_1, \ldots, d_k)$.

2. Define the function $f : \mathbb{Z}_d^k \to \mathbf{B}$ as $f(x) = \alpha_1^{x(1)} \cdots \alpha_k^{x(k)}$, which is a group homomorphism and hides the subgroup $\ker f$ (its own kernel). Apply the Abelian HSP algorithm to compute a set of generators $h_1, \ldots, h_m$ of $\ker f$. This round uses normalizer circuits over $\mathbb{Z}_d^k \times \mathbf{B}$ and classical post-processing (cf. Section 5.2.4).

3. Given the generators h_i of $\ker f$ one can classically compute a $k \times \ell$ matrix A (for some ℓ) such that $(\beta_1, \ldots, \beta_\ell) = (\alpha_1, \ldots, \alpha_k)A$ is a system of linearly independent

[13]Cheung and Mosca's original presentation first used Shor's algorithm to decompose B into Sylow p-subgroups, and then performed the group decomposition on these subgroups. As they commented, that step was not strictly necessary, although it would reduce the computational resources required. Also, their algorithm does not receive a list of generators as an input, but this is not a big difference since such a list can always be computed in probabilistic classical polynomial-time for a uniquely encoded black-box group.

generators [73, theorem 7]. β, A and the orders c_i of the β_is (computed again via an order-finding subroutine) will form part of the output.

4. Finally, we show how to classically compute a valid relationship matrix B. (This step is not part of Cheung-Mosca's original algorithm.) The problem reduces to finding a $k \times \ell$ integer matrix X with two properties:

   (a) X is a solution to the equation $(\alpha_1, \ldots, \alpha_k)X = (\alpha_1, \ldots, \alpha_k)$. Equivalently, every column x_i of X is equal (modulo d) to some element of the coset $e_i + \ker f \subset \mathbb{Z}_d^k$.

   (b) Every column x_i is an element of the image of the matrix A.

   It is easy to see that a matrix X fulfilling (a-b) always exists, since for any α_i, there exists some y_i such that $\alpha_i = (\beta_1, \ldots, \beta_\ell)y_i$ (because the β_is generate the group). It follows that $\alpha_i = (\alpha_1, \ldots, \alpha_k)(Ay_i)$. Then, the matrix with columns $x_i = Ay_i$ has the desired properties.

   Our existence proof for X is constructive, and tells us that X can be computed in quantum polynomial time by solving a multivariate discrete logarithm problem (Lemma 85). However, X can be more straightforwardly obtained classically, by reducing the problem to a **system of linear equations over Abelian groups** (Section 4.5). Let H be a matrix formed column-wise by the generators h_i of $\ker f$. By construction, the image of the map $H : \mathbb{Z}_d^m \to \mathbb{Z}_d^k$ fulfills $\mathrm{im} H = \ker f$. Properties (a-b) imply that the ith column x_i of X must be a particular solution to the equations $x_i = Ay_i$ with $y_i \in \mathbb{Z}^\ell$ and $x_i = e_i + Hz_i \bmod d$, with $z_i \in \mathbb{Z}_d^m$. These equations can be equivalently written as a system of linear equations over $\mathbb{Z}_d^m \times \mathbb{Z}^\ell$:

$$\begin{pmatrix} A & -H \end{pmatrix} \begin{pmatrix} y_i \\ z_i \end{pmatrix} = e_i \mod d, \qquad (y_i, z_i) \in \mathbb{Z}_d^m \times \mathbb{Z}^\ell, \tag{5.2.27}$$

   This system of equations can be solved in classical polynomial time using the mtehods of Section 4.5. The matrix X can then be constructed column wise taking $x_i = Ay_i$.

   Finally, given such an X, it is easy to find a valid B by computing a Hurt-Waid integral pseudo-inverse $A^\#$ of A [148, 112]:

$$\alpha = \alpha X = \alpha(AA^\#)X = (\alpha A)(A^\# X) = (\beta_1, \ldots, \beta_\ell)(A^\# X). \tag{5.2.28}$$

   In the third step, we used that $A^\#$ acts as the inverse of A on inputs $x \in \mathbb{Z}^k$ that live in the image of A [148]. Since integral pseudo-inverses can be computed efficiently using the Smith normal form (see Appendix C.3), we finally set $B := A^\# X$.

5.3 Simulation of black-box normalizer circuits

Our results so far show that the computational power of normalizer circuits over black-box groups (supplemented with classical pre- and post- processing) is *strikingly high*: they can solve several problems believed to be classically intractable and render the RSA, Diffie-Hellman, and elliptic curve public-key cryptosystems vulnerable. In contrast, standard normalizer circuits, which are associated with Abelian groups that are *explicitly decomposed*, can be efficiently simulated classically, as we've seen in Chapter 4 (see Theorem 59).

It is natural to wonder at this point where the computational power of black-box normalizer circuits originates. In this section, we will argue that the hardness of simulating black-box normalizer circuits resides *precisely* in the hardness of decomposing black-box Abelian groups. An equivalence is suggested by the fact that we can use these circuits to solve the group decomposition problem and, in turn, when the group is decomposed, the techniques of Chapter 4 render these circuits classically simulable. In this sense, then, the *quantum speedup* of such circuits appears to be completely encapsulated in the group decomposition algorithm. This intuition can be made precise and be stated as a theorem.

Theorem 87 (Simulation of black-box normalizer circuits). *Black-box normalizer circuits can be efficiently simulated classically using the stabilizer formalism over Abelian groups if a subroutine for solving the group-decomposition problem is provided as an oracle.*

The proof of this theorem is the subject of Section D.2 in the appendix.

Since normalizer circuits can solve the group decomposition problem (Section 5.2.5), we obtain that this problem is complete for the associated normalizer-circuit complexity class, which we now define.

Definition 88 (Black-Box Normalizer). The complexity class **Black-Box Normalizer** is the set of oracle problems that can be solved with bounded error by at most polynomially many rounds of efficient black-box normalizer circuits (as defined in Section 3.7), with polynomial-sized classical computation interspersed between. In other words, if N is an oracle that given an efficient (poly-sized) black-box normalizer circuit as input, samples from its output distribution, then

$$\textbf{Black-Box Normalizer} = BPP^{N}. \tag{5.3.1}$$

Corollary 2 (Completeness of group decomposition). *Group decomposition is a complete problem for the complexity class* **Black-Box Normalizer** *under classical polynomial-time Turing reductions.*

We stress that Theorem 87 tells us even more than the completeness of group decomposition. As we discussed in the introduction, an oracle for group decomposition gives us an efficient classical algorithm to simulate Shor's factoring and discrete-log algorithm (and all the others) *step-by-step* with a stabilizer-picture approach "à la Gottesman-Knill".

We also highlight that Theorem 87 can be restated as a *no-go theorem* for finding new quantum algorithms based on black-box normalizer circuits.

Theorem 89 (No-go theorem for new quantum algorithms). *It is not possible to find "fundamentally new" quantum algorithms within the class of black-box normalizer circuits studied in this chapter, in the sense that any new algorithm would be efficiently simulable using the extended Cheung-Mosca algorithm and classical post-processing.*

This theorem tells us that black-box normalizer circuits cannot give exponential speedups over classical circuits that are not already covered by the extended Cheung-Mosca algorithm; the theorem may thus have applications to algorithm design.

Note, however, that this no-go theorem says nothing about other possible *polynomial* speed-ups for black-box normalizer circuits; there may well be other normalizer circuits that are polynomially faster, conceptually simpler, or easier to implement than the extended

Cheung-Mosca algorithm. Our theorem neither denies that investigating black-box normalizer could be of pedagogical or practical value if, e.g., this led to new interesting complete problems for the class **Black-Box Normalizer**.

Finally, we note that Theorem 87 can be extended to the general Abelian hidden subgroup problem to show that the quantum algorithm for the Abelian HSP becomes efficiently classically simulable if an algorithm for decomposing the oracular group $\mathcal{O}$ is given to us (cf. Section 5.2.4 and refer to Appendix D.3 for a proof). We discuss some implications of this fact in the next sections.

5.4 Universality of short quantum circuits

Since all problems in **Black-Box Normalizer** are solvable by the extended Cheung-Mosca quantum algorithm (supplemented with classical processing), the structure of said quantum algorithm allows us to state the following:

Theorem 90 (Universality of short normalizer circuits). *Any problem in the class* **Black-Box Normalizer** *can be solved by a quantum algorithm composed of polynomially-many rounds of* short *normalizer circuits, each with at most a* **constant** *number of normalizer gates, and additional classical computation. More precisely, in every round, normalizer circuits containing two quantum Fourier transforms and one automorphism gate (and no quadratic phase gate) are already sufficient.*

Proof. This result follows immediately form the fact that group decomposition is complete for this class (Theorem 2) and from the structure of the extended Cheung-Mosca quantum algorithm with this problem, which has precisely this structure. □

Similarly to Theorem 87, Theorem 90 can be extended to the general Abelian HSP setting. For details, we refer the reader to Appendix D.3.

We find the latter result is insightful, in that it actually explains a somewhat intriguing feature present in Deutsch's, Simon's, Shor's and virtually all known quantum algorithms for solving Abelian hidden subgroup problems: they all contain at most two quantum Fourier transforms! Clearly, it follows from this theorem than no more than two are enough.

Also, Theorem 90 tells us that it is actually pretty *useless* to use logarithmically or polynomially long sequences of quantum Fourier transforms for solving Abelian hidden subgroup problems, since just two of them suffice[14]. In this sense, the Abelian HSP quantum algorithm uses an *asymptotically optimal* number of quantum Fourier transforms. Furthermore, the normalizer-gate depth of this algorithm is optimal in general.

5.5 Other Complete problems

We end this chapter by giving two other complete problems for the complexity class **Black Box Normalizer**.

Theorem 91 (Hidden kernel problem is complete). *Let the Abelian hidden kernel problem (Abelian HKP) be the subcase of the hidden subgroup problem where the oracle function f is a group homomorphism from a group of the form $G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_b}$*

[14]This last comment does not imply that building up sequences of Fourier transforms is useless in general. On the contrary, this can be actually be useful, e.g., in QMA amplification [248].

into a black-box group $\mathbf{B}$. *This problem is complete for* **Black Box Normalizer** *under polynomial-time Turing reductions.*

Proof. Clearly group decomposition reduces to this problem, since the quantum steps of the extended Cheung-Mosca algorithm algorithm (steps 1 and 3) are solving instances of the Abelian kernel problem. Therefore, the Abelian HKP problem is hard for **Black Box Normalizer**.

Moreover, Abelian HKP can be solved with the general Abelian HSP quantum algorithm, which manifestly becomes a black-box normalizer circuit for oracle functions f that are group homomorphisms onto black-box groups. This implies that Abelian HKP is inside **Black Box Normalizer**, and therefore, it is complete.

Note. Although we originally stated the Abelian HSP for finite groups, one can first apply the order-finding algorithm to compute a multiple d of the orders of the elements $f(e_i)$, where e_i are the canonical generators of G. This can be used to reduce the original HKP problem to a simplified HKP over the group $\mathbb{Z}_d^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_b}$ $\square$

The latter result can be extended to the HSP setting to show that the Abelian hidden subgroup problem is polynomial-time equivalent to decomposing groups of the form $\mathcal{O}$ (cf. Appendix D.3).

Theorem 92 (System of linear equations over groups). *Let* α *be a group homomorphism from a group* $G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_b}$ *onto a black-box group* $\mathbf{B}$. *An instance of a linear system of equations over* G *and* $\mathbf{B}$ *is given by* α *and an element* $\mathbf{b} \in \mathbf{B}$. *Our task is to find a general* (x_0, K) *solution to the equation*

$$\alpha(x) = \mathbf{b}, \quad x \in G,$$

where x_0 *is any particular solution and* K *is a generating set of the kernel of* α. *This problem is complete for* **Black Box Normalizer** *under polynomial-time Turing reductions.*

Proof. Clearly, this problem is hard for our complexity class, since the Abelian hidden kernel problem reduces to finding K.

Moreover, this problem can be solved with black-box normalizer circuits and classical computation, proving its completeness. First, we find a decomposition $\mathbf{B} = \bigoplus \langle \beta_i \rangle \cong H = \mathbb{Z}_{c_1} \times \cdots \times \mathbb{Z}_{c_\ell}$ with black-box normalizer circuits. Second, we recycle the de-black-boxing idea from the proof of Theorem 87 to compute a matrix representation of α, and solve the multivariate discrete logarithm problem $\mathbf{b} = \beta_1^{b(1)} \cdots \beta_\ell^{b(\ell)}$, $b \in H$, either with black-box normalizer circuits or classically (recall Section 5.2.5). The original system of equations can now be equivalently written as $Ax = b \pmod{H}$. A general solution of this system can be computed with classical algorithms given in Section 4.5. $\square$

Appendix A

Proofs of results in Chapter 2

A.1 Proof of the adversary lower bound for $B(f)$ (Theorem 20)

Before we give the proof of the general result that $B(f) = \Omega(Q(f)^2)$ (Theorem 20)), we will illustrate the proof by means of an example, the special case where f is the AND function.

Theorem 93. *For $\delta < 1/10$, $B_{\epsilon,\delta}(AND) = \Omega(\frac{N}{\epsilon})$.*

Proof. Let $|\psi_t^0\rangle$ be the unnormalized state of the algorithm with $x = 1^n$, and $|\psi_t^k\rangle$ be the unnormalized state with $x = 1\cdots101\cdots1$, $x_k = 0$, right before the $(t+1)$-th call to M_x. Then

$$|\psi_{t+1}^x\rangle = U_{t+1}M_x|\psi_t^x\rangle \tag{A.1.1}$$

for some unitary U_{t+1}. For ease of notation, we'll write $M_0 \equiv M_{1^n}$ and $M_k = M_{1\cdots101\cdots1}$, where the k-th bit is 0 in the latter case. When acting on the control and index bits,

$$\begin{aligned} M_0 &= \sum_{i=1}^{N} |0,i\rangle\langle 0,i| \\ M_k &= \sum_{i=1}^{N} |0,i\rangle\langle 0,i| + |1,k\rangle\langle 1,k|. \end{aligned} \tag{A.1.2}$$

Since the M_i's are projectors, $M_i^2 = M_i$. Define

$$\epsilon_t^i = \langle\psi_t^i|(I - M_i)|\psi_t^i\rangle, \quad i = 0, 1, \cdots, N. \tag{A.1.3}$$

Note that $\langle\psi_{t+1}^i|\psi_{t+1}^i\rangle = \langle\psi_t^i|M_i^2|\psi_t^i\rangle = \langle\psi_t^i|M_i|\psi_t^i\rangle = \langle\psi_t^i|\psi_t^i\rangle - \epsilon_t^i$, for all $i = 0, \cdots, N$ (including 0!), and hence

$$\sum_{t=0}^{T-1} \epsilon_t^i = \langle\psi_0^i|\psi_0^i\rangle - \langle\psi_T^i|\psi_T^i\rangle \le \epsilon. \tag{A.1.4}$$

We now define the progress function. Let

$$W_t^k = \langle\psi_t^0|\psi_t^k\rangle \tag{A.1.5}$$

and let the progress function be a sum over W^k's:

$$W_t = \sum_{k=1}^{N} W_t^k = \sum_{k=1}^{N} \langle \psi_t^0 | \psi_t^k \rangle. \tag{A.1.6}$$

We can lower bound the total change in the progress function by (see [33] for a proof; their proof equally applies to unnormalized states)

$$W_0 - W_T \geq (1 - 2\sqrt{\delta(1-\delta)})N. \tag{A.1.7}$$

We now proceed to upper bound $W_0 - W_T$. Note that

$$\begin{aligned} W_t^k - W_{t+1}^k &= \langle \psi_t^0 | \psi_t^k \rangle - \langle \psi_t^0 | M_0 M_k \big| \psi_t^k \big\rangle \\ &= \langle \psi_t^0 | (I - M_0) M_k \big| \psi_t^k \big\rangle + \langle \psi_t^0 | M_0 (I - M_k) \big| \psi_t^k \big\rangle + \langle \psi_t^0 | (I - M_0)(I - M_k) \big| \psi_t^k \big\rangle \end{aligned} \tag{A.1.8}$$

and since $M_0(I - M_k) = 0$, $(I - M_0)M_k = |1,k\rangle\langle 1,k|$, we have

$$\begin{aligned} W_t^k - W_{t+1}^k &\leq \langle \psi_t^0 | 1,k \rangle \langle 1,k | \psi_t^k \rangle + \| (I - M_0) | \psi_t^0 \rangle \| \, \Big\| (I - M_k) \big| \psi_t^k \big\rangle \Big\| \\ &\leq \| \langle 1,k | \psi_t^0 \rangle \| + \sqrt{\epsilon_t^0 \epsilon_t^k}. \end{aligned} \tag{A.1.9}$$

where we used A.1.3. Summing over k and t, we obtain

$$\begin{aligned} W_0 - W_T &\leq \sum_{t=0}^{T-1} \sum_{k=1}^{N} \left[\| \langle 1,k | \psi_t^0 \rangle \| + \sqrt{\epsilon_t^0 \epsilon_t^k} \right] \\ &\leq \sqrt{TN} \sqrt{\sum_{t=0}^{T-1} \sum_{k=1}^{N} \langle \psi_t^0 | 1,k \rangle \langle 1,k | \psi_t^0 \rangle} + \sum_{t=0}^{T-1} \sum_{k=1}^{N} \frac{\epsilon_t^0 + \epsilon_t^k}{2} \\ &\leq \sqrt{TN} \sqrt{\sum_{t=0}^{T-1} \langle \psi_t^0 | (I - M_0) | \psi_t^0 \rangle} + N\epsilon \\ &\leq \sqrt{TN \sum_{t=0}^{T-1} \epsilon_t^0} + N\epsilon \\ &\leq \sqrt{\epsilon TN} + N\epsilon \end{aligned} \tag{A.1.10}$$

where in the second line we used Cauchy-Schwarz and the AM-GM inequality. Combined with $W_0 - W_T \geq (1 - 2\sqrt{\delta(1-\delta)})N$ (Eq. A.1.7), this immediately gives us

$$T \geq \frac{(1 - 2\sqrt{\delta(1-\delta)} - \epsilon)^2 N}{\epsilon}. \tag{A.1.11}$$

□

We now proceed to prove the general result. This proof follows the presentation given in A. Childs's online lecture notes [249], which we found quite illuminating.

Theorem 20. *For all functions f with boolean input alphabet, and numbers ϵ, δ satisfying $0 < \epsilon \leq \delta \leq 1/10$,*

$$B_{\epsilon,\delta}(f) = \Omega(Q_{0.01}(f)^2/\epsilon). \tag{A.1.12}$$

Proof. We will show that $B_{\epsilon,\delta}(f)$ is lower bounded by $\Omega(\mathrm{Adv}^{\pm}(f)^2/\epsilon)$, where $\mathrm{Adv}^{\pm}(f)$ is the generalized (i.e. allowing negative weights) adversary bound [34] for f. We can then derive our theorem from the result [37] that $Q(f) = O(\mathrm{Adv}^{\pm}(f))$.

We generalize the bound on the $f = AND$ case to an adversary bound for $B_{\epsilon,\delta}$ on arbitrary f. Define the projectors

$$\begin{aligned} \Pi_0 &= \sum_{i=1}^{N} |0,i\rangle\langle 0,i| \\ \Pi_i &= |1,i\rangle\langle 1,i|, \quad i = 1, \cdots, n. \end{aligned} \tag{A.1.13}$$

It is clear that

$$\Pi_0 + \sum_{i=1}^{N} \Pi_i = I. \tag{A.1.14}$$

Note that $M_x = CP_{x,0}$ is

$$M_x = \Pi_0 + \sum_{i:x_i=0} \Pi_i. \tag{A.1.15}$$

Define $|\psi_t^x\rangle$ as the state of the algorithm right before the $(t+1)$-th query with input x; then

$$|\psi_{t+1}^x\rangle = U_{t+1} M_x |\psi_t^x\rangle \tag{A.1.16}$$

for some unitary U_{t+1}. Now if we let

$$\epsilon_t^x = \langle\psi_t^x|(I - M_x)|\psi_t^x\rangle \tag{A.1.17}$$

then it follows that $\langle\psi_t^x|\psi_t^x\rangle - \langle\psi_{t+1}^x|\psi_{t+1}^x\rangle = \epsilon_t^x$, and thus

$$\sum_{t=0}^{T-1} \epsilon_t^x = \langle\psi_0^x|\psi_0^x\rangle - \langle\psi_T^x|\psi_T^x\rangle \leq \epsilon. \tag{A.1.18}$$

We proceed to define the progress function. Let S be the set of allowable input strings x. Let Γ be an *adversary matrix*, i.e. an $S \times S$ matrix such that

1. $\Gamma_{xy} = \Gamma_{yx} \quad \forall x, y \in S$; and
2. $\Gamma_{xy} = 0 \quad$ if $f(x) = f(y)$.

Let a be the normalized eigenvector of Γ with eigenvalue $\pm\|\Gamma\|$, where $\pm\|\Gamma\|$ is the largest (by absolute value) eigenvalue of Γ. Define the progress function

$$W_t = \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \langle\psi_t^x|\psi_t^y\rangle. \tag{A.1.19}$$

For $\epsilon \leq \delta < 1/10$ we have that[1] (see [34] for a proof; their proof applies equally well to unnormalized states)

$$|W_0 - W_T| \geq (1 - 2\sqrt{\delta(1-\delta)} - 2\delta)\|\Gamma\| \tag{A.1.20}$$

We now proceed to upper bound $|W_0 - W_T| \leq \sum_t |W_t - W_{t-1}|$. Note that

$$\begin{aligned}
W_t - W_{t+1} &= \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \left(\langle\psi_t^x|\psi_t^y\rangle - \langle\psi_{t+1}^x|\psi_{t+1}^y\rangle\right) \\
&= \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \left(\langle\psi_t^x|\psi_t^y\rangle - \langle\psi_t^x|M_x M_y|\psi_t^y\rangle\right) \\
&= \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \left(\langle\psi_t^x|(I-M_x)M_y|\psi_t^y\rangle + \langle\psi_t^x|M_x(I-M_y)|\psi_t^y\rangle\right) \\
&\quad + \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \langle\psi_t^x|(I-M_x)(I-M_y)|\psi_t^y\rangle
\end{aligned} \tag{A.1.21}$$

We bound the three terms separately. For the first two terms, use

$$\begin{aligned}
(I - M_x)M_y &= \sum_{i:x_i=1,y_i=0} \Pi_i \\
&= (I - M_x)\sum_{i:x_i\neq y_i} \Pi_i
\end{aligned} \tag{A.1.22}$$

Define the $S \times S$ matrix Γ_i as

$$\Gamma_i = \begin{cases} \Gamma_{xy} & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i \end{cases} \tag{A.1.23}$$

The first term of A.1.21 is

$$\begin{aligned}
\sum_{x,y\in S}\sum_{i:x_i\neq y_i} \Gamma_{xy} a_x^* a_y \langle\psi_t^x|(I-M_x)\Pi_i|\psi_t^y\rangle &= \sum_{x,y\in S}\sum_{i=1}^{N} (\Gamma_i)_{xy}\, a_x^* a_y \langle\psi_t^x|(I-M_x)\Pi_i|\psi_t^y\rangle \\
&= \sum_{i=1}^{N} \operatorname{tr}(Q_i \Gamma_i \tilde{Q}_i^\dagger)
\end{aligned} \tag{A.1.24}$$

where

$$Q_i = \sum_{x\in S} a_x \Pi_i |\psi_t^x\rangle\langle x| \tag{A.1.25}$$

$$\tilde{Q}_i = \sum_{x\in S} a_x \Pi_i (I - M_x)|\psi_t^x\rangle\langle x|. \tag{A.1.26}$$

Although both Q_i and $\tilde{Q}_i$ depend on t, we suppress the t dependence in the notation. Similarly, the second term of A.1.21 is equal to $\sum_{i=1}^{N} \operatorname{tr}(\tilde{Q}_i \Gamma_i Q_i^\dagger)$. We can also rewrite the

[1] As described in [34], the 2δ term can be removed if the output is boolean (0 or 1).

third term of A.1.21 as

$$\sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \langle \psi_t^x | (I - M_x)(I - M_y) | \psi_t^y \rangle = \operatorname{tr}(Q' \Gamma Q'^\dagger) \tag{A.1.27}$$

where

$$Q' = \sum_{x\in S} a_x (I - M_x) |\psi_t^x\rangle\langle x|. \tag{A.1.28}$$

Therefore, adding absolute values,

$$|W_t - W_{t+1}| \leq \sum_{i=1}^{N} \left[\left| \operatorname{tr}(Q_i \Gamma_i \tilde{Q}_i^\dagger) \right| + \left| \operatorname{tr}(\tilde{Q}_i \Gamma_i Q_i^\dagger)) \right| \right] + \left| \operatorname{tr}(Q' \Gamma Q'^\dagger) \right| \tag{A.1.29}$$

To continue, we need the following lemma:

Lemma 94. *For any $m, n > 0$ and matrices $X \in \mathbb{C}^{m\times n}$, $Y \in \mathbb{C}^{n\times n}$, $Z \in \mathbb{C}^{n\times m}$, we have $|\operatorname{tr}(XYZ)| \leq \|X\|_F \|Y\| \|Z\|_F$. Here $\|\cdot\|$ and $\|\cdot\|_F$ denote the spectral norm and Frobenius norm, respectively.*

This lemma can be proved by using that $|\operatorname{tr}(XYZ)| \leq \|Y\| \|ZX\|_{tr}$ and $\|ZX\|_{tr} \leq \|X\|_F \|Z\|_F$, which follows from [250, Exercise IV.2.12 and Corollary IV.2.6]. A more accessible proof is found online at [249].

Then by Lemma 94,

$$\sum_{i=1}^{N} \left| \operatorname{tr}(Q_i \Gamma_i \tilde{Q}_i^\dagger) \right| \leq \sum_{i=1}^{N} \|\Gamma_i\| \|Q_i\|_F \|\tilde{Q}_i\|_F \tag{A.1.30}$$

Since

$$\begin{aligned} \sum_{i=1}^{N} \|Q_i\|_F^2 &= \sum_{i=1}^{N} \sum_{x\in S} |a_x|^2 \, \|\Pi_i |\psi_t^x\rangle\|^2 \\ &= \sum_{x\in S} |a_x|^2 \langle \psi_t^x | \sum_{i=1}^{N} \Pi_i |\psi_t^x\rangle \\ &\leq \sum_{x\in S} |a_x|^2 \\ &= 1 \end{aligned} \tag{A.1.31}$$

and

$$\sum_{i=1}^{N} \|\tilde{Q}_i\|_F^2 = \sum_{i=1}^{N}\sum_{x\in S} |a_x|^2 \, \|\Pi_i(I - M_x)|\psi_t^x\rangle\|^2$$
$$= \sum_{x\in S} |a_x|^2 \langle\psi_t^x|(I - M_x)\left(\sum_{i=1}^{N}\Pi_i\right)(I - M_x)|\psi_t^x\rangle$$
$$\leq \sum_{x\in S} |a_x|^2 \langle\psi_t^x|(I - M_x)|\psi_t^x\rangle$$
$$= \sum_{x\in S} |a_x|^2 \epsilon_t^x \tag{A.1.32}$$

we have, by Cauchy-Schwarz,

$$\sum_{i=1}^{N} \|Q_i\|_F \|\tilde{Q}_i\|_F \leq \sqrt{\sum_{x\in S} |a_x|^2 \epsilon_t^x} \tag{A.1.33}$$

Therefore by A.1.30 and A.1.33,

$$\sum_{i=1}^{N} \left|\mathrm{tr}(Q_i \Gamma_i \tilde{Q}_i^\dagger)\right| \leq \sqrt{\sum_{x\in S} |a_x|^2 \epsilon_t^x} \max_{i\in[N]} \|\Gamma_i\|. \tag{A.1.34}$$

Similartly for $\mathrm{tr}(Q'\Gamma Q'^\dagger)$, we have

$$\|Q'\|_F^2 = \sum_{x\in S} |a_x|^2 \, \|(I - M_x)|\psi_t^x\rangle\|^2$$
$$= \sum_{x\in S} |a_x|^2 \langle\psi_t^x|(I - M_x)|\psi_t^x\rangle$$
$$= \sum_{x\in S} |a_x|^2 \epsilon_t^x \tag{A.1.35}$$

and using Lemma 94,

$$\mathrm{tr}(Q'\Gamma Q'^\dagger) \leq \|Q'\|_F^2 \|\Gamma\| \tag{A.1.36}$$
$$= \sum_{x\in S} |a_x|^2 \epsilon_t^x \|\Gamma\| \tag{A.1.37}$$

Thus continuing from A.1.29, we have that

$$|W_t - W_{t+1}| \leq 2\sqrt{\sum_{x\in S} |a_x|^2 \epsilon_t^x} \max_{i\in[N]} \|\Gamma_i\| + \sum_{x\in S} |a_x|^2 \epsilon_t^x \|\Gamma\| \tag{A.1.38}$$

Finally, if we sum the above over t we obtain

$$|W_0 - W_T| \leq 2 \max_{i\in[N]} \|\Gamma_i\| \sum_{t=0}^{T-1} \sqrt{\sum_{x\in S} |a_x|^2 \epsilon_t^x} + \sum_{t=0}^{T-1}\sum_{x\in S} |a_x|^2 \epsilon_t^x \|\Gamma\| \tag{A.1.39}$$

The first term can be bounded using Cauchy-Schwarz:

$$\sum_{t=0}^{T-1}\sqrt{\sum_{x\in S}|a_x|^2\epsilon_t^x} \leq \sqrt{T\sum_{t=0}^{T-1}\sum_{x\in S}|a_x|^2\epsilon_t^x} \leq \sqrt{\epsilon T} \tag{A.1.40}$$

where we used $\sum_t \epsilon_t^x \leq \epsilon$ and $\sum_x |a_x|^2 = 1$. The second term can be summed easily:

$$\sum_{t=0}^{T-1}\sum_{x\in S}|a_x|^2\epsilon_t^x\|\Gamma\| \leq \sum_{x\in S}|a_x|^2\epsilon\|\Gamma\| = \epsilon\|\Gamma\|. \tag{A.1.41}$$

Therefore

$$|W_0 - W_T| \leq 2\sqrt{\epsilon T}\max_{i\in[N]}\|\Gamma_i\| + \epsilon\|\Gamma\|. \tag{A.1.42}$$

Combined with our lower bound $|W_0 - W_T| \geq (1 - 2\sqrt{\delta(1-\delta)} - 2\delta)\|\Gamma\|$, this immediately gives

$$T \geq \frac{(1 - 2\sqrt{\delta(1-\delta)} - 2\delta - \epsilon)^2}{4\epsilon}\frac{\|\Gamma\|^2}{\max_{i\in[N]}\|\Gamma_i\|^2}. \tag{A.1.43}$$

Recalling that [34]

$$\mathrm{Adv}^{\pm}(f) = \max_{\Gamma}\frac{\|\Gamma\|}{\max_{i\in[N]}\|\Gamma_i\|}, \tag{A.1.44}$$

we obtain[2]

$$T \geq \frac{(1 - 2\sqrt{\delta(1-\delta)} - 2\delta - \epsilon)^2}{4\epsilon}\mathrm{Adv}^{\pm}(f)^2. \tag{A.1.45}$$

We now use the tight characterization of the quantum query complexity by the general weight adversary bound:

Theorem 95 ([37, Theorem 1.1]). *Let $f : D \to E$, where $D \subseteq \{0,1\}^N$. Then $Q_{0.01}(f) = O(\mathrm{Adv}^{\pm}(f))$.*

Combined with our result above, we obtain

$$B_{\epsilon,\delta}(f) = \Omega\left(\frac{Q_{0.01}(f)^2}{\epsilon}\right). \tag{A.1.46}$$

□

[2]For boolean output (0 or 1) the 2δ term can be dropped, as we previously noted (Footnote 1).

A.2 Proof of Theorem 24

We restate and prove Theorem 24:

Theorem 24. *Let $f : D \to E$, where $D \subseteq \{0,1\}^N$. Suppose there is a classical randomized query algorithm $\mathcal{A}$, that makes at most T queries, and evaluates f with bounded error. Let the query results of $\mathcal{A}$ on random seed $s_{\mathcal{A}}$ be $x_{p_1}, x_{p_2}, \cdots, x_{p_{\tilde{T}(x)}}$, $\tilde{T}(x) \le T$, where x is the hidden query string.*

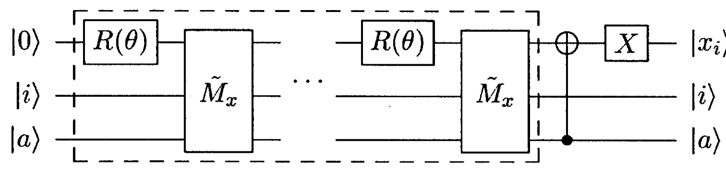
Suppose there is another (not necessarily time-efficient) randomized algorithm $\mathcal{G}$, with random seed $s_{\mathcal{G}}$, which takes as input $x_{p_1}, \cdots, x_{p_{t-1}}$ and $s_{\mathcal{A}}$, and outputs a guess for the next query result x_{p_t} of $\mathcal{A}$. Assume that $\mathcal{G}$ makes no more than an expected total of G mistakes (for all inputs x). In other words,

$$\mathbf{E}_{s_{\mathcal{A}}, s_{\mathcal{G}}} \left\{ \sum_{t=1}^{\tilde{T}(x)} \left| \mathcal{G}(x_{p_1}, \cdots, x_{p_{t-1}}, s_{\mathcal{A}}, s_{\mathcal{G}}) - x_{p_t} \right| \right\} \le G \quad \forall x. \tag{A.2.1}$$

Note that $\mathcal{G}$ is given the random seed $s_{\mathcal{A}}$ of $\mathcal{A}$, so it can predict the next query index of $\mathcal{A}$. Then $B_\epsilon(f) = O(TG/\epsilon)$, and thus (by Theorem 17) $Q(f) = O(\sqrt{TG})$.

Proof. For the purposes of this proof, we use the characterization of B by the modified bomb construction given in Section 2.5.1. This proof is substantially similar to that of theorem 19.

The following circuit finds x_i with zero probability of explosion if $x_i = a$, and with an $O(1/L)$ probability of explosion if $x_i \ne a$ (in both cases the value of x_i found by the circuit is always correct):



L times in total (A.2.2)

where $\theta = \pi/(2L)$ for some large number L to be picked later, and

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{A.2.3}$$

The boxed part of the circuit is then simply $[\tilde{M}_x(R(\theta)\otimes I \otimes I)]^L$, applied to the state $|0,i,a\rangle$. We can analyze this circuit by breaking into cases:

- If $x_i = a$, then $\tilde{M}_x|\psi\rangle|i,a\rangle = |\psi\rangle|i,a\rangle$ for any state $|\psi\rangle$ in the control register. Thus the $\tilde{M}_x$'s act as identities, and the circuit simply applies the rotation $R(\theta)^L = R(\pi/2)$ to the control register, rotating it from 0 to 1. We thus obtain the state $|1,i,a\rangle$; the final CNOT and X gates add $a \oplus 1 = x_i \oplus 1$ to the first register, giving $|x_i, i, a\rangle$.
- If $x_i \ne a$, then

$$\tilde{M}_x|0,i,a\rangle = |0,i,a\rangle, \quad \tilde{M}_x|1,i,a\rangle = 0 \quad (\text{for } x_i \ne a) \tag{A.2.4}$$

Therefore after each rotation $R(\theta)$, the projection $\tilde{M}_x$ projects the control qubit back to 0:

$$\tilde{M}_x(R(\theta)\otimes I\otimes I)|0,i,a\rangle = \tilde{M}_x(\cos\theta|0\rangle+\sin\theta|1\rangle)|i,a\rangle = \cos\theta|0,i,a\rangle \quad (\text{for } x_i\neq a) \tag{A.2.5}$$

In this case the effect of $\tilde{M}_x(R(\theta)\otimes I\otimes I)$ is to shrink the amplitude by $\cos(\theta)$; L applications results in the state $\cos^L(\theta)|0,i,a\rangle$. The final CNOT and X gates add $a\oplus 1 = x_i$ to the first register, giving $|x_i,i,a\rangle$.

The probability of explosion is 0 if $x_i = a$. If $x_i\neq a$, the probability of explosion is

$$1-\cos^{2L}\left(\frac{\pi}{2L}\right)\leq\frac{\pi^2}{4L}. \tag{A.2.6}$$

Pick

$$L=\left\lceil\frac{\pi^2 G}{4\epsilon}\right\rceil. \tag{A.2.7}$$

Then the probability of explosion is 0 if $x_i = a$, and no more than ϵ/G if $x_i\neq a$. If the bomb does not explode, then the circuit *always* finds the correct value of x_i.

We now construct the bomb query algorithm based on $\mathcal{A}$ and $\mathcal{G}$. The bomb query algorithm follows $\mathcal{A}$, with each classical query replaced by the above construction. There are no more than $TL\approx\pi^2TG/(4\epsilon)$ bomb queries. At each classical query, we pick the guess a to be the guess provided by $\mathcal{G}$. The bomb only has a chance of exploding if the guess is incorrect; hence for all x, the total probability of explosion is no more than

$$\frac{\epsilon}{G}\mathbf{E}_{s_{\mathcal{A}},s_{\mathcal{G}}}\left\{\sum_{t=1}^{\tilde{T}(x)}\left|\mathcal{G}(x_{p_1},\cdots,x_{p_{t-1}},s_{\mathcal{A}},s_{\mathcal{G}})-x_{p_t}\right|\right\}\leq\epsilon \tag{A.2.8}$$

Thus replacing the classical queries of $\mathcal{A}$ with our construction gives a bomb query algorithm with probability of explosion no more than ϵ; aside from the probability of explosion, this bomb algorithm makes no extra error over the classical algorithm $\mathcal{A}$. The number of queries this algorithm uses is

$$\tilde{B}_{\epsilon,\delta+\epsilon}(f)\leq\left\lceil\frac{\pi^2G}{4\epsilon}\right\rceil T, \tag{A.2.9}$$

where δ is the error rate of the classical algorithm. Therefore by Lemma 21 and Lemma 18,

$$B_\epsilon(f)=O(B_{\epsilon,\delta+\epsilon}(f))=O(\tilde{B}_{\epsilon,\delta+\epsilon}(f))=O\left(TG/\epsilon\right) \tag{A.2.10}$$

□

A.3 Proof of Theorem 26

We restate and prove Theorem 26:

Theorem 26 (Finding the first marked element in a list). *Suppose there is an ordered list of N elements, and each element is either marked or unmarked. Then there is a bounded-error quantum algorithm for finding the **first** marked element in the list, or determines that no marked elements exist, such that:*

- *If the first marked element is the d-th element of the list, then the algorithm uses an expected $O(\sqrt{d})$ time and queries.*
- *If there are no marked elements, then the algorithm uses $O(\sqrt{N})$ time and queries.*

Proof. We give an algorithm that has the stated properties. We first recall a quantum algorithm for finding the minimum in a list of items:

Theorem 96 ([63]). *Given a function g on a domain of N elements, there is a quantum algorithm that finds the minimum of g with expected $O(\sqrt{N})$ time and evaluations of g, making $\delta < 1/10$ error.*

We now give our algorithm for finding the first marked element in a list. For simplicity, assume that N is a power of 2 (i.e. $\log_2 N$ is an integer).

Algorithm 97.

1. For $\ell = 2^0, 2^1, 2^2, \cdots, 2^{\log_2 N} = N$:
   - Find the first marked element within the first ℓ elements, or determine no marked element exists. This can be done by defining
   $$g(i) = \begin{cases} \infty & \text{if } i \text{ is unmarked} \\ i & \text{if } i \text{ is marked,} \end{cases} \tag{A.3.1}$$
   and using Theorem 96 to find the minimum of g. This takes $O(\sqrt{\ell}) = O(\sqrt{d})$ queries and makes $\delta < 1/10$ error for each ℓ. If a marked element i^* is found, the algorithm outputs i^* and stops.
2. If no marked element was found in Step 1, the algorithm decides that no marked element exists.

We now claim that Algorithm 97 has the desired properties. Let us break into cases:

- If no marked items exist, then no marked item can possibly be found in Step 1, so the algorithm correctly determines that no marked items exist in Step 2. The number of queries used is
$$\sum_{i=0}^{\log_2 N} \sqrt{2^i} = O(\sqrt{N}) \tag{A.3.2}$$
as desired.
- Suppose the first marked item is the d-th item in the list. Then in Step 1(a), if $\ell \geq d$, there is at least a $1-\delta$ probability that the algorithm will detect that a marked item exists in the first ℓ elements and stop the loop. Letting $\alpha = \lceil \log_2 d \rceil$, the total expected number of queries is thus
$$\sum_{i=0}^{\alpha-1} \sqrt{2^i} + \sum_{i=\alpha}^{\log_2 N} \delta^{i-\alpha}\sqrt{2^i} + O(\sqrt{d}) \leq \frac{2^{\alpha/2}-1}{\sqrt{2}-1} + \sqrt{2^\alpha}\frac{1}{1-\sqrt{2}\delta} + O(\sqrt{d}) \tag{A.3.3}$$
$$= O(\sqrt{2^\alpha}) + O(\sqrt{d}) \tag{A.3.4}$$
$$= O(\sqrt{d}). \tag{A.3.5}$$

The probability of not finding the marked item at the first $\ell \geq d$ is at most δ , and thus the total error of the algorithm is bounded by δ.

$\square$

Appendix B

Supplementary material for Chapter 3

B.1 Supplementary material for Section 3.9

Proof of Lemma 49

First we prove (a). Note that it follows from the assumptions that $\alpha(g+h) = A(g+h) = Ag + Ah \pmod{H}$, $\beta(x+y) = B(x+y) = Bx + By \pmod{J}$, for every $g, h \in G$, $x, y \in H$. Hence, $\beta \circ \alpha(g+h) = \beta(Ag + Ah + \text{zero}_H) = BAg + BAh + \text{zero}_J \pmod{J}$, where zero_X denotes some string congruent to the neutral element 0 of the group X. As in the last equation zero_J vanishes modulo J, BA is a matrix representation of $\beta \circ \alpha$.

We prove (b). From the definitions of character, bullet group and bullet map it follows that

$$\chi_\mu(\alpha(g)) = \exp\left(2\pi\mathrm{i}\sum_{ij} \mu^\bullet(i)A(i,j)g(j)\right) = \exp\left(2\pi\mathrm{i}(A^T\mu^\bullet)\cdot g\right) \text{ for every } g \in G. \quad \text{(B.1.1)}$$

Let f be the function $f(g) := \exp(2\pi\mathrm{i}(A^T\mu^\bullet)\cdot g)$. Then it follows from (B.1.1) that f is continuous and that $f(g+h) = f(g)f(h)$, since the function $\chi_\mu \circ \alpha$ has these properties. As a result, f is a continuous character $f = \chi_\nu$, where $\nu \in G^*$ satisfies $\nu^\bullet = \Upsilon_G \nu = A^T\mu^\bullet \pmod{G^\bullet}$. Moreover, since $f = \chi_\mu \circ \alpha = \chi_{\alpha^*(\mu)}$ it follows that $\alpha^*(\mu) = \nu \pmod{G^*}$ and, consequently,

$$\alpha^*(\mu) = \Upsilon_G^{-1}(A^T\mu^\bullet) \pmod{G^*} = \Upsilon_G^{-1}A^T\Upsilon_H\mu \pmod{G^*}. \quad \text{(B.1.2)}$$

Finally, since $\chi_\mu(\alpha(g)) = \chi_x(\alpha(g))$ for any $x \in \mathbb{R}^n$ congruent to μ, we get that $\alpha^*(\mu) = \Upsilon_G^{-1}A^T\Upsilon_H x \pmod{G^*}$ for any such x, which proves the second part of the lemma. □

Proof of Lemma 50

We will show that each of the homomorphisms α_{XY} as considered in Lemma 47 has a matrix representation, say A_{XY}. Then it will follow from (3.9.3) in Lemma 47 that

$$A := \begin{pmatrix} A_{\mathbb{ZZ}} & 0 & 0 & 0 \\ A_{\mathbb{RZ}} & A_{\mathbb{RR}} & 0 & 0 \\ A_{F\mathbb{Z}} & 0 & A_{FF} & 0 \\ A_{\mathbb{TZ}} & A_{\mathbb{TR}} & A_{\mathbb{T}F} & A_{\mathbb{TT}}. \end{pmatrix}, \tag{B.1.3}$$

as in (3.9.10), is a matrix representation of α.

First, note that if the group Y is finitely generated, then the tuples e_i form a generating set of Y. It is then easy to find a matrix representation A_{XY} of α_{XY}: just choose the jth column of A_{XY} to be the element $\alpha(e_j)$ of X. Expanding $g = \sum_i g(i)e_i$ (where the coefficients $g(i)$ are integral), it easily follows that A_{XY} satisfies the requirements for being a proper matrix representation as given in definition 48. Thus, all homomorphisms α_{XY} with Y of the types $\mathbb{Z}^a$ or F have matrix representations; by duality and Lemma 49(b), all homomorphisms α_{XY} with X of type $\mathbb{T}^a$ or F have matrix representations too.

The only non-trivial α_{XY} left to consider is $\alpha_{\mathbb{RR}}$. Recall that the latter is a continuous map from $\mathbb{R}^m$ to $\mathbb{R}^n$ satisfying $\alpha_{\mathbb{RR}}(x+y) = \alpha_{\mathbb{RR}}(x) + \alpha_{\mathbb{RR}}(y)$ for all $x, y \in \mathbb{R}$. We claim that every such map must be linear, i.e. in addition we have

$$\alpha_{\mathbb{RR}}(rx) = r\alpha_{\mathbb{RR}}(x) \tag{B.1.4}$$

for all $r \in \mathbb{R}$. To see this, first note that $d\alpha_{\mathbb{RR}}(kx/d) = k\alpha_{\mathbb{RR}}(x)$, where k/d is any fraction (k, d are integers). Thus (B.1.4) holds for all rational numbers $r = k/d$. Using that $\alpha_{\mathbb{RR}}$ is continuous and that the rationals are dense in the reals then implies that (B.1.4) holds for all $r \in \mathbb{R}$. This shows that $\alpha_{\mathbb{RR}}$ is a linear map; the existence of a matrix representation readily follows. □

Proof of Theorem 51

It suffices to show (a), that any matrix representation A of α must be an element of Rep and fulfill the consistency conditions (3.9.9); (b), that these consistency conditions imply that A is of the form (3.9.10) and fulfills propositions 1-4; and (c), that every such matrix defines a group homomorphism.

We will first prove (a). Let H_i is of the form $\mathbb{Z}$ or $\mathbb{Z}_{d_i}$. Then, for every $j = 1, \ldots, m$, the definition of matrix representation 48 requires that $(Ae_j)(i) = A(i,j) \pmod{H_i}$ must be an element of H_i. This shows that the ith row of A must be integral and, thus, A belongs to Rep. Moreover, since $x := c_j e_j \equiv 0 \pmod{G}$ and $y := d_i^* e_i \equiv 0 \pmod{H^*}$, (due to the definition of characteristic) it follows that $Ax = 0 \pmod{H}$ and $Ay = 0 \pmod{G^*}$, leading to the consistency conditions (3.9.9).

Next, we will now prove (b).

First, the block form (3.9.10) almost follows from (3.9.3) in Lemma 47: we only have to show, in addition, that the zero matrix is the only valid matrix representation for any trivial group homomorphism $\alpha_{XY} = \mathit{0}$ in (3.9.3). It is, however, easy to check case-by-case that, if A_{XY} is a matrix representation of α_{XY} with $A_{XY} \neq 0$, then α_{XY} cannot be trivial.

Secondly, we prove Propositions 1-4. In Proposition 1, $A_{\mathbb{ZZ}}$ must have integer entries since $A_{\mathbb{ZZ}}e_j(i) \in \mathbb{Z}$ (where, with abuse of notation, i, j index the rows and columns of $A_{\mathbb{ZZ}}$). By duality the same holds for $A_{\mathbb{TT}}$ (it can be shown using Lemma 49(b)). In Proposition 2 the consistency conditions (including dual ones) are vacuously fulfilled and tell us nothing about $A_{\mathbb{RZ}}$, $A_{\mathbb{RR}}$. In Proposition 3, both matrices have to be integral to fulfill that $A_{XY}(e_i)$ (mod Y) is an element of X, which is of type $\mathbb{Z}^a$ or F; moreover, for $Y = F$, the consistency conditions directly impose that the coefficients must be of the form (3.9.11). (The derivation is similar to that of lemma 11 in [2]. Lastly, in Proposition 4, all consistency conditions associated to $A_{\mathbb{TZ}}$ and $A_{\mathbb{TR}}$ are, again, vacuous and tell us nothing about the matrix; however, the first consistency condition tells us that $A_{\mathbb{T}F}$ must be fractional with coefficients of the form $\alpha_{i,j}/c_j$.

Finally we will show (c). First, it is manifest that if A fulfills 1-4 then $A \in$ Rep. Second, to show that A is a matrix representation of a group homomorphism it is enough to prove that every A_{XY} fulfilling 1-4 is the matrix representation of a group homomorphism from Y to X. This can be checked straightforwardly for the cases $A_{\mathbb{ZZ}}$, $A_{\mathbb{RZ}}$, $A_{\mathbb{RR}}$, $A_{F\mathbb{Z}}$, $A_{\mathbb{TZ}}$, $A_{\mathbb{TR}}$ applying properties 1-4 of A and using that, in all cases, there are no non-zero real vectors congruent to the zero element of Y. Obviously, for the cases where A_{XY} must be zero the proof is trivial. It remains to consider the cases A_{FF}, $A_{\mathbb{T}F}$, $A_{\mathbb{TT}}$. In all of this cases, it holds due to properties 1,3,4 that the first consistency condition in (3.9.9) is fulfilled. Then, for the case A_{FF}, Lemma 2 in [2] can be applied to show that A_{FF} is a group homomorphism; moreover, for any $g \in F$ and x congruent to g, the first consistency condition implies that $Ax = Ag$ (mod G), which is what we wanted to show. For the cases $A_{\mathbb{T}F}$, $A_{\mathbb{TT}}$, Lemma 2 in [2] can still be generalized (the proof given in [2] for the second statement of Lemma 2 can be directly adapted) and the same argument applies. □

B.2 Supplementary material for Section 3.10

Proof of Lemma 52

The lemma is a particular case of proposition 1.1 in [251]. We reproduce a shortened proof of the result in [251] (modified to suit our notation) here.

If β is an continuous homomorphism from G into G^* then $B(g, h) = \chi_{\beta(g)}(h)$ is continuous, since composition preserves continuity. Also, it follows using the linearity of this map and of the character functions that $B(g, h)$ is bilinear, and hence a bicharacter. Conversely, consider an arbitrary bicharacter B. The condition that B is a character on the second argument says that for every g the function $f_g : h \rightarrow B(g, h)$ is a character. Consequently $f_g(h) = B(g, h) = \chi_{\mu_g}(h)$ for all $h \in G$ and some $\mu_g \in G^*$ that is determined by g. We denote by β be the map which sends g to μ_g. Using that $g \rightarrow B(g, h)$ is also a character it follows that $\chi_{\beta(g+g')}(h) = \chi_{\beta(g)}(h)\chi_{\beta(g')}(h)$ for all $h \in G$, so that $\beta : G \rightarrow G^*$ is a group homomorphism. It remains to show that β is continuous; for this we refer to the proof in [251], where the author analyzes how neighborhoods are transformed under this map.

Proof of Lemma 53

We obtain (a) by combining (3.9.7) with the normal form (3.10.3): the matrix M is of the form ΥX where X is a matrix representation of β; (b) follows from this construction. (c) follows from the normal form in Lemma 52, property (a) and Lemma 45.

To prove (d) we bring together (a) and the relationship $B(h,g) = B(g,h)$, and derive

$$g^T M h = g^T M^T h \mod \mathbb{Z}, \quad \text{for every } g,\, h \in G. \tag{B.2.1}$$

Write $G = G_1 \times \cdots \times G_m$ with G_i of primitive type. If G_i is either finite or equal to $\mathbb{Z}$ or $\mathbb{R}$ then the canonical basis vector e_i belongs to G. If $G_i = \mathbb{T}$ then $te_i \in G$ for all $t \in [0,1)$. If neither G_i nor G_j is equal to $\mathbb{T}$, taking $g = e_i$, $h = e_j$ in equation (B.2.1) yields $M(i,j) \equiv M(j,i) \bmod \mathbb{Z}$. If G_i and G_j are equal to $\mathbb{T}$, setting $g = te_i$ and $h = se_j$ yields $stM(i,j) \equiv stM(j,i) \bmod \mathbb{Z}$ for all $s,t \in [0,1)$, which implies that

$$st(M(i,j) - M(j,i)) \in \mathbb{Z} \tag{B.2.2}$$

for all $s,t \in [0,1)$. This can only happen if $M(i,j) = M(j,i)$. The other cases are treated similarly. In conclusion, we find that M is symmetric modulo $\mathbb{Z}$. This proves (d).

Lastly, we prove (e). Note that we have just shown that $M(i,j) = M(j,i)$ if $G_i = G_j = \mathbb{T}$; the same argument can be repeated (with minor modifications) to show $M(i,j) = M(j,i)$ if either one of G_i or G_j is of the form $\mathbb{R}$ or $\mathbb{T}$. Hence, $M(i,j) \neq M(j,i)$ can only happen if G_i, G_j are of the form $\mathbb{Z}$ or $\mathbb{Z}_d$. In this case, we denote by Δ_{ij} the number such that $M(j,i) = M(i,j) + \Delta_{i,j}$. (d) tells us that Δ_{ij} is an integer. Moreover, by choosing $g = g(i)e_i$, $h = h(j)e_j$ in (B.2.1) it follows that

$$M(j,i)g(i)h(j) = M(i,j)g(i)h(j) + \Delta_{i,j}g(i)h(j) \mod \mathbb{Z}, \tag{B.2.3}$$

As $g(i)$ and $h(j)$ are integers the factor $\Delta_{i,j}g(i)h(j)$ gets cancelled modulo $\mathbb{Z}$ and produces no effect. Finally, we define a new symmetric matrix M' as $M'(i,j) = M(i,j)$ if $i \geq j$, and $M'(i,j) = M(j,i)$ if $i < j$. It follows from our discussion that $g^T M' h = g^T M h \bmod \mathbb{Z}$ for every $g, h \in G$, so that M' manifestly fulfills (a).

It remains to show that M' fulfills (b)-(c). Keep in mind that $h \to Mh \pmod{G^\bullet}$ defines a group homomorphism into $G^\bullet$. From our last equations, it follows that either $M(i,j)h(j) = M'(i,j)h(j)$ or $M(i,j))h(j) = M'(i,j)h(j) \bmod \mathbb{Z}$ if both G_i and G_j are discrete groups. From the definition of bullet group (3.8.16), it is now easy to derive that $Mh = M'h \pmod{G^\bullet}$ for every h, and to extend this equation to all tuples x congruent to h (this reduces to analyzing all possible combinations of primitive factors). As a result, M' is a matrix representation that defines the same map as M, which implies (b). The fact that M' satisfies (c) follows using the same argument we used for M.

Proof of Lemma 54

We prove that the function $f(g) := \xi_1(g)/\xi_2(g)$ is a character, implying that there exists $\mu \in G^*$ such that $\chi_\mu = f$:

$$f(g+h) := \frac{\xi_1(g)}{\xi_2(g)} \frac{\xi_1(h)}{\xi_2(h)} \frac{B(g,h)}{B(g,h)} = f(g)f(h). \tag{B.2.4}$$

Proof of Lemma 55

Define the function $q : G \to \mathbb{R}$ as

$$q(g) := g^T M g + C^T g. \tag{B.2.5}$$

We prove that $q(g)$ is a quadratic form modulo $2\mathbb{Z}$ with associated bilinear form $b_q(g,h) := 2g^TMh$; or, in other words, that the following equality holds for every $g,h \in G$:

$$q(g+h) = q(g) + q(h) + 2g^TMh \pmod{2\mathbb{Z}}. \tag{B.2.6}$$

Assuming that (B.2.6) is correct, it follows readily that the function $Q(g) = \exp(\pi i q(g))$ is quadratic and also a B-representation, which is what we wanted:

$$Q(g+h) = Q(g)Q(h)\exp(2\pi i\, g^TMh) \tag{B.2.7}$$

We prove (B.2.6) by direct evaluation of the statement. First we define $q_M(g) := g^TMg$ and $q_C(g) := C^Tg$, so that $q(g) = q_M(g) + q_C(g)$. We will also (temporarily, i.e. only within the scope of this proof) use the notation $g \oplus h$ to denote the group operation in G and reserve $g+h$ for the case when we sum over the reals. Also, denoting $G = G_1 \times \dots G_m$ with G_i primitive, we define $c := (c_1, \dots, c_m)$ to be a tuple containing all the characteristics $c := \mathrm{char}(G_i)$. With these conventions we have $g \oplus h = g + h + \lambda \circ c$, where λ is a vector of integers and $\circ$ denotes the entrywise product: $\lambda \circ c = (\lambda_1 c_1, \dots, \lambda_m c_m)$. Note that $\lambda \circ c$ is the most general form of any string of real numbers that is congruent to $0 \in G$ (the neutral element of the group). We then have (using that $M = M^T$):

$$\begin{aligned} q_M(g \oplus h) &= q_M(g) + q_M(h) + 2g^TMh \\ &\quad + 2g^TM(\lambda\circ c) + 2h^TM(\lambda \circ c) + (\lambda\circ c)^TM(\lambda\circ c), \end{aligned} \tag{B.2.8}$$

$$q_C(g \oplus h) = q_C(g) + q_C(h) + \sum_i M(i,i)\lambda(i)c_i^2. \tag{B.2.9}$$

Consider an $x \in \mathbb{R}^m$ for which there exists $g \in G$ such that $x \equiv g \bmod G$. Then $x^TM(\lambda\circ c)$ with $x \in G$ must be an integer. Indeed, we have

$$1 = B(g,0) = \exp(2\pi i x^TM(\lambda\circ c)), \tag{B.2.10}$$

where in the second identity we used Lemma 53 together with the property $\lambda\circ c \equiv 0 \bmod G$. This shows that $x^TM(\lambda \circ c)$ is an integer. It follows that the fourth and fifth terms on the right hand side of eq. (B.2.8) must be equal to an even integer and thus cancel modulo $2\mathbb{Z}$. Combining results we end up with the expression

$$q(g \oplus h) = q(g) + q(h) + 2g^TMh + \Delta \pmod{2\mathbb{Z}}, \tag{B.2.11}$$

where

$$\Delta := (\lambda\circ c)^TM(\lambda\circ c) + \sum_i M(i,i)\lambda(i)c_i^2. \tag{B.2.12}$$

We finish our proof by showing that Δ is an even integer too, which proves (B.2.6).

First, we note that, due to the symmetry of M, we can expand $(\lambda\circ c)^TM(\lambda\circ c)$ as

$$(\lambda\circ c)^TM(\lambda\circ c) = \sum_{i,j\,:\,i<j} 2M(i,j)\lambda(i)\lambda(j)c_ic_j + \sum_i M(i,i)\lambda(i)^2c_i^2. \tag{B.2.13}$$

Revisiting (B.2.10) and choosing $x = e_i$ and $\lambda = e_j$ for all different values of i, j, we obtain the following consistency equation for M

$$c_j M(i,j) = c_i M(i,j) = 0 \pmod{\mathbb{Z}} \tag{B.2.14}$$

It follows that all terms of the form $2M(i,j)\lambda(i)\lambda(j)c_i c_j$ are *even* integers. We can thus remove these terms from (B.2.13) by taking modulo $2\mathbb{Z}$, yielding

$$\Delta = \sum_i M(i,i)\lambda(i)^2 c_i^2 + \sum_i M(i,i)\lambda(i)c_i^2 \pmod{2\mathbb{Z}} \tag{B.2.15}$$

$$= \sum_i M(i,i)c_i^2\lambda(i)(\lambda(i)+1) = 0 \pmod{2\mathbb{Z}}, \tag{B.2.16}$$

where in the last equality we used the fact that $\lambda(i)(\lambda(i)+1)$ is necessarily even.

Proof of Lemma 58

The fact that $\xi_{M,v} \circ \alpha$ is quadratic follows immediately from the fact that $\xi_{M,v}$ is quadratic and that α is a homomorphism. Composed continuous functions lead to continuous functions. As a result, Theorem 57 applies and we know $\xi_{M',v'} = \xi_{M,v} \circ \alpha$ for some choice of M', v'.

Let $B_M(g,h) = \exp(2\pi i g^T M h)$ be the bicharacter associated with $\xi_{M,v}$. One can show by direct evaluation (and using Lemma 49(a) and lemma 53) that $B_{M'}$ with $M' := A^T M A$ is the bicharacter associated to $\xi_{M,v} \circ \alpha$. Let $Q_{M'}(g) := \exp(\pi i\,(g^T M' g + C_{M'}^T g))$, be the quadratic function in Lemma 54. By construction, both $\xi_{M,v} \circ \alpha$ and $Q_{M'}$ are $B_{M'}$-representations of the bicharacter $B_{M'}$. As a result, Lemma 54 tells us that the function $f(g) := \xi_{M,v} \circ \alpha(g)/Q_{M'}(g)$ is a character of G, so that there exists $v' \in G^*$ such that $\chi_{v'}(g) = f(g)$. We can compute v' by direct evaluation of this expression:

$$\chi_{v'}(g) = \exp\left(\pi i\,(A^T C_M - C_{A^T M A})\,g\right)\exp\left(2\pi i\,(A^T v)\cdot g\right). \tag{B.2.17}$$

It can be checked that the function $\exp(2\pi i\,(A^T v)\,g)$ is a character, using that it is the composition of a character $\exp(2\pi v^T g)$ (Theorem 57) and a continuous group homomorphism α. Since $\chi_{v'}$ is also a character, the function $\exp(\pi i\,(A^T C_M - C_{A^T M A})\,g)$ is a character too (as characters are a group under multiplication), and it follows that $v_{A,M} = (A^T C_M - C_{A^T M A})/2$ is congruent to some element of $G^\bullet$ [1]; we obtain that $v' = A^T v + v_{A,M}$ is an element of $G^\bullet$. Finally, we obtain that $\xi_{M',v'}$ is a normal form of $\xi_{M,v} \circ \alpha$, using the relationship $\xi_{M,v} \circ \alpha(g) = Q_{M'}(g) f(g) = Q_{M'}(g)\chi_{v'}(g) = \xi_{M',v'}(g)$.

[1]This statement can also be proven (more laboriously) by explicit evaluation, using arguments similar to those in the proof of Lemma 55.

Appendix C

Proofs of results in Chapter 4

C.1 Existence of general solutions of systems given by (4.5.1)

In this section we show that general-solutions of systems of linear equations over elementary Abelian groups always exist (given that the systems admit at least one solution).

We start by recalling an important property of elementary Abelian groups.

Lemma 98 (See theorem 21.19 in [98] or section 7.3.3 in [86]). *The class of elementary Abelian groups*[1] *is closed with respect to forming closed subgroups, quotients by these, and finite products.*[2]

In our setting, the kernel of a continuous group homomorphism $A : G \to H$ as in (4.5.1) is always closed: this follows from the fact that the singleton $\{0\} \subset H$ is closed (because elementary Abelian groups are Hausdorff [98]), which implies that $\ker A = A^{-1}(\{0\})$ is closed (due to continuity of A). Hence, it follows from Lemma 98 that $\ker A$ is topologically isomorphic to some elementary Abelian group $H' := \mathbb{R}^a \times \mathbb{T}^b \times \mathbb{Z}^c \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$; consequently, there exists a continuous group isomorphism φ from H' to H.

Next, we write the group H' as a quotient group X/K of the group $X := \mathbb{R}^{a+b} \times \mathbb{Z}^{c+d}$ by the subgroup K generated by the elements of the form $\mathrm{char}(X_i)e_i$. The quotient group X/K is the image of the quotient map $q : X \to X/K$ and the latter is a continuous group homomorphism [98]. By composing φ and q we obtain a continuous group homomorphism $\mathcal{E} := \varphi \circ q$ from X onto H. The map $\mathcal{E}$ together with any particular solution x_0 of (4.5.1) constitutes a general solution of (4.5.1), proving the statement.

C.2 Proof of Theorem 66

In this section we show that systems of linear equations over groups (4.5.1) can be reduced to systems of mixed real-integer linear equations (4.5.5).

Start with two elementary groups of general form G, H. First, notice that we can write G and H as $G = G_1 \times \cdots \times G_m$, $H = H_1 \times \cdots \times H_n$ where each factor G_i, H_j is of the form $G_i = \mathbf{X}_i/c_i\mathbb{Z}$, $H_j = \mathbf{Y}_i/d_i\mathbb{Z}$ with $\mathbf{X}_i$, $\mathbf{Y}_i \in \{\mathbb{Z}, \mathbb{R}\}$; the numbers c_i, d_j are the characteristics of the primitive factors. We assume w.l.o.g. that the primitive factors of G, H are *ordered*

[1]Beware that in [98] the class of elementary groups is referred as "the category CGAL", which stands for Compactly Generated Abelian Lie groups.

[2]In fact, as mentioned in [98], corollary 21.20 elementary LCA groups constitute the smallest subclass of LCA containing $\mathbb{R}$ and fulfilling all these properties.

such that both groups are of the form $\mathbb{Z}^a \times F \times \mathbb{T}^b$: in other words, the finitely generated factors come first.

We now define a new group $\mathbf{X} := \mathbf{X}_1 \times \cdots \times \mathbf{X}_m$ (recall that with the ordering adopted X is of the form $\mathbb{Z}^a \times \mathbb{R}^b$) which will play the role of an *enlarged solution space*, in the following sense. Let $\mathbf{V}$ be the subgroup of $\mathbf{X}$ generated by the elements $c_1 e_1, \ldots, c_m e_m$. Observe that the group G—the solution space in system (4.5.1)—is precisely the quotient group $\mathbf{X}/\mathbf{V}$, and thus can be *embedded* inside the larger group $\mathbf{X}$ via the quotient group homomorphism $\mathbf{q} : \mathbf{X} \to G = \mathbf{X}/\mathbf{V}$:

$$\mathbf{q}(\mathbf{x}) := (\mathbf{x}(1) \bmod c_1, \ldots, \mathbf{x}(m) \bmod c_m) = \mathbf{x} \pmod{G}; \qquad \text{(C.2.1)}$$

remember also that $\ker \mathbf{q} = \mathbf{V}$. Now let $\alpha : \mathbf{X} \to H$ be the group homomorphism defined as $\alpha := A \circ \mathbf{q}$. Then it follows from the definition that $\alpha(\mathbf{x}) = A\mathbf{x} \pmod{H}$, and A is a matrix representation of α. (This is also a consequence of the composition property of matrix representations (Lemma 49.(a), since the $m \times m$ identity matrix I_m is a matrix representation of $\mathbf{q}$.) We now consider the relaxed[3] system of equations

$$\alpha(\mathbf{x}) = b \pmod{H}, \quad \text{where } \mathbf{x} \in \mathbf{X} = \mathbb{Z}^a \times \mathbb{R}^b. \qquad \text{(C.2.2)}$$

Note that the problem of solving (4.5.1) reduces to solving (C.2.2), which looks closer to a system of mixed real-integer linear equations. Indeed, let $\mathbf{X}_{\text{sol}}$ denote the set of all solutions of system (C.2.2); then[4]

$$G_{\text{sol}} = \mathbf{q}(\mathbf{X}_{\text{sol}}) \quad \Longrightarrow \quad G_{\text{sol}} = \mathbf{q}(\mathbf{x}_0) + \mathbf{q}(\ker \alpha) = \mathbf{x}_0 + \ker \alpha \pmod{G}, \qquad \text{(C.2.3)}$$

Hence, our original system (4.5.1) admits solutions iff (C.2.2) also does, and the former can be obtained from the latter via the homomorphism $\mathbf{q}$. We further show next that (C.2.2) is equivalent to a system of form (4.5.5). First, note that the matrix A has a block form $A = \begin{pmatrix} A_{\mathbb{Z}} & A_{\mathbb{R}} \end{pmatrix}$ where $A_{\mathbb{Z}}$, $A_{\mathbb{R}}$ act, respectively, in integer and real variables. Since the constraint (mod H) is equivalent to the modular constraints mod $d_1, \ldots,$ mod d_n, it follows that $\mathbf{x} = \begin{pmatrix} \mathbf{x}_{\mathbb{Z}} & \mathbf{x}_{\mathbb{R}} \end{pmatrix} \in \mathbf{X}_{\text{sol}}$ if and only if

$$A_{\mathbb{Z}}\mathbf{x}_{\mathbb{Z}} + A_{\mathbb{R}}\mathbf{x}_{\mathbb{R}} + D\mathbf{y} = c, \quad \text{where } D = \operatorname{diag}(d_1, \ldots, d_n),\ \mathbf{y} \in \mathbb{Z}^n. \qquad \text{(C.2.4)}$$

Clearly, if we rename $A' := \begin{pmatrix} A_{\mathbb{Z}} & D \end{pmatrix}$, $\mathbf{x}' := \begin{pmatrix} \mathbf{x}_{\mathbb{Z}} & \mathbf{y} \end{pmatrix}$, $B = A_{\mathbb{R}}$ and $\mathbf{y}' := \mathbf{x}_{\mathbb{R}}$, system (C.2.4) is a system of mixed-integer linear equations as in (4.5.5). Also, system (4.5.5) can be seen as a system of linear equations over Abelian groups: note that in the last step the solution space $\mathbf{X}$ is increased by introducing new extra integer variables $\mathbf{y} \in \mathbb{Z}^n$. If we let $\mathbf{G}$ denote the group $\mathbf{X} \times \mathbb{Z}^n$ that describes this new space of solutions, then (C.2.4) can be rewritten as

$$\mathbf{A}\mathbf{g} := \begin{pmatrix} A & D \end{pmatrix} \mathbf{g} = c, \quad \text{where } \mathbf{g} \in \mathbf{G} \qquad \text{(C.2.5)}$$

and c represents an element of $\mathbf{Y}$.

Mind that (C.2.4) (or equivalently (C.2.5)) admits solutions if and only if both of (C.2.2) and (4.5.1) admit solutions. Indeed, the solutions of (C.2.4) and (C.2.2) are—again—related

[3]Notice that the new system is less constrained, as we look for solutions in a larger space than beforehand.

[4]It is easy to prove $G_{\text{sol}} = \mathbf{q}(\mathbf{X}_{\text{sol}})$ by showing $G_{\text{sol}} \supset \mathbf{q}(\mathbf{X}_{\text{sol}})$ and the reversed containment for the preimage $\mathbf{q}^{-1}(G_{\text{sol}}) \subset \mathbf{X}_{\text{sol}}$; then surjectivity of $\mathbf{q}$ implies $G_{\text{sol}} = \mathbf{q}(\mathbf{q}^{-1}(G_{\text{sol}})) \subset \mathbf{q}(\mathbf{X}_{\text{sol}})$.

via a surjective group homomorphism $\pi : \mathbf{X} \times \mathbb{Z}^n \to \mathbf{X} : (\mathbf{x}, \mathbf{y}) \to \mathbf{x}$. It follows from the derivation of (C.2.4) that $\pi(\mathbf{G}_{\mathrm{sol}}) = \mathbf{X}_{\mathrm{sol}}$ and, consequently, $\mathbf{q} \circ \pi(\mathbf{G}_{\mathrm{sol}}) = G_{\mathrm{sol}}$; these relationships show that either all systems admit solutions or none of them do.

Finally, we use existing algorithms to find a general solution $(\mathbf{g}_0, \mathbf{P})$ of system (C.2.5) and show how to use this information to compute a general solution of our original problem (4.5.1).

First, we recall that algorithms presented in [112] can be used to: (a) *check* whether a system of the form (4.5.5,C.2.5) admits a solution[5]; (b) *find* a particular solution $\mathbf{g}_0$ (if there is any) and a matrix $\mathbf{P}$ that defines a group endomorphism of $\mathbf{G} = \mathbf{X} \times \mathbb{Z}^n$ whose image $\operatorname{im}\mathbf{P}$ is precisely the kernel[6] of $\mathbf{A} = \begin{pmatrix} A & D \end{pmatrix}$ (for details see theorem 1 in [112]).

Assume now that (C.2.4) admits solutions and that we have already found a general solution $(\mathbf{g}_0 = (\mathbf{x}_0, \mathbf{y}_0), \mathbf{P})$. We show next how a general solution (x_0, P) of (4.5.1) can be computed by making use of the map $\mathbf{q} \circ \pi$. We also discuss the overall worst-time running time we need to compute (x_0, P), as a function of the sizes of the matrix A and the tuple b given as an input in our original problem (4.5.1) (the bit-size or simply *size* of an array of real numbers—tuple, vector or matrix—is defined as the minimum number of bits needed to store it with infinite precision), size(G) and size(H):

- First, note that $(\mathbf{g}_0 = (\mathbf{x}_0, \mathbf{y}_0), \mathbf{P})$ can be computed in polynomial-time in size(A), size(b), size(G) and size(H), since there is only a polynomial number of additional variables and constrains in (C.2.4) and the worst-time scaling of the algorithms in [112] is also polynomial in the mentioned variables. (We discussed the complexity of these methods in Section 4.5.)

- Second, a particular solution x_0 of (4.5.1) can be easily computed just by taking $x_0 := \mathbf{q} \circ \pi((\mathbf{x}_0, \mathbf{y}_0)) = \pi(\mathbf{x}_0) \pmod{G}$: this computation is clearly efficient in size($\mathbf{x}_0$) and size(G).

- Third, note that the composed map $P := \mathbf{q} \circ \pi \circ \mathbf{P}$ defines a group homomorphism $P : \mathbf{G} \to G$ whose image is precisely the subgroup $\ker A$; a matrix representation of P (that we denote with the same symbol) can be efficiently computed, since

$$\text{if} \quad \mathbf{P} = \begin{pmatrix} \mathbf{P}_{\mathbf{XX}} & \mathbf{P}_{\mathbf{X}\mathbb{Z}} \\ \mathbf{P}_{\mathbb{Z}\mathbf{X}} & \mathbf{P}_{\mathbb{Z}\mathbb{Z}} \end{pmatrix} \quad \text{then} \quad P := \begin{pmatrix} \mathbf{P}_{\mathbf{XX}} & \mathbf{P}_{\mathbf{X}\mathbb{Z}} \end{pmatrix} \tag{C.2.6}$$

  is a matrix representation of $\mathbf{q} \circ \pi \circ \mathbf{P}$ that we can take without further effort.

The combination of all steps above yields a deterministic polynomial-time algorithm to compute a general solution $(x_0, P;\ \mathbf{G})$ of system (4.5.1), with worst-time scaling as a polynomial in the variables m, n, $\log \|A\|_{\mathbf{b}}$, $\log \|b\|_{\mathbf{b}}$, $\log c_i$, $\log d_j$. This proves Theorem 66.

[5]Mind that this step is actually not essential for our purposes, since in the applications we are interested on all such systems admits solutions by promise.

[6]In fact, the matrix P is also idempotent and defines a projection map on $\mathbf{G}$ and $\ker \mathbf{A}$ is the image of a projection map: subgroups satisfying this property are called *retracts*. Though the authors never mention the fact that $\mathbf{P}$ is a projection, this follows immediately from their equations (10a,10b).

C.3 Efficiency of Bowman-Burdet's algorithm

In this appendix we briefly discuss the time performance of Bowman-Burdet's algorithm [112] and argue that, using current algorithms to compute certain matrix normal forms (namely, Smith normal forms) as subroutines), their algorithm can be implemented in worst-time polynomial time.

An instance of the problem $Ax + By = C$, of the form (4.5.5), is specified by the rational matrices A, B and the rational vector C. Let A, B, C have $c \times a$, $c \times b$ and c entries. Bowman-Burdet's algorithm (explained in [112, Section 3]) involves different types of steps, of which the most time consuming are (see [112, Equations 8-10]):

1. The calculation of a constant number of certain types of generalized inverses introduced by Hurt and Waid [148];
2. A constant number of matrix multiplications.

A Hurt-Waid generalized inverse $M^{\#}$ of a rational matrix M can be computed with an algorithm given in [148], equations 2.3-2.4. The worst-running time of this procedure is dominated by the computation of a Smith Normal form $S = UMV$ of M with pre- and post- multipliers U, V. This subroutine becomes the bottleneck of the entire algorithm, since existing algorithms for this problem are slightly slower than those for multiplying matrices (cf. [154] for a slightly outdated review). Furthermore, S, U and V can be computed in polynomial time (we refer the reader to [154] again).

The analysis above shows that Bowman-Burdet's algorithm runs in worst-time polynomial in the variables $\log \|A\|_{\mathbf{b}}$, $\log \|B\|_{\mathbf{b}}$, $\log \|C\|_{\mathbf{b}}$, a, b, c, which is enough for our purposes.

C.4 Proof of Lemma 67

As a preliminary, recall that group homomorphism form an Abelian group with the point-wise addition operation. Clearly, matrix representations inherit this group operation and form a group too. This follows from the following formula,

$$(\alpha + \beta)(g) = \alpha(g) + \beta(g) = Ag + Bg = (A + B)g \pmod{G}, \tag{C.4.1}$$

which also states that the sum $(A+B)$ of the matrix representations A, B of two homomorphisms α, β is a matrix representation of the homomorphism $\alpha + \beta$. The group structure of the matrices is, in turn, inherited by their *columns*, a fact that will be exploited in the rest of the proof; we will denote by X_j the Abelian group formed by the jth columns of all matrix representations with addition rule inherited from the matrix addition operation.

A consequence of Theorem 51 is that the group X_j is always an elementary Abelian group, namely

$$\begin{aligned}
&G_j = \mathbb{Z} && \mathbb{R}ightarrow && X_j = G = \mathbb{Z}^a \times \mathbb{R}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbb{T}^d; \\
&G_j = \mathbb{R} && \mathbb{R}ightarrow && X_j = \{0\}^a \times \mathbb{R}^b \times \{0\}^c \times \mathbb{R}^d; \\
&G_j = \mathbb{Z}_{N_j} && \mathbb{R}ightarrow && X_j = \{0\}^a \times \{0\}^b \times (\eta_{1,j}\mathbb{Z} \times \cdots \times \eta_{c,j}\mathbb{Z}) \times \left(\tfrac{1}{N_j}\mathbb{Z}\right)^d; \\
&G_j = \mathbb{T} && \mathbb{R}ightarrow && X_j = \{0\}^{m_z} \times \{0\}^{m_r} \times \{0\}^{m_f} \times \mathbb{Z}^{m_t};
\end{aligned} \tag{C.4.2}$$

where $\eta_{i,j} := N_i / \gcd(N_i, N_j)$.

We will now prove the statement of the lemma.

First, we reduce the problem of computing a valid matrix representation X of α^{-1} to that of solving the equation $\alpha \circ \beta = \mathrm{id}$ (α) is now the given automorphism) where β stands for any continuous group homomorphism $\beta : G \to G$. It is easy to show that this equation admits $\beta = \alpha^{-1}$ as unique solution, since

$$\alpha \circ \beta = \mathrm{id} \quad \Longrightarrow \quad \beta = (\alpha^{-1} \circ \alpha) \circ \beta = \alpha^{-1} \circ (\alpha \circ \beta) = \alpha^{-1}. \tag{C.4.3}$$

Hence, our task will be to find a matrix X such that $g \to Xg \pmod G$ is a continuous group *homomorphism* and such that AX is a matrix representation of the identity automorphism. The latter condition reads $AXg = g \pmod G$ for every $g \in G$ and is equivalent to

$$AX\left(\sum_j g(j)e_j\right) = \sum_j g(j)Ax_j = \sum_j g(j)e_j \pmod G, \text{ for every } g \in G, \tag{C.4.4}$$

where x_j denotes the jth column of X. Since (C.4.4) holds, in particular, when all but one number $g(j)$ are zero, it can be re-expressed as an equivalent system of equations:

$$g(j)Ax_j = g(j)e_j \pmod G, \text{ for any } g(j) \in G_j, \text{ for } j = 1, \ldots, m. \tag{C.4.5}$$

Finally, we will reduce each individual equation in (C.4.5) to a linear system of equations of the form (4.5.1). This will let us apply the algorithm in Theorem 66 to compute every individual column x_j of X.

We begin by finding some simpler equivalent form for (C.4.5) for the different types of primitive factors:

(a) If $G_j = \mathbb{Z}$ or $G_j = \mathbb{Z}_{N_j}$ the coefficient $g(j)$ is integral and can take the value 1. Hence, equation (C.4.5) holds iff $Ax_j = e_j \pmod G$.

(b) If $G_j = \mathbb{R}$ or $G_j = \mathbb{T}$ we show that (C.4.5) is equivalent to $Ax_j = e_j \pmod{X_j}$. Clearly, (C.4.5) implies $g(j)Ax_j = g(j)e_j + \mathit{zero}$ where $\mathit{zero} = 0 \pmod G$ and where we fix a value of $g(j) \in G_j$. Since G_j is divisible, $g(j)' = g(j)/d$ is also an element of G_j for any positive integer d. For this value we get $\frac{g(j)}{d}Ax_j = \frac{g(j)}{d}e_j + \mathit{zero}'$. These two equations combined show that $\mathit{zero} = d\,\mathit{zero}'$ must hold for every positive integer $d \in \mathbb{Z}$. Since both zero and zero' are integral, it follows that the entries of zero are *divisible* by all positive integers; this can only happen if $\mathit{zero} = 0$ and, consequently, (C.4.5) is equivalent to $Ax_j = e_j$. Since both Ax_j and e_j are jth columns of matrix representations, the latter equation can be written as $Ax_j = e_j \pmod{X_j}$ with X_j as in (C.4.2).

Finally, we argue that the final systems (a) $Ax_j = e_j \pmod G$ and (b) $Ax_j = e_j \pmod{X_j}$ are linear systems of the form (4.5.1). First notice that for any two homomorphisms β, β' with matrix representations X, Y, it follows from (C.4.1) and Lemma 49.(a) that $A(X+Y) = AX + AY$ is a matrix representation of the homomorphism $\alpha \circ (\beta + \beta') = \alpha \circ \beta + \alpha \circ \beta'$. Consequently,

$$A(X+Y)g = (AX + AY)g \pmod G, \text{for every } g \in G. \tag{C.4.6}$$

The argument we used to reduce $AXg = g \pmod G$ to the cases (a) and (b) can be applied again to find a simpler form for (C.4.6). Applying the same procedure step-by-step (the derivation is omitted), we obtain that, if $G_j = \mathbb{Z}$ or $G_j = \mathbb{Z}_{N_j}$, then (C.4.6) is equivalent to $A(x_j + y_j) = Ax_j + Ay_j \pmod{G}$; if $G_j = \mathbb{R}$ and $G_j = \mathbb{T}$, we get $A(x_j + y_j) = Ax_j + Ay_j$ (mod X_j) instead. It follows that the map $x_j \to Ax_j$ is a group homomorphism from X_j to G in case (a) and from X_j to X_j in case (b). This shows that systems (a) and (b) are of the form (4.5.1).

Appendix D

Supplementary material for Chapter 5

D.1 Proof of Theorem 80

To prove the result we can assume that we know a group isomorphism $\varphi : \mathbb{Z}_N^\times \to G$ that decomposes the black-box group as a product of cyclic factors $G = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_d}$. Let $U_\varphi : \mathcal{H}_\mathbf{B} \to \mathcal{H}_G$ be the unitary that implements the isomorphism $U_\varphi|b\rangle = |\varphi(b)\rangle$ for any $b \in \mathbb{Z}_N^\times$. It is easy to check that $\mathcal{C}$ is a normalizer circuit over $\mathbb{Z}_M \times G$ if and only if $(I \otimes U_\varphi)\mathcal{C}(I \otimes U_\varphi)^\dagger$ is a normalizer circuit over $\mathbb{Z}_M \times \mathbb{Z}_N^\times$: automorphism (resp. quadratic phase) gates get mapped to automorphism (resp. quadratic phase) gates and vice-versa; isomorphic groups have isomorphic character groups [97], and therefore Fourier transforms get mapped to Fourier transforms.

As a result, it is enough to prove the result in the basis labeled by elements of $\mathbb{Z}_M^\times \times G$. The advantage now is that we can use results from [1, 2]. In fact, the rest of the proof will be similar to the proof of theorem 2 in [1].

We define $\alpha := \varphi(a)$. The action of U_{me} in the G-basis reads $U_{\text{me}}|m, g\rangle = |m, m\alpha + g\rangle$, in additive notation. Define a fuction $F(m, g) = (m, m\alpha + g)$. We now assume that M is not divisible by $|a|$ and that there exists a normalizer circuit $\mathcal{C}$ such that $\|\mathcal{C} - U_{\text{me}}\| \leq \delta$ with $\delta = 1 - 1/\sqrt{2}$ and try to arrive to a contradiction. This property implies that $\|\mathcal{C}|m, g\rangle - U_{\text{me}}|m, g\rangle\| \leq \delta$ for any standard basis state, and consequently

$$|\langle F(m, g)|\mathcal{C}|m, g\rangle| \geq 1 - \delta = \tfrac{1}{\sqrt{2}} \qquad \text{(D.1.1)}$$

It was shown in [1] that $\mathcal{C}|m, g\rangle$ is a uniform superposition over some subset $x + H$ of G, being H a subgroup. If H has more than two elements, then $\mathcal{C}|m, g\rangle$ is a uniform superposition over more than two computational basis states. It follows that $\langle n, h|\mathcal{C}|m, g\rangle \leq \frac{1}{\sqrt{2}}$ for any basis state $|n, h\rangle$ in contradiction with D.1.1, so that we can assume $H = \{0\}$ and that $\mathcal{C}|m, g\rangle$ is a standard basis state. Then (D.1.1) implies that $|F(m, g)\rangle$ and $\mathcal{C}|m, g\rangle$ must coincide for every $(m, g) \in \mathbb{Z}_M \times G$, so that $\mathcal{C}$ must perfectly realize the transformation $|(m, g)\rangle \to |F(m, g)\rangle$; however, the only classical functions that can be implemented by normalizer circuits of this form are affine maps [1], meaning that $F(m, g) = f(m, g) + b$ for some group automorphism $f : \mathbb{Z}_M \times G \to \mathbb{Z}_M \times G$ and some $b \in \mathbb{Z}_M \times G$.

Finally, we arrive to a contradiction showing that $F(m, g)$ is not affine unless M is a multiple of $|a|$. First, by evaluating $F(m, g) = f(m, g) + b = (m, m\alpha + g)$ at $(0, 0)$,$(1, 0)$ and elements of the form $(0, g)$, we can compute b and a matrix representation A of the automorphism f [1]: we obtain $b = 0$, so that $F(m, g)$ must be an automorphism, and $A = \begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix}$. However, for the matrix A to be a matrix representation of a group automorphism, the first column a_1 needs to fulfill the equation: $Ma_1 \pmod{\mathbb{Z}_M \times G}$ [2, lemma 2]. Expanding this equation, we finally get that $M\alpha = 0 \pmod{G}$, which means that M needs to be a multiple of the order of α.

D.2 Proof of Theorem 87

In this section we will prove Theorem 87, which we restate here for convenience:

Theorem 87 (Simulation of black-box normalizer circuits). *Black-box normalizer circuits can be efficiently simulated classically using the stabilizer formalism over Abelian groups if a subroutine for solving the group-decomposition problem is provided as an oracle.*

The proof uses results of Section 5.2.5; the reader may wish to review that section before proceeding with this proof.

Given a black-box normalizer circuit acting on a black-box group $G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbf{B}$, there are two things we need to do to de-blackbox it, so that the circuit can be classically simulated:

1. Decompose the black-box portion of G, $\mathbf{B}$, into cyclic subgroups: $\mathbf{B} = \mathbb{Z}_{N_{c+1}} \times \cdots \times \mathbb{Z}_{N_{c+d}}$.
2. Calculate *normal forms* (Sections 3.9, 3.10) for each of the normalizer gates in the computation.

Assuming we have access to an oracle for Group Decomposition, Task 1 can already be done. In this proof we will concentrate on tackling task 2, for group automorphisms and quadratic phase gates (a quantum Fourier transform is easily specified by the subgroup it acts on).

D.2.1 Group automorphisms

Suppose we have an abelian group $G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbb{T}^b$; recall that we can represent each element $g \in G$ as an $a + b + c$-tuple of real numbers $g = (g_1, \cdots, g_m)$, where each of the g_i's are only defined modulo the characteristic $\mathrm{char}(G_i)$ of the group G_i, multiplied by some integer. (We have $\mathrm{char}(\mathbb{Z}) = 0$, $\mathrm{char}(\mathbb{T}) = 1$, and $\mathrm{char}(\mathbb{Z}_N) = N$.) Let us first remember the normal form for group automorphisms:

Theorem 51 (Normal form of a matrix representation). *Let $G = G_1 \times \cdots \times G_m$ be an elementary Abelian group. A real $n \times m$ matrix A is a valid matrix representation of some group automorphism $\alpha : G \to G$ iff A is of the form*

$$A := \begin{pmatrix} A_{\mathbb{Z}\mathbb{Z}} & 0 & 0 \\ A_{F\mathbb{Z}} & A_{FF} & 0 \\ A_{\mathbb{T}\mathbb{Z}} & A_{\mathbb{T}F} & A_{\mathbb{T}\mathbb{T}} \end{pmatrix} \tag{D.2.1}$$

with the following restrictions:

1. *$A_{\mathbb{Z}\mathbb{Z}}$ and $A_{\mathbb{T}\mathbb{T}}$ are arbitrary integer matrices.*

2. *$A_{F\mathbb{Z}}$, A_{FF} are integer matrices: the first can be arbitrary, while the coefficients of the second must be of the form*

$$A(i,j) = \alpha_{i,j} \frac{N_i}{\gcd(N_i, N_j)} \tag{D.2.2}$$

   where $\alpha_{i,j}$ can be arbitrary integers, and N_i is the order of the i-th cyclic subgroup of F, $\mathbb{Z}_{N_i}$. The coefficients of the i-th rows of these matrices can be chosen w.l.o.g. to lie in the range $[0, N_i)$ (by taking remainders).

3. *$A_{\mathbb{T}\mathbb{Z}}$ and $A_{\mathbb{T}F}$ are real matrices: the former is arbitrary, while the coefficients of the latter are of the form $A(i,j) = \alpha_{i,j}/N_j$, where $\alpha_{i,j}$ can be arbitrary integers, and N_i is the order of the i-th cyclic subgroup of F, $\mathbb{Z}_{N_i}$. (Due to the periodicity of the torus, the coefficients of $A_{\mathbb{T}\mathbb{Z}}$, $A_{\mathbb{T}F}$ can be chosen to lie in the range $[0, 1)$.)*

Recall the underlying group is $G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbf{B}$ with $\mathbf{B} \cong \mathbb{Z}_{N_{c+1}} \times \cdots \times \mathbb{Z}_{N_{c+d}}$. Assume we are given black box access to a group automorphism $\alpha : G \to G$ implemented as a classical function (a uniformly generated circuit family, say). We wish to find a matrix representation for A for f. We will assume (for the efficiency of this algorithm) that the size and precision of the coefficients are upper bounded by some known constant D, i.e. each element of M can be written as $A_{i,j} = \alpha_{i,j}/\beta_{i,j}$ for integers $\alpha_{i,j}, \beta_{i,j}$ with absolute value no more than D.[1]

We will show how to find the matrix representation A in two steps:

1. Given access to α, we show how to switch the input and output of α from the black-box encoding (where the group action is implemented as a black-box circuit) to the decomposed group encoding (where elements of the group are given as a list of numbers, and the group action is simply addition of vectors), and vice versa.

2. Once we have achieved this, we can implement classically a rational function $f : \mathbb{Q}^{\ell+m+k} \to \mathbb{Q}^{\ell+m+k}$ that implements α in decomposed form. We will show how to obtain the matrix representation A from this f.

Switching from black-box encoding to decomposed group encoding.

We need to be able to convert elements back and forth from the black-box encoding and the decomposed group encoding. We assume our black box group $\mathbf{B}$ has already been decomposed, i.e. we have found linearly independent generators $b_1, \cdots, b_{k'}$ of $\mathbf{B}$ such that $\mathbf{B} = \langle \beta_1 \rangle \oplus \cdots \oplus \langle \beta_\ell \rangle$; moreover, we know the order $N_i' = N_{c+i}$ of β_i. Define the explicitly decomposed group $\mathbb{Z}_{\mathbf{B}} = \mathbb{Z}_{N_1'} \times \cdots \times \mathbb{Z}_{N_d'}$; then we need to be able to perform the following tasks:

[1]Note that D can be inferred from the precision bound (see Section 3.7) of the automorphism gate. If the automorphism gate increases the input size by at most n bits, then it follows that the size of the denominator or numerator of every matrix element can increase by at most $D = 2^n$. A similar argument will hold for quadratic phase gates.

(a) Our first task is to map an element from the decomposed group $\mathbb{Z}_{\mathbf{B}}$ to the black box group $\mathbf{B}$. In otherwords, we need to be able to compute the following group homomorphism φ:

$$\varphi : \mathbb{Z}_{\mathbf{B}} \to \mathbf{B}, \qquad \varphi(g) = b_1^{g(1)} \cdots b_d^{g(d)}, \quad \text{for any } g \in \mathbb{Z}_{\mathbf{B}}.$$

(b) Our second task is to convert elements from the original black-box group encoding to the new encoding defined by $\mathbb{Z}_{\mathbf{B}}$. In other words, given an arbitrary $\mathbf{b} \in \mathbf{B}$, we need to be able to compute $\varphi^{-1}(\mathbf{b})$.

Note that it is always possible to compute $\varphi(g) = b_1^{g(1)} \cdots b_d^{g(d)}$ for any $g \in \mathbb{Z}_{\mathbf{B}}$, since this can be done using a polynomial number of queries to the black-box group oracle (using repeated squaring if necessary for the exponentiation). Task (a) is therefore immediate.

As for Task (b), we note that computing $\varphi^{-1}(\mathbf{b})$ for an element $\mathbf{b} \in \mathbf{B}$ is equivalent to finding a list of integers $(g(1), \cdots, g(d))$ such that $b_1^{g(1)} \cdots b_d^{g(d)} = \mathbf{b}$. This is a special case of the multivariate discrete logarithm problem, defined in Lemma 85; from Lemma 85 we see that Task (b) can be solved efficiently with a polynomial number of calls to the Group Decomposition oracle.

Finding the matrix representation A

Now by converting the input and output of α to the decomposed group representation, we may assume that we instead have a classical rational function $f : \mathbb{Q}^{a+b+c+d} \to \mathbb{Q}^{a+b+c+d}$ such that f can be treated as a group automorphism on G:

$$f(x) \equiv f(x') \bmod G \quad \text{if } x \equiv x' \bmod G. \tag{D.2.3}$$

(Here we say two vectors are equal modulo G if each pair of corresponding entries are equal modulo $\operatorname{char}(G_i)$.) We wish to find a matrix representation A for f. For most entries of A this is trivial: note that we have

$$A_{i,j} \equiv f(e_j)_i \bmod c_i \tag{D.2.4}$$

where $c_i = \operatorname{char}(G_i)$. Hence by evaluating f on the unit vectors e_i, we can determine $A_{i,j}$ modulo c_i. Thus we can evalute $A_{\mathbb{T}F}$ exactly, the coefficients of the i-th rows of $A_{F\mathbb{Z}}$ and A_{FF} modulo $\mathbb{Z}_{N_i}$, and the coefficients of $A_{\mathbb{T}\mathbb{Z}}$ and $A_{\mathbb{T}F}$ modulo 1. This is sufficient for the cases listed above; the only case we still need to treat is $A_{\mathbb{T}\mathbb{T}}$, whose entries are arbitrary integers (and $c_i = \operatorname{char}(\mathbb{T}) = 1$ in this case). We can instead evaluate $f(e_j/\alpha)$ for some large integer α:

$$A_{i,j}/\alpha \equiv f(e_j/\alpha)_i \bmod c_i \tag{D.2.5}$$

which allows us to determine $A_{i,j}$ modulo αc_i for our choice of α. Choosing $\alpha > 2D$ then allows us to determine $A_{i,j}$ exactly for the case of $A_{\mathbb{T}\mathbb{T}}$.

D.2.2 Quadratic phase functions

We will continue to use the matrix representation referenced in the last section. Recall the following theorem:

Theorem 57 (Normal form of a quadratic function, extended). *Let $G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times \mathbb{T}^b$ be an elementary Abelian group. Define $\mathbb{Z}_N^\bullet = \{0, 1/N, \cdots, (N-1)/N\}$ to be a group under addition modulo 1, and let $G^\bullet = \mathbb{T}^a \times \mathbb{Z}_{N_1}^\bullet \times \cdots \times \mathbb{Z}_{N_c}^\bullet \times \mathbb{Z}^b$. Then a function $\xi : G \to U(1)$ is quadratic if and only if*

$$\xi(g) = \mathrm{e}^{\pi \mathrm{i}\left(g^T M g + C^T g + 2v^T g\right)} \tag{D.2.6}$$

where C, v, M are, respectively, two vectors and a matrix that satisfy the following:

- *v is an element of $G^\bullet$;*
- *M is the matrix representation of a group homomorphism from G to $G^\bullet$, which necessarily has the form*

$$M := \begin{pmatrix} M_{\mathbb{T}\mathbb{Z}} & M_{\mathbb{T}F} & M_{\mathbb{T}\mathbb{T}} \\ M_{F^\bullet\mathbb{Z}} & M_{F^\bullet F} & 0 \\ M_{\mathbb{Z}\mathbb{Z}} & 0 & 0 \end{pmatrix} \tag{D.2.7}$$

  with the following restrictions:

  - *$M_{\mathbb{Z}\mathbb{Z}}$ and $M_{\mathbb{T}\mathbb{T}}$ are arbitrary integer matrices.*
  - *$M_{F^\bullet\mathbb{Z}}$ and $M_{\mathbb{T}F}$ are rational matrices, the former with the form $M(i,j) = \alpha_{i,j}/N_i$ and the latter with the form $M(i,j) = \alpha_{i,j}/N_j$, where $\alpha_{i,j}$ are arbitrary integers, and N_i is the order of the i-th cyclic subgroup $\mathbb{Z}_{N_i}$.*
  - *$M_{F^\bullet F}$ is a rational matrix with coefficients of the form*

$$M(i,j) = \frac{\alpha_{i,j}}{\gcd(N_i, N_j)} \tag{D.2.8}$$

    where $\alpha_{i,j}$ are arbitrary integers, and N_i is the order of the i-th cyclic subgroup $\mathbb{Z}_{N_i}$.
  - *$M_{\mathbb{T}\mathbb{Z}}$ is an arbitrary real matrix.*

  The entries of $M_{F^\bullet\mathbb{Z}}$, $M_{\mathbb{T}F}$, $M_{F^\bullet F}$, and $M_{\mathbb{T}\mathbb{Z}}$ can be assumed to lie in the interval $[0,1)$. Moreover, M can be assumed to be symmetric, i.e. $M_{\mathbb{Z}\mathbb{Z}}^T = M_{\mathbb{T}\mathbb{T}}$, $M_{F^\bullet\mathbb{Z}}^T = M_{\mathbb{T}F}$, $M_{F^\bullet F}^T = M_{F^\bullet F}$, and $M_{\mathbb{T}\mathbb{Z}}^T = M_{\mathbb{T}\mathbb{Z}}$.

- *C is an integer vector dependent on M, defined component-wise as $C(i) = M(i,i)c_i$, where c_i is the characteristic of the group G_i. (Recall that* char$(\mathbb{Z}) = 0$, char$(\mathbb{T}) = 1$, *and* char$(\mathbb{Z}_N) = N$.*)*

Recall the underlying group is $G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c} \times B$ with $B \cong \mathbb{Z}_{N_{c+1}} \times \cdots \times \mathbb{Z}_{N_{c+d}}$. Assume we are given a quadratic phase gate ξ, implemented as a classical circuit family $q : G \to \mathbb{Q}$ such that

$$\xi(g) = \mathrm{e}^{2\pi \mathrm{i} q(g)} \quad \forall g \in G. \tag{D.2.9}$$

Since we can switch between the black-box group and decomposed group encodings (see Section D.2.1), we can assume the elements of G are treated as a vector in $\mathbb{Q}^{a+b+c+d}$.

We wish to write the quadratic function $\xi(g)$ in the normal form given by Theorem 57, i.e. find M, c, v as in Theorem 57 such that

$$\xi(g) = \mathrm{e}^{\pi \mathrm{i}\left(g^T M g + C^T g + 2v^T g\right)}. \tag{D.2.10}$$

q, and hence M, c, and v, are rational by assumption. We will furthermore assume, as before, that the size and precision of the coefficients are upper bounded by some known constant D, i.e. each element of M can be written as $M_{i,j} = \alpha_{i,j}/\beta_{i,j}$ for integers $\alpha_{i,j}, \beta_{i,j}$ with absolute value no more than D.

To do this, let us first determine the entries of M. This can be done in the following manner: it should be straightforward to verify that

$$\xi(x+y) = \xi(x)\xi(y)\mathrm{e}^{2\pi\mathrm{i}\, x^T M y} \tag{D.2.11}$$

for any $x, y \in G$, and therefore

$$x^T M y \equiv q(x+y) - q(x) - q(y) \bmod \mathbb{Z}. \tag{D.2.12}$$

We can use this method to determine nearly all the entries of M exactly, by taking x and y to be unit vectors e_i and e_j; this would determine M_{ij} up to an integer, i.e.

$$M_{i,j} = e_i^T M e_j \equiv q(e_i + e_j) - q(e_i) - q(e_j) \bmod \mathbb{Z}. \tag{D.2.13}$$

This determines all entries of M except for those in $M_{\mathbb{Z}\mathbb{Z}}$ and $M_{\mathbb{T}\mathbb{T}}$ (the other entries can be assumed to lie in $[0,1)$). To deal with $M_{\mathbb{Z}\mathbb{Z}}$ we take $x = \alpha^{-1}e_i$, and $y = e_j$, such that the coefficient $M(i,j)$ is in the submatrix $M_{\mathbb{Z}\mathbb{Z}}$ and $1/\alpha$ is an element of the circle group with $\alpha < 2D$, where D is the precision bound. We obtain an analogous equation

$$\left(\frac{e_i^T}{\alpha} M e_j\right) \equiv \frac{M_{i,j}}{\alpha} \equiv q(\alpha^{-1}e_i + e_j) - q(\alpha^{-1}e_i) - q(e_j) \bmod \mathbb{Z}, \tag{D.2.14}$$

which allows us to determine $M_{i,j}$: since the number $M_{i,j}/\alpha$ is smaller than $1/2$ in absolute value, the coefficient is not truncated modulo 1. One can apply the same argument to obtain the coefficients of $M_{\mathbb{T}\mathbb{T}}$, choosing $x = e_i$, and $y = \alpha^{-1}e_j$.

Once we determine all the entries of M in this manner, we get immediately the vector C as well (since $C(i) = c_i M(i,i)$). It is then straightforward to calculate the vector $\tilde{v}$. Thus we can efficiently find the normal form of $\xi(g)$ through polynomially many uses of the classical function q.

D.3 Extending Theorem 87 to the Abelian HSP setting

In this appendix, we briefly discuss that Theorem 87 (and some of the results that follow from this theorem) can be re-proven in the general hidden subgroup problem oracular setting that we studied in Section 5.2.4. This fact supports our view (discussed in the main text) that the oracle models in the HSP and in the black-box setting are very close to each other.

Recall that the main result in this section (Theorem 84) states that the quantum algorithm Abelian HSP is a normalizer circuits over a group of the form $\mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_m} \times \mathcal{O}$, where $\mathcal{O}$ is a group associated with the Abelian HSP oracle f via the formula (5.2.17). The group $\mathcal{O}$ is not a black-box group, because no oracle to multiply in $\mathcal{O}$ was provided. However, we discussed at the end of Section 5.2.4 that one can use the hidden subgroup problem oracle to perform certain multiplications implicitly.

We show next that Theorem 87 can be re-casted in the HSP setting as "*the ability to decompose the oracular group $\mathcal{O}$ renders normalizer circuits over $\mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_m} \times \mathcal{O}$ efficiently classically simulable*". To see this, assume a group decomposition table (α, β, A, B, c) is given. Then we know $\mathcal{O} \cong \mathbb{Z}_{c_1} \times \cdots \times \mathbb{Z}_{c_m}$. Let us now view the function $\alpha(g) = (g, f(g))$ used in the HSP quantum algorithm as a group automorphism of $G \times Z_{c_1} \times \cdots \times \mathbb{Z}_{c_m}$, where we decompose $\mathcal{O}$. Then, it is easy to check that $\begin{pmatrix} 1 & 0 \\ B & 1 \end{pmatrix}$ is a matrix representation of this map. It follows that the group decomposition table can be used to de-black-box the HSP oracle, and this fact allows us to adapt the proof of Theorem 87 step-by-step to this case.

We point out further that the extended Cheung-Mosca algorithm can be adapted to the HSP setting, showing that normalizer circuits over $G \times \mathcal{O}$ can be used to decompose $\mathcal{O}$. This follows from the fact that the function f that we need to query to decompose $\mathbf{B}$ using the extended Cheung-Mosca algorithm (algorithm 86) has precisely the same form as the HSP oracle. Using the HSP oracle as a subroutine in algorithm 86 (which we can query *by promise*), the algorithm computes a group decomposition tuple for $\mathcal{O}$.

Finally, we can combine these last observations with Theorem 91 and conclude that the problem of decomposing groups of the form $\mathcal{O}$ is classically polynomial-time equivalent to the Abelian hidden subgroup problem. The proof is analogous to that of Theorem 91.

Bibliography

[1] M. Van den Nest, "Efficient classical simulations of quantum Fourier transforms and normalizer circuits over Abelian groups," *Quantum Information and Computation* **0** no. 1, (2012) , arXiv:1201.4867v1 [quant-ph].

[2] J. Bermejo-Vega and M. Van Den Nest, "Classical simulations of Abelian-group normalizer circuits with intermediate measurements," *Quantum Information and Computation* **14** no. 3-4, (2014) , arXiv:1210.3637 [quant-ph].

[3] M. A. Neilsen and I. L. Chuang, *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[4] P. R. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing.* Oxford University Press, 2007.

[5] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London Mathematical Society* **2(42)** (1937) 230–265.

[6] A. Church, "An unsolvable problem of elementary number theory," *American Journal of Mathematics* **58** no. 345-363, (1936) .

[7] D. Hilbert and W. Ackermann, *Grundzüge der theoretischen Logik.* Springer-Verlag, 1928.

[8] T. Cubitt, D. Perez-Garcia, and M. M. Wolf, "Undecidability of the spectral gap (short version)," arXiv:1502.04135 [quant-ph].

[9] T. Cubitt, D. Perez-Garcia, and M. M. Wolf, "Undecidability of the spectral gap (full version)," arXiv:1502.04573 [quant-ph].

[10] J. Eisert, M. P. Mueller, and C. Gogolin, "Quantum measurement occurrence is undecidable," arXiv:1111.3965 [quant-ph].

[11] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES is in P," *Annals of Mathematics* **160** no. 2, (2004) 781–793.

[12] R. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics* **21 (6-7)** (1982) 467–488.

[13] D. Deutsch, "Quantum theory, the church-turing principle and the universal quantum computer," *Proceedings of the Royal Society A* **400 (1818)** (1985) 97–117.

[14] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Scientific and Statistical Computing* **26** (1997) , arXiv:quant-ph/9508027.

[15] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM* **21** (1978) .

[16] D. Gottesman, "The Heisenberg representation of quantum computers," in *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics.* International Press, 1999. arXiv:quant-ph/9807006v1.

[17] D. Gottesman, *Stabilizer Codes and Quantum Error Correction.* PhD thesis, California Institute of Technology, 1997. arXiv:quant-ph/9705052v1.

[18] S. Jordan, "Quantum algorithm zoo." http://math.nist.gov/quantum/zoo/.

[19] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC).* May, 1996. arXiv:quant-ph/9605043.

[20] A. Y. Kitaev, "Quantum measurements and the Abelian stabilizer problem," arXiv:quant-ph/9511026.

[21] C. Lomont, "The hidden subgroup problem - review and open problems," *arXiv* (2004) , arXiv:arXiv:quant-ph/0411037v1.

[22] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals and Systems.* Prentice-Hall, 2nd ed., 1996.

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

[24] A. M. Childs and W. van Dam, "Quantum algorithms for algebraic problems," *Reviews of Modern Physics* **82** (2010) , arXiv:0812.0380v1 [quant-ph].

[25] M. Ettinger, P. Hoyer, and E. Knill, "The quantum query complexity of the hidden subgroup problem is polynomial," *Information Processing Letters* **91** no. 1, (2004) , arXiv:quant-ph/0401083.

[26] A. Ambainis, "Quantum walk algorithm for element distinctness," *SIAM Journal on Computing* **37** no. 1, (2007) 210–239, arXiv:quant-ph/0311001.

[27] M. Szegedy, "Quantum speed-up of Markov chain based algorithms," in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS).* 2004.

[28] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via quantum walk," *SIAM Journal on Computing* **40** no. 1, (2011) 142–164, arXiv:quant-ph/0608026.

[29] S. Jeffery, R. Kothari, and F. Magniez, "Nested quantum walks with quantum data structures," in *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1474–1485. 2013. arXiv:1210.1199 [quant-ph].

[30] A. Belovs, "Span programs for functions with constant-sized 1-certificates," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 77–84. 2012. arXiv:1105.4024 [quant-ph].

[31] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM Journal on Computing* **26** no. 5, (1997) 1510–1523, arXiv:quant-ph/9701001.

[32] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, "Quantum lower bounds by polynomials," in *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, p. 352. 1998. arXiv:quant-ph/9802049.

[33] A. Ambainis, "Quantum lower bounds by quantum arguments," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 636–643. 2000. arXiv:quant-ph/0002066.

[34] P. Høyer, T. Lee, and R. Špalek, "Negative weights make adversaries stronger," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 526–535. 2007. arXiv:quant-ph/0611054.

[35] B. W. Reichardt, "Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function," in *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 544–551. 2009. arXiv:0904.2759 [quant-ph].

[36] B. W. Reichardt, "Reflections for quantum query algorithms," in *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 560–569. 2011. arXiv:1005.1601 [quant-ph].

[37] T. Lee, R. Mittal, B. W. Reichardt, R. Špalek, and M. Szegedy, "Quantum query complexity of state conversion," in *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 344–353. 2011. arXiv:1011.3020 [quant-ph].

[38] C. Y.-Y. Lin and H.-H. Lin, "Upper bounds on quantum query complexity inspired by the Elitzur-Vaidman bomb tester," arXiv:1410.0932 [quant-ph].

[39] P. Kwiat, H. Weinfurter, T. Herzog, A. Zeilinger, and M. A. Kasevich, "Interaction-free measurement," *Physical Review Letters* **74** no. 24, (1995) 4763.

[40] B. Furrow, "A panoply of quantum algorithms," *Quantum Information and Computation* **8** no. 8, (September, 2008) 834–859, arXiv:quant-ph/0606127.

[41] A. Ambainis and R. Špalek, "Quantum algorithms for matching and network flows," in *Lecture Notes in Computer Science*, vol. 3884, pp. 172–183. Springer, 2006. arXiv:quant-ph/0508205.

[42] S. Dörn, "Quantum algorithms for matching problems," *Theory of Computing Systems* **45** no. 3, (October, 2009) 613–628.

[43] O. Regev and L. Schiff, "Impossibility of a quantum speed-up with a faulty oracle," in *Lecture Notes in Computer Science*, vol. 5125, pp. 773–781. Springer, 2008. arXiv:1202.1027 [quant-ph].

[44] G. Mitchison and R. Jozsa, "Counterfactual computation," *Proceedings of the Royal Society A* **457** no. 2009, (2001) 1175–1194, arXiv:quant-ph/9907007.

[45] O. Hosten, M. T. Rakher, J. T. Barreiro, N. A. Peters, and P. G. Kwiat, "Counterfactual quantum computation through quantum interrogation," *Nature* **439** (February, 2006) 949–952.

[46] G. Mitchison and R. Jozsa, "The limits of counterfactual computation," arXiv:quant-ph/0606092.

[47] O. Hosten, M. T. Rakher, J. T. Barreiro, N. A. Peters, and P. Kwiat, "Counterfactual computation revisited," arXiv:quant-ph/0607101.

[48] L. Vaidman, "The impossibility of the counterfactual computation for all possible outcomes," arXiv:quant-ph/0610174.

[49] O. Hosten and P. G. Kwiat, "Weak measurements and counterfactual computation," arXiv:quant-ph/0612159.

[50] H. Salih, Z.-H. Li, M. Al-Amri, and M. S. Zubairy, "Protocol for direct counterfactual quantum communication," *Physical Review Letters* **110** (2013) 170502, arXiv:1206.2042 [quant-ph].

[51] L. Vaidman, "Comment on "protocol for direct counterfactual quantum communication" [arxiv:1206.2042]," arXiv:1304.6689 [quant-ph].

[52] T.-G. Noh, "Counterfactual quantum cryptography," *Physical Review Letters* **103** (2009) 230501, arXiv:0809.3979 [quant-ph].

[53] A. Brodutch, D. Nagaj, O. Sattath, and D. Unruh, "An adaptive attack on Wiesner's quantum money," arXiv:1404.1507 [quant-ph].

[54] S. Wiesner, "Conjugate coding," *ACM SIGACT News* **15** no. 1, (1983) .

[55] R. Kothari, "An optimal quantum algorithm for the oracle identification problem," in *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS)*, E. W. Mayr and N. Portier, eds., vol. 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 482–493. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2014. arXiv:1311.7685 [quant-ph].

[56] S. Aaronson. Personal communication, 2014.

[57] S. Micali and V. V. Vazirani, "An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs," in *Proceedings of the 21st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 17–27. 1980.

[58] S. Cook, C. Dwork, and R. Reischuk, "Upper and lower time bounds for parallel random access machines without simultaneous writes," *SIAM Journal on Computing* **15** no. 1, (1986) 87–97.

[59] N. Nisan, "CREW PRAMs and decision trees," *SIAM Journal on Computing* **20** no. 6, (1991) 999–1007.

[60] H. Buhrman and R. D. Wolf, "Complexity measures and decision tree complexity: A survey," *Theoretical Computer Science* **288** (1999) 2002.

[61] A. C. Elitzur and L. Vaidman, "Quantum mechanical interaction-free measurements," *Foundations of Physics* **23** no. 7, (July, 1993) 987–997, arXiv:hep-th/9305002.

[62] B. Misra and E. C. G. Sudarshan, "The Zeno's paradox in quantum theory," *Journal of Mathematical Physics* **18** no. 4, (1977) 756.

[63] C. Dürr and P. Høyer, "A quantum algorithm for finding the minimum," arXiv:quant-ph/9607014.

[64] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla, "Quantum query complexity of some graph problems," arXiv:quant-ph/0401091.

[65] T. H. Cormen, C. E. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press and McGraw-Hill, 3rd ed., 2009.

[66] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing* **2** no. 4, (1973) 225–231.

[67] C. Berge, "Two theorems in graph theory," in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 43, pp. 842–844. 1957.

[68] A. Berzina, A. Dubrovsky, R. Freivalds, L. Lace, and O. Scegulnaja, "Quantum query complexity for some graph problems," in *Lecture Notes in Computer Science*, vol. 2932, pp. 140–150. Springer, 2004.

[69] S. Zhang, "On the power of Ambainis's lower bounds," *Theoretical Computer Science* **339** no. 2-3, (2005) 241–256, arXiv:quant-ph/0311060.

[70] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Math Doklady* **11** (1970) 1277–1280.

[71] A. V. Karzanov, "O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh," in *Matematicheskie Voprosy Upravleniya Proizvodstvom*, L. Lyusternik, ed., vol. 5, pp. 81–94. Moscow State University Press, 1973.

[72] S. Even and R. E. Tarjan, "Network flow and testing graph connectivity," *SIAM Journal on Computing* **4** no. 4, (1975) 507–518.

[73] K. K. H. Cheung and M. Mosca, "Decomposing finite Abelian groups," *Quantum Information and Computation* **1** no. 3, (2001) , arXiv:cs/0101004.

[74] J. Bermejo-Vega, C. Y.-Y. Lin, and M. V. den Nest, "Normalizer circuits and a Gottesman-Knill theorem for infinite-dimensional systems," arXiv:1409.3208 [quant-ph].

[75] J. Bermejo-Vega, C. Y.-Y. Lin, and M. V. den Nest, "The computational power of normalizer circuits over black-box groups," arXiv:1409.4800 [quant-ph].

[76] E. Knill, "Non-binary unitary error bases and quantum codes," tech. rep., Los Alamos National Laboratory, 1996. arXiv:quant-ph/9608048.

[77] D. Gottesman, "Fault-tolerant quantum computation with higher-dimensional systems," in *Selected papers from the First NASA International Conference on Quantum Computing and Quantum Communications.* Springer, 1998. arXiv:quant-ph/9802007v1.

[78] G. Brassard and P. Høyer, "An exact quantum polynomial-time algorithm for simon's problem," in *Proceedings of the Fifth Israel Symposium on the Theory of Computing Systems (ISTCS '97)*, ISTCS '97. IEEE Computer Society, Washington, DC, USA, 1997. arXiv:quant-ph/9704027.

[79] P. Høyer, "Conjugated operators in quantum algorithms," *Physical Review A* **59** (1999) .

[80] M. Mosca and A. Ekert, "The hidden subgroup problem and eigenvalue estimation on a quantum computer," in *Selected Papers from the First NASA International Conference on Quantum Computing and Quantum Communications*, QCQC '98. Springer, 1998. arXiv:quant-ph/9903071.

[81] I. Damgård, "QIP note: on the quantum Fourier transform and applications." http://www.daimi.au.dk/~ivan/fourier.ps.

[82] L. Babai and R. Beals, "A polynomial-time theory of black-box groups I," in *Groups St Andrews 1997 in Bath*, vol. I of *London Mathematical Society Lecture Note Series.* Cambridge University Press, 1999.

[83] L. Babai and E. Szemerédi, "On the complexity of matrix group problems I," in *Proceedings of the 25th Annual Symposium onFoundations of Computer Science, 1984*, SFCS '84. IEEE Computer Society, 1984.

[84] M. Mosca, *Quantum computer algorithms.* PhD thesis, University of Oxford, 1999.

[85] D. Gross, "Hudson's theorem for finite-dimensional quantum systems," *Journal of Mathematical Physics* **47** no. 12, (2006) , arXiv:quant-ph/0602001.

[86] D. Dikranjan, "Introduction to topological groups." 2010.

[87] J. F. Humphreys, *A course in group theory.* Oxford University Press, 1996.

[88] V. Shoup, *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press, 2nd ed., 2008.

[89] R. P. Brent and P. Zimmermann, *Modern Computer Arithmetic.* Cambridge University Press, 2010.

[90] F. Fontein and P. Wocjan, "On the probability of generating a lattice," *Journal of Symbolic Computation* **64** (2014) 3 – 15, arXiv:1211.6246 [quant-ph].

[91] F. Bruhat, "Distributions sur un groupe localement compact et applications à l etude des représentations des groupes p-adiques," *Bulletin de la Société Mathématique de France* **89** (1961) .

[92] M. S. Osborne, "On the Schwartz-Bruhat space and the Paley-Wiener theorem for locally compact Abelian groups," *Journal of Functional Analysis* **19** (1975) .

[93] R. de la Madrid, "The role of the rigged hilbert space in quantum mechanics," *European Journal of Physics* **26** no. 2, (2005) , arXiv:quant-ph/0502053.

[94] J. P. Antoine, "Quantum mechanics beyond Hilbert space," in *Irreversibility and Causality Semigroups and Rigged Hilbert Spaces*, vol. 504 of *Lecture Notes in Physics.* Springer, 1998.

[95] M. Gadella and F. Gómez, "A unified mathematical formalism for the Dirac formulation of quantum mechanics," *Foundations of Physics* **32** no. 6, (2002) .

[96] M. Gadella, F. Gómez, and S. Wickramasekara, "Riggings of locally compact Abelian groups.," *Journal of Geometry and Symmetry in Physics* **11** (2008) .

[97] S. A. Morris, *Pontryagin Duality and the Structure of Locally Compact Abelian Groups.* Cambridge University Press, 1977.

[98] M. Stroppel, *Locally Compact Groups.* EMS Textbooks in Mathematics. European Mathematical Society, 2006.

[99] W. Rudin, *Fourier analysis on groups.* No. 12 in Interscience Tracts in Pure and Applied Mathematics. Interscience Publishers, 1962.

[100] K. H. Hofmann and S. A. Morris, *The Structure of Compact Groups.* No. 25 in de Gruyter Studies in Mathematics. Walter de Gruyter.

[101] D. L. Armacost, *The structure of locally compact abelian groups.* M. Dekker New York, 1981.

[102] J. Baez, "The n-category café: Locally compact Hausdorff Abelian groups," 2008. http://golem.ph.utexas.edu/category/2008/11/locally_compact_hausdorff_abel.html.

[103] H. Cohen, *A Course in Computational Algebraic Number Theory.* Springer, 1993.

[104] P. Kaye, "Optimized quantum implementation of elliptic curve arithmetic over binary fields," *Quantum Information and Computation* **5** no. 6, (2005) , arXiv:quant-ph/0407095.

[105] C. J. Moreno, *Advanced Analytic Number Theory: L-Functions*, vol. 15 of *Mathematical Surveys and Monographs.* American Mathematical Society, 2005.

[106] A. Prasad and M. K. Vemuri, "Decomposition of phase space and classification of Heisenberg groups," arXiv:0806.4064 [quant-ph].

[107] N. B. Backhouse and C. J. Bradely, "Projective representations of Abelian groups," in *Proceedings of the American Mathematical Society*, vol. 36. American Mathematical Society, 1972.

[108] J. Dehaene and B. De Moor, "Clifford group, stabilizer states, and linear and quadratic operations over GF(2)," *Physical Review A* **68** (2003) , arXiv:quant-ph/0304125v1.

[109] E. Hostens, J. Dehaene, and B. De Moor, "Stabilizer states and Clifford operations for systems of arbitrary dimensions and modular arithmetic," *Physical Review A* **71** (2005) , arXiv:quant-ph/0408190v2.

[110] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A* **70** (2004) , arXiv:quant-ph/0406196.

[111] N. de Beaudrap, "A linearized stabilizer formalism for systems of finite dimension," *Quantum Information and Computation* **13** no. 1-2, (2013) , arXiv:1102.3354v3 [quant-ph].

[112] V. J. Bowman and C.-A. Burdet, "On the general solution to systems of mixed-integer linear equations," *SIAM Journal on Applied Mathematics* **26** no. 1, (1974) .

[113] S. L. Braunstein, "Error correction for continuous quantum variables," *Physical Review Letters* **80** (1998) , arXiv:quant-ph/9711049.

[114] S. Lloyd and J.-J. E. Slotine, "Analog quantum error correction," *Physical Review Letters* **80** (1998) , arXiv:quant-ph/9711021.

[115] D. Gottesman, A. Kitaev, and J. Preskill, "Encoding a qubit in an oscillator," *Physical Review A* **64** (2001) .

[116] S. D. Bartlett, B. C. Sanders, S. L. Braunstein, and K. Nemoto, "Efficient classical simulation of continuous variable quantum information processes," *Physical Review Letters* **88** (2002) , arXiv:quant-ph/0109047.

[117] S. D. Bartlett and B. C. Sanders, "Efficient classical simulation of optical quantum information circuits," *Physical Review Letters* **89** (2002) , arXiv:quant-ph/0204065.

[118] R. L. Barnes, "Stabilizer codes for continuous-variable quantum error correction," arXiv:quant-ph/0405064.

[119] S. Lloyd and S. L. Braunstein, "Quantum computation over continuous variables," *Physical Review Letters* **82** (1999) , arXiv:quant-ph/9810082.

[120] S. L. Braunstein and P. van Loock, "Quantum information with continuous variables," *Reviews of Modern Physics* **77** (2005) , arXiv:quant-ph/0410100.

[121] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, and S. Lloyd, "Gaussian quantum information," *Reviews of Modern Physics* **84** (2012) .

[122] N. C. Menicucci, "Fault-tolerant measurement-based quantum computing with continuous-variable cluster states," *Physical Review Letters* **112** (2014) , arXiv:1310.7596 [quant-ph].

[123] J. Zhang and S. L. Braunstein, "Continuous-variable gaussian analog of cluster states," *Physical Review A* **73** (2006) .

[124] N. C. Menicucci, P. van Loock, M. Gu, C. Weedbrook, T. C. Ralph, and M. A. Nielsen, "Universal quantum computation with continuous-variable cluster states," *Physical Review Letters* **97** (2006) . http://link.aps.org/doi/10.1103/PhysRevLett.97.110501.

[125] M. Gu, C. Weedbrook, N. C. Menicucci, T. C. Ralph, and P. van Loock, "Quantum computing with continuous-variable clusters," *Physical Review A* **79** (2009) , arXiv:0903.3233 [quant-ph].

[126] G. Aruldhas, *Quantum Mechanics: 500 Problems with Solutions.* Prentice-Hall of India, 2010.

[127] M. Van den Nest, "Classical simulation of quantum computation, the gottesman-knill theorem, and slightly beyond," *Quantum Information and Computation* **10** no. 3, (2010) , arXiv:0811.0898 [quant-ph].

[128] V. Veitch, C. Ferrie, D. Gross, and J. Emerson, "Negative quasi-probability as a resource for quantum computation," *New Journal of Physics* **14** no. 11, (2012) , arXiv:1201.1256 [quant-ph].

[129] A. Mari and J. Eisert, "Positive wigner functions render classical simulation of quantum computation efficient," *Physical Review Letters* **109** (2012) , arXiv:1208.3660 [quant-ph].

[130] K. S. Gibbons, M. J. Hoffman, and W. K. Wootters, "Discrete phase space based on fi- nite fields," *Phys. Rev. A* **70** (2004) .

[131] D. Gross, "Computational power of quantum many-body states and some results on discrete phase spaces," 2008.

[132] N. Delfosse, P. A. Guerin, J. Bian, and R. Raussendorf, "Wigner function negativity and contextuality in quantum computation on rebits," *arXiv preprint* (2014) , arXiv:1409.5170 [quant-ph].

[133] N. D. Mermin, "Extreme quantum entanglement in a superposition of macroscopically distinct states," *Physical Review Letters* **65** (1990) .

[134] V. Scarani, A. Acín, E. Schenck, and M. Aspelmeyer, "Nonlocality of cluster states of qubits," *Physical Review A* **71** (2005) , arXiv:quant-ph/0405119.

[135] O. Gühne, G. Tóth, P. Hyllus, and H. J. Briegel, "Bell inequalities for graph states," *Physical Review Letters* **95** (2005) , arXiv:quant-ph/0410059.

[136] I. Rigas, L. L. Sánchez-Soto, A. B. Klimov, J. Řeháček, and Z. Hradil, "Non-negative Wigner functions for orbital angular momentum states," *Physical Review A* **81** (2010) , arXiv:0909.1887 [quant-ph].

[137] I. Rigas, L. Sánchez-Soto, A. Klimov, J. Řeháček, and Z. Hradil, "Orbital angular momentum in phase space," *Annals of Physics* **326** no. 2, (2011) , arXiv:1011.6184 [quant-ph].

[138] M. Hinarejos, A. Pérez, and M. C. Bañuls, "Wigner function for a particle in an infinite lattice," *New Journal of Physics* **14** no. 10, (2012) , arXiv:1205.3925 [quant-ph].

[139] A. Kitaev, "Protected qubit based on a superconducting current mirror," arXiv:cond-mat/0609441.

[140] P. Brooks, A. Kitaev, and J. Preskill, "Protected gates for superconducting qubits," *Physical Review A* **87** (2013) .

[141] A. M. Steane, "Overhead and noise threshold of fault-tolerant quantum error correction," *Physical Review A* **68** (2003) , arXiv:quant-ph/0207119.

[142] E. Knill, "Quantum computing with realistically noisy devices," *Nature* **434** (2005) , arXiv:arXiv:quant-ph/0410199.

[143] D. P. DiVincenzo and P. Aliferis, "Effective fault-tolerant quantum computation with slow measurements," *Physical Review Letters* **98** (2007) , arXiv:quant-ph/0607047.

[144] A. Paler, S. Devitt, K. Nemoto, and I. Polian, "Software-based Pauli tracking in fault-tolerant quantum circuits," in *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '14. European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 2014. arXiv:1401.5872 [quant-ph].

[145] P. Kok and B. W. Lovett, *Introduction to Optical Quantum Information Processing.* Cambridge University Press.

[146] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," in *Quantum Information Science and Its Contributions to Mathematics*, vol. 68 of *Proceedings of Symposia in Applied Mathematics.* American Physical Society, 2009. arXiv:0904.2557 [quant-ph].

[147] M. V. den Nest, "A monomial matrix formalism to describe quantum many-body states," *New Journal of Physics* **13** no. 12, (2011) , arXiv:1108.0531v1 [quant-ph].

[148] M. F. Hurt and C. Waid, "A generalized inverse which gives all the integral solutions to a system of linear equations," *SIAM Journal on Applied Mathematics* **19** no. 3, (1970) .

[149] P. Hayden, D. Leung, P. W. Shor, and A. Winter, "Randomizing quantum states: Constructions and applications," *Communications in Mathematical Physics* **250** no. 2, (2004) , arXiv:quant-ph/0307104.

[150] Y. Shi and X. Wu, "Epsilon-net method for optimizations over separable states," in *Automata, Languages, and Programming*, vol. 7391 of *Lecture Notes in Computer Science.* Springer, 2012. arXiv:1112.0808 [quant-ph].

[151] X. Ni and M. Van den Nest, "Commuting quantum circuits: efficiently classical simulations versus hardness results," *Quantum Information and Computation* **13** no. 1&2, (2013) , arXiv:1204.4570 [quant-ph].

[152] M. M. Deza and E. Deza, *Encyclopedia of Distances.* Springer, 2nd ed., 2013.

[153] R. A. Mollin, *Advanced Number Theory with Applications.* Discrete Mathematics and its Applications. CRC Press, 2010.

[154] A. Storjohann, *Algorithms for Matrix Canonical Forms.* PhD thesis, University of Waterloo, 2000.

[155] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," *Quantum Information and Computation* **3** no. 4, (2003) , arXiv:quant-ph/0301141.

[156] D. Cheung, D. Maslov, J. Mathew, and D. Pradhan, "On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography," in *Theory of Quantum Computation, Communication, and Cryptography*, vol. 5106 of *Lecture Notes in Computer Science.* Springer, 2008. arXiv:0710.1093 [quant-ph].

[157] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* **400** no. 1818, (1985) .

[158] D. R. Simon, "On the power of quantum computation," *SIAM Journal on Computing* **26** (1994) .

[159] D. Boneh and R. Lipton, "Quantum cryptanalysis of hidden linear functions," in *Advances in Cryptology — CRYPTO' 95*, D. Coppersmith, ed., vol. 963 of *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, 1995.

[160] D. Grigoriev, "Testing shift-equivalence of polynomials by deterministic, probabilistic and quantum machines," *Theor. Comput. Sci.* **180** no. 1-2, (1997) .

[161] A. Y. Kitaev, "Quantum computations: algorithms and error correction," *Russian Mathematical Surveys* **52** no. 6, (1997) .

[162] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on* **22** no. 6, (1976) .

[163] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography.* CRC Press, 1st ed., 1996.

[164] J. A. Buchmann, *Introduction to Cryptography.* Springer, 1st ed., 2000.

[165] S. Anders and H. J. Briegel, "Fast simulation of stabilizer circuits using a graph-state representation," *Physical Review A* **73** (2006) , arXiv:quant-ph/0504117.

[166] R. Jozsa and M. Van Den Nest, "Classical simulation complexity of extended clifford circuits," *Quantum Information and Computation* **14** no. 7&8, (2014) , arXiv:1305.6190 [quant-ph].

[167] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal clifford gates and noisy ancillas," *Physical Review A* **71** (2005) , arXiv:quant-ph/0403025. http://link.aps.org/doi/10.1103/PhysRevA.71.022316.

[168] R. Jozsa and A. Miyake, "Matchgates and classical simulation of quantum circuits," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* **464** no. 2100, (2008) , arXiv:0804.4050 [quant-ph].

[169] S. Clark, R. Jozsa, and N. Linden, "Generalized clifford groups and simulation of associated quantum circuits," *Quantum Information and Computation* **8** no. 1, (2008) , arXiv:quant-ph/0701103.

[170] S. Hallgren, A. Russell, and A. Ta-Shma, "Normal subgroup reconstruction and quantum computation using group representations," *SIAM Journal on Computing* **32** no. 4, (2003) .

[171] G. Kuperberg, "A subexponential-time quantum algorithm for the dihedral hidden subgroup problem," *SIAM Journal on Computing* **35** no. 1, (2005) , arXiv:quant-ph/0302112.

[172] O. Regev, "A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space," arXiv:quant-ph/0406151.

[173] G. Kuperberg, "Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem," in *Proceedings of TQC13.* 2013. arXiv:1112.3333 [quant-ph].

[174] M. Roetteler and T. Beth, "Polynomial-time solution to the hidden subgroup problem for a class of non-abelian groups," arXiv:quant-ph/9812070.

[175] G. Ivanyos, F. Magniez, and M. Santha, "Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem," arXiv:quant-ph/0102014.

[176] C. Moore, D. Rockmore, A. Russell, and L. Schulman, "The power of basis selection in fourier sampling: the hidden subgroup problem in affine groups.," in *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms.* 2004. arXiv:quant-ph/0211124.

[177] Y. Inui and F. Le Gall, "Efficient quantum algorithms for the hidden subgroup problem over a class of semi-direct product groups," *Quantum Information and Computation* **7** no. 5/6, (2007) , arXiv:0412033 [quant-ph].

[178] D. Bacon, A. M. Childs, and W. van Dam, "From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups," in *Proceedings of the 46th IEEE Symposium on Foundations of Computer Science.* 2005. arXiv:0504083 [quant-ph].

[179] D. P. Chi, J. S. Kim, and S. Lee, "Notes on the hidden subgroup problem on some semi-direct product groups," *Physical Letters A* **359** no. 2, (2006) , arXiv:quant-ph/0604172.

[180] G. Ivanyos, L. Sanselme, and M. Santha, "An efficient quantum algorithm for the hidden subgroup problem in extraspecial groups," in *Proceedings of the 24th Symposium on Theoretical Aspects of Computer Science.* 2007. arXiv:quant-ph/0701235.

[181] C. Magno, M. Cosme, and R. Portugal, "Quantum algorithm for the hidden subgroup problem on a class of semidirect product groups," arXiv:quant-ph/0703223.

[182] G. Ivanyos, L. Sanselme, and M. Santha, "An efficient quantum algorithm for the hidden subgroup problem in nil-2 groups," in *LATIN 2008: Theoretical Informatics*, vol. 4957 of *LNCS.* Springer, 2008. arXiv:0707.1260 [quant-ph].

[183] K. Friedl, G. Ivanyos, F. Magniez, M. Santha, and P. Sen, "Hidden translation and translating coset in quantum computing," in *Proceedings of the 35th ACM Symposium on Theory of Computing*. 2003. arXiv:quant-ph/0211091.

[184] D. Gavinsky, "Quantum solution to the hidden subgroup problem for poly-near-hamiltonian-groups," *Quantum Information and Computation* **4** (2004) .

[185] A. M. Childs and W. van Dam, "Quantum algorithm for a generalized hidden shift problem," in *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*. 2007. arXiv:quant-ph/0507190.

[186] A. Denney, C. Moore, and A. Russell, "Finding conjugate stabilizer subgroups in psl(2;q) and related groups," *Quantum Information and Computation* **10** no. 3, (2010) , arXiv:0809.2445 [quant-ph].

[187] N. Wallach, "A quantum polylog algorithm for non-normal maximal cyclic hidden subgroups in the affine group of a finite field," arXiv:1308.1415 [quant-ph].

[188] A. Childs, *Lecture Notes on Quantum Algorithms*. University of Waterloo, 2011. Published online.

[189] W. van Dam and Y. Sasaki, *Quantum algorithms for problems in number theory, algebraic geometry, and group theory*. World Scientific, 2012. arXiv:1206.6126 [quant-ph].

[190] V. Arvind and N. Vinodchandran, "Solvable black-box group problems are low for pp," *Theoretical Computer Science* **180** (1997) .

[191] L. Babai, "Local expansion of vertex-transitive graphs and random generation in finite groups," in *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91. ACM, 1991.

[192] L. Babai, "Bounded round interactive proofs in finite groups," *SIAM Journal on Discrete Mathematics* **5** no. 1, (Feb., 1992) .

[193] L. Babai, "Randomization in group algorithms: conceptual questions," in *Groups and Computation II*, vol. 28 of *Discrete Mathematics & Theoretical Computer Science*. 1997.

[194] Y. Zhang, "Quantum algorithm for decomposing black-box finite abelian groups," in *Proceedings of the 7th Annual International Conference on Foundations of Computer Science*. 2011.

[195] J. Watrous, "Quantum algorithms for solvable groups," in *Proceedings of the 33rd ACM Symposium on Theory of Computing*. 2001. arXiv:quant-ph/0011023.

[196] F. Magniez and A. Nayak, "Quantum complexity of testing group commutativity," in *Proceedings of 32nd International Colloquium on Automata, Languages and Programming*, vol. 3580 of *LNCS*. 2005. arXiv:quant-ph/0506265.

[197] S. A. Fenner and Y. Zhang, "Quantum algorithms for a set of group theoretic problems," in *Proceedings of the 9th Italian Conference on Theoretical Computer Science*, ICTCS'05. Springer, 2005. http://dx.doi.org/10.1007/11560586_18.

[198] F. Le Gall, "An efficient quantum algorithm for some instances of the group isomorphism problem," in *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS).* 2010. arXiv:1001.0608 [quant-ph].

[199] K. C. Zatloukal, "Classical and quantum algorithms for testing equivalence of group extensions," arXiv:1305.1327 [quant-ph].

[200] L. G. Valiant, "Quantum circuits that can be simulated classically in polynomial time," *SIAM J. Comput.* **31** no. 4, (2002) .

[201] E. Knill, "Fermionic linear optics and matchgates," arXiv:quant-ph/0108033.

[202] B. M. Terhal and D. P. DiVincenzo, "Classical simulation of noninteracting-fermion quantum circuits," *Physical Review A* **65** (2002) , arXiv:quant-ph/0108010.

[203] S. Bravyi, "Lagrangian representation for fermionic linear optics," *Quantum Information and Computation* **5** no. 3, (2005) , arXiv:quant-ph/0404180.

[204] R. Jozsa, B. Kraus, A. Miyake, and J. Watrous, "Matchgate and space-bounded quantum computations are equivalent," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* **466** no. 2115, (2010) .

[205] M. Van den Nest, "Quantum matchgate computations and linear threshold gates," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* **467** no. 2127, (2011) , arXiv:1005.1143 [quant-ph].

[206] S. Bravyi and R. König, "Classical simulation of dissipative fermionic linear optics," *Quantum Information and Computation* **12** no. 11-12, (2012) , arXiv:1112.2184 [quant-ph].

[207] F. de Melo, P. Ćwikliński, and B. M. Terhal, "The power of noisy fermionic quantum computation," *New Journal of Physics* **15** no. 1, (2013) , arXiv:1208.5334 [quant-ph].

[208] A. Ambainis, L. J. Schulman, and U. Vazirani, "Computing with highly mixed states," *J. ACM* **53** no. 3, (2006) , arXiv:quant-ph/0003136.

[209] D. Poulin, R. Laflamme, G. J. Milburn, and J. P. Paz, "Testing integrability with a single bit of quantum information," *Physical Review A* **68** (2003) , arXiv:quant-ph/0303042.

[210] D. Poulin, R. Blume-Kohout, R. Laflamme, and H. Ollivier, "Exponential speedup with a single bit of quantum information: Measuring the average fidelity decay," *Physical Review Letters* **92** (2004) , arXiv:quant-ph/0310038.

[211] D. Shepherd, "Computation with unitaries and one pure qubit," arXiv:quant-ph/0608132.

[212] P. W. Shor and S. P. Jordan, "Estimating Jones polynomials is a complete problem for one clean qubit," *Quantum Information and Computation* **8** no. 8, (2008) , arXiv:0707.2831 [quant-ph].

[213] S. P. Jordan and P. Wocjan, "Estimating Jones and Homfly polynomials with one clean qubit," *Quantum Information and Computation* **9** no. 3, (2009) , arXiv:0807.4688 [quant-ph].

[214] S. P. Jordan and G. Alagic, "Approximating the turaev-viro invariant of mapping tori is complete for one clean qubit," in *Theory of Quantum Computation, Communication, and Cryptography.* Springer Berlin Heidelberg, 2014. arXiv:1105.5100 [quant-ph].

[215] T. Morimae, K. Fujii, and J. F. Fitzsimons, "Hardness of classically simulating the one-clean-qubit model," *Physical Review Letters* **112** (2014) , arXiv:1312.2496 [quant-ph].

[216] S. Aaronson and A. Arkhipov, "The computational complexity of linear optics," in *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing,* STOC '11. ACM, 2011. arXiv:1011.3245 [quant-ph].

[217] V. Veitch, N. Wiebe, C. Ferrie, and J. Emerson, "Efficient simulation scheme for a class of quantum optics experiments with non-negative wigner representation," *New Journal of Physics* **15** no. 1, (2013) , arXiv:1210.1783 [quant-ph].

[218] D. J. Shepherd, *Quantum Complexity: restrictions on algorithms and architectures.* PhD thesis, 2010. arXiv:1005.1425 [quant-ph].

[219] D. Shepherd and M. J. Bremner, "Temporally unstructured quantum computation," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* (2009) , arXiv:0809.0847 [quant-ph].

[220] M. J. Bremner, R. Jozsa, and D. J. Shepherd, "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* **467** (2011) , arXiv:1005.1407 [quant-ph].

[221] M. Van Den Nest, "Universal quantum computation with little entanglement," *Physical Review Letters* **110** (2012) , arXiv:1204.3107 [quant-ph].

[222] R. Jozsa and N. Linden, "On the role of entanglement in quantum-computational speed-up," *Proceedings of the Royal Society of London. Series A. Mathematical, Physical and Engineering Sciences* **459** (2003) , arXiv:quant-ph/0201143.

[223] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Physical Review Letters* **91** (2003) .

[224] B. M. Terhal and D. P. DiVincenzo, "Adaptive Quantum Computation, Constant Depth Quantum Circuits and Arthur-Merlin Games," *Quantum Information and Computation* **4** no. 2, (2014) , arXiv:quant-ph/0205133.

[225] I. Markov and Y. Shi, "Simulating quantum computation by contracting tensor networks," *SIAM Journal on Computing* **38** no. 3, (2008) , arXiv:quant-ph/0511069.

[226] D. Aharonov, Z. Landau, and J. Makowsky, "The quantum FFT can be classically simulated," *arXiv* (2006) , arXiv:quant-ph/0611156v2.

[227] N. Yoran and A. J. Short, "Efficient classical simulation of the approximate quantum Fourier transform," *Physical Review A* **76** (2007) , arXiv:quant-ph/0611241v1.

[228] D. E. Browne, "Efficient classical simulation of the quantum fourier transform," *New Journal of Physics* **9** no. 5, (2007) , arXiv:quant-ph/0612021.

[229] N. Yoran, "Efficiently contractable quantum circuits cannot produce much entanglement," arXiv:0802.1156 [quant-ph].

[230] M. Van den Nest, "Simulating quantum computers with probabilistic methods," *Quantum Information and Computation* **11** no. 9-10, (2011) , arXiv:0911.1624v3 [quant-ph].

[231] D. Stahlke, "Quantum interference as a resource for quantum speedup," *Physical Review A* **90** (2014) , arXiv:1305.2186 [quant-ph].

[232] S. P. Jordan, "Permutational quantum computing," *Quantum Information and Computation* **10** no. 5, (2010) , arXiv:0906.2508 [quant-ph].

[233] M. Schwarz and M. V. d. Nest, "Simulating quantum circuits with sparse output distributions," *Electronic Colloquium on Computational Complexity* (2013) , arXiv:1310.6749 [quant-ph].

[234] J. Bermejo-Vega, "Classical simulations of non-abelian quantum Fourier transforms," Master's thesis, Technical University of Munich, 2011.

[235] P. Sarvepalli and P. Wocjan, "Quantum algorithms for one-dimensional infrastructures," *Quantum Information and Computation* **14** no. 1-2, (2014) , arXiv:1106.6347 [quant-ph].

[236] F. Fontein and P. Wocjan, "Quantum algorithm for computing the period lattice of an infrastructure," *arXiv* (2011) , arXiv:1111.1348 [quant-ph].

[237] S. Hallgren, "Polynomial-time quantum algorithms for pell's equation and the principal ideal problem," *J. ACM* **54** no. 1, (2007) .

[238] R. Jozsa, "Quantum computation in algebraic number theory: Hallgren's efficient quantum algorithm for solving pell's equation," *Annals of Physics* **306** no. 2, (2003) , arXiv:quant-ph/0302134.

[239] A. Schmidt and U. Vollmer, "Polynomial time quantum algorithm for the computation of the unit group of a number field," in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05. ACM, 2005.

[240] S. Hallgren, "Fast quantum algorithms for computing the unit group and class group of a number field," in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05. ACM, 2005.

[241] A. M. Childs and G. Ivanyos, "Quantum computation of discrete logarithms in semigroups," arXiv:1310.6238 [quant-ph].

[242] W. van Dam and G. Seroussi, "Efficient quantum algorithms for estimating Gauss sums," arXiv:quant-ph/0207131.

[243] H. J. Briegel and R. Raussendorf, "Persistent entanglement in arrays of interacting particles," *Physical Review Letters* **86** (2001) , arXiv:quant-ph/0004051v2.

[244] R. Raussendorf and H. J. Briegel, "A one-way quantum computer," *Physical Review Letters* **86** (2001) .

[245] E. Knill, "On Shor's quantum factor finding algorithm: Increasing the probability of success and tradeoffs involving the Fourier transform modulus," 1995. http://www.c3.lanl.gov/~knill/papers/on_shors_alg.ps.gz.

[246] D. Lorenzini, *An Invitation to Arithmetic Geometry.* Graduate Studies in Mathematics. American Mathematical Society, 1997.

[247] H. Cohen, *Advanced Topics in Computational Number Theory.* Graduate Texts in Mathematics. Springer.

[248] D. Nagaj, P. Wocjan, and Y. Zhang, "Fast amplification of QMA," *Quantum Information and Computation* **9** no. 11, (2009) , arXiv:0904.1549 [quant-ph].

[249] A. Childs. http://www.math.uwaterloo.ca/~amchilds/teaching/w13/l15.pdf, 2013.

[250] R. Bhatia, *Matrix Analysis.* Springer-Verlag, 1997.

[251] A. Kleppner, "Multipliers on Abelian groups," *Mathematische Annalen* **158** no. 1, (1965) .